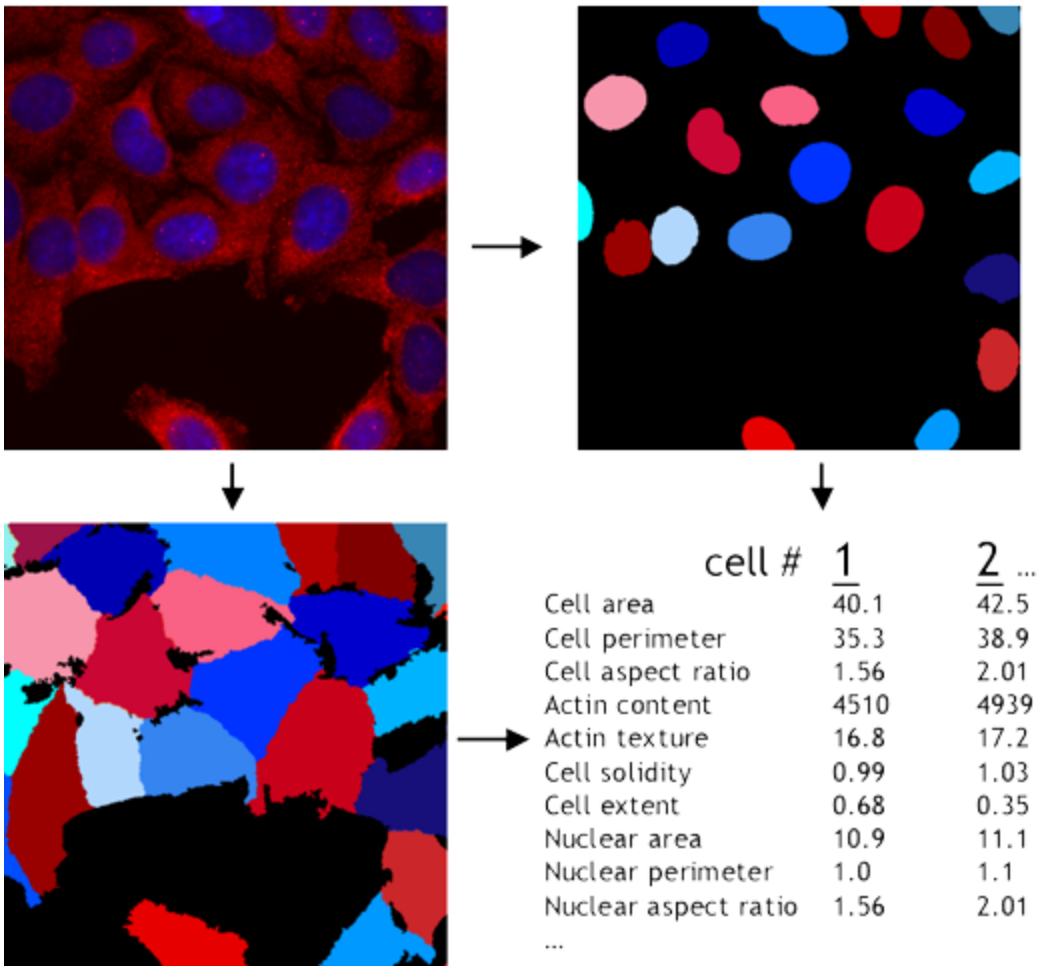
CellProfiler

cell image analysis software



CellProfiler cell image analysis software

Created by
Anne E. Carpenter and Thouis R. Jones

In the laboratories of
David M. Sabatini and Polina Golland at

Whitehead Institute for Biomedical Research

CSAIL
Computer Sciences/ Artificial Intelligence Laboratory at the Massachusetts Institute of Technology

And now based at

CellProfiler is free and open-source!

If you find it useful, please credit CellProfiler in publications

1. Cite the website.
2. Cite the publication.
3. Post the reference for your publication on the CellProfiler forum so that we are aware of it.

These steps will help us to maintain funding for the project and continue to improve and support it.

This manual accompanies version 2014-07-23T17:45:00 6c2d896 of CellProfiler. The most recent manual is available here.

Table of contents

Using CellProfiler

Help for CellProfiler Modules

- MakeProjection
- MaskImage
- Morph
- OverlayOutlines
- RescaleIntensity
- Resize
- RunImageJ
- Smooth
- Tile
- UnmixColors

- **Measurement**
  - CalculateImageOverlap
  - MeasureCorrelation
  - MeasureGranularity
  - MeasureImageAreaOccupied
  - MeasureImageIntensity
  - MeasureImageQuality
  - MeasureNeurons
  - MeasureObjectIntensity
  - MeasureObjectNeighbors
  - MeasureObjectRadialDistribution
  - MeasureObjectSizeShape
  - MeasureTexture
- **Object Processing**
  - ClassifyObjects
  - ConvertObjectsToImage
  - EditObjectsManually
  - ExpandOrShrinkObjects
  - FilterObjects
  - IdentifyObjectsInGrid
  - IdentifyObjectsManually
  - IdentifyPrimaryObjects
  - IdentifySecondaryObjects
  - IdentifyTertiaryObjects
  - MaskObjects
  - ReassignObjectNumbers
  - RelateObjects
  - StraightenWorms
  - TrackObjects
  - UntangleWorms
- **Other**
  - ConserveMemory
  - CreateWebPage
  - DefineGrid
  - IdentifyDeadWorms
  - InputExternal
  - LabelImages
  - OutputExternal
  - SendEmail
- **Worm Toolbox**
  - IdentifyDeadWorms
  - StraightenWorms
  - UntangleWorms

Why Use CellProfiler?

Most laboratories studying biological processes and human disease use light/fluorescence microscopes to image cells and other biological samples. There is strong and growing demand for software to analyze these images, as automated microscopes collect images faster than can be examined by eye and the information sought from images is increasingly quantitative and complex.

CellProfiler is a versatile, open-source software tool for quantifying data from biological images, particularly in high-throughput experiments. CellProfiler is designed for modular, flexible, high-throughput analysis of images, measuring size, shape, intensity, and texture of every cell (or other object) in every image. Using the point-and-click graphical user interface (GUI), users construct an image analysis "pipeline", a sequential series of modules that each perform an image processing function such as illumination correction, object identification (segmentation), and object measurement. Users mix and match modules and adjust their settings to measure the phenotype of interest. While originally designed for high-throughput images, it is equally appropriate for low-throughput assays as well (i.e., assays of < 100 images).

CellProfiler can extract valuable biological information from images quickly while increasing the objectivity and statistical power of assays. It helps researchers approach a variety of biological questions quantitatively, including standard assays (e.g., cell count, size, per-cell protein levels) as well as complex morphological assays (e.g., cell/organelle shape or subcellular patterns of DNA or protein staining).

The wide variety of measurements produced by CellProfiler serves as useful "raw material" for machine learning algorithms. CellProfiler's companion software, CellProfiler Analyst, has an interactive machine learning tool called Classifier which can learn to recognize a phenotype of interest based on your guidance. Once you complete the training phase, CellProfiler Analyst will score every object in your images based on CellProfiler's measurements. CellProfiler Analyst also contains tools for the interactive visualization of the data produced by CellProfiler.

In summary, CellProfiler contains:

- Advanced algorithms for image analysis that are able to accurately identify crowded cells and non-mammalian cell types.
- A modular, flexible design allowing analysis of new assays and phenotypes.
- Open-source code so the underlying methodology is known and can be modified or improved by others.
- A user-friendly interface.
- The capability to make use of clusters of computers when available.
- A design that eliminates the tedium of the many steps typically involved in image analysis, many of which are not easily transferable from one project to another (for example, image formatting, combining several image analysis steps, or repeating the analysis with slightly different parameters).

References

For a full list of references, visit our citation page.

- Carpenter AE, Jones TR, Lamprecht MR, Clarke C, Kang IH, Friman O, Guertin DA, Chang JH, Lindquist RA, Moffat J, Golland P, Sabatini DM (2006) "CellProfiler: image analysis software for identifying and quantifying cell phenotypes" *Genome Biology* 7:R100 (link)
- Kamentsky L, Jones TR, Fraser A, Bray MA, Logan D, Madden K, Ljosa V, Rueden C, Harris GB, Eliceiri K, Carpenter AE (2011) "Improved structure, function, and compatibility for CellProfiler: modular high-throughput image analysis software" *Bioinformatics* 27(8):1179-1180 (link)

- Lamprecht MR, Sabatini DM, Carpenter AE (2007) "CellProfiler: free, versatile software for automated biological image analysis" *Biotechniques* 42(1):71-75. [link)
- Jones TR, Carpenter AE, Lamprecht MR, Moffat J, Silver S, Grenier J, Root D, Golland P, Sabatini DM (2009) "Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning" *PNAS* 106(6):1826-1831 (link)
- Jones TR, Kang IH, Wheeler DB, Lindquist RA, Papallo A, Sabatini DM, Golland P, Carpenter AE (2008) "CellProfiler Analyst: data exploration and analysis software for complex image-based screens" *BMC Bioinformatics* 9(1):482 (link)

Using the File Menu

The *File* menu provides options for loading and saving your pipelines and performing an analysis run.

- **New project:** Clears the current project by removing all the analysis modules and resetting the input modules.
- **Open Project...:** Open a previously saved CellProfiler project (*.cpproj* file) from your hard drive.
- **Open Recent:** Displays a list of the most recent projects used. Select any one of these projects to load it.
- **Save Project:** Save the current project to your hard drive as a *.cpproj* file. If it has not been saved previously, you will be asked for a file name to give the project. Thereafter, any changes to the project will be automatically saved to that filename unless you choose **Save as...**.
- **Save Project As...:** Save the project to a new file name.
- **Revert to Saved:** Restore the currently open project to the settings it had when it was first opened.
- **Import Pipeline:** Gives you the choice of importing a CellProfiler pipeline file from your hard drive (*From file...*) or from a web address (*From URL...*). If importing from a file, you can point it to a pipeline (*.cppipe*) file or have it extract the pipeline from a project (*.cpproj*) file.
- **Export:** You have the choice of exporting the pipeline you are currently working on as a CellProfiler *.cppipe* pipeline file (*Pipeline*), or the image set list as a CSV (*Image set listing*).
- **Clear Pipeline:** Removes all modules from the current pipeline.
- **View Image:** Opens a dialog box prompting you to select an image file for display. Images listed in the File list panel in the **Images** module can be also be displayed by double-clicking on the filename.
- **Analyze Images:** Executes the current pipeline using the current pipeline and Default Input and Output folder settings.
- **Stop Analysis:** Stop the current analysis run.
- **Run Multiple Pipelines:** Execute multiple pipelines in sequential order. This option opens a dialog box allowing you to select the pipelines you would like to run, as well as the associated input and output folders. See the help in the *Run Multiple Pipelines* dialog for more details.
- **Resume Pipeline:** Resume a partially completed analysis run from where it left off. You will be prompted to choose the output *.h5/.mat* file containing the partially complete measurements and the analysis run will pick up starting with the last cycle that was processed.
- **Preferences...:** Displays the Preferences window, where you can change many options in CellProfiler.
- **Exit:** End the current CellProfiler session. You will be given the option of saving your current pipeline if you have not done so.

Using the Edit Menu

The *Edit* menu provides options for modifying modules in your current pipeline.

- **Undo:** Undo the last module modification. You can undo multiple actions by using *Undo* repeatedly.
- **Cut:** If a module text setting is currently active, remove the selected text.
- **Copy:** Copy the currently selected text to the clipboard.
- **Paste:** Paste clipboard text to the cursor location, if a text setting is active.
- **Select All:** If a text setting is active, select all the text in the setting. If the module list is active, select all the modules in the module list.
- **Move Module Up:** Move the currently selected module(s) up. You can also use the  button located below the Pipeline panel.
- **Move Module Down:** Move the currently selected module(s) down. You can also use the  button located below the Pipeline panel.
- **Delete Module:** Remove the currently selected module(s). Pressing the Delete key also removes the module(s). You can also use the  button located under the Pipeline panel.
- **Duplicate Module:** Duplicate the currently selected module(s) in the pipeline. The current settings of the selected module(s) are retained in the duplicate.
- **Add Module:** Select a module from the pop-up list to inster into the current pipeline. You can also use the  button located under the Pipeline panel.

You can select multiple modules at once for moving, deletion and duplication by selecting the first module and using Shift-click on the last module to select all the modules in between.

Using the Test Menu

Before starting an analysis run, you can test the pipeline settings on a selected image cycle using the *Test* mode option on the main menu. Test mode allows you to run the pipeline on a selected image, preview the results and adjust the module settings on the fly.

To enter Test mode once you have built a pipeline, choose *Test* > *Start Test Mode* from the menu bar in the main window. At this point, you will see the following features appear:

- The module view will have a slider bar appearing on the far left.
- A Pause icon  will appear to the left of each module.
- A series of buttons will appear at the bottom of the pipeline panel above the module adjustment buttons.
- The grayed-out items in the *Test* menu will become active, and the *Analyze Images* button will become inactive.

You can run your pipeline in Test mode by selecting *Test* > *Step to Next Module* or clicking the *Run* or *Step* buttons at the bottom of the pipeline panel. The pipeline will execute normally, but you will be able to back up to a previous module or jump to a downstream module, change module settings to see the results, or execute the pipeline on the image of your choice. The additional controls allow you to do the following:

- *Slider:* Start/resume execution of the pipeline at any time by moving the slider. However, if the selected module depends on objects and/or images generated by prior modules, you will see an error message indicating that the data has not been produced yet. To avoid this, it is best to actually run the pipeline up to the module of interest, and move the slider to modules already executed.
- *Pause:* Clicking the pause icon will cause the pipeline test run to halt execution when that module is reached (the paused module itself is not executed). The icon changes from  to to indicate that a pause has been inserted at that point.
- *Run:* Execution of the pipeline will be started/resumed until the next module pause is reached. When all modules have been executed for a given image cycle, execution will stop.
- *Step:* Execute the next module (as indicated by the slider location)
- *Next Image:* Skip ahead to the next image cycle as determined by the image order in the Input modules. The slider will automatically return to the first module in the pipeline.

From the *Test* menu, you can choose additional options:

- *Exit Test Mode:* Exit *Test* mode. Loading a new pipeline or adding/subtracting modules will also automatically exit test mode.
- *Step to Next Module:* Execute the next module (as indicated by the slider location)
- *Next Image Set:* Step to the next image set in the current image group.
- *Next Image Group:* Step to the next group in the image set. The slider will then automatically return to the first module in the pipeline.
- *Random Image Set:* Randomly select and jump to an image set in the current image group.
- *Choose Image Set:* Choose the image set to jump to. The slider will then automatically return to the first module in the pipeline.
- *Choose Image Group:* Choose an image group to jump to. The slider will then automatically return to the first module in the pipeline.
- *Reload Modules Source (enabled only if running from source code):* This option will reload the module source code, so any changes to the code will be reflected immediately.

Note that if movies are being loaded, the individual movie is defined as a group automatically. Selecting

Choose Image Group will allow you to choose the movie file, and *Choose Image Set* will let you choose the individual movie frame from that file.

Please see the **Groups** module for more details on the proper use of metadata for grouping

Using the Window Menu

The *Windows* menu provides options for showing and hiding the module display windows.

- **Close All Open Windows:** Closes all display windows that are currently open.
- **Show All Windows On Run:** Select to show all display windows during the current test run or next analysis run. The display mode icons next to each module in the pipeline panel will switch to  .
- **Hide All Windows On Run:** Select to show no display windows during the current test run or next analysis run. The display mode icons next to each module in the pipeline panel will switch to  .

If there are any open windows, the window titles are listed underneath these options. Select any of these window titles to bring that window to the front.

Using the Parameter Sampling Menu

The *Sampling* menu is an interface for Paramorama, a plugin for an interactive visualization program for exploring the parameter space of image analysis algorithms.

This menu option is only shown if specified in the Preferences. Note that if this preference setting is changed, CellProfiler must be restarted.

Using this plugin will allow you sample a range of setting values in **IdentifyPrimaryObjects** and save the object identification results for later inspection. Upon completion, the plug-in will generate a text file, which specifies: (1) all unique combinations of the sampled parameter values; (2) the mapping from each combination of parameter values to one or more output images; and (3) the actual output images.

More information on how to use the plugin can be found here.

References

- Pretorius AJ, Bray MA, Carpenter AE and Ruddle RA. (2011) "Visualization of parameter space for image analysis" *IEEE Transactions on Visualization and Computer Graphics* 17(12), 2402-2411.

Using the Data Tools Menu

The *Data Tools* menu provides tools to allow you to plot, view, export or perform specialized analyses on your measurements.

Each data tool has a corresponding module with the same name and functionality. The difference between the data tool and the module is that the data tool takes a CellProfiler output file (i.e., a *.mat or .h5* file) as input, which contains measurements from a previously completed analysis run. In contrast, a module uses measurements received from the upstream modules during an in-progress analysis run.

Opening a data tool will present a prompt in which the user is asked to provide the location of the output file. Once specified, the user is then prompted to enter the desired settings. The settings behave identically as those from the corresponding module.

Help for each *Data Tool* is available under *Help* > *Data Tool Help* or the corresponding module help.

Using The Display Window Menu Bar

From the menu bar of each module display window, you have the following options:

- **File**
  - *Save:* You can save the figure window to an image file. Note that this will save the entire contents of the window, not just the individual subplot(s) or images.
  - *Save table:* This option is only enabled on windows which are displaying tabular output, such as that from a **Measure** module. This allows you to save the tabular data to a comma-delimited file (CSV).
- **Tools**
  - *Measure length:* Select this option to measure distances within an image window. If you click on an image and drag, a line will appear between the two endpoints, and the distance between them shown at the right-most portion of the bottom panel. This is useful for measuring distances in order to obtain estimates of typical object diameters for use in **IdentifyPrimaryObjects**.
- **Subplots:** If the module display window has multiple subplots (such as **IdentifyPrimaryObjects**), the Image Tool options for the individual subplots are displayed here. See *Help > Using Module Display Windows > How To Use The Image Tools* for more details.

Using The Interactive Navigation Toolbar

All figure windows come with a navigation toolbar, which can be used to navigate through the data set.

- **Home, Forward, Back buttons:** *Home*  always takes you to the initial, default view of your data. The *Forward*  and *Back*  buttons are akin to the web browser forward and back buttons in that they are used to navigate back and forth between previously defined views, one step at a time. They will not be enabled unless you have already navigated within an image else using the **Pan** and **Zoom** buttons, which are used to define new views.
- **Pan/Zoom button:** This button has two modes: pan and zoom. Click the toolbar button  to activate panning and zooming, then put your mouse somewhere over an axes, where it will turn into a hand icon.
  - *Pan:* Press the left mouse button and hold it to pan the figure, dragging it to a new position. Press Ctrl+Shift with the pan tool to move in one axis only, which one you have moved farther on.
    Keep in mind that that this button will allow you pan outside the bounds of the image; if you get lost, you can always use the **Home** to back you back to the initial view.
  - *Zoom:* You can zoom in and out of a plot by pressing Ctrl (Mac) or holding down the right mouse button (Windows) while panning.
    Once you're done, the right click menu will pop up when you're done with the action; dismiss it by clicking off the plot. This is a known bug to be corrected in the next release.
- **Zoom-to-rectangle button:** Click this toolbar button  to activate this mode. To zoom in, press the left mouse button and drag in the window to draw a box around the area you want to zoom in on. When you release the mouse button, the image is re-drawn to display the specified area. Remember that you can always use *Backward* button to go back to the previous zoom level, or use the *Home* button to reset the window to the initial view.
- **Save:** Click this button  to launch a file save dialog. You can save the figure window to an image file. Note that this will save the entire contents of the window, not just the individual subplot(s) or images.

How To Use The Image Tools

Right-clicking in an image displayed in a window will bring up a pop-up menu with the following options:

- *Open image in new window:* Displays the image in a new display window. This is useful for getting a closer look at a window subplot that has a small image.
- *Show image histogram:* Produces a new window containing a histogram of the pixel intensities in the image. This is useful for qualitatively examining whether a threshold value determined by **IdentifyPrimaryObjects** seems reasonable, for example. Image intensities in CellProfiler typically range from zero (dark) to one (bright). If you have an RGB image, the histogram shows the intensity values for all three channels combined, even if one or more channels is turned off for viewing.
- *Image contrast:* Presents three options for displaying the color/intensity values in the images:
    - *Raw:* Shows the image using the full colormap range permissible for the image type. For example, for a 16-bit image, the pixel data will be shown using 0 as black and 65535 as white. However, if the actual pixel intensities span only a portion of the image intensity range, this may render the image unviewable. For example, if a 16-bit image only contains 12 bits of data, the resulting image will be entirely black.
    - *Normalized (default):* Shows the image with the colormap "autoscaled" to the maximum and minimum pixel intensity values; the minimum value is black and the maximum value is white.
    - *Log normalized:* Same as *Normalized* except that the color values are then log transformed. This is useful for when the pixel intensity spans a wide range of values but the standard deviation is small (e.g., the majority of the interesting information is located at the dim values). Using this option increases the effective contrast.
- *Interpolation:* Presents three options for displaying the resolution in the images. This is useful for specifying the amount of detail that you want to be visible if you zoom in:
    - *Nearest neighbor:* Use the intensity of the nearest image pixel when displaying screen pixels at sub-pixel resolution. This produces a blocky image, but the image accurately reflects the data.
    - *Linear:* Use the weighted average of the four nearest image pixels when displaying screen pixels at sub-pixel resolution. This produces a smoother, more visually-appealing image, but makes it more difficult to find pixel borders.
    - *Cubic:* Perform a bicubic interpolation of the nearby image pixels when displaying screen pixels at sub-pixel resolution. This produces the most visually-appealing image but is the least faithful to the image pixel values.
- *Save subplot:* Save the clicked subplot as an image file. If there is only one p lot in the figure, this option will save that one.
- *Channels:* For color images only. You can show any combination of the red, green, and blue color channels.

Introduction to Projects

What is a project?

In CellProfiler, a *project* is comprised of two elements:

- An *image file list* which is the list of files and their locations that are selected by the user as candidates for analysis.
- The *pipeline*, which is a series of modules put together used to analyze a set of images.
- Optionally, the associated information about the images (*metadata*). This information may be part of the images themselves, or imported externally by the user.

The project is the container for image information associated with a CellProfiler analysis. It stores such details as:

- What type of image(s) are the input files?
- Where are the input images located?
- What distinguishes multiple image channels from each other? How are these relationships represented?
- What information about the images and/or experiment is linked to the images, and how?
- Are certain groups of images to be processed differently from other groups?

By using projects, the above information is stored along with the analysis pipeline and is available on demand.

Working with projects

Creating a project

Upon starting CellProfiler, you will be presented with a new, blank project. At this point, you may start building your project by using the modules located in the "Input modules" panel on the upper-left. The modules are:

- **Images**: Assemble the relevant images for analysis (required).
- **Metadata**: Asociate metadata with the images (optional).
- **NamesAndTypes**: Assign names to channels and define their relationship (required).
- **Groups**: Define sub-divisions between groups of images for processing (optional).

Detailed help for each module is provided by selecting the module and clicking the "?" button on the bottom of CellProfiler.

Saving a project

As you work in CellProfiler, the project is updated automatically, so there is no need to save it unless you are saving the project to a new name or location. You can always save your current work to a new project file by selecting *File > Save Project As...*, which will save your project, complete with the current image file list and pipeline, to a file with with the extension *.cpproj*.

You also have the option of automatically saving the associated pipeline file and the file list in addition to the project file. See *File > Preferences...* for more details.

For those interested, some technical details:

- The *.cpproj* file stores collected information using the HDF5 format. Documentation on how measurements are stored and handled in CellProfiler using this format can be found here.
- All information is cached in the project file after it is computed. It is either re-computed or retrieved from the cache when an analysis run is started, when entering Test mode, or when the user requests a refreshed view of the information (e.g., when a setting has been changed).

Legacy modules: LoadImages and LoadData

Historically, two modules were used for project creation: **LoadImages** and **LoadData**. While the approach described above partly supercedes these modules, you have the option of preserving these modules if you load old pipelines into CellProfiler that contain them; these pipelines will operate exactly as before.

Alternately, the user can choose to convert these modules into the project equivalent as closely as possible. Both **LoadImages** and **LoadData** remain accessible via the "Add module" and  buttons at the bottom of the pipeline panel.

Selecting Images for Input

Any image analysis project using CellProfiler begins with providing the program with a set of image files to be analyzed. You can do this by clicking on the **Images** module to select it (located in the Input modules panel on the left); this module is responsible for collecting the names and locations of the files to be processed.

The most straightforward way to provide files to the **Images** module is to simply drag-and-drop them from your file manager tool (e.g., Windows Explorer, Finder) onto the file list panel (the blank space indicated by the text "Drop files and folders here"). Both individual files and entire folders can be dragged onto this panel, and as many folders and files can be placed onto this panel as needed. As you add files, you will see a listing of the files appear in the panel.

CellProfiler supports a wide variety of image formats, including most of those used in imaging, by using a library called Bio-Formats; see here for the formats available. Some image formats are better than others for image analysis. Some are "lossy" (information is lost in the conversion to the format) like most JPG/JPEG files; others are lossless (no image information is lost). For image analysis purposes, a lossless format like TIF or PNG is recommended.

If you have a subset of files that you want to analyze from the full list shown in the panel, you can also filter the files according to a set of rules that you specify. This is useful when, for example, you have dragged a folder of images onto the file list panel, but the folder contains the images from one experiment that you want to process along with images from another experiment that you want to ignore for now. You may specify as many rules as necessary to define the desired list of images.

For more information on this module and how to configure it for the best performance, please see the detailed help by selecting the module and clicking the  button at the bottom of the pipeline panel, or check out the Input module tutorials on our Tutorials page.

Configuring Images for Analysis

Once you have used the **Images** module to produce a list of images to be analyzed, you can use the other Input modules to define how images are related to one another, give them a memorable name for future reference, attach additional image information about the experiment, among other things.

After **Images**, you can use the following Input modules:

Module	Description	Use required?	Usage notes
Metadata	Associate image information (metadata) with the images	No	With this module, you can extract metadata from various sources and append it to the measurements that your pipeline will collect, or use it to define how the images are related to each other. The metadata can come from the image filename or location, or from a spreadsheet that you provide. If your assay does not require or have such information, this module can be safely skipped.
NamesAndTypes	Assign names to images and/or channels and define their relationship.	Yes	This module gives each image a meaningful name by which modules in the analysis pipeline will refer to it. The most common usage for this module is to define a collection of channels that represent a single field of view. By using this module, each of these channels will be loaded and processed together for each field of view.
Groups	Define sub-divisions between groups of images for processing.	No	For some assays, you will need the option of further sub-dividing an image set into *groups* that share a common feature. An example of this is a time-lapse movie that consists of individual files; each group of files that define a single movie needs to be processed independently of the others. This module allows you to specify what distinguishes one group of images from another. If your assay does not require this sort of behavior, this module can be safely skipped.

For more information on these modules and how to configure them for the best performance, please see the detailed help by selecting the module and clicking the  button at the bottom of the pipeline panel, or check out the Input module tutorials on our Tutorials page.

Loading Image Stacks and Movies

Introdution

In this context, the term *image sequence* is used to refer to a collection of images which can be from a time-lapse assay, a three-dimensional (3-D) Z-stack assay, or both. This section will instruct you how to load these collections in order to properly represent your data for processing.

Sequences of individual files

For some microscopes, the simplest method of capturing image sequences is to simply acquire them as a series of individual images, where each image represents a single timepoint/z-slice (for simplicity, we will refer to *timepoints* in the rest of this example). Typically, the image filename reflects the timepoint, such that the alphabetical image listing corresponds to the proper sequence, e.g., *img000.png*, *img001.png*, *img002.png*, etc

. It is also not uncommon to store the movie such that one movie's worth of files is stored in a single folder.

*Example:*You have a time-lapse movie of individual files set up as follows:

- Three folders, one for each image channel, named *fluo2*, *fluor* and *phase*.
- In each folder, the files are named as follows:
  - *fluo2*: calibrate2-P01.001.TIF, calibrate2-P01.002.TIF,..., calibrate2-P01.287.TIF
  - *fluor*: calibrated-P01.001.TIF, calibrated-P01.002.TIF,..., calibrated-P01.287.TIF
  - *phase*: phase-P01.001.TIF, phase-P01.002.TIF,..., phase-P01.287.TIF

  where the file names are in the format *<Stain>-<Well>.<Timepoint>.TIF*.
- There are 287 timepoints per movie, and a movie of the 3 channels above is acquired from each well.

In this case, the procedure to set up the input modules to handle these files is as follows:

- In the **Images** module, drag-and-drop your folders of images into the File list panel. If necessary, set your rules accordingly in order to filter out any files that are not part of a movie sequence.

  In the above example, you would drag-and-drop the *fluo2*, *fluor* and *phase* folders into the File list panel.

- In the **Metadata** module, check the box to enable metadata extraction. The key step here is to obtain the metadata tags necessary to do two things:
  - Distinguish the movies from each other. This information is typically encapsulated in the filename and/or the folder name.
  - For each movie, distinguish the timepoints from each other, ensuring their proper ordering. This information is usually contained in the filename.

  To accomplish this, do the following:
  - Select "Extract from file/folder names" or "Import from file" as the metadata extraction method. You will use these to extract the movie and timepoint tags from the images.
  - Use "Extract from file/folder names" to create a regular expression to extract the metadata from the filename and/or path name.
  - Or, use "Import from file" if you have a comma-delimted file (CSV) of the necessary metadata columns (including the movie and timepoint tags) for each image.

  If there are multiple channels for each movie, this step may need to be performed for each channel.

In this example, you could do the following:

  - Select "Extract from file/folder names" as the method, "From file name" as the source, and `.*-(?P<Well>[A-P][0-9]{2})\.(?P<Timepoint>[0-9]{3})` as the regular expression. This step will extract the well ID and timepoint from each filename.
  - Click the "Add" button to add another extraction method.
  - In the new group of extraction settings, select "Extract from file/folder names" as the method, "From folder name" as the source, and `.*[\/](?P<Stain>.*)[\/].*$` as the regular expression. This step will extract the stain name from each folder name.
  - Click the "Update" button below the divider and check the output in the table to confirm that the proper metadata values are being collected from each image.

- In the **NamesAndTypes** module, assign the channel(s) to a name of your choice. If there are multiple channels, you will need to do this for each channel.
  For this example, you could do the following:
  - Select "Assign images matching rules".
  - Make a new rule `[Metadata][Does][Have Stain matching][fluor]` and name it *OrigFluor*.
  - Click the "Add" button to define another image with a rule.
  - Make a new rule `[Metadata][Does][Have Stain matching][fluo2]` and name it *OrigFluo2*.
  - Click the "Add" button to define another image with a rule.
  - Make a new rule `[Metadata][Does][Have Stain matching][phase]` and name it *OrigPhase*.
  - In the "Image set matching method" setting, select "Metadata".
  - Select "Well" for the *OrigFluor*, *OrigFluo2*, and *OrigPhase* channels.
  - Click the  button to the right to add another row, and select "Timepoint" for each channel.
  - Click the "Update" button below the divider to view the resulting table and confirm that the proper files are listed and matched across the channels. The corresponding well and frame for each channel should now be matched to each other.
- In the **Groups** module, enable image grouping for these images in order to select the metadata that defines a distinct movie of data.
  For the example above, do the following:
  - Select "Well" as the metadata category.
  - The tables below this setting will update themselves, and you should be able to visually confirm that each well is defined as a group, each with 287 frames' worth of images.

  Without this step, CellProfiler would not know where one movie ends and the next one begins, and would process the images in all movies together as if they were constituents of only one movie.

Basic image sequences consisting of a single file

Another common means of storing time-lapse/Z-stack data is as a single file containing the movie. Examples of this approach include image formats such as:

- Multi-frame TIF
- Metamorph stack: STK
- Evotec/PerkinElmer Opera Flex
- Zeiss ZVI, LSM
- Standard movie formats: AVI, Quicktime MOV, etc

CellProfiler uses the Bio-Formats library for reading various image formats. For more details on supported files, see this webpage. In general, we recommend saving stacks and movies in a format such as .TIF.

*Example:*You have two image stacks in the following format:

- The stacks are Opera's FLEX format.

- Each FLEX file contains 8 fields of view, with 3 channels at each site (DAPI, GFP, Texas Red).
- Each channel is in grayscale format.

In this case, the procedure to set up the input modules to handle these files is as follows (please note that this procedure is basically identical whether the file is for a time-lapse assay or a Z-stack assay):

- In the **Images** module, drag-and-drop your folders of images into the File list panel. If necessary, set your rules accordingly in order to filter out any files that are not files that are not images to be processed.
  In the above example, you would drag-and-drop the FLEX files into the File list panel.
- In the **Metadata** module, enable metadata extraction in order to obtain metadata from these files. The key step here is to obtain the necessary metadata tags to do two things:
  - Distinguish the stacks from each other. This information is contained as the file itself, that is, each file represents a different stack.
  - For each stack, distinguish the frames from each other. This information is usually contained in the image's internal metadata, in contrast to the image sequence described above.

  To accomplish this, do the following:
  - Select "Extract from image file headers" as the metadata extraction method. In this case, CellProfiler will extract the requisite information from the metadata stored in the image headers.
  - Click the "Update metadata" button. A progress bar will appear showing the time elapsed; depending on the number of files present, this step may take a while to complete.
  - Click the "Update" button below the divider.
  - The resulting table should show the various metadata contained in the file. In this case, the relevant information is contained in the *C* and *Series* columns. In the figure shown, the *C* column shows three unique values for the channels represented, numbered from 0 to 2. The *Series* column shows 8 values for the slices collected in each stack, numbered from 0 to 7, followed by the slices for other stacks.
- In the **NamesAndTypes** module, assign the channel to a name of your choice. If there are multiple channels, you will need to do this for each channel. For this example, you could do the following:
  - Select "Assign images matching rules".
  - Make a new rule `[Metadata][Does][Have C matching][0]`
  - Click the + button to the right of the rule to add another set of rules underneath.
  - Add the rule `[Image][Is][Stack frame]`. This combination tells CellProfiler not to treat the image as a single file, but rather as a series of frames.
  - Name the image *DAPI*.
  - Click the "Add another image" button to define a second image with a set of rules.
  - Make a new rule `[Metadata][Does][Have C matching][1]`
  - Click the + button to the right of the rule to add another set of rules underneath.
  - Add the rule `[Image][Is][Stack frame]`.
  - Name the image *GFP*.
  - Click the "Add another image" button to define a third image with a set of rules.
  - Make a new rule `[Metadata][Does][Have C matching][2]`
  - Click the + button to the right of the rule to add another set of rules underneath.
  - Add the rule `[Image][Is][Stack frame]`.
  - Name the image *TxRed*.
  - In the "Image set matching method" setting, select "Metadata".
  - Select "FileLocation" for the DAPI, GFP and TxRed channels. The FileLocation metadata tag identifies the individual stack, and selecting this parameter ensures that the channels are first matched within each stack, rather than across stacks.
  - Click the + button to the right to add another row, and select *Series* for each channel.
  - Click the "Update" button below the divider to confirm that the proper image slices are listed and matched across the channels. The corresponding *FileLocation* and *Series* for each

channel should now be matched to each other.
- In the **Groups** module, select the metadata that defines a distinct image stack. For the example above, do the following:
    - Select "FileLocation" as the metadata category.
    - The tables below this setting will update themselves, and you should be able to visually confirm that each of the two image stacks are defined as a group, each with 8 slices' worth of images.

  Without this step, CellProfiler would not know where one stack ends and the next one begins, and would process the slices in all stacks together as if they were constituents of only one stack.

*Example:*You have two Z-stacks in the following format:

- The stacks are in Zeiss' CZI format.
- Each stack consists of a number of slices with 4 channels (DAPI, GFP, Texas Red and Cy3) at each slice.
- One stack has 9 slices, two stacks have 7 slices and the fourth has 12 slices. Even though the stacks were collected with differing numbers of slices, the pipeline to be constructed is intended to analyze all stacks in the same manner.
- Each slice is in grayscale format.

In this case, the procedure to set up the input modules to handle these this file is as follows (please note that this procedure is basically identical whether the file is for a time-lapse assay or a Z-stack assay):

- In the **Images** module, drag-and-drop your folders of images into the File list panel. If necessary, set your rules accordingly in order to filter out any files that are not images to be processed.
  In the above example, you would drag-and-drop the LSM files into the File list panel. In this case, the default "Images only" filter is sufficient to capture the necessary files.
- In the **Metadata** module, enable metadata extraction in order to obtain metadata from these files.
  The key step here is to obtain the metadata tags necessary to do two things:
    - Distinguish the stacks from each other. This information is contained as the file itself, that is, each file represents a different stack.
    - For each stack, distinguish the timepoints from each other, ensuring proper ordering. This information is usually contained in the image file's internal metadata.

  To accomplish this, do the following:
    - Select "Extract from image file headers" as the metadata extraction method. In this case, CellProfiler will extract the requisite information from the metadata stored in the image headers.
    - Click the "Update metadata" button. A progress bar will appear showing the time elapsed; depending on the number of files present, this step may take a while.
    - Click the "Update" button below the divider.
    - The resulting table should show the various metadata contained in the file. In this case, the relevant information is contained in the C and Z columns. The *C* column shows four unique values for the channels represented, numbered from 0 to 3. The *Z* column shows nine values for the slices represented from the first stack, numbered from 0 to 8.
    - Of note in this case, for each file there is a single row summarizing this information. The *sizeC* column reports a value of 4 and *sizeZ* column shows a value of 9. You may need to scroll down the table to see this summary for the other stacks.
- In the **NamesAndTypes** module, assign the channel(s) to a name of your choice. If there are multiple channels, you will need to do this for each channel.

  For the above example, you could do the following:

    - Select "Assign images matching rules".
    - Make a new rule `[Metadata][Does][Have C matching][0]`

- Click the + button to the right of the rule to add another set of rule options.
- Add the rule `[Image][Is][Stack frame].`
- Name the image *DAPI*.
- Click the "Add another image" button to define a second image with a set of rules.
- Make a new rule `[Metadata][Does][Have C matching][1]`
- Click the + button to the right of the rule to add another set of rule options.
- Add the rule `[Image][Is][Stack frame].`
- Name the second image *GFP*.
- Click the "Add another image" button to define a third image with a set of rules.
- Make a new rule `[Metadata][Does][Have C matching][2].`
- Click the + button to the right of the rule to add another set of rule options.
- Add the rule `[Image][Is][Stack frame].`
- Name the third image *TxRed*.
- Click the "Add another image" button to define a fourth image with set of rules.
- Make a new rule `[Metadata][Does][Have C matching][3].`
- Click the + button to the right of the rule to add another set of rule options.
- Add the rule `[Image][Is][Stack frame].`
- Name the fourth image *Cy3*.
- In the "Image set matching method" setting, select "Metadata".
- Select "FileLocation" for the *DAPI*,*GFP*,*TxRed*, and *Cy3*channels. The *FileLocation* identifies the individual stack, and selecting this parameter insures that the channels are matched within each stack, rather than across stacks.
- Click the + button to the right to add another row, and select "Z" for each channel.
- Click "Update table" to confirm the channel matching. The corresponding *FileLocation* and *Z* for each channel should be matched to each other.

- In the **Groups** module, select the metadata that defines a distinct image stack. For the example above, do the following:
  - Select "FileLocation" as the metadata category.
  - The tables below this setting will update themselves, and you should be able to visually confirm that each of the four image stacks are defined as a group, with 9, 7, 7 and 12 slices' worth of images.

  Without this step, CellProfiler would not know where one stack ends and the next one begins, and would process the slices in all stacks together as if they were constituents of only one stack.

How To Build A Pipeline

A *pipeline* is a sequential set of image analysis modules. The best way to learn how to use CellProfiler is to load an example pipeline from the CellProfiler website's Examples page and try it out, then adapt it for your own images. You can also build a pipeline from scratch. Click the *Help*  button in the main window to get help for a specific module.

Loading an existing pipeline

1. Put the images and pipeline into a folder on your computer.
2. Set the Default Output Folder (press the "View output settings") to the folder where you want to place your output (preferably a different location than in the input images).
3. Load the pipeline using *File* > *Import Pipeline* > *From File...* in the main menu of CellProfiler.
4. Click the *Analyze Images* button to start processing.
5. Examine the measurements using *Data tools*. The *Data tools* options are accessible in the main menu of CellProfiler and allow you to plot, view, or export your measurements (e.g., to Excel).
6. If you modify the modules or settings in the pipeline, you can save the pipeline using *File* > *Export* > *Pipeline...*. Alternately, you can save the project as a whole using *File* > *Save Project* or *Save Project As...* which also saves the file list.
7. To learn how to use a cluster of computers to process large batches of images, see *Help* > *Other Features* > *Batch Processing*.

Building a pipeline from scratch

Constructing a pipeline involves placing individual modules into a pipeline. The list of modules in the pipeline is shown in the *pipeline panel* (located on the left-hand side of the CellProfiler window).

1. *Place analysis modules in a new pipeline.*

   Choose image analysis modules to add to your pipeline by clicking the *Add*  button (located underneath the pipeline panel) or right-clicking in the pipeline panel itself and selecting a module from the pop-up box that appears.

   You can learn more about each module by clicking *Module Help* in the "Add modules" window or the *?* button after the module has been placed and selected in the pipeline. Modules are added to the end of the pipeline or after the currently selected module, but you can adjust their order in the main window by dragging and dropping them, or by selecting a module (or modules, using the *Shift* key) and using the *Move Module Up*  and *Move Module Down*  buttons. The *Remove Module* button will delete the selected module(s) from the pipeline.

   Most pipelines depend on one major step: identifying the objects. In CellProfiler, the objects you identify are called *primary*, *secondary*, or *tertiary*:

   - **IdentifyPrimary** modules identify objects without relying on any information other than a single grayscale input image (e.g., nuclei are typically primary objects).
   - **IdentifySecondary** modules require a grayscale image plus an image where primary objects have already been identified, because the secondary objects are determined based on the primary objects (e.g., cells can be secondary objects when their identification is based on the location of nuclei).
   - **IdentifyTertiary** modules require images in which two sets of objects have already been identified (e.g., nuclei and cell regions are used to define the cytoplasm objects, which are

tertiary objects).

2. *Adjust the settings in each module.*
   In the CellProfiler main window, click a module in the pipeline to see its settings in the settings panel. To learn more about the settings for each module, select the module in the pipeline and click the *Help* button to the right of each setting, or at the bottom of the pipeline panel for the help for all the settings for that module.

   If there is an error with the settings (e.g., a setting refers to an image that doesn't exist yet), a  icon will appear next to the module name. If there is a warning (e.g., a special notification attached to a choice of setting), a  icon will appear. Errors will cause the pipeline to fail upon running, whereas a warning will not. Once the errors/warnings have been resolved, a  icon will appear indicating that the module is ready to run.

3. *Set your Default Input Folder, Default Output Folder and output filename.*
   For more help, click their nearby *Help* buttons in the main window.

4. *Click Analyze images to start processing.*
   All of the images in your selected folder(s) will be analyzed using the modules and settings you have specified. A status window will appear which has the following:

   - A *progress bar* which gives the elapsed time and estimates the time remaining to process the full image set.
   - A *pause button*  which pauses execution and allows you to subsequently resume the analysis.
   - A *stop button*  which cancels execution after prompting you for a place to save the measurements collected to that point.
   - A *save measurements* button  which will prompt you for a place to save the measurements collected to that point while continuing the analysis run.

   At the end of each cycle, CellProfiler saves the measurements in the output file.

5. *ode to preview results.*
   You can optimize your pipeline by selecting the *Test* option from the main menu. Test mode allows you to run the pipeline on a selected image, preview the results, and adjust the module settings on the fly. See *Help* > *Testing Your Pipeline* for more details.

6. Save your project (which includes your pipeline) via *File* > *Save Project*.

Saving images in your pipeline: Due to the typically high number of intermediate images produced during processing, images produced during processing are not saved to the hard drive unless you specifically request it, using a **SaveImages** module.

Saving data in your pipeline: You can include an **Export** module to automatically export data in a format you prefer. See *Help* > *Using Your Output* for more details.

Testing Your Pipeline

Before starting an analysis run, you can test the pipeline settings on a selected image cycle using the *Test* mode option on the main menu. Test mode allows you to run the pipeline on a selected image, preview the results and adjust the module settings on the fly.

To enter Test mode once you have built a pipeline, choose *Test* > *Start Test Mode* from the menu bar in the main window. At this point, you will see the following features appear:

- The module view will have a slider bar appearing on the far left.
- A Pause icon will appear to the left of each module.
- A series of buttons will appear at the bottom of the pipeline panel above the module adjustment buttons.
- The grayed-out items in the *Test* menu will become active, and the *Analyze Images* button will become inactive.

You can run your pipeline in Test mode by selecting *Test* > *Step to Next Module* or clicking the *Run* or *Step* buttons at the bottom of the pipeline panel. The pipeline will execute normally, but you will be able to back up to a previous module or jump to a downstream module, change module settings to see the results, or execute the pipeline on the image of your choice. The additional controls allow you to do the following:

- *Slider:* Start/resume execution of the pipeline at any time by moving the slider. However, if the selected module depends on objects and/or images generated by prior modules, you will see an error message indicating that the data has not been produced yet. To avoid this, it is best to actually run the pipeline up to the module of interest, and move the slider to modules already executed.
- *Pause:* Clicking the pause icon will cause the pipeline test run to halt execution when that module is reached (the paused module itself is not executed). The icon changes from to to indicate that a pause has been inserted at that point.
- *Run:* Execution of the pipeline will be started/resumed until the next module pause is reached. When all modules have been executed for a given image cycle, execution will stop.
- *Step:* Execute the next module (as indicated by the slider location)
- *Next Image:* Skip ahead to the next image cycle as determined by the image order in the Input modules. The slider will automatically return to the first module in the pipeline.

From the *Test* menu, you can choose additional options:

- *Exit Test Mode:* Exit *Test* mode. Loading a new pipeline or adding/subtracting modules will also automatically exit test mode.
- *Step to Next Module:* Execute the next module (as indicated by the slider location)
- *Next Image Set:* Step to the next image set in the current image group.
- *Next Image Group:* Step to the next group in the image set. The slider will then automatically return to the first module in the pipeline.
- *Random Image Set:* Randomly select and jump to an image set in the current image group.
- *Choose Image Set:* Choose the image set to jump to. The slider will then automatically return to the first module in the pipeline.
- *Choose Image Group:* Choose an image group to jump to. The slider will then automatically return to the first module in the pipeline.
- *Reload Modules Source (enabled only if running from source code):* This option will reload the module source code, so any changes to the code will be reflected immediately.

Note that if movies are being loaded, the individual movie is defined as a group automatically. Selecting

Choose Image Group will allow you to choose the movie file, and *Choose Image Set* will let you choose the individual movie frame from that file.

Please see the **Groups** module for more details on the proper use of metadata for grouping

Running Your Pipeline

Once you have tested your pipeline using Test mode and you are satisfied with the module settings, you are ready to run the pipeline on your entire set of images. To do this:

- Exit Test mode by clicking the "Exit Test Mode" button or selecting *Test* > *Exit Test Mode*.
- Click the " Analyze Images" button and begin processing your data sets.

During the analysis run, the progress will appear in the status bar at the bottom of CellProfiler. It will show you the total number of image sets, the number of image sets completed, the time elapsed and the approximate time remaining in the run.

If you need to pause analysis, click the " Pause" button, then click the "Resume" button to continue. If you want to terminate analysis, click the " Stop Analysis" button.

If your computer has multiple processors, CellProfiler will take advantage of them by starting multiple copies of itself to process the image sets in parallel. You can set the number of *workers* (i.e.,copies of CellProfiler activated) under *File* > *Preferences...*

How Measurements are Named

In CellProfiler, measurements are exported as well as stored internally using the following general nomenclature: *`MeasurementType_Category_SpecificFeatureName_Parameters`*

Below is the description for each of the terms:

- `MeasurementType`: The type of data contained in the measurement, which can be one of three forms:
  - *Per-image:* These measurements are image-based (e.g., thresholds, counts) and are specified with the name "Image" or with the measurement (e.g., "Mean") for per-object measurements aggregated over an image.
  - *Per-object:* These measurements are per-object and are specified as the name given by the user to the identified objects (e.g., "Nuclei" or "Cells")
  - *Experiment:* These measurements are produced for a particular measurement across the entire analysis run (e.g., Z'-factors), and are specified with the name "Experiment". See **CalculateStatistics** for an example.
- `Category`: Typically, this information is specified in one of two ways
  - A descriptive name indicative of the type of measurement taken (e.g., "Intensity")
  - No name if there is no appropriate `Category` (e.g., if the *SpecificFeatureName* is "Count", no `Category` is specfied).
- `SpecificFeatureName`: The specific feature recorded by a module (e.g., "Perimeter"). Usually the module recording the measurement assigns this name, but a few modules allow the user to type in the name of the feature (e.g., the **CalculateMath** module allows the user to name the arithmetic measurement).
- `Parameters`: This specifier is to distinguish measurements obtained from the same objects but in different ways. For example, **MeasureObjectIntensity** can measure intensities for "Nuclei" in two different images. This specifier is used primarily for data obtained from an individual image channel specified by the **Images** module or a legacy **Load** module (e.g., "OrigBlue" and "OrigGreen") or a particular spatial scale (e.g., under the category "Texture" or "Neighbors"). Multiple parameters are separated by underscores.

  Below are additional details specific to various modules:

  - Measurements from the *AreaShape* and *Math* categories do not have a `Parameter` specifier.
  - Measurements from *Intensity*, *Granularity*, *Children*, *RadialDistribution*, *Parent* and *AreaOccupied* categories will have an associated image as the Parameter.
  - *Measurements from the Neighbors and Texture category will have a spatial scale `Parameter`.*
  - *Measurements from the Texture and RadialDistribution categories will have both a spatial scale and an image `Parameter`.*

As an example, consider a measurement specified as `Nuclei_Texture_DifferenceVariance_ER_3`:

- *`MeasurementType` is "Nuclei," the name given to the detected objects by the user.*
- *`Category` is "Texture," indicating that the module **MeasureTexture** produced the measurements.*
- *`SpecificFeatureName` is "DifferenceVariance," which is one of the many texture measurements made by the **MeasureTexture** module.*
- *There are two `Parameters`, the first of which is "ER". "ER" is the user-provided name of the image in which this texture measurement was made.*
- *The second `Parameter` is "3", which is the spatial scale at which this texture measurement was made.*

See also the Available measurements heading under the main help for many of the modules, as well as

ExportToSpreadsheet *and* ***ExportToDatabase*** *modules.*

Using Spreadsheets and Databases

The most common form of output for cellular analysis is a *spreadsheet*, *which is an application which tabulates data values. CellProfiler can also output data into a database, which is a collection of data that is stored for retrieval by users. Which format you use will depend on some of the considerations below:*

- *Assessibility: Spreadsheet applications are typically designed to allow easy user interaction with the data, to edit values, make graphs and the like. In contrast, the values in databases are typically not modified after the fact. Instead, database applications typically allow for viewing a specific data range.*
- *Capacity and speed: Databases are designed to hold larger amounts of data than spreadsheets. Spreadsheets may contain hundreds to a few thousand rows of data, whereas databases can hold mnay millions of rows of data. Due to the high capacity, accessing a particular portion of data in a database is optimized for speed.*
- *Learning curve: The applications that access spreadsheets are usually made for ease-of-use to allow for user edits. Databases are more sophisticated and are not typically edited or modified; to do so require knowledge of specialized languages made for this purpose (e.g., MySQL, Oracle, etc).*

For spreadsheets, the most widely used program to open these files is Microsoft's Excel program. Since the file is plain text, other editors can also be used, such as Calc or Google Docs. For databases, a popular freeware access tool is SQLyog.

Using the Output File

Please note that the output file will be deprecated in the future. This setting is temporarily present for those needing HDF5 or MATLAB formats, and will be moved to Export modules in future versions of CellProfiler.

The *output file* is a file where all information about the analysis as well as any measurements will be stored to the hard drive. **Important note:** This file does *not* provide the same functionality as the Export modules. If you want to produce a spreadsheet of measurements easily readable by Excel or a database viewer (or similar programs), please refer to the **ExportToSpreadsheet** or **ExportToDatabase** modules and the associated help.

The options associated with the output file are accessible by pressing the "View output settings" button at the bottom of the pipeline panel. In the settings panel to the left, in the *Output Filename* box, you can specify the name of the output file.

The output file can be written in one of two formats:

- A *.mat file* which is readable by CellProfiler and by MATLAB (Mathworks).
- An *.h5 file* which is readable by CellProfiler, MATLAB and any other program capable of reading the HDF5 data format. Documentation on how measurements are stored and handled in CellProfiler using this format can be found here.

Results in the output file can also be accessed or exported using **Data Tools** from the main menu of CellProfiler. The pipeline with its settings can be be loaded from an output file using *File > Load Pipeline...*

The output file will be saved in the Default Output Folder unless you type a full path and file name into the file name box. The path must not have spaces or characters disallowed by your computer's platform.

If the output filename ends in *OUT.mat* (the typical text appended to an output filename), CellProfiler will prevent you from overwriting this file on a subsequent run by generating a new file name and asking if you want to use it instead. You can override this behavior by checking the *Allow overwrite?* box to the right.

For analysis runs that generate a large number of measurements, you may notice that even though the analysis completes, CellProfiler continues to use an inordinate amount of your CPU and RAM. This is because the output file is written after the analysis is completed and can take a very long time for a lot of measurements. If you do not need this file and/or notice this behavior, select "*Do not write measurements*" from the "Measurements file format" drop-down box.

Troubleshooting Memory and Speed Issues

If you find that you are running into out-of-memory errors and/or speed issues associated with your analysis run, we have detailed a number of solutions on our forum FAQ on this issue. We will continue to add more tips and tricks to this page over time.

Load Modules

Historically, two modules served the same functionality as the current project structure: **LoadImages** and **LoadData**. While the approach described above supercedes these modules in part, old pipelines loaded into CellProfiler that contain these modules will provide the option of preserving them; these pipelines will operate exactly as before.

Alternately, the user can choose to convert these modules into the project equivalent as closely as possible. Both modules remain accesible via the "Add module" and  button at the bottom of the pipeline panel. The section details information relevant for users who would like to continue using these modules. Please note, however, that these modules are deprcated and may be removed in the future.

Associating metadata with images

Metadata (i.e., additional data about image data) is sometimes available for input images. This information can be:

1. Used by CellProfiler to group images with common metadata identifiers (or "tags") together for particular steps in a pipeline;
2. Stored in the output file along with CellProfiler-measured features for annotation or sample-tracking purposes;
3. Used to name additional input/output files.

Two sources of metadata are:

- *Metadata provided in the image filename or location (pathname).* For example, images produced by an automated microscope can be given names similar to "Experiment1_A01_w1_s1.tif" in which the metadata about the plate ("Experiment1"), the well ("A01"), the wavelength number ("w1") and the imaging site ("s1") are encapsulated. The name of the folder in which the images are saved may be meaningful and may also be considered metadata as well. If this is the case for your data, use **LoadImages** to extract this information for use in the pipeline and storage in the output file.
- *Metadata provided as a table of information*. Often, information associated with each image (such as treatment, plate, well, etc) is available as a separate spreadsheet. If this is the case for your data, use **LoadData** to load this information.

Details for the metadata-specific help is given next to the appropriate settings in **LoadImages** and **LoadData**, as well the specific settings in other modules which can make use of metadata. However, here is an overview of how metadata is obtained and used.

In **LoadImages**, metadata can be extracted from the filename and/or folder location using regular expression, a specialized syntax used for text pattern-matching. These regular expressions can be used to identify different parts of the filename / folder. The syntax *(?P<fieldname>expr)* will extract whatever matches *expr* and assign it to the image's *fieldname* measurement. A regular expression tool is available which will allow you to check the accuracy of your regular expression.

For instance, say a researcher has folder names with the date and subfolders containing the images with the run ID (e.g., *./2009_10_02/1234/*). The following regular expression will capture the plate, well and site in the fields *Date* and *Run*:

.*[\\/](?P<Date>.*)[\\/](?P<Run>.*)$	
.*[\\/]	Skip characters at the beginning of the pathname until either a slash (/) or backslash (\) is encountered (depending on the OS). The extra slash for the backslash is used as an escape

	sequence.
(?P<Date>	Name the captured field *Date*
.*	Capture as many characters that follow
[\\/]	Discard the slash/backslash character
(?P<Run>	Name the captured field *Run*
.*	Capture as many characters as follow
$	The *Run* field must be at the end of the path string, i.e. the last folder on the path. This also means that the *Date* field contains the parent folder of the *Date* folder.

In **LoadData**, metadata is extracted from a CSV (comma-separated) file (a spreadsheet). Columns whose name begins with "Metadata" can be used to group files loaded by **LoadData** that are associated with a common metadata value. The files thus grouped together are then processed as a distinct image set.

For instance, an experiment might require that images created on the same day use an illumination correction function calculated from all images from that day, and furthermore, that the date be captured in the file names for the individual image sets and in a CSV file specifying the illumination correction functions.

In this case, if the illumination correction images are loaded with the **LoadData** module, the file should have a "Metadata_Date" column which contains the date metadata tags. Similarly, if the individual images are loaded using the **LoadImages** module, **LoadImages** should be set to extract the metadata tag from the file names. The pipeline will then match the individual images with their corresponding illumination correction functions based on matching "Metadata_Date" fields.

Using image grouping

To use grouping, you must define the relevant metadata for each image. This can be done using regular expressions in **LoadImages** or having them pre-defined in a CSV file for use in **LoadData**.

To use image grouping in **LoadImages**, please note the following:

- *Metadata tags must be specified for all images listed.* You cannot use grouping unless an appropriate regular expression is defined for all the images listed in the module.
- *Shared metadata tags must be specified with the same name for each image listed.* For example, if you are grouping on the basis of a metadata tag "Plate" in one image channel, you must also specify the "Plate" metadata tag in the regular expression for the other channels that you want grouped together.

Setting the Default Input Folder

Please note that the Default Input Folder will be deprecated in the future. The location of non-image files needed by some modules will be set to an absolute path in future versions of CellProfiler. For specifying the location of image files, please use the *Input modules* panel starting with the **Images** module.

The *Default Input Folder* is enabled only if a legacy pipeline is loaded into CellProfiler and is accessible by pressing the "View output settings" button at the botton of the pipeline panel. The folder designated as the *Default Input Folder* contains the input image or data files that you want to analyze. Several File Processing modules (e.g., **LoadImages** or **LoadData**) provide the option of retrieving images from this folder on a default basis unless you specify, within the module, an alternate, specific folder on your computer. Within modules, we recommend selecting the Default Input Folder as much as possible, so that your pipeline will work even if you transfer your images and pipeline to a different computer. If, instead, you type specific folder path names into a module's settings, your pipeline will not work on someone else's computer until you adjust those pathnames within each module.

Use the *Browse* button  to specify the folder you would like to use as the Default Input Folder, or type the full folder path in the edit box. If you type a folder path that cannot be found, the message box below will indicate this fact until you correct the problem. If you want to specify a folder that does not yet exist, type the desired name and click on the *New folder* button  . The folder will be created according to the pathname you have typed.

Setting the Default Output Folder

Please note that the Default Output Folder will be deprecated in the future. The location of files written by the various output modules will be set to an absolute path in future versions of CellProfiler.

The *Default Output Folder* is accessible by pressing the "View output settings" button at the botton of the pipeline panel. The Default Output Folder is the folder that CellProfiler uses to store the output file it creates. Also, several File Processing modules (e.g., **SaveImages** or **ExportToSpreadsheet**) provide the option of saving analysis results to this folder on a default basis unless you specify, within the module, an alternate, specific folder on your computer. Within modules, we recommend selecting the Default Output Folder as much as possible, so that your pipeline will work even if you transfer your images and pipeline to a different computer. If, instead, you type specific folder path names into a module's settings, your pipeline will not work on someone else's computer until you adjust those pathnames within each module.

Use the *Browse* button (to the right of the text box) to specify the folder you would like to use as the Default Output Folder, or type the full folder path in the edit box. If you type a folder path that cannot be found, the message box below will indicate this fact until you correct the problem. If you want to specify a folder that does not yet exist, type the desired name and click on the *New folder* icon to the right of the *Browse folder* icon. The folder will be created according to the pathname you have typed.

Setting the Output Filename

Please note that the output file will be deprecated in the future. This setting is temporarily present for those needing HDF5 or MATLAB formats, and will be moved to Export modules in future versions of CellProfiler.

The *output file* is a file where all information about the analysis as well as any measurements will be stored to the hard drive. **Important note:** This file does *not* provide the same functionality as the Export modules. If you want to produce a spreadsheet of measurements easily readable by Excel or a database viewer (or similar programs), please refer to the **ExportToSpreadsheet** or **ExportToDatabase** modules and the associated help.

The options associated with the output file are accessible by pressing the "View output settings" button at the bottom of the pipeline panel. In the settings panel to the left, in the *Output Filename* box, you can specify the name of the output file.

The output file can be written in one of two formats:

- A *.mat file* which is readable by CellProfiler and by MATLAB (Mathworks).
- An *.h5 file* which is readable by CellProfiler, MATLAB and any other program capable of reading the HDF5 data format. Documentation on how measurements are stored and handled in CellProfiler using this format can be found here.

Results in the output file can also be accessed or exported using **Data Tools** from the main menu of CellProfiler. The pipeline with its settings can be be loaded from an output file using *File > Load Pipeline...*

The output file will be saved in the Default Output Folder unless you type a full path and file name into the file name box. The path must not have spaces or characters disallowed by your computer's platform.

If the output filename ends in *OUT.mat* (the typical text appended to an output filename), CellProfiler will prevent you from overwriting this file on a subsequent run by generating a new file name and asking if you want to use it instead. You can override this behavior by checking the *Allow overwrite?* box to the right.

For analysis runs that generate a large number of measurements, you may notice that even though the analysis completes, CellProfiler continues to use an inordinate amount of your CPU and RAM. This is because the output file is written after the analysis is completed and can take a very long time for a lot of measurements. If you do not need this file and/or notice this behavior, select "*Do not write measurements*" from the "Measurements file format" drop-down box.

Batch Processing

CellProfiler is designed to analyze images in a high-throughput manner. Once a pipeline has been established for a set of images, CellProfiler can export batches of images to be analyzed on a computing cluster with the pipeline.

It is possible to process tens or even hundreds of thousands of images for one analysis in this manner. We do this by breaking the entire set of images into separate batches, then submitting each of these batches as individual jobs to a cluster. Each individual batch can be separately analyzed from the rest.

Submitting files for batch processing

Below is a basic workflow for submitting your image batches to the cluster.

1. *Create a folder for your project on your cluster.* For high-throughput analysis, it is recommended to create a separate project folder for each run.
2. Within this project folder, create the following folders (both of which must be connected to the cluster computing network):
    - Create an input folder, then transfer all of our images to this folder as the input folder. The input folder must be readable by everyone (or at least your cluster) because each of the separate cluster computers will read input files from this folder.
    - Create an output folder where all your output data will be stored. The output folder must be writeable by everyone (or at least your cluster) because each of the separate cluster computers will write output files to this folder.

   If you cannot create folders and set read/write permissions to these folders (or don't know how), ask your Information Technology (IT) department for help.
3. Press the "View output settings" button. In the panel that appears, set the Default Input and Default Output Folders to the *images* and *output* folders created above, respectively. The Default Input Folder setting will only appear if a legacy pipeline is being run.
4. *Create a pipeline for your image set.* You should test it on a few example images from your image set (if you are unfamilar with the concept of an image set, please see the help for the **Input** modules). The module settings selected for your pipeline will be applied to *all* your images, but the results may vary depending on the image quality, so it is critical to insure that your settings be robust against your "worst-case" images.
   For instance, some images may contain no cells. If this happens, the automatic thresholding algorithms will incorrectly choose a very low threshold, and therefore "find" spurious objects. This can be overcome by setting a lower limit on the threshold in the **IdentifyPrimaryObjects** module. The Test mode in CellProfiler may be used for previewing the results of your settings on images of your choice. Please refer to *Help* > *Testing Your Pipeline* for more details on how to use this utility.
5. *Add the **CreateBatchFiles** module to the end of your pipeline.* This module is needed to resolve the pathnames to your files with respect to your local machine and the cluster computers. If you are processing large batches of images, you may also consider adding **ExportToDatabase** to your pipeline, after your measurement modules but before the CreateBatchFiles module. This module will export your data either directly to a MySQL/SQLite database or into a set of comma-separated files (CSV) along with a script to import your data into a MySQL database. Please refer to the help for these modules in order learn more about which settings are appropriate.
6. *Run the pipeline to create a batch file.* Click the *Analyze images* button and the analysis will begin processing locally. Do not be surprised if this initial step takes a while since CellProfiler must first create the entire image set list based on your settings in the **Input** modules (this process can be sped up by creating your list of images as a CSV and using the **LoadData** module to load it). With the **CreateBatchFiles** module in place, the pipeline will not process all the images, but instead will

creates a batch file (a file called *Batch_data.h5*) and save it in the Default Output Folder (Step 1). The advantage of using **CreateBatchFiles** from the researcher's perspective is that the Batch_data.h5 file generated by the module captures all of the data needed to run the analysis. You are now ready to submit this batch file to the cluster to run each of the batches of images on different computers on the cluster.

7. *Submit your batches to the cluster.* Log on to your cluster, and navigate to the directory where you have installed CellProfiler on the cluster.
   A single batch can be submitted with the following command:
   `./python-2.6.sh CellProfiler.py -p <Default_Output_Folder_path>/Batch_data.h5 -c -r -b -f <first_image_set_number> -l <last_image_set_number>` This command submits the batch file to CellProfiler and specifies that CellProfiler run in a batch mode without its user interface to process the pipeline. This run can be modified by using additional options to CellProfiler that specify the following (type "CellProfiler.py -h" to see a list of available options):
   - `-p <Default_Output_Folder_path>/Batch_data.h5`: The location of the batch file, where <Default_Output_Folder_path> is the output folder path as seen by the cluster computer.
   - `-c`: Run "headless", i.e., without the GUI
   - `-r`: Run the pipeline specified on startup, which is contained in the batch file.
   - `-b`: Do not build extensions, since by this point, they should already be built.
   - `-f <first_image_set_number>`: Start processing with the image set specified, <first_image_set_number>
   - `-l <last_image_set_number>` : Finish processing with the image set specified, <last_image_set_number>

   Typically, a user will break a long image set list into pieces and execute each of these pieces using the command line switches, `-f` and `-l` to specify the first and last image sets in each job. A full image set would then need a script that calls CellProfiler with these options with sequential image set numbers, e.g, 1-50, 51-100, etc to submit each as an individual job.
   >If you need help in producing the batch commands for submitting your jobs, use the `--get-batch-commands` along with the `-p` switch to specify the Batch_data.h5 file output by the CreateBatchFiles module. When specified, CellProfiler will output one line to the terminal per job to be run. This output should be further processed to generate a script that can invoke the jobs in a cluster-computing context.
   The above notes assume that you are running CellProfiler using our source code (see "Developer's Guide" under Help for more details). If you are using the compiled version, you would replace `./python-2.6.sh CellProfiler.py` with the CellProfiler executable file itself and run it from the installation folder.

Once all the jobs are submitted, the cluster will run each batch individually and output any measurements or images specified in the pipeline. Specifying the output filename using the `-o` switch when calling CellProfiler will also produce an output file containing the measurements for that batch of images in the output folder. Check the output from the batch processes to make sure all batches complete. Batches that fail for transient reasons can be resubmitted.

For additional help on batch processing, refer to our wiki if installing CellProfiler on a Unix system, our wiki on adapting CellProfiler to a LIMS environment, or post your questions on the CellProfiler CPCluster forum.

Running Multiple Pipelines

The **Run multiple pipelines** dialog lets you select several pipelines which will be run consecutively. Please note the following:

- CellProfiler 2.1 project files are not currently supported.
- Pipelines from CellProfiler 2.0 and lower are supported.
- If you want to use a pipeline made using CellProfiler 2.1, then you need to include the project file list with the pipeline, by selecting *Export* > *Pipeline...*, and under the "Save as type" dropdown, select "CellProfiler pipeline and file list".

You can invoke **Run multiple pipelines** by selecting it from the file menu. The dialog has three parts to it:

- *File chooser*: The file chooser lets you select the pipeline files to be run. The *Select all* and *Deselect all* buttons to the right will select or deselect all pipeline files in the list. The *Add* button will add the pipelines to the pipeline list. You can add a pipeline file multiple times, for instance if you want to run that pipeline on more than one input folder.
- *Directory chooser*: The directory chooser lets you navigate to different directories. The file chooser displays all pipeline files in the directory chooser's current directory.
- *Pipeline list*: The pipeline list has the pipelines to be run in the order that they will be run. Each pipeline has a default input and output folder and a measurements file. You can change any of these by clicking on the file name - an appropriate dialog will then be displayed. You can click the remove button to remove a pipeline from the list

CellProfiler will run all of the pipelines on the list when you hit the "OK" button.

Configuring Logging

CellProfiler prints diagnostic messages to the console by default. You can change this behavior for most messages by configuring logging. The simplest way to do this is to use the command-line switch, "-L", to set the log level. For instance, to show error messages or more critical events, start CellProfiler like this:

```
CellProfiler -L ERROR
```

The following is a list of log levels that can be used:

- **DEBUG:** Detailed diagnostic information
- **INFO:** Informational messages that confirm normal progress
- **WARNING:** Messages that report problems that might need attention
- **ERROR:** Messages that report unrecoverable errors that result in data loss or termination of the current operation.
- **CRITICAL:** Messages indicating that CellProfiler should be restarted or is incapable of running.

You can tailor CellProfiler's logging with much more control using a logging configuration file. You specify the file name in place of the log level on the command line, like this:

```
CellProfiler -L ~/CellProfiler/my_log_config.cfg
```

Files are in the Microsoft .ini format which is grouped into categories enclosed in square brackets and the key/value pairs for each category. Here is an example file:

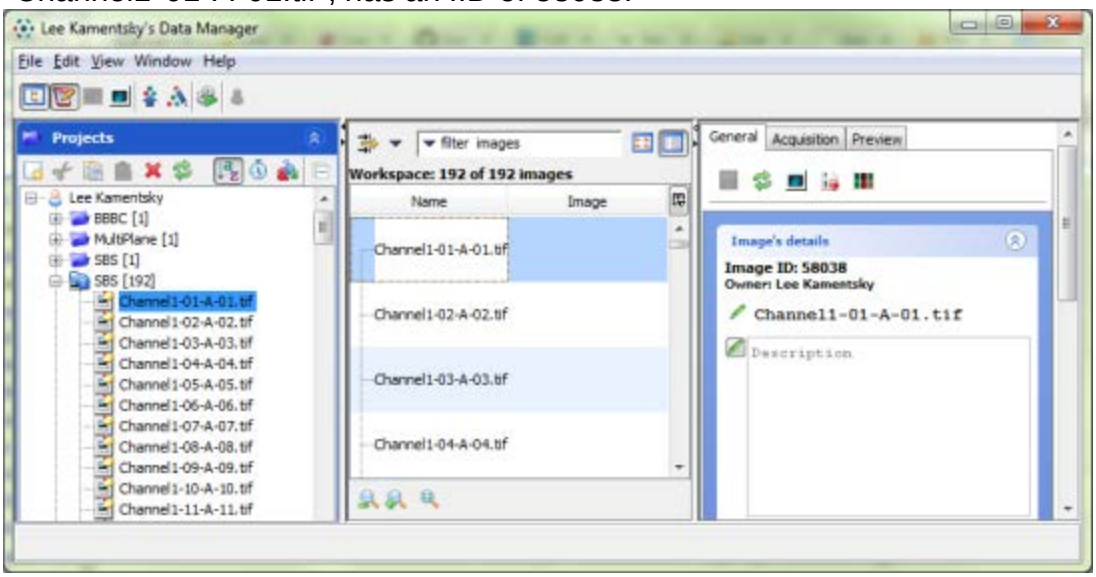
```
[loggers]
keys=root,pipelinestatistics

[handlers]
keys=console,logfile

[formatters]
keys=detailed

[logger_root]
level=WARNING
handlers=console

[logger_pipelinestatistics]
level=INFO
handlers=logfile
qualname=pipelineStatistics
propagate=0

[handler_console]
class=StreamHandler
formatter=detailed
level=WARNING
args=(sys.stderr)

[handler_logfile]
class=FileHandler
level=INFO
args=('~/CellProfiler/logfile.log','w')

[formatter_detailed]
format=[%(asctime)s] %(name)s %(levelname)s %(message)s
datefmt=
```

The above file would print warnings and errors to the console for all messages but "pipeline statistics" which are configured using the *pipelineStatistics* logger are written to a file instead.. The pipelineStatistics logger is the logger that is used to print progress messages when the pipeline is run. You can find out which loggers are being used to write particular messages by printing all messages with a formatter that prints the logger name ("%(name)s").
The format of the file is described in greater detail here.

Accessing Images From OMERO

CellProfiler has first-class support for loading images from OMERO. The input modules and the LoadData module can refer to images by URL, for instance, the example pipeline on the welcome page loads its images from `http://cellprofiler.org/ExampleFlyImages`. The first part of a URL (the part before the colon) is the schema. CellProfiler decides which communication protocol to use, depending on the schema; in the case of the example on the welcome page, the schema is HTTP and CellProfiler uses the HTTP protocol to get the image data. For OMERO, the schema that should be used is "omero" and we use the OMERO client library to fetch and load the data.

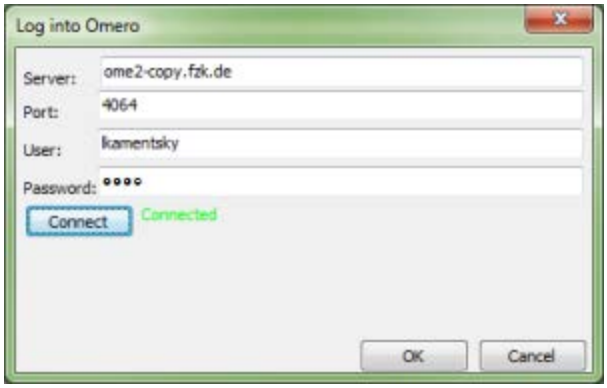
OMERO URLs have the form, "omero:iid=". You can find the image IDs using either the OMERO web client or the Insight software. As an example, the screen capture below indicates that the image, "Channel1-01-A-01.tif", has an IID of 58038:



At present, manually curating the URL list can be somewhat time-consuming, but we are planning to develop plug-ins for Insight that will automatically generate these lists for CellProfiler from within the Insight user interface. The plugin will allow you to select a screen or plate and export an image set list that can be used with CellProfiler's LoadData module.

OMERO login credentials

CellProfiler will ask you for your OMERO login credentials when you first access an OMERO URL, either by viewing it from the file list or by loading it in a pipeline. CellProfiler will create and maintain a session for you based on these credentials until you close the application. You should only need to enter your credentials once. To use the "Log into Omero" dialog, enter your server's name or IP address, the port (usually 4064), your user name and password and press the "Connect" button. The "Connect" button should turn green and the OK button of the dialog should become enabled (see below). Press OK to complete the login.



Currently, CellProfiler cannot establish a connection to OMERO when running headless - to do that, we would need to store the user password where it might be otherwise visible. We would like to provide a secure mechanism for establishing a session when headless and would like to work with you to make this work in your environment; please contact us for further information on how to modify CellProfiler yourself to do this or with suggestions for us to implement.

Using OMERO URLs with the Input modules

The **Images** module has a file list panel of all of the image files in a project. This file list supports URLs including OMERO URLs. You can drag URLs from a text document and drop them into the file list. The URLs do not end with image file extensions (like .TIF), so you need to change the "Filter images?" setting to "No filtering" to allow the OMERO URLs to be processed further. You should be able to view the image by double-clicking on it and you should be able to extract plate, well, site and channel name metadata from each image using the "Extract from image file headers" method in the **Metadata** module (press the "Update Metadata" button after selecting the "Extract from image file headers" method). If your experiment has more than one image channel, you can use the "ChannelName" metadata extracted from the OMERO image to create image sets containing all of your image channels. In the **NamesAndTypes** module, change the "Assign a name to" setting to "Images matching rules". For the rule criteria, select "Metadata does have ChannelName matching" and enter the name that appears under "Channels" in the OMERO Insight browser. Add additional channels to **NamesAndTypes** using the "Add another image" button.

OMERO URLs and LoadData

The LoadData module reads image sets from a .CSV file. The CSV file has a one-line header that tells LoadData how to use each of the columns of the file. You can load channels from a URL by adding a "URL" tag to this header. The OMERO URLs themselves appear in rows below. For instance, here is a .CSV that loads a DNA and GFP channel:

```
URL_DNA,URL_GFP
omero:iid=58134,omero:iid=58038
omero:iid=58135,omero:iid=58039
omero:iid=58136,omero:iid=58040
```

Plate Viewer

Plate Viewer help

The plate viewer is a data tool that displays the images in your experiment in plate format. Your project must define an image set list with metadata annotations for the image's well and, optionally its plate and site. The plate viewer will then group your images by well and display a plate map for you. If you have defined a plate metadata tag (with the name, "Plate"), the plate viewer will group your images by plate and display a choice box that lets you pick the plate to display.

Click on a well to see the images for that well. Channels are composited together if there is more than one channel and you can control the color used to display each channel. If you have more than one site per well and have site metadata (with the name, "Site"), the plate viewer will tile the sites when displaying.

Module: CalculateMath

Calculate Math takes measurements produced by previous modules and performs basic arithmetic operations.

The arithmetic operations available in this module include addition, subtraction, multiplication, and division. The result can be log-transformed or raised to a power and can be used in further calculations if another **CalculateMath** module is added to the pipeline.

The module can make its calculations on a per-image basis (for example, multiplying the area occupied by a stain in the image by the total intensity in the image) or on an object-by-object basis (for example, dividing the intensity in the nucleus by the intensity in the cytoplasm for each cell).

Available measurements

- **Image measurements:** If both input measurements are whole-image measurements, then the result will also be a whole-image measurement.
- **Object measurements:** Object measurements can be produced in two ways:
    - If both input measurements are individual object measurements, then the result will also be an object measurement. In these cases, the measurement will be associated with *both* objects that were involved in the measurement.
    - If one measure is object-based and one image-based, then the result will be an object measurement.

The result of these calculations is a new measurement in the "Math" category.

See also all **Measure** modules.

Settings:

Name the output measurement

Enter a name for the measurement calculated by this module.

Operation

Choose the arithmetic operation would you like to perform. *None* is useful if you simply want to select some of the later options in the module, such as multiplying or exponentiating your image by a constant.

Select the first operand measurement type

Indicate whether the operand is an image or object measurement.

Select the first operand objects

Choose the objects you want to measure for this operation.

Select the first operand measurement

Enter the category that was used to create the measurement. You will be prompted to add additional

information depending on the type of measurement that is requested.

Multiply the above operand by

Enter the number by which you would like to multiply the above operand.

Raise the power of above operand by

Enter the power by which you would like to raise the above operand.

Select the second operand measurement type

Indicate whether the operand is an image or object measurement.

Select the second operand objects

Choose the objects you want to measure for this operation.

Select the second operand measurement

Enter the category that was used to create the measurement. You will be prompted to add additional information depending on the type of measurement that is requested.

Take log10 of result?

Select *Yes* if you want the log (base 10) of the result.

Multiply the result by

(Used only for operations other than None)
Enter the number by which you would like to multiply the result.

Raise the power of result by

(Used only for operations other than None)
Enter the power by which you would like to raise the result.

Add to the result

Enter the number you like to add to the result.

Constrain the result to a lower bound?

Select *Yes* if you want the result to be constrained to a lower bound.

Enter the lower bound

Enter the lower bound of the result here.

Constrain the result to an upper bound?

Select *Yes* if you want the result to be constrained to an upper bound.

Enter the upper bound

Enter the upper bound of the result here.

Module: CalculateStatistics

Calculate Statistics calculates measures of assay quality (V and Z' factors) and dose response data (EC50) for all measured features made from images.

The V and Z' factors are statistical measures of assay quality and are calculated for each per-image measurement and for each average per-object measurement that you have made in the pipeline. Placing this module at the end of a pipeline in order to calculate these values allows you to identify which measured features are most powerful for distinguishing positive and negative control samples, or for accurately quantifying the assay's response to dose. These measurements will be calculated for all measured values (Intensity, AreaShape, Texture, etc.). These measurements can be exported as the "Experiment" set of data.

Available measurements

- **Experiment features:** Whereas most CellProfiler measurements are calculated for each object (per-object) or for each image (per-image), this module produces *per-experiment* values; for example, one Z' factor is calculated for each measurement, across the entire analysis run.
    - *Zfactor:* The Z'-factor indicates how well separated the positive and negative controls are. A Z'-factor > 0 is potentially screenable; a Z'-factor > 0.5 is considered an excellent assay. The formula is $1 - 3 \times (\sigma_p + \sigma_n)/|\mu_p - \mu_n|$ where σ_p and σ_n are the standard deviations of the positive and negative controls, and μ_p and μ_n are the means of the positive and negative controls.
    - *Vfactor:* The V-factor is a generalization of the Z'-factor, and is calculated as $1 - 6 \times \text{mean}(\sigma)/|\mu_p - \mu_n|$ where σ are the standard deviations of the data, and μ_p and μ_n are defined as above.
    - *EC50:* The half maximal effective concentration (EC50) is the concentration of a treatment required to induce a response which is 50%% of the maximal response.
    - *OneTailedZfactor:* This measure is an attempt to overcome a limitation of the original Z'-factor formulation (it assumes a Gaussian distribution) and is informative for populations with moderate or high amounts of skewness. In these cases, long tails opposite to the mid-range point lead to a high standard deviation for either population, which results in a low Z' factor even though the population means and samples between the means may be well-separated. Therefore, the one-tailed Z' factor is calculated with the same formula but using only those samples that lie between the positive/negative population means. **This is not yet a well established measure of assay robustness, and should be considered experimental.**

For both Z' and V factors, the highest possible value (best assay quality) is 1, and they can range into negative values (for assays where distinguishing between positive and negative controls is difficult or impossible). The Z' factor is based only on positive and negative controls. The V factor is based on an entire dose-response curve rather than on the minimum and maximum responses. When there are only two doses in the assay (positive and negative controls only), the V factor will equal the Z' factor.

Note: If the standard deviation of a measured feature is zero for a particular set of samples (e.g., all the positive controls), the Z' and V factors will equal 1 despite the fact that the assay quality is poor. This can occur when there is only one sample at each dose. This also occurs for some non-informative measured features, like the number of cytoplasm compartments per cell, which is always equal to 1.

This module can create MATLAB scripts that display the EC50 curves for each measurement. These scripts will require MATLAB and the statistics toolbox in order to run. See *Create dose/response plots?* below.

References

- *Z' factor:* Zhang JH, Chung TD, et al. (1999) "A simple statistical parameter for use in evaluation and validation of high throughput screening assays" *J Biomolecular Screening* 4(2): 67-73. (link)
- *V factor:* Ravkin I (2004): Poster #P12024 - Quality Measures for Imaging-based Cellular Assays. *Society for Biomolecular Screening Annual Meeting Abstracts*.
- Code for the calculation of Z' and V factors was kindly donated by Ilya Ravkin. Carlos Evangelista donated his copyrighted dose-response-related code.

*Example format for a file to be loaded by **LoadData** for this module:*

LoadData loads information from a CSV file. The first line of this file is a header that names the items. Each subsequent line represents data for one image cycle, so your file should have the header line plus one line per image to be processed. You can also make a file for **LoadData** to load that contains the positive/negative control and dose designations *plus* the image file names to be processed, which is a good way to guarantee that images are matched with the correct data. The control and dose information can be designated in one of two ways:

- As metadata (so that the column header is prefixed with the "Metadata_" tag). "Metadata" is the category and the name after the underscore is the measurement.
- As some other type of data, in which case the header needs to be of the form <*prefix*>_<*measurement*>. Select <*prefix*> as the category and <*measurement*> as the measurement.

Here is an example file:

```
Image_FileName_CY3, Image_PathName_CY3, Data_Control, Data_Dose
"Plate1_A01.tif",   "/images",          -1,           0
"Plate1_A02.tif",   "/images",          1,            1E10
"Plate1_A03.tif",   "/images",          0,            3E4
"Plate1_A04.tif",   "/images",          0,            5E5
```

See also the **Metadata** and legacy **LoadData** modules.

Settings:

Select the image measurement describing the positive and negative control status

The Z' factor, a measure of assay quality, is calculated by this module based on measurements from images that are specified as positive controls and images that are specified as negative controls. (Images that are neither are ignored.) The module assumes that all of the negative controls are specified by a minimum value, all of the positive controls are specified by a maximum value, and all other images have an intermediate value; this might allow you to use your dosing information to also specify the positive and negative controls. If you don't use actual dose data to designate your controls, a common practice is to designate -1 as a negative control, 0 as an experimental sample, and 1 as a positive control. In other words, positive controls should all be specified by a single high value (for instance, 1) and negative controls should all be specified by a single low value (for instance, 0). Other samples should have an intermediate value to exclude them from the Z' factor analysis.

The typical way to provide this information in the pipeline is to create a text comma-delimited (CSV) file outside of CellProfiler and then load that file into the pipeline using the **Metadata** module or the legacy

LoadData module. In that case, choose the measurement that matches the column header of the measurement in the input file. See the **Metadata** module help for an example text file.

Select the image measurement describing the treatment dose

The V and Z' factor, a measure of assay quality, and the EC50, indicating dose/response, are calculated by this module based on each image being specified as a particular treatment dose. Choose a measurement that gives the dose of some treatment for each of your images.

The typical way to provide this information in the pipeline is to create a comma-delimited text file (CSV) outside of CellProfiler and then load that file into the pipeline using **Metadata** or the **LoadData**. In that case, choose the measurement that matches the column header of the measurement in the CSV input file. See **LoadData** help for an example text file.

Log-transform the dose values?

Select *Yes* if you have dose-response data and you want to log-transform the dose values before fitting a sigmoid curve.

Select *No* if your data values indicate only positive vs. negative controls.

Create dose/response plots?

Select *Yes* if you want to create and save dose response plots. You will be asked for information on how to save the plots.

Figure prefix

(Used only when creating dose/response plots)
CellProfiler will create a file name by appending the measurement name to the prefix you enter here. For instance, if you have objects named, "Cells", the "AreaShape_Area measurement", and a prefix of "Dose_", CellProfiler will save the figure as *Dose_Cells_AreaShape_Area.m*. Leave this setting blank if you do not want a prefix.

Output file location

(Used only when creating dose/response plots)
This setting lets you choose the folder for the output files. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

For *Elsewhere...*, *Default Input Folder sub-folder* and *Default Output Folder sub-folder*, if you have metadata associated with your images via **Metadata** module, you can name the folder using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have a metadata tag named "Plate", you can create a per-plate folder by selecting one of the subfolder options and then specifying the subfolder name as "\g<Plate>". The module will substitute the metadata values for the current image set for any metadata tags in the folder name. Please see the **Metadata** module for more details on metadata collection and usage.

Module: DisplayDataOnImage

Display Data On Image produces an image with measured data on top of identified objects.

This module displays either a single image measurement on an image of your choosing, or one object measurement per object on top of every object in an image. The display itself is an image which you can save to a file using **SaveImages**.

Settings:

Display object or image measurements?

- *Object* displays measurements made on objects.
- *Image* displays a single measurement made on an image.

Select the input objects

(Used only when displaying object measurements)
Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**).

Measurement to display

Choose the measurement to display. This will be a measurement made by some previous module on either the whole image (if displaying a single image measurement) or on the objects you selected.

Select the image on which to display the measurements

Choose the image to be displayed behind the measurements. This can be any image created or loaded by a previous module. If you have chosen not to display the background image, the image will only be used to determine the dimensions of the displayed image

Text color

This is the color that will be used when displaying the text.

Name the output image that has the measurements displayed

The name that will be given to the image with the measurements superimposed. You can use this name to refer to the image in subsequent modules (such as **SaveImages**).

Image elements to save

This setting controls the level of annotation on the image:

- *Image:* Saves the image with the overlaid measurement annotations.
- *Axes:* Adds axes with tick marks and image coordinates.
- *Figure:* Adds a title and other decorations.

Annotation offset (in pixels)

Add a pixel offset to the measurement. Normally, the text is placed at the object (or image) center, which can obscure relevant features of the object. This setting adds a specified offset to the text, in a random direction.

Display mode

(Used only when displaying object measurements)
Choose how to display the measurement information. If you choose Text, **DisplayDataOnImage** will display the numeric value on top of each object. If you choose Color, **DisplayDataOnImage** will convert the image to grayscale, if necessary, and display the portion of the image within each object using a hue that indicates the measurement value relative to the other objects in the set using the default color map.

Color map

(Used only when displaying object measurements)
This is the color map used as the color gradient for coloring the objects by their measurement values.

Display background image?

Choose whether or not to display the measurements on a background image. Usually, you will want to see the image context for the measurements, but it may be useful to save just the overlay of the text measurements and composite the overlay image and the original image later. Choose "Yes" to display the measurements on top of a background image or "No" to display the measurements on a black background.

Module: DisplayDensityPlot

Display Density Plot plots measurements as a two-dimensional density plot.

A density plot displays the relationship between two measurements (that is, features) but instead of showing each data point as a dot, as in a scatter plot, the data points are binned into an equally-spaced grid of points, where the color of each point in the grid represents the tabulated frequency of the measurements within that region of the grid. A density plot is also known as a 2-D histogram; in a conventional histogram the height of a bar indicates how many data points fall in that region. By contrast, in a density plot (2-D histogram), the color of a portion of the plot indicates the number of data points in that region.

The module shows the values generated for the current cycle. However, this module can also be run as a Data Tool, in which case you will first be asked for the output file produced by the analysis run. The resultings plot is created from all the measurements collected during the run.

See also **DisplayScatterPlot**, **DisplayHistogram**.

Settings:

Select the object to display on the X-axis

Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed on the X-axis.

Select the object measurement to plot on the X-axis

Choose the object measurement made by a previous module to display on the X-axis.

Select the object to display on the Y-axis

Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed on the Y-axis.

Select the object measurement to plot on the Y-axis

Choose the object measurement made by a previous module to display on the Y-axis.

Select the grid size

Enter the number of grid regions you want used on each axis. Increasing the number of grid regions increases the resolution of the plot.

How should the X-axis be scaled?

The X-axis can be scaled either with a *linear* scale or with a *log* (base 10) scaling.

Using a log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

How should the Y-axis be scaled?

The Y-axis can be scaled either with a *linear* scale or with a *log* (base 10) scaling.

Using a log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

How should the colorbar be scaled?

The colorbar can be scaled either with a *linear* scale or with a *log* (base 10) scaling.

Using a log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

Select the color map

Select the color map for the density plot. See this page for pictures of the available colormaps.

Enter a title for the plot, if desired

Enter a title for the plot. If you leave this blank, the title will default to *(cycle N)* where *N* is the current image cycle being executed.

Module: DisplayHistogram

Display Histogram plots a histogram of the desired measurement.

A histogram is a plot of tabulated data frequencies (each of which is shown as a bar) created by binning measurement data for a set of objects. A two-dimensional histogram can be created using the **DisplayDensityPlot** module.

The module shows the values generated for the current cycle. However, this module can also be run as a Data Tool, in which you will first be asked for the output file produced by the analysis run. The resultant plot is created from all the measurements collected during the run.

See also **DisplayDensityPlot**, **DisplayScatterPlot**.

Settings:

Select the object whose measurements will be displayed

Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed.

Select the object measurement to plot

Choose the object measurement made by a previous module to plot.

Number of bins

Enter the number of equally-spaced bins that you want used on the X-axis.

Transform the data prior to plotting along the X-axis?

The measurement data can be scaled with either a linear scale (*No*) or a *log* (base 10) scaling.

Log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

How should the Y-axis be scaled?

The Y-axis can be scaled either with either a *linear* scale or a *log* (base 10) scaling.

Log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

Enter a title for the plot, if desired

Enter a title for the plot. If you leave this blank, the title will default to *(cycle N)* where *N* is the current image cycle being executed.

Specify min/max bounds for the X-axis?

Select *Yes* to specify minimum and maximum values for the plot on the X-axis. This is helpful if an outlier bin skews the plot such that the bins of interest are no longer visible.

Module: DisplayPlatemap

Display Platemap displays a desired measurement in a plate map view.

Display Platemap is a tool for browsing image-based data laid out on multi-well plates common to high-throughput biological screens. The display window for this module shows a plate map with each well color-coded according to the measurement chosen.

As the pipeline runs, the measurement information displayed is updated, so the value shown for each well is current up to the image cycle currently being processed; wells which have no corresponding measurements as yet as shown as blank.

See also **DisplayDensityPlot**, **DisplayHistogram**, **DisplayScatterPlot**.

Settings:

Display object or image measurements?

- *Image* allows you to select an image measurement to display for each well.
- *Object* allows you to select an object measurement to display for each well.

Select the object whose measurements will be displayed

Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed.

Select the measurement to plot

Choose the image or object measurement made by a previous module to plot.

Select your plate metadata

Choose the metadata tag that corresponds to the plate identifier. That is, each plate should have a metadata tag containing a specifier corresponding uniquely to that plate.

Please see the **Metadata** module for more details on metadata collection and usage.

Multiwell plate format

The module assumes that your data is laid out in a multi-well plate format common to high-throughput biological screens. Supported formats are:

- *96:* A 96-well plate with 8 rows × 12 columns
- *384:* A 384-well plate with 16 rows × 24 columns

Select your well metadata

Choose the metadata tag that corresponds to the well identifier. The row-column format of these entries should be an alphabetical character (specifying the plate row), followed by two integer characters (specifying the plate column). For example, a standard format 96-well plate would span from "A1" to

"H12", whereas a 384-well plate (16 rows and 24 columns) would span from well "A01" to well "P24"."

Please see the **Metadata** module for more details on metadata collection and usage.

Select your well row metadata

Choose the metadata tag that corresponds to the well row identifier, typically specified as an alphabetical character. For example, a standard format 96-well plate would span from row "A" to "H", whereas a 384-well plate (16 rows and 24 columns) would span from row "A" to "P".

Please see the **Metadata** module for more details on metadata collection and usage.

Select your well column metadata

Choose the metadata tag that corresponds to the well column identifier, typically specified with two integer characters. For example, a standard format 96-well plate would span from column "01" to "12", whereas a 384-well plate (16 rows and 24 columns) would span from column "01" to "24".

Please see the **Metadata** module for more details on metadata collection and usage.

How should the values be aggregated?

Measurements must be aggregated to a single number for each well so that they can be represented by a color. Options are:

- *avg:* Average
- *stdev:* Standard deviation
- *median*
- *cv%:* Coefficient of variation, defined as the ratio of the standard deviation to the mean. This is useful for comparing between data sets with different units or widely different means.

Enter a title for the plot, if desired

Enter a title for the plot. If you leave this blank, the title will default to *(cycle N)* where *N* is the current image cycle being executed.

Well metadata format

- *Well name* allows you to select an image measurement to display for each well.
- *Row & Column* allows you to select an object measurement to display for each well.

Module: DisplayScatterPlot

Display Scatter Plot plots the values for two measurements.

A scatter plot displays the relationship between two measurements (that is, features) as a collection of points. If there are too many data points on the plot, you should consider using **DisplayDensityPlot** instead.

The module will show a plot shows the values generated for the current cycle. However, this module can also be run as a Data Tool, in which you will first be asked for the output file produced by the analysis run. The resultant plot is created from all the measurements collected during the run.

See also **DisplayDensityPlot**, **DisplayHistogram**.

Settings:

Type of measurement to plot on X-axis

You can plot two types of measurements:

- *Image:* For a per-image measurement, one numerical value is recorded for each image analyzed. Per-image measurements are produced by many modules. Many have **MeasureImage** in the name but others do not (e.g., the number of objects in each image is a per-image measurement made by the **IdentifyObject** modules).
- *Object:* For a per-object measurement, each identified object is measured, so there may be none or many numerical values recorded for each image analyzed. These are usually produced by modules with **MeasureObject** in the name.

Select the object to plot on the X-axis

(Used only when plotting objects)
Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed on the X-axis.

Select the measurement to plot on the X-axis

Choose the measurement (made by a previous module) to plot on the X-axis.

Type of measurement to plot on Y-axis

You can plot two types of measurements:

- *Image:* For a per-image measurement, one numerical value is recorded for each image analyzed. Per-image measurements are produced by many modules. Many have **MeasureImage** in the name but others do not (e.g., the number of objects in each image is a per-image measurement made by **IdentifyObject** modules).
- *Object:* For a per-object measurement, each identified object is measured, so there may be none or many numerical values recorded for each image analyzed. These are usually produced by modules with **MeasureObject** in the name.

Select the object to plot on the Y-axis

(Used only when plotting objects)
Choose the name of objects identified by some previous module (such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**) whose measurements are to be displayed on the Y-axis.

Select the measurement to plot on the Y-axis

Choose the measurement (made by a previous module) to plot on the Y-axis.

How should the X-axis be scaled?

The X-axis can be scaled with either a *linear* scale or a *log* (base 10) scaling.

Log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

How should the Y-axis be scaled?

The Y-axis can be scaled with either a *linear* scale or with a *log* (base 10) scaling.

Log scaling is useful when one of the measurements being plotted covers a large range of values; a log scale can bring out features in the measurements that would not easily be seen if the measurement is plotted linearly.

Enter a title for the plot, if desired

Enter a title for the plot. If you leave this blank, the title will default to *(cycle N)* where *N* is the current image cycle being executed.

Module: ExportToDatabase

Export To Database exports data directly to a database, or in database readable format, including an imported file with column names and a CellProfiler Analyst properties file, if desired.

This module exports measurements directly to a database or to a SQL-compatible format. It allows you to create and import MySQL and associated data files into a database and gives you the option of creating a properties file for use with CellProfiler Analyst. Optionally, you can create an SQLite database file if you do not have a server on which to run MySQL itself.

This module must be run at the end of a pipeline, or second to last if you are using the **CreateBatchFiles** module. If you forget this module, you can also run the *ExportDatabase* data tool after processing is complete; its functionality is the same.

The database is set up with two primary tables. These tables are the *Per_Image* table and the *Per_Object* table (which may have a prefix if you specify):

- The Per_Image table consists of all the per-image measurements made during the pipeline, plus per-image population statistics (such as mean, median, and standard deviation) of the object measurements. There is one per_image row for every "cycle" that CellProfiler processes (a cycle is usually a single field of view, and a single cycle usually contains several image files, each representing a different channel of the same field of view).
- The Per_Object table contains all the measurements for individual objects. There is one row of object measurements per object identified. The two tables are connected with the primary key column *ImageNumber*, which indicates the image to which each object belongs. The Per_Object table has another primary key called *ObjectNumber*, which is unique to each image.

Typically, if multiple types of objects are identified and measured in a pipeline, the numbers of those objects are equal to each other. For example, in most pipelines, each nucleus has exactly one cytoplasm, so the first row of the Per-Object table contains all of the information about object #1, including both nucleus- and cytoplasm-related measurements. If this one-to-one correspondence is *not* the case for all objects in the pipeline (for example, if dozens of speckles are identified and measured for each nucleus), then you must configure **ExportToDatabase** to export only objects that maintain the one-to-one correspondence (for example, export only *Nucleus* and *Cytoplasm*, but omit *Speckles*).

If you have extracted "Plate" and "Well" metadata from image filenames or loaded "Plate" and "Well" metadata via the **Metadata** or **LoadData** modules, you can ask CellProfiler to create a "Per_Well" table, which aggregates object measurements across wells. This option will output a SQL file (regardless of whether you choose to write directly to the database) that can be used to create the Per_Well table. At the secure shell where you normally log in to MySQL, type the following, replacing the italics with references to your database and files:

```
mysql -h hostname -u username -p databasename < pathtoimages/perwellsetupfile.SQL
```

The commands written by CellProfiler to create the Per_Well table will be executed.

Oracle is not fully supported at present; you can create your own Oracle DB using the .csv output option and writing a simple script to upload to the database.

Available measurements

For details on the nomenclature used by CellProfiler for the exported measurements, see *Help > General*

Help > How Measurements Are Named.

See also **ExportToSpreadsheet**.

Settings:

Database type

Specify the type of database you want to use:

- *MySQL:* Writes the data directly to a MySQL database. MySQL is open-source software; you may require help from your local Information Technology group to set up a database server.
- *MySQL / CSV:* Writes a script file that contains SQL statements for creating a database and uploading the Per_Image and Per_Object tables. This option will write out the Per_Image and Per_Object table data to two CSV files; you can use these files can be used to import the data directly into an application that accepts CSV data.
- *SQLite:* Writes SQLite files directly. SQLite is simpler to set up than MySQL and can more readily be run on your local computer rather than requiring a database server. More information about SQLite can be found here.

  If running this module on a computing cluster, there are a few considerations to note:

  - The *MySQL* option is well-suited for cluster use, since multiple jobs can write to the database simultaneously.
  - The *SQLite* option is not as appropriate; a SQLite database only allows access by one job at a time.

Experiment name

Select a name for the experiment. This name will be registered in the database and linked to the tables that **ExportToDatabase** creates. You will be able to select the experiment by name in CellProfiler Analyst and will be able to find the experiment's tables through database queries.

Database name

Select a name for the database you want to use

SQL file prefix

(Used if MySQL / CSV is selected as the database type)
Enter the prefix to be used to name the SQL file.

Name the SQLite database file

(Used if SQLite selected as database type)
Enter the name of the SQLite database filename to which you want to write.

Overwrite without warning?

ExportToDatabase creates tables and databases at the start of a run when writing directly to a MySQL or SQLite database. It writes SQL scripts and CSVs when not writing directly. It also can write CellProfiler

Analysis property files. In some cases, it is appropriate to run CellProfiler and append to or overwrite the data in existing tables, for instance when running several CellProfiler instances which each process a range of the experiment's image sets. In other cases, such as when the measurements to be written have changed, the data tables must be dropped completely.
You can choose fromm three options to conrtol overwriting behavior:

- *Never:* **ExportToDatabase** will ask before dropping and recreating tables unless you are running headless. CellProfiler will exit if running headless if the tables exist and this option is chosen.
- *Data only:* **ExportToDatabase** will keep the existing tables if present and will overwrite the data. Choose *Data only* if you are breaking your experiment into ranges of image sets and running each range on a separate instance of CellProfiler.
- *Data and schema:* **ExportToDatabase** will drop previous versions of tables at the start of a run. This option is appropriate if you are using the **CreateBatchFiles** module; your tables will be created by the run that creates the batch data file. The actual analysis runs that utilize the `Batch_data` file will use the existing tables without trying to recreate them.

Add a prefix to table names?

Select whether you want to add a prefix to your table names. The default table names are *Per_Image* for the per-image table and *Per_Object* for the per-object table. Adding a prefix can be useful for bookkeeping purposes.

- Select *Yes* to add a user-specified prefix to the default table names. If you want to distinguish multiple sets of data written to the same database, you probably want to use a prefix.
- Select *No* to use the default table names. For a one-time export of data, this option is fine.

Whether you chose to use a prefix or not, CellProfiler will warn you if your choice entails overwriting an existing table.

Table prefix

(Used if Add a prefix to table names? is selected)
Enter the table prefix you want to use.

MySQL has a 64 character limit on the full name of the table. If the combination of the table name and prefix exceeds this limit, you will receive an error associated with this setting.

Create a CellProfiler Analyst properties file?

Select *Yes* to generate a template properties file that will allow you to use your new database with CellProfiler Analyst (a data exploration tool which can also be downloaded from http://www.cellprofiler.org/). The module will attempt to fill in as many as the entries as possible based on the pipeline's settings, including the server name, username and password if MySQL is used.

Which objects should be used for locations?

(Used only if creating a properties file)
CellProfiler Analyst displays cells during classification. This setting determines which object centers will be used as the center of the cells to be displayed. Choose one of the listed objects and CellProfiler will save that object's location columns in the properties file so that CellProfiler Analyst centers cells using that object's center.

You can manually change this choice in the properties file by edting the *cell_x_loc* and *cell_y_loc*

properties.

Note that if there are no objects defined in the pipeline (e.g. if only using MeasureImageQuality and/or Illumination Correction modules), a warning will diplay until you choose *'None'* for the subsequent setting: 'Export measurements for all objects to the database?'.

Access CPA images via URL?

(Used only if creating a properties file)
The image paths written to the database will be the absolute path the the image files on your computer. If you plan to make these files accessible via the web, you can have CellProfiler Analyst prepend a URL to your file name. Eg: If an image is loaded from the path "/cellprofiler/images/" and you use a url prepend of "http://mysite.com/", CellProfiler Analyst will look for your file at "http://mysite.com/cellprofiler/images/"

Enter an image url prepend if you plan to access your files via http

(Used only if accessing CellProfiler Analyst images via URL)
The image paths written to the database will be the absolute path the the image files on your computer. If you plan to make these files accessible via the web, you can enter a url prefix here. Eg: If an image is loaded from the path "/cellprofiler/images/" and you use a url prepend of "http://mysite.com/", CellProfiler Analyst will look for your file at "http://mysite.com/cellprofiler/images/"

If you are not using the web to access your files (i.e., they are locally aceesible by your computer), leave this setting blank.

Select the plate type

(Used only if creating a properties file)
If you are using a multi-well plate or microarray, you can select the plate type here. Supported types in CellProfiler Analyst are 96- and 384-well plates, as well as 5600-spot microarrays. If you are not using a plate or microarray, select *None*.

Select the plate metadata

(Used only if creating a properties file)
If you are using a multi-well plate or microarray, you can select the metadata corresponding to the plate here. If there is no plate metadata associated with the image set, select *None*.

Please see the **Metadata** module for more details on metadata collection and usage.

Select the well metadata

(Used only if creating a properties file)
If you are using a multi-well plate or microarray, you can select the metadata corresponding to the well here. If there is no well metadata associated with the image set, select *None*.

Please see the **Metadata** module for more details on metadata collection and usage.

Include information for all images, using default values?

(Used only if creating a properties file)
Select *Yes* to include information in the properties file for all images. This option will do the following:

- All images loaded using the **Input** modules or saved in **SaveImages** will be included.
- The CellProfiler image name will be used for the *image_name* field.
- A channel color listed in the *image_channel_colors* field will be assigned to the image by default order.

Select *No* to specify which images should be included or to override the automatic values.

Select an image to include

(Used only if creating a properties file and specifiying the image information)
Choose image name to include it in the properties file of images.

The images in the drop-down correspond to images that have been:

- Loaded using one of the **Load** modules.
- Saved with the **SaveImages** module, with the corresponding file and path information stored.

If you do not see your desired image listed, check the settings on these modules.

Use the image name for the display?

(Used only if creating a properties file and specifiying the image information)
Select *Yes* to use the image name as given above for the displayed name.

Select *No* to name the file yourself.

Image name

(Used only if creating a properties file, specifiying the image information and naming the image)
Enter a name for the specified image

Channel color

(Used only if creating a properties file and specifiying the image information)
Enter a color to display this channel.

Do you want to add group fields?

(Used only if creating a properties file)
Please note that "groups" as defined by CellProfiler Analyst has nothing to do with "grouping" as defined by CellProfiler in the Groups module.

Select *Yes* to define a "group" for your image data (for example, when several images represent the same experimental sample), by providing column(s) that identify unique images (the *image key*) to another set of columns (the *group key*).

The format for a group in CPA is:
`group_SQL_<XXX> = <MySQL SELECT statement that returns image-key columns followed by group-key columns>` For example, if you wanted to be able to group your data by unique plate names, you could define a group called *SQL_Plate* as follows:
`group_SQL_Plate = SELECT ImageNumber, Image_Metadata_Plate FROM Per_Image`

Grouping is useful, for example, when you want to aggregate counts for each class of object and their

scores on a per-group basis (e.g., per-well) instead of on a per-image basis when scoring with Classifier. It will also provide new options in the Classifier fetch menu so you can fetch objects from images with specific values for the group columns.

Enter the name of the group

(Used only if creating a properties file and specifiying an image data group)
Enter a name for the group. Only alphanumeric characters and underscores are permitted.

Enter the per-image columns which define the group, separated by commas

(Used only if creating a properties file and specifiying an image data group)
To define a group, enter the image key columns followed by group key columns, each separated by commas.

In CellProfiler, the image key column is always given the name as *ImageNumber*; group keys are typically metadata columns which are always prefixed with *Image_Metadata_*. For example, if you wanted to be able to group your data by unique plate and well metadata tags, you could define a group with the following MySQL statement:
```
group_SQL_Plate = SELECT ImageNumber, Image_Metadata_Plate, Image_Metadata_Well FROM Per_Image
```
For this example, the columns to enter in this setting would be:
```
ImageNumber, Image_Metadata_Plate, Image_Metadata_Well
```

Groups are specified as MySQL statements in the properties file, but please note that the full SELECT and FROM clauses will be added automatically, so there is no need to enter them here.

Do you want to add filter fields?

(Used only if creating a properties file)
Select *Yes* to specify a subset of the images in your experiment by defining a *filter*. Filters are useful, for example, for fetching and scoring objects in Classifier or making graphs using the plotting tools that satisfy a specific metadata contraint.

Automatically create a filter for each plate?

(Used only if creating a properties file and specifiying an image data filter)
If you have specified a plate metadata tag, selecting *Yes* to create a set of filters in the properties file, one for each plate.

Enter a phenotype class table name if using the classifier tool

(Used only if creating a properties file)
If you are using the machine-learning tool in CellProfiler Analyst, you can create an additional table in your database which contains the per-object phenotype labels. This table is produced after scoring all the objects in your data set and will be named with the label given here.

You can manually change this choice in the properties file by edting the *class_table* field. Leave this field blank if you are not using the classifier or do not need the table written to the database.

Output file location

(Used only when using a CSV or a SQLite database, and/or creating a properties or workspace file)
This setting determines where the CSV files or SQLite database is saved if you decide to write

measurements to files instead of writing them directly to the database. If you request a CellProfiler Analyst properties file or workspace file, it will also be saved to this location. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

For *Elsewhere...*, *Default Input Folder sub-folder* and *Default Output Folder sub-folder*, if you have metadata associated with your images via **Metadata** module, you can name the folder using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have a metadata tag named "Plate", you can create a per-plate folder by selecting one of the subfolder options and then specifying the subfolder name with the "Plate" metadata tag. The module will substitute the metadata values for the last image set processed for any metadata tags in the folder name. Please see the **Metadata** module for more details on metadata collection and usage.

Create a CellProfiler Analyst workspace file?

Select *Yes* to generate a workspace file for use with CellProfiler Analyst, a data exploration tool which can also be downloaded from http://www.cellprofiler.org/. A workspace file allows you to open a selected set of measurements with the display tools of your choice. This is useful, for example, if you want examine a standard set of quality control image measurements for outliers.

Select the measurement display tool

(Used only if creating a workspace file)

Select what display tool in CPA you want to use to open the measurements.

- ScatterPlot
- Histogram
- DensityPlot
- PlateViewer
- BoxPlot

Type of measurement to plot on the X-axis

(Used only if creating a workspace file)
You can plot two types of measurements:

- *Image:* For a per-image measurement, one numerical value is recorded for each image analyzed. Per-image measurements are produced by many modules. Many have **MeasureImage** in the name but others do not (e.g., the number of objects in each image is a per-image measurement made by **IdentifyObject** modules).
- *Object:* For a per-object measurement, each identified object is measured, so there may be none or many numerical values recorded for each image analyzed. These are usually produced by modules with **MeasureObject** in the name.

Enter the object name

(Used only if creating a workspace file)
Select the object that you want to measure from the list. This should be an object created by a previous module such as **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, or **IdentifyTertiaryObjects**.

Select the X-axis measurement

(Used only if creating a workspace file)
Select the measurement to be plotted on the desired axis.

Type of measurement to plot on the Y-axis

(Used only if creating a workspace file)
You can plot two types of measurements:

- *Image:* For a per-image measurement, one numerical value is recorded for each image analyzed. Per-image measurements are produced by many modules. Many have **MeasureImage** in the name but others do not (e.g., the number of objects in each image is a per-image measurement made by **IdentifyObject** modules).
- *Object:* For a per-object measurement, each identified object is measured, so there may be none or many numerical values recorded for each image analyzed. These are usually produced by modules with **MeasureObject** in the name.

Select the Y-axis measurement

(Used only if creating a workspace file)
Select the measurement to be plotted on the desired axis.

Calculate the per-image mean values of object measurements?

Select *Yes* for **ExportToDatabase** to calculate population statistics over all the objects in each image and store the results in the database. For instance, if you are measuring the area of the Nuclei objects and you check the box for this option, **ExportToDatabase** will create a column in the Per_Image table called "Mean_Nuclei_AreaShape_Area".

You may not want to use **ExportToDatabase** to calculate these population statistics if your pipeline generates a large number of per-object measurements; doing so might exceed database column limits. These columns can be created manually for selected measurements directly in MySQL. For instance, the following SQL command creates the Mean_Nuclei_AreaShape_Area column:

```
ALTER TABLE Per_Image ADD (Mean_Nuclei_AreaShape_Area); UPDATE Per_Image SET
Mean_Nuclei_AreaShape_Area = (SELECT AVG(Nuclei_AreaShape_Area) FROM Per_Object WHERE
Per_Image.ImageNumber = Per_Object.ImageNumber);
```

Calculate the per-well mean values of object measurements?

Select *Yes* for **ExportToDatabase** to calculate statistics over all the objects in each well and store the results as columns in a "per-well" table in the database. For instance, if you are measuring the area of the Nuclei objects and you check the aggregate mean box in this module, **ExportToDatabase** will create a table in the database called "Per_Well_avg", with a column called "Mean_Nuclei_AreaShape_Area". Selecting all three aggregate measurements will create three per-well tables, one for each of the measurements.

The per-well functionality will create the appropriate lines in a .SQL file, which can be run on your Per-Image and Per-Object tables to create the desired per-well table.

Note: this option is only available if you have extracted plate and well metadata from the filename using the **Metadata** or **LoadData** modules. It will write out a .sql file with the statements necessary to create the Per_Well table, regardless of the option chosen above. Please see the **Metadata** module for more details on metadata collection and usage

Calculate the per-well median values of object measurements?

Select *Yes* for **ExportToDatabase** to calculate statistics over all the objects in each well and store the results as columns in a "per-well" table in the database. For instance, if you are measuring the area of the Nuclei objects and you check the aggregate median box in this module, **ExportToDatabase** will create a table in the database called "Per_Well_median", with a column called "Median_Nuclei_AreaShape_Area". Selecting all three aggregate measurements will create three per-well tables, one for each of the measurements.

The per-well functionality will create the appropriate lines in a .SQL file, which can be run on your Per-Image and Per-Object tables to create the desired per-well table.

Note: this option is only available if you have extracted plate and well metadata from the filename using the **Metadata** or **LoadData** modules. It will write out a .sql file with the statements necessary to create the Per_Well table, regardless of the option chosen above. Please see the **Metadata** module for more details on metadata collection and usage

Calculate the per-well standard deviation values of object measurements?

Select *Yes* for **ExportToDatabase** to calculate statistics over all the objects in each well and store the results as columns in a "per-well" table in the database. For instance, if you are measuring the area of the Nuclei objects and you check the aggregate standard deviation box in this module, **ExportToDatabase** will create a table in the database called "Per_Well_std", with a column called

"Mean_Nuclei_AreaShape_Area". Selecting all three aggregate measurements will create three per-well tables, one for each of the measurements.

The per-well functionality will create the appropriate lines in a .SQL file, which can be run on your Per-Image and Per-Object tables to create the desired per-well table.

Note: this option is only available if you have extracted plate and well metadata from the filename using the **Metadata** or **LoadData** modules. It will write out a .sql file with the statements necessary to create the Per_Well table, regardless of the option chosen above. Please see the **Metadata** module for more details on metadata collection and usage

Export measurements for all objects to the database?

This option lets you choose the objects whose measurements will be saved in the Per_Object and Per_Well(s) database tables.

- *All:* Export measurements from all objects.
- *None:* Do not export data to a Per_Object table. Save only Per_Image or Per_Well measurements (which nonetheless include population statistics from objects).
- *Select...:* Select the objects you want to export from a list.

Select the objects

(Used only if Select is chosen for adding objects)
Choose one or more objects from this list (click using shift or command keys to select multiple objects). The list includes the objects that were created by prior modules. If you choose an object, its measurements will be written out to the Per_Object and/or Per_Well(s) tables, otherwise, the object's measurements will be skipped.

Create one table per object, a single object table or a single object view?

ExportToDatabase can create either one table for each type of object exported or a single object table.

- *One table per object type* creates one table for each object type you export. The table name will reflect the name of your objects. The table will have one row for each of your objects. You can write SQL queries that join tables using the "Number_ObjectNumber" columns of parent objects (such as those created by **IdentifyPrimaryObjects**) with the corresponding "Parent_... column" of the child objects. Choose *One table per object type* if parent objects can have more than one child object, if you want a relational representation of your objects in the database, or if you need to split columns among different tables and shorten column names because of database limitations.
- *Single object table* creates a single database table that records the object measurements. **ExportToDatabase** will prepend each column name with the name of the object associated with that column's measurement. Each row of the table will have measurements for all objects that have the same image and object number. Choose *Single object table* if parent objects have a single child, or if you want a simple table structure in your database. You can combine the measurements for all or selected objects in this way.
- *Single object view* creates a single database view to contain the object measurements. A *view* is a virtual database table which can be used to package together multiple per-object tables into a single structure that is accessed just like a regular table. Choose *Single object view* if you want to combine multiple objects but using *Single object table* would produce a table that hits the database size limitations.
  An important note is that only objects that are related as primary, secondary or tertiary objects to each other should be combined in a view. This is because the view expects a one-to-one

relationship between the combined objects. If you are selecting objects for the view, the module will warn you if they are not related in this way.

Maximum # of characters in a column name

This setting limits the number of characters that can appear in the name of a field in the database. MySQL has a limit of 64 characters per field, but also has an overall limit on the number of characters in all of the columns of a table. **ExportToDatabase** will shorten all of the column names by removing characters, at the same time guaranteeing that no two columns have the same name.

Write image thumbnails directly to the database?

(Used only if MySQL or SQLite are selected as database type)
Select Yes if you'd like to write image thumbnails directly into the database. This will slow down the writing step, but will enable new functionality in CellProfiler Analyst such as quickly viewing images in the Plate Viewer tool by selecting "thumbnail" from the "Well display" dropdown.

Select the images for which you want to save thumbnails

(Used only if MySQL or SQLite are selected as database type and writing thumbnails is selected)
Select the images that you wish to save as thumbnails to the database. Make multiple selections by using Ctrl-Click (Windows) or Command-Click (Mac);

Auto-scale thumbnail pixel intensities?

(Used only if MySQL or SQLite are selected as database type and writing thumbnails is selected)
Select *Yes* if you'd like to automatically rescale the thumbnail pixel intensities to the range 0-1, where 0 is black/unsaturated, and 1 is white/saturated.

Module: ExportToSpreadsheet

Export To Spreadsheet exports measurements into one or more files that can be opened in Excel or other spreadsheet programs.

This module will convert the measurements to a comma-, tab-, or other character-delimited text format and save them to the hard drive in one or several files, as requested.

Using metadata tags for output

ExportToSpreadsheet can write out separate files for groups of images based on their metadata tags. This is controlled by the directory and file names that you enter. For instance, you might have applied two treatments to each of your samples and labeled them with the metadata names "Treatment1" and "Treatment2", and you might want to create separate files for each combination of treatments, storing all measurements with a given "Treatment1" in separate directories. You can do this by specifying metadata tags for the folder name and file name:

- Choose *Elsewhere...* or *Default Input/Output Folder sub-folder* for the output file location.
- Insert the metadata tag of choice into the output path. You can insert a previously defined metadata tag by either using:
  - The insert key
  - A right mouse button click inside the control
  - In Windows, the Context menu key, which is between the Windows key and Ctrl key

  The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:
  - Use the up and down arrows to cycle through possible values.
  - Right-click on the tag to display and select the available values.

  In this instance, you would select the metadata tag "Treatment1"
- Uncheck "Export all measurements?"
- Uncheck *Use the object name for the file name?*.
- Using the same approach as above, select the metadata tag "Treatment2", and complete the filename by appending the text ".csv".

Here's an example table of the files that would be generated:

Treatment1	Treatment2	Path
1M_NaCl	20uM_DMSO	1M_NaCl/20uM_DMSO.csv
1M_NaCl	40uM_DMSO	1M_NaCl/40uM_DMSO.csv
2M_NaCl	20uM_DMSO	2M_NaCl/20uM_DMSO.csv
2M_NaCl	40uM_DMSO	2M_NaCl/40uM_DMSO.csv

Available measurements

For details on the nomenclature used by CellProfiler for the exported measurements, see *Help* > *General Help* > *How Measurements Are Named*.

See also **ExportToDatabase**.

Settings:

Select the column delimiter

Select the delimiter to use, i.e., the character that separates columns in a file. The two default choices are tab and comma, but you can type in any single character delimiter you would prefer. Be sure that the delimiter you choose is not a character that is present within your data (for example, in file names).

Add image metadata columns to your object data file?

" Image_Metadata_" columns are normally exported in the Image data file, but if you select *Yes*, they will also be exported with the Object data file(s).

Limit output to a size that is allowed in Excel?

If your output has more than 256 columns, select *Yes* will open a window allowing you to select the columns you'd like to export. If your output exceeds 65,000 rows, you can still open the CSV in Excel, but not all rows will be visible.

Select the measurements to export

Select *Yes* to provide a button that allows you to select which measurements you want to export. This is useful if you know exactly what measurements you want included in the final spreadheet(s).

Calculate the per-image mean values for object measurements?

Select *Yes* for **ExportToSpreadsheet** to calculate population statistics over all the objects in each image and save that value as an aggregate measurement in the Image file. For instance, if you are measuring the area of the Nuclei objects and you check the box for this option, **ExportToSpreadsheet** will create a column in the Image file called "Mean_Nuclei_AreaShape_Area".

You may not want to use **ExportToSpreadsheet** to calculate these measurements if your pipeline generates a large number of per-object measurements; doing so might exceed Excel's limits on the number of columns (256).

Output file location

This setting lets you choose the folder for the output files. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the

Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

For *Elsewhere...*, *Default Input Folder sub-folder* and *Default Output Folder sub-folder*, if you have metadata associated with your images via **Metadata** module, you can name the folder using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have a metadata tag named "Plate", you can create a per-plate folder by selecting one of the subfolder options and then specifying the subfolder name as "\g<Plate>". The module will substitute the metadata values for the current image set for any metadata tags in the folder name. Please see the **Metadata** module for more details on metadata collection and usage.

Create a GenePattern GCT file?

Select *Yes* to create a GCT file compatible with GenePattern. The GCT file format is a tab-delimited text file format that describes a gene expression dataset; the specifics of the format are described here. By converting your measurements into a GCT file, you can make use of GenePattern's data visualization and clustering methods.

Each row in the GCT file represents (ordinarily) a gene and each column represents a sample (in this case, a per-image set of measurements). In addition to any other spreadsheets desired, enabling this option will produce a GCT file with the extension .gct, prepended with the text selection above. If per-image aggregate measurements are requested above, those measurements are included in the GCT file as well.

Select source of sample row name

(Used only if a GenePattern file is requested)
The first column of the GCT file is the unique identifier for each sample, which is ordinarily the gene name. This information may be specified in one of two ways:

- *Metadata:* If you used the **Metadata** modules to add metadata to your images, you may specify a metadata tag that corresponds to the identifier for this column. Please see the **Metadata** module for more details on metadata collection and usage.
- *Image filename:* If the gene name is not available, the image filename can be used as a surrogate identifier.

Select the image to use as the identifier

(Used only if a GenePattern file is requested and image filename is used to name each row)

Select which image whose filename will be used to identify each sample row.

Select the metadata to use as the identifier

(Used only if a GenePattern file is requested and metadata is used to name each row)
Choose the measurement that corresponds to the identifier, such as metadata from the **Metadata** module. Please see the **Metadata** module for more details on metadata collection and usage.

Export all measurement types?

Select *Yes* to export every category of measurement. **ExportToSpreadsheet** will create one data file for each object produced in the pipeline, as well as per-image, per-experiment and object relationships, if relevant. See *Help > Using Your Output > How Measurements are Named* for more details on the various measurement types. The module will use the object name as the file name, optionally prepending the output file name if specified above.

Select *No* if you want to do either (or both) of two things:

- Specify which objects should be exported;
- Override the automatic nomenclature of the exported files.

Press button to select measurements to export

(Used only when selecting the columns of measurements to export)
This setting controls the columns to be exported. Press the button and check the measurements or categories to export.

Representation of Nan/Inf

This setting controls the output for numeric fields if the calculated value is infinite (*Inf*) or undefined (*NaN*). CellProfiler will produce Inf or NaN values under certain rare circumstances, for instance when calculating the mean intensity of an object within a masked region of an image.

- *Null:* Output these values as empty fields.
- *NaN:* Output them as the strings "NaN", "Inf" or "-Inf".

Add a prefix to file names?

This setting lets you choose whether or not to add a prefix to each of the .CSV filenames produced by **ExportToSpreadsheet**. A prefix may be useful if you use the same directory for the results of more than one pipeline; you can specify a different prefix in each pipeline. Select *Yes* to add a prefix to each file name (e.g. "MyExpt_Images.csv"). Select *No* to use filenames without prefixes (e.g. "Images.csv").

Filename prefix:

(*Used only if "Add a prefix to file names?" is Yes*) The text you enter here is prepended to the names of each file produced by **ExportToSpreadsheet**.

Overwrite without warning?

This setting either prevents or allows overwriting of old .CSV files by **ExportToSpreadsheet** without confirmation. Select *Yes* to overwrite without warning any .CSV file that already exists. Select *No* to

prompt before overwriting when running CellProfiler in the GUI and to fail when running headless.

Data to export

(Used only when "Export all measurements?" is set to "No")
Choose *Image*, *Experiment*, *Object relationships* or an object name from the list. **ExportToSpreadsheet** will write out a file of measurements for the given category. See *Help > Using Your Output > How Measurements are Named* for more details on the various measurement types.

Combine these object measurements with those of the previous object?

(Used only when "Export all measurements?" is set to "No")
Select *Yes* to create a file composed of measurements made on this object and the one directly above it.

Select *No* to create separate files for this and the previous object.

File name

(Used only when "Export all measurements?" is set to "No")
Enter a file name for the named objects' measurements. **ExportToSpreadsheet** will prepend the name of the measurements file to this if you asked to do so above. If you have metadata associated with your images, this setting will also substitute metadata tags if desired. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

Please see the **Metadata** module for more details on metadata collection and usage.

Use the object name for the file name?

(Used only when "Export all measurements?" is set to "No")
Select *Yes* to use the object name as selected above to generate a file name for the spreadsheet. For example, if you selected *Image*, above and have not checked the *Prepend output file name* option, your output file will be named "Image.csv".

Select *No* to name the file yourself.

Module: FlagImage

Flag Image allows you to flag an image based on properties that you specify, for example, quality control measurements.

This module allows you to assign a flag if an image meets certain measurement criteria that you specify (for example, if the image fails a quality control measurement). The value of the flag is 1 if the image meets the selected criteria (for example, if it fails QC), and 0 if it does not meet the criteria (if it passes QC). The flag can be used in post-processing to filter out images you do not want to analyze, e.g., in CellProfiler Analyst. In addition, you can use **ExportToSpreadsheet** to generate a file that includes the flag as a metadata measurement associated with the images. The **Metadata** module can then use this flag to put images that pass QC into one group and images that fail into another.

A flag can be based on one or more measurements. If you create a flag based on more than one measurement, you can choose between setting the flag if all measurements are outside the bounds or if one of the measurements is outside of the bounds.

This module must be placed in the pipeline after the relevant measurement modules upon which the flags are based.

Settings:

Name the flag's category

Name a measurement category by which to categorize the flag. The *Metadata* category is the default used in CellProfiler to store information about images (referred to as *metadata*).

The flag is stored as a per-image measurement whose name is a combination of the flag's category and feature name, underscore delimited. For instance, if the measurement category is *Metadata* and the feature name is *QCFlag*, then the default measurement name would be *Metadata_QCFlag*. Please see the **Metadata** module for more details on metadata collection and usage

Name the flag

The flag is stored as a per-image measurement whose name is a combination of the flag's category and feature name, separated by underscores. For instance, if the measurement category is *Metadata* and the feature name is *QCFlag*, then the default measurement name would be *Metadata_QCFlag*.

Flag if any, or all, measurement(s) fails to meet the criteria?

- *Flag if any fail:* An image will be flagged if any of its measurements fail. This can be useful for flagging images possessing multiple QC flaws; for example, you can flag all bright images and all out of focus images with one flag.
- *Flag if all fail:* A flag will only be assigned if all measurements fail. This can be useful for flagging images that possess only a combination of QC flaws; for example, you can flag only images that are both bright and out of focus.

Skip image set if flagged?

Select *Yes* to skip the remainder of the pipeline for image sets that are flagged. CellProfiler will not run

subsequent modules in the pipeline on the images in any image set that is flagged. Select *No* for CellProfiler to continue to process the pipeline regardless of flagging.

You may want to skip processing in order to filter out unwanted images. For instance, you may want to exclude out of focus images when running **CorrectIllumination_Calculate**. You can do this with a pipeline that measures image quality and flags inappropriate images before it runs **CorrectIllumination_Calculate**.

Flag is based on

- *Whole-image measurement:* A per-image measurement, such as intensity or granularity.
- *Average measurement for all objects in each image:* The average of all object measurements in the image.
- *Measurements for all objects in each image:* All the object measurements in an image, without averaging. In other words, if *any* of the objects meet the criteria, the image will be flagged.
- *Rules:*Use a text file of rules produced by CellProfiler Analyst. If you choose *Rules*, you will have to ensure that this pipeline makes every measurement in the rules file prior to this module.

Select the object to be used for flagging

(Used only when flag is based on an object measurement)
Select the objects whose measurements you want to use for flagging.

Flag images based on low values?

Select *Yes* to flag images with measurements below the specified cutoff.

Flag images based on high values?

Select *Yes* to flag images with measurements above the specified cutoff.

Rules file location

(Used only when flagging using Rules)
Select the location of the rules file that will be used for filtering. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.

- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

Rules file name

(Used only when flagging using Rules)
The name of the rules file. This file should be a plain text file containing the complete set of rules.

Each line of this file should be a rule naming a measurement to be made on an image, for instance:

```
IF (Image_ImageQuality_PowerLogLogSlope_DNA < -2.5, [0.79, -0.79], [-0.94, 0.94])
```

The above rule will score +0.79 for the positive category and -0.94 for the negative category for images whose power log slope is less than -2.5 pixels and will score the opposite for images whose slope is larger. The filter adds positive and negative and flags the images whose positive score is higher than the negative score.

Class number

(Used only when flagging using Rules)
Select which classes to flag when filtering. The CellProfiler Analyst classifier user interface lists the names of the classes in order. By default, these are the positive (class 1) and negative (class 2) classes. **FlagImage** uses the first class from CellProfiler Analyst if you choose "1", etc.

Please note the following:

- The flag is set if the image falls into the selected class.
- You can make multiple class selections. If you do so, the module will set the flag if the image falls into any of the selected classes.

Module: MergeOutputFiles

MergeOutputFiles merges several output .mat files into one.

This data tool lets you collect the output .mat files from several runs, for instance, as might be created by running CellProfiler in batch mode.

MergeOutputFiles is a pure data tool; *you cannot use it as a module*, and it will generate an error if you try to do so. To use it as a data tool, choose it from the *Data Tools* menu to bring up the **MergeOutputFiles** dialog.

The dialog has the following parts:

- *Destination file:* This is the name of the file that will be created. The file will contain all merged input data files in MATLAB format.
- *File list:* The file list is the box with the columns, "Folder" and "File". It will be empty until you add files using the "Add..." button. Measurement files are written out to the destination file in the order they appear in this list. You can select multiple files in this box to move them up or down or to remove them.
- *Add button:* Brings up a file chooser when you press it. You can select multiple files from the file chooser and they will be added in alphabetical order to the bottom of the current list of files.
- *Remove button:* Removes all currently selected files from the list.
- *Up button:* Moves the currently selected files up in the list.
- *Down button:* Mves the currently selected files down in the list.
- *OK button:*Accepts the file list and writes it to the output.
- *Cancel button:* Closes the dialog without performing any operation.

Once merged, this output file will be compatible with other data tools. Output files can be quite large, so prior to merging, be sure that the total size of the merged output file is of a reasonable size to be opened on your computer (based on the amount of memory available on your computer). It may be preferable instead to import data from individual output files directly into a database using **ExportDatabase** as a data tool.

See also **CreateBatchFiles**, **ExportToDatabase**.

Module: CreateBatchFiles

Create Batch Files produces files that allow individual batches of images to be processed separately on a cluster of computers.

This module creates files that can be submitted in parallel to a cluster for faster processing. It should be placed at the end of an image processing pipeline.

If your computer mounts the file system differently than the cluster computers, **CreateBatchFiles** can replace the necessary parts of the paths to the image and output files. For instance, a Windows machine might access files images by mounting the file system using a drive letter, like this:

```
Z:\your_data\images
```

and the cluster computers access the same file system like this:

```
/server_name/your_name/your_data/images
```

In this case, the local root path is `Z:\your_data\images` and the cluster root path is `/server_name/your_name/your_data/images`.

For more details on batch processing, please see *Help* > *Other Features* > *Batch Processing*.

Settings:

Store batch files in default output folder?

Select *Yes* to store batch files in the Default Output folder.
Select *No* to enter the path to the folder that will be used to store these files.

Output folder path

Enter the path to the output folder.

Are the cluster computers running Windows?

Select *Yes* if the cluster computers are running one of the Microsoft Windows operating systems. In this case, **CreateBatchFiles** will modify all paths to use the Windows file separator (backslash \).
Select *No* for **CreateBatchFiles** to modify all paths to use the Unix or Macintosh file separator (slash /).

Local root path

Enter the path to files on this computer. This is the root path on the local machine (i.e., the computer setting up the batch files). If **CreateBatchFiles** finds any pathname that matches the local root path at the begining, it will replace the start with the cluster root path.

For example, if you have mapped the remote cluster machine like this:

`Z:\your_data\images` (on a Windows machine, for instance)

and the cluster machine sees the same folder like this:

`/server_name/your_name/your_data/images`

you would enter `z:` here and /server_name/your_name/ for the cluster path in the next setting.

Cluster root path

Enter the path to files on the cluster. This is the cluster root path, i.e., how the cluster machine sees the top-most folder where your input/output files are stored.

For example, if you have mapped the remote cluster machine like this:

`Z:\your_data\images` (on a Windows machine, for instance)

and the cluster machine sees the same folder like this:

`/server_name/your_name/your_data/images`

you would enter `z:` in the previous setting for the local machine path and /server_name/your_name/ here.

Module: Groups

The **Groups** module organizes sets of images into groups.

Once the images have been identified with the **Images** module, have had metadata associated with them using the **Metadata** module, and have been assigned names by the **NamesAndTypes** module, you have the option of further sub-dividing the image sets into groups that share a common feature. Some downstream modules of CellProfiler are capable of processing groups of images in useful ways (e.g., object tracking within a set of images comprising a time series, illumination correction within a set of images comprising an experimental batch, data export for a set of images comprising a plate).

What is an image "group"?

The key to understanding why grouping may be necessary is that CellProfiler processes the input images sequentially and in the order given by the NamesAndTypes module. If you have multiple collections (or "groups") of images that should be processed independently from each other, CellProfiler will simply finish processing one collection and proceed to the next, ignoring any distinction between them unless told otherwise via the **Groups** module.

To illustrate this idea, below are two examples where the grouping concept can be useful or important:

- If you have time-lapse movie data that is in the form of individual image files, and you are performing object tracking, it is important to indicate to CellProfiler that the end of a movie indicates the end of a distinct data set. Without doing so, CellProfiler will simply take the first frame of the next movie as a continuation of the previous one. If each set of files that comprise a movie is defined using the **Metadata** module, the relevant metadata can be used in this module to insure that object tracking only takes place within each movie.
- If you are performing illumination correction for a screening experiment, we recommend that the illumination function (an image which represents the overall background fluorescence) be calculated on a per-plate basis. Since the illumination function is an aggregate of images from a plate, running a pipeline must yield a single illumination function for each plate. Running this pipeline multiple times, once for each plate, will give the desired result but would be tedious and time-consuming. In this case, CellProfiler can use image grouping for this purpose; if plate metadata can be defined by the **Metadata** module, grouping will enable you to process images that have the same plate metadata together.

What are the inputs?

Using this module assumes that you have already adjusted the following Input modules:

- Used the **Images** module to produce a list of images to analyze.
- Used the **Metadata** module to produce metadata defining the distinct sub-divisions between groups of images.
- Used the **NamesAndTypes** module to assign names to individual channels and create image sets.

What do the settings mean?

See below for help on the individual settings. Selecting this module will display a panel, allowing you to select whether you want to create groups or not. A grouping may be defined as according to any or as many of the metadata categories as defined by the **Metadata** module. By selecting a metadata tag from the drop-down for the metadata category, the **Groups** module will sub-divide and assemble the image sets according to their unique metadata value. Upon adding a metadata category, the two tables underneath will update to show the resultant organization of the image sets for each group.

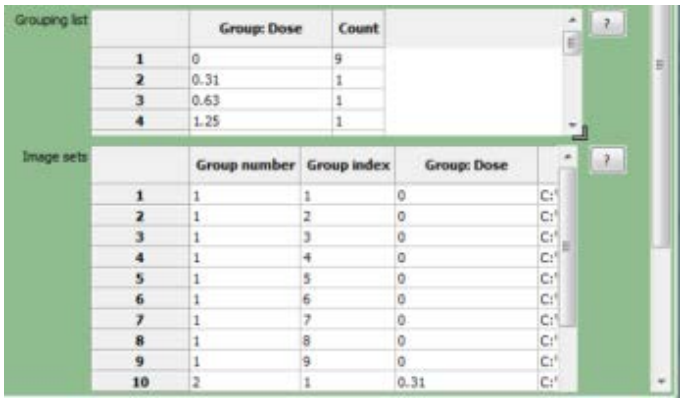
What do I get as output?

The final product of the **Groups** module is a list defining subsets of image sets that will be processed independently of the other subsets.

- If no groups are defined, the Analysis modules in the rest of the pipeline will be applied to all images in exactly the same way.
- If groups are defined in the **Groups** module, then organizationally (and transparently to you), CellProfiler will begin the analyses with the first image set of the group, end with the last image set of the group, and then proceed to the next group.

The two tables at the bottom provide the following information when a metadata category is selected:

- The *grouping list*(top table) shows the unique values of the selected metadata under the "Group" column; each of the unique values comprises a group. The "Count" column shows the number of image sets included in a given group; this is useful as a "sanity check" to make sure that the expected numbers of images are present.
- The *image set list* (bottom table) shows the file name and location of each of the image sets that comprise the groups.



Available measurements

- *Group_Number:* The index of each grouping, as defined by the unique combinations of the metadata tags specified. These are written to the per-image table.
- *Group_Index:* The index of each imaget set within each grouping, as defined by the *Group_Number*. These are written to the per-image table.

Technical notes

To perform grouping, only one analysis worker (i.e., copy of CellProfiler) will be allocated to handle each group. This means that you may have multiple workers created (as set under the Preferences), but only a subset of them may actually be active, depending on the number of groups you have.

Settings:

Do you want to group your images?

Select *Yes* if you need to split your images into image subsets (or *groups*) such that each group is processed independently of each other. See the main module help for more details.

Metadata category

Specify the metadata category with which to define a group. Once a selection is made, the two listings below will display the updated values:

- The *grouping list* (top table) shows the unique values of the selected metadata under the "Group" column; each of the unique values comprises a group. The "Count" column shows the number of image sets included in a given group; this is useful as a "sanity check" to make sure that the expected numbers of images are present.
- The *image set list* (bottom table) shows the file name and location of each of the image sets that comprise the groups. In this example, the table has 26 rows, one for each of the DNA and GFP image sets defined by the **NamesAndTypes** module.

You may specify multiple metadata tags to group with by clicking the "Add" button. This would be necessary if a combination of metadata is required in order to define a group. Upon adding a metadata category, the two tables will update in the panels below showing the resulting organization of the image data for each group.

As an example, an time-lapse experiment consists of a set of movie images (indexed by a frame number), collected on a per-well basis. The plate, well, wavelength and frame number metadata have been extracted using the **Metadata** module. Using the **NamesAndTypes** module, the two image channels (OrigBlue, *w1* and OrigGreen, *w2*) have been set up in the following way:

Image set number	OrigBlue (w1) file name	OrigGreen (w2) file name	Plate	Well	FrameNumber
1	P-12345_A01_t001_w1.tif	P-12345_A01_t001_w2.tif	P-12345	A01	t001
2	P-12345_A01_t002_w1.tif	P-12345_A01_t002_w2.tif	P-12345	A01	t002
3	P-12345_B01_t001_w1.tif	P-12345_B01_t001_w2.tif	P-12345	B01	t001
4	P-12345_B01_t002_w1.tif	P-12345_B01_t002_w2.tif	P-12345	B01	t002
5	2-ABCDF_A01_t001_w1.tif	2-ABCDF_A01_t001_w2.tif	2-ABCDF_	A01	t001
6	2-ABCDF_A01_t002_w1.tif	2-ABCDF_A01_t002_w2.tif	2-ABCDF_	A01	t002
7	2-ABCDF_B01_t001_w1.tif	2-ABCDF_B01_t001_w2.tif	2-ABCDF_	B01	t001
8	2-ABCDF_B01_t002_w1.tif	2-ABCDF_B01_t002_w2.tif	2-ABCDF_	B01	t002

We would like to perform object tracking for each movie, i.e., for each plate and well. Without the use of groups, even though image sets 1–2, 3–4, 5–6, and 7–8 represent different movies, image set 3 will get processed immediately after image set 2, image set 5 after 4, and so on. For an object tracking assay, failure to recognize where the movies start and end would lead to incorrect tracking results.

Selecting the *Plate* followed by the *Well* metadata as the metadata categories will create four groups based on the unique plate and well combinations:

Grouping tags		Image set tags				Channels	
Group number	Group index	Image set number	Plate	Well	FrameNumber	OrigBlue	OrigGreen
1	1	1	P-12345	A01	t001	P-12345_A01_t001_w1.tif	P-12345_A01_t001_w2.tif
	2	2	P-12345	A01	t002	P-12345_A01_t002_w1.tif	P-12345_A01_t002_w2.tif

2	1	3	P-12345	B01	t001	P-12345_B01_t001_w1.tif	P-12345_B01_t001_w2.tif
	2	4	P-12345	B01	t002	P-12345_B01_t002_w1.tif	P-12345_B01_t002_w2.tif
3	1	5	2-ABCDF	A01	t001	2-ABCDF_A01_t001_w1.tif	2-ABCDF_A01_t001_w2.tif
	2	6	2-ABCDF	A01	t002	2-ABCDF_A01_t002_w1.tif	2-ABCDF_A01_t002_w2.tif
4	1	7	2-ABCDF	B01	t001	2-ABCDF_B01_t001_w1.tif	2-ABCDF_B01_t001_w2.tif
	2	8	2-ABCDF	B01	t002	2-ABCDF_B01_t002_w1.tif	2-ABCDF_B01_t002_w2.tif

Each group will be processed independently from the others, which is the desired behavior.

Grouping list

This list shows the unique values of the selected metadata under the "Group" column; each of the unique values comprises a group. The "Count" column shows the number of image sets that included in a given group; this is useful as a "sanity check", to make sure that the expected number of images are present. For example, if you are grouping by per-plate metadata from a 384-well assay with 2 sites per well consisting of 3 plates, you would expect to see 3 groups (each from the 3 unique plate IDs), with 384 wells × 2 sites/well = 768 image sets in each.

Image sets

This list displays the file name and location of each of the image sets that comprise the group. For example, if you are grouping by per-plate metadata from a 384-well assay with 2 sites per well consisting of 3 plates, you would expect to see a table consisting of 3 plates × 384 wells/plate ×2 sites/well = 2304 rows.

Module: Images

The **Images** module specifies the location of image files to be analyzed by your pipeline.

The **Images** module allows you to specify the location of files to be analyzed by the pipeline; setting this module correctly is the first step in creating a new project in CellProfiler. These files can be located on your hard drive, on a networked computer elsewhere, or accessible with a URL. You can also provide rules to specify only those files that you want analyzed out of a larger collection (for example, from a folder containing both images for analysis and non-image files that should be disregarded).

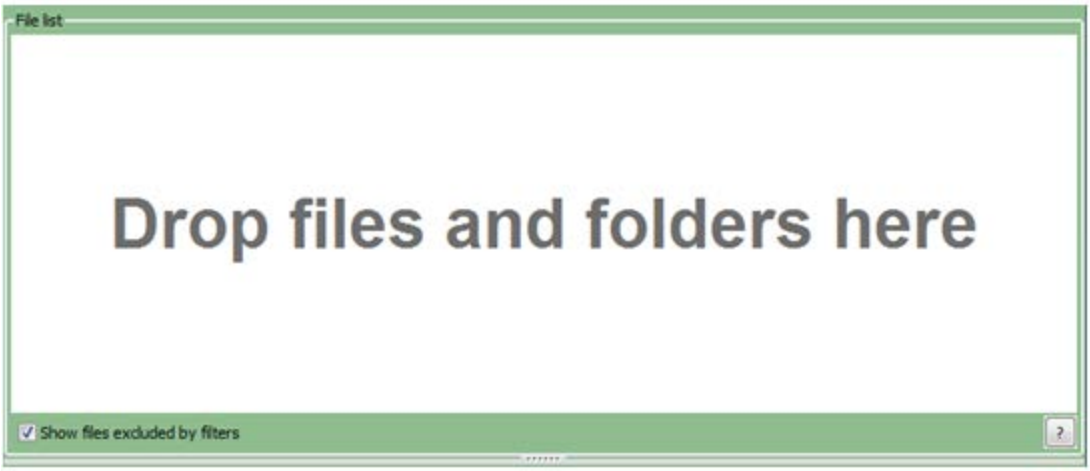
What is a "digital image"?

A *digital image* is a set of numbers arranged into a two-dimensional format of rows and columns; a pixel refers to the row/column location of a particular point in the image. Pixels in grayscale or monochrome (black/white) images contain a single intensity value, whereas in color images, each pixel contains a red, green, and blue (RGB) triplet of intensity values. Additionally, the term image can be used as short-hand for an image sequence, that is, an image collection such as a time-lapse series (2-D + *t*), confocal Z-stacks (3-D), etc.

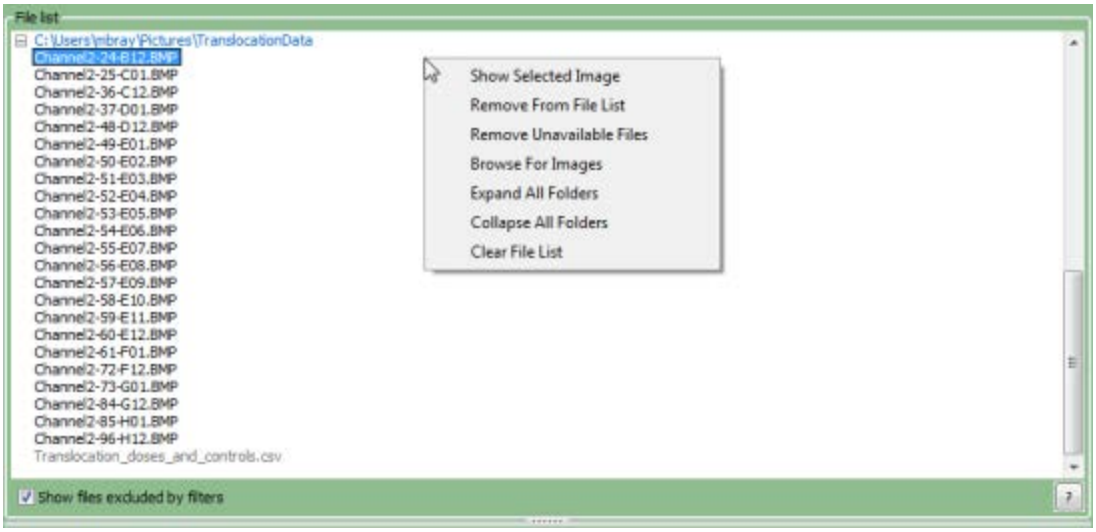
CellProfiler can read a wide variety of image formats by using a library called Bio-Formats; see here for the formats available. Some image formats are better than others for use in image analysis. Some are "lossy" (information is lost in the conversion to the format) like most JPG/JPEG files; others are "lossless" (no image information is lost). For image analysis purposes, a lossless format like TIF or PNG is recommended.

What do I need as input?

The most straightforward way to provide image files to the **Images** module is to simply drag-and-drop them on the file list panel (the blank space indicated by the text "Drop files and folders here").



Using the file explorer tool of your choice (e.g., Explorer in Windows, Finder in Mac), you can drag-and-drop individual files and/or entire folders into this panel. You can also right-click in the File list panel to bring up a file selection window to browse for individual files; on the Mac, folders can be drag-and-dropped from this window and you can select multiple files using Ctrl-A (Windows) or Cmd-A (Mac).



Right-clicking on the file list panel will provide a context menu with options to modify the file list:

- *Show Selected Image:* Selecting this option (or double-clicking on the file) will open the image in a new window.
- *Remove From List:* Removes the selected file or folder from the list. Note that this does not remove the file/folder from the hard drive.
- *Remove Unavailable Files:* Refresh the list by checking for existence of file. Note that this does not remove the file from the hard drive.
- *Browse For Images:* Use a dialog box to select an image file (though drag-and-drop is recommended).
- *Refresh:* Shown only if folder is selected. Refresh the list of files from the folder. Files that were manually removed from the list for that folder are restored.
- *Expand All Folders:* Expand all trees shown in the file list panel.
- *Collapse All Folders:* Collapse all folder trees shown in the file list panel.
- *Clear File List:* Remove all files/folders in the file list panel. You will be prompted for confirmation beforehand.

What do the settings mean?

If you have a subset of files that you want to analyze from the full listing shown in the panel, you can filter the files according to a set of rules. This is useful in cases such as:

- You have dragged a folder of images onto the file list panel, but the folder contains images you want to analyze along with non-image files that you want to disregard.
- You have dragged a folder of images onto the file list panel, but the folder contains the images from one experiment that you want to process along with images from another experiment that you want to ignore for now.

You may specify as many rules as necessary to define the desired list of images.

After you have filtered the file list, press the "Apply" button to update the view of the file list. You can also toggle the "Show file excluded by filters" box to modify the display of the files:

- Checking this box will show all the files in the list, with the files that have been filtered out shown as grayed-out entries.
- Not checking this box will only show the files in the list that pass the filter(s).

What do I get as output?

The final product of the **Images** module is a file list in which any files that are not intended for further processing have been removed, whether manually or using filtering. This list will be used when collecting metadata (if desired) and when assembling the image sets in NamesAndTypes. The list can be filtered further in NamesAndTypes to specify, for example, that a subset of these images represents a particular wavelength.

Settings:

Filter images?

The **Images** module will pass all the files specified in the file list panel downstream to have a meaningful name assigned to it (so other modules can access it) or optionally, to define the relationships between images and associated metadata. Enabling file filtering will allow you to specify a subset of the files from the file list panel by defining rules to filter the files. This approach is useful if, for example, you drag-and-dropped a folder onto the file list panel which contains a mixture of images that you want to analyze and other files that you want to ignore.

Several options are available for this setting:

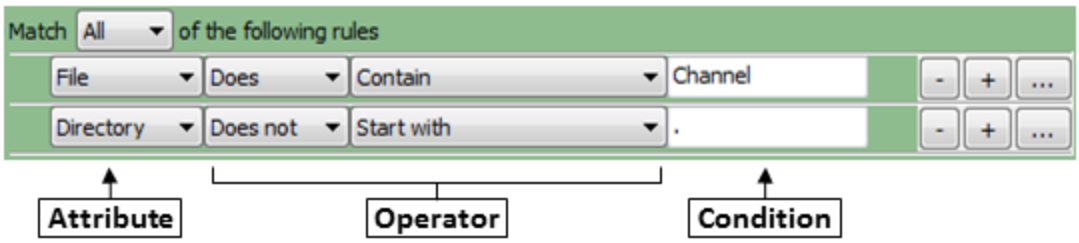
- *No filtering:* Do not enable filtering; all files in the File list panel will be passed to downstream modules for processing. This option can be selected if you are sure that only images are specified in the list.
- *Images only:* Only image files will be passed to downstream modules. The permissible image formats are provided by a library called Bio-Formats; see here for the formats available.
- *Custom:* Specify custom rules for selecting a subset of the files from the File list panel. This approach is useful if, for example, you drag-and-dropped a folder onto the File list panel which contains a mixture of images that you want to analyze and other files that you want to ignore.

Select the rule criteria

Specify a set of rules to narrow down the files to be analyzed.

Clicking the rule menus shows you all the file *attributes*, *operators* and *conditions* you can specify to narrow down the image list.

1. For each rule, first select the *attribute* that the rule is to be based on. For example, you can select "File" to define a rule that will filter files on the basis of their filename.
2. The *operator* drop-down is then updated with operators applicable to the attribute you selected. For example, if you select "File" as the attribute, the operator menu includes text operators such as *Contain* or *Starts with*. On the other hand, if you select "Extension" as the attribute, you can choose the logical operators "Is" or "Is not" from the menu.
3. In the operator drop-down menu, select the operator you want to use. For example, if you want to match data exactly, you may want the "Exactly match" or the "Is" operator. If you want the condition to be more loose, select an operator such as "Contains".
4. Use the *condition* box to type the condition you want to match. The more you type, the more specific the condition is.
    - As an example, if you create a new filter and select *File* as the attribute, then select "Does" and "Contain" as the operators, and type "Channel" as the condition, the filter finds all files that include the text "Channel", such as "Channel1.tif" "Channel2.jpg", "1-Channel-A01.BMP" and so on.
    - If you select "Does" and "Start with" as the operators and "Channel1" in the Condition box, the rule will includes such files as "Channel1.tif" "Channel1-A01.png", and so on.



You can also create regular expressions (an advanced syntax for pattern matching; see below) in order to select particular files.

To add another rule, click the plus buttons to the right of each rule. Subtract an existing rule by clicking the minus button.

You can also link a set of rules by choosing the logical expression *All* or *Any*. If you use *All* logical expression, all the rules be true for a file to be included in the File list. If you use the *Any* option, only one of the conditions has to be met for a file to be included.

If you want to create more complex rules (e.g, some criteria matching all rules and others matching any), you can create sets of rules, by clicking the ellipsis button (to the right of the plus button). Repeat the above steps to add more rules to the filter until you have all the conditions you want to include.

Details on regular expressions

A *regular expression* is a general term refering to a method of searching for pattern matches in text. There is a high learning curve to using them, but are quite powerful once you understand the basics.

Patterns are specified using combinations of metacharacters and literal characters. There are a few classes of metacharacters, partially listed below. Some helpful links follow:

- A more extensive explanation of regular expressions can be found here
- A helpful quick reference can be found here
- Pythex provides quick way to test your regular expressions. Here is an example to capture information from a common microscope nomenclature.

The following metacharacters match exactly one character from its respective set of characters:

Metacharacter	Meaning
.	Any character
[]	Any character contained within the brackets
[^]	Any character not contained within the brackets
\w	A word character [a-z_A-Z0-9]
\W	Not a word character [^a-z_A-Z0-9]
\d	A digit [0-9]
\D	Not a digit [^0-9]
\s	Whitespace [\t\r\n\f\v]
\S	Not whitespace [^ \t\r\n\f\v]

The following metacharacters are used to logically group subexpressions or to specify context for a position in the match. These metacharacters do not match any characters in the string:

Metacharacter	Meaning
( )	Group subexpression
|	Match subexpression before or after the |
^	Match expression at the start of string
$	Match expression at the end of string
\<	Match expression at the start of a word
\>	Match expression at the end of a word

The following metacharacters specify the number of times the previous metacharacter or grouped subexpression may be matched:

Metacharacter	Meaning
*	Match zero or more occurrences
+	Match one or more occurrences
?	Match zero or one occurrence
{n,m}	Match between n and m occurrences

Characters that are not special metacharacters are all treated literally in a match. To match a character that is a special metacharacter, escape that character with a '\'. For example '.' matches any character, so to match a '.' specifically, use '\.' in your pattern. Examples:

- `[trm]ail` matches 'tail' or 'rail' or 'mail'.
- `[0-9]` matches any digit between 0 to 9.
- `[^Q-S]` matches any character other than 'Q' or 'R' or 'S'.
- `[[]A-Z]` matches any upper case alphabet along with square brackets.
- `[ag-i-9]` matches characters 'a' or 'g' or 'h' or 'i' or '-' or '9'.
- `[a-p]*` matches '' or 'a' or 'aab' or 'p' etc.
- `[a-p]+` matches 'a' or 'abc' or 'p' etc.
- `[^0-9]` matches any string that is not a number.
- `^[0-9]*$` matches either a blank string or a natural number.
- `^-[0-9]+$|^\+?[0-9]+$` matches any integer.

Module: LoadData

Load Data loads text or numerical data to be associated with images, and can also load images specified by file names.

This module loads a file that supplies text or numerical data associated with the images to be processed, e.g., sample names, plate names, well identifiers, or even a list of image filenames to be processed in the analysis run.

Disclaimer: Please note that the Input modues (i.e., **Images**, **Metadata**, **NamesAndTypes** and **Groups**) largely supercedes this module. However, old pipelines loaded into CellProfiler that contain this module will provide the option of preserving them; these pipelines will operate exactly as before.

The module currently reads files in CSV (comma-separated values) format. These files can be produced by saving a spreadsheet from Excel as "Windows Comma Separated Values" file format. The lines of the file represent the rows, and each field in a row is separated by a comma. Text values may be optionally enclosed by double quotes. The **LoadData** module uses the first row of the file as a header. The fields in this row provide the labels for each column of data. Subsequent rows provide the values for each image cycle.

There are many reasons why you might want to prepare a CSV file and load it via **LoadData**. Below, we describe how the column nomenclature allows for special functionality for some downstream modules:

- *Columns with any name:* Any data loaded via **LoadData** will be exported as a per-image measurement along with CellProfiler-calculated data. This is a convenient way for you to add data from your own sources to the files exported by CellProfiler.

- *Columns whose name begins with Image_FileName or Image_PathName:* A column whose name begins with "Image_FileName" or "Image_PathName" can be used to supply the file name and path name (relative to the base folder) of an image that you want to load. The image's name within CellProfiler appears afterward. For instance, "Image_FileName_CY3" would supply the file name for the CY3-stained image, and choosing the *Load images based on this data?* option allows the CY3 images to be selected later in the pipeline. "Image_PathName_CY3" would supply the path names for the CY3-stained images. The path name column is optional; if all image files are in the base folder, this column is not needed.

- *Columns whose name begins with Image_ObjectsFileName or Image_ObjectsPathName:* The behavior of these columns is identical to that of "Image_FileName" or "Image_PathName" except that it is used to specify an image that you want to load as objects.

- *Columns whose name begins with Metadata:* A column whose name begins with "Metadata" can be used to group or associate files loaded by **LoadData**.

  For instance, an experiment might require that images created on the same day use an illumination correction function calculated from all images from that day, and furthermore, that the date be captured in the file names for the individual image sets and in a CSV file specifying the illumination correction functions.

  In this case, if the illumination correction images are loaded with the **LoadData** module, the file should have a "Metadata_Date" column which contains the date metadata tags. Similarly, if the individual images are loaded using the **LoadImages** module, **LoadImages** should be set to extract the metadata tag from the file names (see **LoadImages** for more details on how to do so). The pipeline will then match the individual image with their corresponding illumination correction functions based on matching "Metadata_Date" tags. This is useful if the same data is associated with several images (for example, multiple images obtained from a single well).

- *Columns whose name begins with Series or Frame:* A columns whose name begins with "Series" or "Frame" refers to CSVs containing information about image stacks or movies. The name of the image within CellProfiler appears afterward an underscore character. For example, "Frame_DNA" would supply the frame number for the movie/image stack file specified by the "Image_FileName_DNA" and "Image_PathName_DNA" columns.

  Using a CSV for loading frames and/or series from an movie/image stack allows you more flexibility in assembling image sets for operations that would difficult or impossible using the Input modules alone. For example, if you wanted to analyze a movie of 1,000 frames by computing the difference between frames, you could create two image columns in a CSV, one for loading frames 1,2,...,999, and the other for loading frames 2,3,...,1000. In this case, CellProfiler would load the frame and its predecessor for each cycle and **ImageMath** could be used to create the differece image for downstream use.

- *Columns that contain dose-response or positive/negative control information:* The **CalculateStatistics** module can calculate metrics of assay quality for an experiment if provided with information about which images represent positive and negative controls and/or what dose of treatment has been used for which images. This information is provided to **CalculateStatistics** via the **LoadData** module, using particular formats described in the help for **CalculateStatistics**. Again, using **LoadData** is useful if the same data is associated with several images (for example, multiple images obtained from a single well).

Example CSV file:

```
Image_FileName_FITC, Image_PathName_FITC, Metadata_Plate, Titration_NaCl_uM
"04923_d1.tif",      "2009-07-08",        "P-12345",      750
"51265_d1.tif",      "2009-07-09",        "P-12345",      2750
```

After the first row of header information (the column names), the first image-specific row specifies the file, "2009-07-08/04923_d1.tif" for the FITC image (2009-07-08 is the name of the subfolder that contains the image, relative to the Default Input Folder). The plate metadata is "P-12345" and the NaCl titration used in the well is 750 uM. The second image-specific row has the values "2009-07-09/51265_d1.tif", "P-12345" and 2750 uM. The NaCl titration for the image is available for modules that use numeric metadata, such as **CalculateStatistics**; "Titration" will be the category and "NaCl_uM" will be the measurement.

Using metadata in LoadData

If you would like to use the metadata-specific settings, please see *Help > General help > Using metadata in CellProfiler* for more details on metadata usage and syntax. Briefly, **LoadData** can use metadata provided by the input CSV file for grouping similar images together for the analysis run and for metadata-specfic options in other modules; see the settings help for *Group images by metadata* and, if that setting is selected, *Select metadata tags for grouping* for details.

Using MetaXpress-acquired images in CellProfiler

To produce a CSV file containing image location and metadata from a MetaXpress imaging run, do the following:

- Collect image locations from all files that match the string *.tif* in the desired image folder, one row per image.
- Split up the image pathname and filename into separate data columns for **LoadData** to read.
- Remove data rows corresponding to:
  - Thumbnail images (do not contain imaging data)
  - Duplicate images (will cause metadata mismatching)

    - Corrupt files (will cause failure on image loading)
- The image data table may be linked to metadata contained in plate maps. These plate maps should be stored as flat files, and may be updated periodically via queries to a laboratory information management system (LIMS) database.
- The complete image location and metadata is written to a CSV file where the headers can easily be formatted to match **LoadData**'s input requirements (see column descriptions above). Single plates split across multiple directories (which often occurs in MetaXpress) are written to separate files and then merged, thereby removing the discontinuity.

For a GUI-based approach to performing this task, we suggest using Pipeline Pilot.

For more details on configuring CellProfiler (and LoadData in particular) for a LIMS environment, please see our wiki on the subject.

Available measurements

- *Pathname, Filename:* The full path and the filename of each image, if image loading was requested by the user.
- Per-image information obtained from the input file provided by the user.
- *Scaling:* The maximum possible intensity value for the image format.
- *Height, Width:* The height and width of the current image.

See also the **Input** modules, **LoadImages** and **CalculateStatistics**.

Settings:

Input data file location

Select the folder containing the CSV file to be loaded. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

An additional option is the following:

- *URL*: Use the path part of a URL. For instance, an example .CSV file is hosted at *https://svn.broadinstitute.org/CellProfiler/trunk/ExampleImages/ExampleSBSImages/1049_Metadata.csv*

To access this file, you would choose *URL* and enter *https://svn.broadinstitute.org/CellProfiler/trunk/ExampleImages/ExampleSBSImages* as the path location.

Name of the file

Provide the file name of the CSV file containing the data.

Load images based on this data?

Select *Yes* to have **LoadData** load images using the *Image_FileName* field and the *Image_PathName* fields (the latter is optional).

Base image location

The parent (base) folder where images are located. If images are contained in subfolders, then the file you load with this module should contain a column with path names relative to the base image folder (see the general help for this module for more details). You can choose among the following options:

- *Default Input Folder:* Use the Default Input Folder.
- *Default Output Folder:* Use the Default Output Folder.
- *None:* You have an *Image_PathName* field that supplies an absolute path.
- *Elsewhere...*: Use a particular folder you specify.

Process just a range of rows?

Select *Yes* if you want to process a subset of the rows in the CSV file. Rows are numbered starting at 1 (but do not count the header line). **LoadData** will process up to and including the end row.

Rows to process

(Used only if a range of rows is to be specified)
Enter the row numbers of the first and last row to be processed.

Group images by metadata?

Select *Yes* to break the image sets in an experiment into groups that can be processed by different nodes on a computing cluster. Each set of files that share your selected metadata tags will be processed together. See **CreateBatchFiles** for details on submitting a CellProfiler pipeline to a computing cluster for processing.

Select metadata tags for grouping

(Used only if images are to be grouped by metadata)
Select the tags by which you want to group the image files here. You can select multiple tags. For example, if a set of images had metadata for "Run", "Plate", "Well", and "Site", selecting *Run* and *Plate* will create groups containing images that share the same [*Run*,*Plate*] pair of tags.

Rescale intensities?

This option determines whether image metadata should be used to rescale the image's intensities. Some image formats save the maximum possible intensity value along with the pixel data. For instance, a microscope might acquire images using a 12-bit A/D converter which outputs intensity values between zero

and 4095, but stores the values in a field that can take values up to 65535.

Select *Yes* to rescale the image intensity so that saturated values are rescaled to 1.0 by dividing all pixels in the image by the maximum possible intensity value.

Select *No* to ignore the image metadata and rescale the image to 0 – 1.0 by dividing by 255 or 65535, depending on the number of bits used to store the image.

Module: LoadImages

Load Images allows you to specify which images or movies are to be loaded and in which order.

This module tells CellProfiler where to retrieve images and gives each image a meaningful name by which other modules can access it. You can also use **LoadImages** to extract or define the relationships between images and their associated metadata. For example, you could load a group of images (such as three channels that represent the same field of view) together for processing in a single CellProfiler cycle. Finally, you can use this module to retrieve a label matrix and give the collection of objects a meaningful name.

Disclaimer: Please note that the Input modues (i.e., **Images**, **Metadata**, **NamesAndTypes** and **Groups**) largely supercedes this module. However, old pipelines loaded into CellProfiler that contain this module will provide the option of preserving them; these pipelines will operate exactly as before.

When used in combination with a **SaveImages** module, you can load images in one file format and save them in another, using CellProfiler as a file format converter.

Using metadata in LoadImages

If you would like to use the metadata-specific settings, please see *Help* > *General help* > *Using metadata in CellProfiler* for more details on metadata usage and syntax. Briefly, **LoadImages** can extract metadata from the image filename using pattern-matching strings, for grouping similar images together for the analysis run and for metadata-specfic options in other modules; see the settings help for *Where to extract metadata*, and if an option for that setting is selected, *Regular expression that finds metadata in the file name* for the necessary syntax.

Available measurements

- *Pathname, Filename:* The full path and the filename of each image.
- *Metadata:* The metadata information extracted from the path and/or filename, if requested.
- *Scaling:* The maximum possible intensity value for the image format.
- *Height, Width:* The height and width of the current image.

See also the **Input** modules, **LoadData**, **LoadSingleImage**, **SaveImages**.

Settings:

File type to be loaded

CellProfiler accepts the following image file types. For movie file formats, the files are opened as a stack of images and each image is processed individually, although **TrackObjects** can be used to relate objects across timepoints.

- *individual images:* Each file represents a single image. Some methods of file compression sacrifice image quality ("lossy") and should be avoided for automated image analysis if at all possible (e.g., .jpg). Other file compression formats retain exactly the original image information but in a smaller file ("lossless") so they are perfectly acceptable for image analysis (e.g., .png, .tif, .gif). Uncompressed file formats are also fine for image analysis (e.g., .bmp).
- *avi,mov movies:* AVIs (Audio Video Interleave) and MOVs (QuicktTime) files are types of movie files. Only uncompressed AVIs are supported; supported MOVs are listed here. Note that .mov files are

not supported on 64-bit systems.

- *stk movies:* STKs are a proprietary image format used by MetaMorph (Molecular Devices). It is typically used to encode 3D image data, e.g. from confocal microscopy, and is a special version of the TIF format.
- *tif,tiff,flex,zvi movies:* A TIF/TIFF movie is a file that contains a series of images as individual frames. The same is true for the FLEX file format (used by Evotec Opera automated microscopes). ZVIs are a proprietary image format used by Zeiss. It is typically used to encode 3D image data, e.g. from fluorescence microscopy.

File selection method

Three options are available:

- *Text-Exact match:* Used to load image (or movie) files that have a particular piece of text in the name. The specific text that is entered will be searched for in the filenames and the files that contain that text exactly will be loaded and given the name you specify. The search for the text is case-sensitive.
- *Text-Regular expressions:* Used to load image (or movie) files that match a pattern of regular expressions. Patterns are specified using combinations of metacharacters and literal characters. There are a few classes of metacharacters, partially listed below. Some helpful links follow:
  - A more extensive explanation of regular expressions can be found here
  - A helpful quick reference can be found here
  - Pythex provides quick way to test your regular expressions. Here is an example to capture information from a common microscope nomenclature.

  The following metacharacters match exactly one character from its respective set of characters:

Metacharacter	Meaning
.	Any character
[]	Any character contained within the brackets
[^]	Any character not contained within the brackets
\w	A word character [a-z_A-Z0-9]
\W	Not a word character [^a-z_A-Z0-9]
\d	A digit [0-9]
\D	Not a digit [^0-9]
\s	Whitespace [\t\r\n\f\v]
\S	Not whitespace [^ \t\r\n\f\v]

  The following metacharacters are used to logically group subexpressions or to specify context for a position in the match. These metacharacters do not match any characters in the string:

Metacharacter	Meaning
( )	Group subexpression
|	Match subexpression before or after the |
^	Match expression at the start of string
$	Match expression at the end of string
\<	Match expression at the start of a word

\>	Match expression at the end of a word

The following metacharacters specify the number of times the previous metacharacter or grouped subexpression may be matched:

Metacharacter	Meaning
*	Match zero or more occurrences
+	Match one or more occurrences
?	Match zero or one occurrence
{n,m}	Match between n and m occurrences

Characters that are not special metacharacters are all treated literally in a match. To match a character that is a special metacharacter, escape that character with a '\'. For example '.' matches any character, so to match a '.' specifically, use '\.' in your pattern. Examples:

- `[trm]ail` matches 'tail' or 'rail' or 'mail'.
- `[0-9]` matches any digit between 0 to 9.
- `[^Q-S]` matches any character other than 'Q' or 'R' or 'S'.
- `[[]A-Z]` matches any upper case alphabet along with square brackets.
- `[ag-i-9]` matches characters 'a' or 'g' or 'h' or 'i' or '-' or '9'.
- `[a-p]*` matches '' or 'a' or 'aab' or 'p' etc.
- `[a-p]+` matches 'a' or 'abc' or 'p' etc.
- `[^0-9]` matches any string that is not a number.
- `^[0-9]*$` matches either a blank string or a natural number.
- `^-[0-9]+$|^\+?[0-9]+$` matches any integer.

- *Order:* Used when image (or movie) files are present in a repeating order, like "DAPI, FITC, Red; DAPI, FITC, Red;" and so on. Images are loaded based on the order of their location on the hard disk, and they are assigned an identity based on how many images are in each group and what position within each group the file is located (e.g., three images per group; DAPI is always first).

Number of images in each group?

(Used only when Order is selected for file loading)
Enter the number of images that comprise a group. For example, for images given in the order: *DAPI, FITC, Red; DAPI, FITC, Red* and so on, the number of images that in each group would be 3.

Exclude certain files?

(Used only if "Text-Exact match" for loading files is selected)
The image/movie files specified with the *Text* options may also include files that you want to exclude from analysis (such as thumbnails created by an imaging system). Select *Yes* to enter text to match against such files for exclusion.

Type the text that the excluded images have in common

(Used only if file exclusion is selected)
Specify text that marks files for exclusion. **LoadImages** looks for this text as an exact match within the filename and not as a regular expression.

Analyze all subfolders within the selected folder?

This setting determines whether **LoadImages** analyzes just the images in the specified folder or whether it analyzes images in subfolders as well:

- *All:* Analyze all matching image files in subfolders under your specified image folder location.
- *None:* Only analyze files in the specified location.
- *Some:* Select which subfolders to analyze.

Select subfolders to analyze

Use this control to select some subfolders and exclude others from analysis. Press the button to see the folder tree and check or uncheck the checkboxes to enable or disable analysis of the associated folders.

Check image sets for unmatched or duplicate files?

(Used only if metadata is extracted from the image file and not loading by order)
Select *Yes* to examine the filenames for unmatched or duplicate files based on extracted metadata. This is useful for images generated by HCS systems where acquisition may produce a corrupted image and create a duplicate as a correction or may miss an image entirely. See the *Extract metadata from where?* setting for more details on obtaining, extracting, and using metadata tags.

Group images by metadata?

(Used only if metadata is extracted from the image file or if movies are used)
Select *Yes* to process those images that share a particular metadata tag as a group. For example, if you are performing per-plate illumination correction and the plate metadata is part of the image file name, image grouping will enable you to process those images that have the same plate field together (the alternative would be to place the images from each plate in a separate folder). The next setting allows you to select the metadata tags by which to group.Please see the **Groups** module for more details on the proper use of metadata for grouping

Please note that if you are loading a movie file(e.g., TIFs, FLEX, STKs, AVIs, ZVIs), each movie is already treated as a group of images, so there is no need to enable here.

Specify metadata fields to group by

(Used only if grouping images by metadata)
Select the fields by which you want group the image files. You can select multiple tags. For example, if a set of images had metadata for "Run", "Plate", "Well", and "Site", selecting *Run* and *Plate* will create groups containing images that share the same [*Run*,*Plate*] pair of fields.

Text that these images have in common (case-sensitive)

(Used only for the image-loading Text options)
For *Text-Exact match*, type the text string that all the images have in common. For example, if all the images for the given channel end with the text "D.TIF", type `D.TIF` here.

For *Text-Regular expression*, type the regular expression that would capture all the images for this channel. See the module help for more information on regular expressions.

Position of this image in each group

(Used only for the image-loading Order option)
Enter the number in the image order that this image channel occupies. For example, if the order is "DAPI, FITC, Red; DAPI, FITC, Red" and so on, the DAPI channel would occupy position 1.

Load the input as images or objects?

This setting determines whether you load an image as image data or as segmentation results (i.e., objects):

- *Images:* The input image will be given a user-specified name by which it will be refered downstream. This is the most common usage for this module.
- *Objects:* Use this option if the input image is a label matrix and you want to obtain the objects that it defines. A *label matrix* is a grayscale or color image in which the connected regions share the same label, and defines how objects are represented in CellProfiler. The labels are integer values greater than or equal to 0. The elements equal to 0 are the background, whereas the elements equal to 1 make up one object, the elements equal to 2 make up a second object, and so on. This option allows you to use the objects without needing to insert an **Identify** module to extract them first. See **IdentifyPrimaryObjects** for more details.

Name this loaded image

What do you want to call the images you are loading for use downstream in the pipeline? Give your images a meaningful name that you can use to refer to these images in later modules. Keep the following points in mind:

- Image names can consist of any combination of characters (e.g., letters, digits, and other non-alphanumeric characters). However, if you are using **ExportToDatabase**, these names will become part of the measurement column name, and some characters are not permitted in MySQL (e.g., slashes).
- Names are not case sensitive. Therefore, *OrigBlue*, *origblue*, and *ORIGBLUE* will all correspond to the same name, and unexpected results may ensue.
- Although CellProfiler can accept names of any length, you may want to avoid making the name too long, especially if you are uploading to a database. The name is used to generate the column header for a given measurement, and in MySQL the total bytes used for all column headers cannot exceed 64K. A warning will be generated later if this limit has been exceeded.

Name this loaded object

(Used only if objects are output)
This is the name for the objects loaded from your image

Retain outlines of loaded objects?

(Used only if objects are output)
Select *Yes* if you want to create an image of the outlines of the loaded objects.

Name the outline image

(Used only if objects are output and outlines are saved)
Enter a name that will allow the outlines to be selected later in the pipeline.

Special note on saving images: You can use the settings in this module to pass object outlines along to

the module **OverlayOutlines**, and then save them with the **SaveImages** module.

Channel number

(Used only if a movie image format is selected as file type and movie frame grouping is selected)
The channels of a multichannel image are numbered starting from 1. Each channel is a greyscale image, acquired using different illumination sources and/or optics. Use this setting to pick the channel to associate with the above image name.

Rescale intensities?

This option determines whether image metadata should be used to rescale the image's intensities. Some image formats save the maximum possible intensity value along with the pixel data. For instance, a microscope might acquire images using a 12-bit A/D converter which outputs intensity values between zero and 4095, but stores the values in a field that can take values up to 65535.

Select *Yes* to rescale the image intensity so that saturated values are rescaled to 1.0 by dividing all pixels in the image by the maximum possible intensity value.

Select *No* to ignore the image metadata and rescale the image to 0 – 1.0 by dividing by 255 or 65535, depending on the number of bits used to store the image.

Extract metadata from where?

Metadata fields can be specified from the image filename, the image path (including subfolders), or both. The metadata entered here can be used for image grouping (see the *Group images by metadata?* setting) or simply used as additional columns in the exported measurements (see the **ExportToSpreadsheet** module).

Regular expression that finds metadata in the file name

(Used only if you want to extract metadata from the file name)
The regular expression to extract the metadata from the file name is entered here. Note that this field is available whether you have selected *Text-Regular expressions* to load the files or not. Please see the general module help for more information on construction of a regular expression.

Clicking the magnifying glass icon to the right will bring up a tool for checking the accuracy of your regular expression. The regular expression syntax can be used to name different parts of your expression. The syntax *(?P<fieldname>expr)* will extract whatever matches *expr* and assign it to the measurement,*fieldname* for the image.

For instance, a researcher uses plate names composed of a string of letters and numbers, followed by an underscore, then the well, followed by another underscore, followed by an "s" and a digit representing the site taken within the well (e.g., *TE12345_A05_s1.tif*). The following regular expression will capture the plate, well, and site in the fields "Plate", "Well", and "Site":

^(?P<Plate>.*)_(?P<Well>[A-P][0-9]{1,2})_s(?P<Site>[0-9])	
^	Start only at beginning of the file name
(?P<Plate>	Name the captured field *Plate*
.*	Capture as many characters as follow
_	Discard the underbar separating plate from well

(?P<Well>	Name the captured field *Well*
[A-P]	Capture exactly one letter between A and P
[0-9]{1,2}	Capture one or two digits that follow
_s	Discard the underbar followed by *s* separating well from site
(?P<Site>	Name the captured field *Site*
[0-9]	Capture one digit following

The regular expression can be typed in the upper text box, with a sample file name given in the lower text box. Provided the syntax is correct, the corresponding fields will be highlighted in the same color in the two boxes. Press *Submit* to enter the typed regular expression.

You can create metadata tags for any portion of the filename or path, but if you are specifying metadata for multiple images in a single **LoadImages** module, an image cycle can only have one set of values for each metadata tag. This means that you can only specify the metadata tags which have the same value across all images listed in the module. For example, in the example above, you might load two wavelengths of data, one named *TE12345_A05_s1_w1.tif* and the other *TE12345_A05_s1_w2.tif*, where the number following the *w* is the wavelength. In this case, a "Wavelength" tag *should not* be included in the regular expression because while the "Plate", "Well" and "Site" metadata is identical for both images, the wavelength metadata is not.

Note that if you use the special fieldnames <*WellColumn*> and <*WellRow*> together, LoadImages will automatically create a <*Well*> metadata field by joining the two fieldname values together. For example, if <*WellRow*> is "A" and <*WellColumn*> is "01", a field <*Well*> will be "A01". This is useful if your well row and column names are separated from each other in the filename, but you want to retain the standard well nomenclature.

Type the regular expression that finds metadata in the subfolder path

(Used only if you want to extract metadata from the path)
Enter the regular expression for extracting the metadata from the path. Note that this field is available whether you have selected *Text-Regular expressions* to load the files or not.

Clicking the magnifying glass icon to the right will bring up a tool that will allow you to check the accuracy of your regular expression. The regular expression syntax can be used to name different parts of your expression. The syntax *(?<fieldname>expr)* will extract whatever matches *expr* and assign it to the image's *fieldname* measurement.

For instance, a researcher uses folder names with the date and subfolders containing the images with the run ID (e.g., *./2009_10_02/1234/*) The following regular expression will capture the plate, well, and site in the fields *Date* and *Run*:

.*[\\/](?P<Date>.*)[\\/](?P<Run>.*)$	
.*[\\/]	Skip characters at the beginning of the pathname until either a slash (/) or backslash (\) is encountered (depending on the operating system)
(?P<Date>	Name the captured field *Date*
.*	Capture as many characters that follow
[\\/]	Discard the slash/backslash character
(?P<Run>	Name the captured field *Run*

.*	Capture as many characters as follow
$	The *Run* field must be at the end of the path string, i.e., the last folder on the path. This also means that the Date field contains the parent folder of the Date folder.

Group the movie frames?

(Used only if a movie image format is selected as file type)
LoadImages can load several frames from a movie into different images within the same cycle. For example, a movie's first frame might be an image of the red fluorescence channel at time zero, the second might be the green channel at time zero, the third might be the red channel at time one, etc. Select *Yes* to extract both channels for this movie as separate images within the same cycle.

LoadImages refers to the individual images in a group as *channels*. Channels are numbered consecutively, starting at channel 1. To set up grouping, first specify how the channels are grouped (interleaving and number of channels per group), then assign image names to each of the channels individually.

Grouping method

(Used only if a movie image format is selected as file type and movie frame grouping are selected)
Channels in a movie can be interleaved or separated.

In an interleaved movie, the first frame is channel 1, the second is channel 2 and so on up to the number of channels per group for a given image cycle. In a separated movie, all of the frames for channel 1 are processed as the first image cycle, then the frames for channel 2 for the second image cycle, and so on.

For example, a movie may consist of 6 frames and we would like to process the movie as two channels per group. An interleaved movie would be processed like this:

Frame #	Channel #	Image cycle #
1	1	1
2	2	1
3	1	2
4	2	2
5	1	3
6	2	3

For a separated movie, the channels would be processed like this:

Frame #	Channel #	Image cycle #
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3

Note the difference in which frames are processed in which image cycle between the two methods.

Number of channels per group

(Used only if a movie image format is selected as file type and movie frame grouping is selected)
This setting controls the number of frames to be grouped together. As an example, for an interleaved movie with 12 frames and three channels per group, the first, fourth, seventh and tenth frame will be assigned to channel 1, the 2nd, 5,th 8th and 11th frame will be assigned to channel 2 and the 3rd, 6th, 9th, and 12th will be assigned to channel 3. For a separated movie, frames 1 through 4 will be assigned to channel 1, 5 through 8 to channel 2 and 9 through 12 to channel 3.

Input image file location

Select the folder containing the images to be loaded. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

Module: LoadSingleImage

Load Single Image loads a single image for use in all image cycles.

This module tells CellProfiler where to retrieve a single image and gives the image a meaningful name by which the other modules can access it. The module executes only the first time through the pipeline; thereafter the image is accessible to all subsequent processing cycles. This is particularly useful for loading an image like an illumination correction image for use by the **CorrectIlluminationApply** module, when that single image will be used to correct all images in the analysis run.

Disclaimer: Please note that the Input modues (i.e., **Images**, **Metadata**, **NamesAndTypes** and **Groups**) largely supercedes this module. However, old pipelines loaded into CellProfiler that contain this module will provide the option of preserving them; these pipelines will operate exactly as before.

Available measurements

- *Pathname, Filename:* The full path and the filename of each image.
- *Metadata:* The metadata information extracted from the path and/or filename, if requested.
- *Scaling:* The maximum possible intensity value for the image format.
- *Height, Width:* The height and width of the current image.

Technical notes

For most purposes, you will probably want to use the **LoadImages** module, not **LoadSingleImage**. The reason is that **LoadSingleImage** does not actually create image sets (or even a single image set). Instead, it adds the single image to every image cycle for an *already existing* image set. Hence **LoadSingleImage** should never be used as the only image-loading module in a pipeline; attempting to do so will display a warning message in the module settings.

If you have a single file to load in the pipeline (and only that file), you will want to use **LoadImages** or **LoadData** with a single, hardcoded file name.

See also the **Input** modules, **LoadImages**,**LoadData**.

Settings:

Input image file location

Select the folder containing the image(s) to be loaded. Generally, it is best to store the image you want to load in either the Default Input or Output Folder, so that the correct image is loaded into the pipeline and typos are avoided. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

For *Elsewhere...*, *Default Input Folder sub-folder* and *Default Output Folder sub-folder*, if you have metadata associated with your images via **Metadata** module, you can name the folder using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have a "Plate" metadata tag, and your single files are organized in subfolders named with the "Plate" tag, you can select one of the subfolder options and then specify a subfolder name of "\g<Plate>" to get the files from the subfolder associated with that image's plate. The module will substitute the metadata values for the current image set for any metadata tags in the folder name. Please see the **Metadata** module for more details on metadata collection and usage.

Filename of the image to load (Include the extension, e.g., .tif)

The filename can be constructed in one of two ways:

- As a fixed filename (e.g., *Exp1_D03f00d0.tif*).
- Using the metadata associated with an image set in **LoadImages** or **LoadData**. This is especially useful if you want your output given a unique label according to the metadata corresponding to an image group. The name of the metadata to substitute is included in a special tag format embedded in your file specification. You can insert a previously defined metadata tag by either using:
    - The insert key
    - A right mouse button click inside the control
    - In Windows, the Context menu key, which is between the Windows key and Ctrl key

  The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:
    - Use the up and down arrows to cycle through possible values.
    - Right-click on the tag to display and select the available values.

  Please see the **Metadata** module for more details on metadata collection and usage.

Keep in mind that in either case, the image file extension, if any, must be included.

Load as images or objects?

This setting determines whether you load an image as image data or as segmentation results (i.e., objects):

- *Images:* The input image will be given a user-specified name by which it will be refered downstream. This is the most common usage for this module.
- *Objects:* Use this option if the input image is a label matrix and you want to obtain the objects that it defines. A *label matrix* is a grayscale or color image in which the connected regions share the same label, and defines how objects are represented in CellProfiler. The labels are integer values greater than or equal to 0. The elements equal to 0 are the background, whereas the elements equal to 1 make up one object, the elements equal to 2 make up a second object, and so on. This option allows you to use the objects without needing to insert an **Identify** module to extract them first. See **IdentifyPrimaryObjects** for more details.

Name the image that will be loaded

(Used only if an image is output)
Enter the name of the image that will be loaded. You can use this name to select the image in downstream modules.

Rescale intensities?

(Used only if an image is output)
This option determines whether image metadata should be used to rescale the image's intensities. Some image formats save the maximum possible intensity value along with the pixel data. For instance, a microscope might acquire images using a 12-bit A/D converter which outputs intensity values between zero and 4095, but stores the values in a field that can take values up to 65535.

Select *Yes* to rescale the image intensity so that saturated values are rescaled to 1.0 by dividing all pixels in the image by the maximum possible intensity value.

Select *No* to ignore the image metadata and rescale the image to 0 – 1.0 by dividing by 255 or 65535, depending on the number of bits used to store the image.

Name this loaded object

(Used only if objects are output)
This is the name for the objects loaded from your image

Retain outlines of loaded objects?

(Used only if objects are output)
Select *Yes* if you want to save an image of the outlines of the loaded objects.

Name the outlines

(Used only if objects are output)
Enter a name that will allow the outlines to be selected later in the pipeline.

Module: Metadata

The **Metadata** module connects information about the images (i.e., metadata) to your list of images for processing in CellProfiler.

The **Metadata** module allows you to extract and associate metadata with your images. The metadata can be extracted from the image file itself, from a part of the file name or location, and/or from a text file you provide.

What is "metadata"?

The term *metadata* refers to "data about data." For many assays, metadata is important in the context of tagging images with various attributes, which can include (but is not limited to) items such as the following:

- The row and column of the microtiter plate that the image was acquired from.
- The experimental treatment applied to the well that the image was acquired from.
- The number of timepoints or channels contained in the image file.
- The image type, i.e., RGB, indexed or separate channels.
- The height and width of an image, in pixels.
- Etc.

It can be helpful to inform CellProfiler about certain metadata in order to define a specific relationship between the images and the associated metadata. For instance:

- You want images with a common tag to be matched together so they are processed together during the pipeline run. E.g., the filenames for fluorescent DAPI and GFP images contain different tags indicating the wavelength but share '_s1' in the filename if they were acquired from site #1, '_s2' from site #2, and so on.
- You want certain information attached to the output measurements and filenames for annotation or sample-tracking purposes. E.g., some images are to be identified as acquired from DMSO treated wells, whereas others were collected from wells treated with Compound 1, 2,... and so forth.

The underlying assumption in matching metadata values to image sets is that there is an exact pairing (i.e., a one-to-one match) for a given combination of metadata tags. A common example is that for a two-channel microtiter plate assay, the values of the plate, well, and site tags from one channel get matched uniquely to the plate, well, and site tag values from the other channel.

What are the inputs?

If you do not have metadata that is relevant to your analysis, you can leave this module in the default setting, and continue on to the **NamesAndTypes**module If you do have relevant metadata, the **Metadata** module receives the file list produced by the **Images** module. It then associates information to each file in the File list, which can be obtained from several sources:

- From the image file name or location (e.g., as assigned by a microscope). In this case, you will provide the text search pattern to obtain this information.
- In a text file created and filled out by you or a laboratory information management system. In this case, you will point the module to the location of this file.
- In the image file itself.

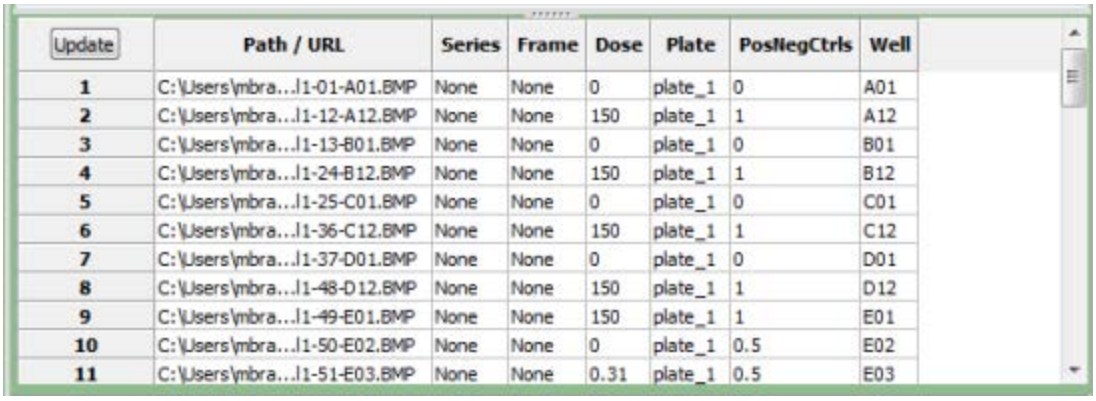
What do the settings mean?

See below for help on the individual settings. In general, the settings serve in various forms of metadata extraction. You can extract metadata from all images from **Images** modules or a subset of them by using rules to filter the list.

What do I get as output?

The final product of the **Metadata** module is a list of files from the **Images**module, accompanied by the associated metadata retrieved from the source(s) provided and matched to the desired images.

As you are extracting metadata from your various sources, you can click the "Update" button below the divider to display a table of results using the current settings. Each row corresponds to an image file from the **Images** module, and the columns display the metadata obtained for each tag specified. You can press this button as many times as needed to display the most current metadata obtained.



Update	Path / URL	Series	Frame	Dose	Plate	PosNegCtrls	Well
1	C:\Users\mbra...\1-01-A01.BMP	None	None	0	plate_1	0	A01
2	C:\Users\mbra...\1-12-A12.BMP	None	None	150	plate_1	1	A12
3	C:\Users\mbra...\1-13-B01.BMP	None	None	0	plate_1	0	B01
4	C:\Users\mbra...\1-24-B12.BMP	None	None	150	plate_1	1	B12
5	C:\Users\mbra...\1-25-C01.BMP	None	None	0	plate_1	0	C01
6	C:\Users\mbra...\1-36-C12.BMP	None	None	150	plate_1	1	C12
7	C:\Users\mbra...\1-37-D01.BMP	None	None	0	plate_1	0	D01
8	C:\Users\mbra...\1-48-D12.BMP	None	None	150	plate_1	1	D12
9	C:\Users\mbra...\1-49-E01.BMP	None	None	150	plate_1	1	E01
10	C:\Users\mbra...\1-50-E02.BMP	None	None	0	plate_1	0.5	E02
11	C:\Users\mbra...\1-51-E03.BMP	None	None	0.31	plate_1	0.5	E03

Some downstream use cases for metadata include the following:

- If the metadata establishes how channels are related to one another, you can use them in the **NamesAndTypes** module to aid in creating an image set.
- If the images need to be further sub-divided into groups of images that share a common metadata value, the **Groups** module can be used to specify which metadata is needed for this purpose.
- You can also use metadata to reference their values in later modules. Since the metadata is stored as an image measurement and can be assigned as an integer or floating-point number, any module which allows measurements as input can make use of it.
- Several modules are also capable of using metadata for more specific purposes. Refer to the module setting help for additional information on how to use them in the context of the specific module.

If the metadata originates from an external source such as a CSV, there are some caveats in the cases when metadata is either missing or duplicated for the referenced images; see the **NamesAndTypes** module for more details.

Available measurements

- *Metadata:* The prefix of each metadata tag in the per-image table.

Settings:

Extract metadata?

Select *Yes* if your file or path names or file headers contain information (i.e., metadata) you would like to

extract and store along with your measurements. See the main module help for more details.

Metadata data type

Metadata can be stored as either a text or numeric value:

- *Text:* Save all metadata item as text.
- *Choose for each:* Choose the data type separately for each metadata entry. An example of when this approach would be necessary would be if a whole filename is captured as metadata but the file name is numeric, e.g., "0001101". In this situation, if the file name needs to be used for an arithmetic calculation or index, the name would need to be converted to a number and you would select "Integer" as the data type. On the other hand, if it important that the leading zeroes be retained, setting it to an integer would them upon conversion to a number. In this case, storing the metadata values as "Text" would be more appropriate.

Metadata types

(Used only when Choose for each is selected for the metadata data type)
This setting determines the data type of each metadata field when stored as a measurement.

- *Text:* Save the metadata as text.
- *Integer:* Save the metadata as an integer.
- *Float:* Save the metadata as a decimal number.
- *None:* Do not save the metadata as a measurement.

Metadata extraction method

Metadata can be stored in either or both of two ways:

- *Internally:* This method is often through the file naming, directory structuring, or the file header information.
- *Externally:* This is through an external index, such as spreadsheet or database of some kind.

The **Metadata** module can extract internal or external metadata from the images in any of three ways:

- *Extract from file/folder names*: This approach retrieves information based on the file nomenclature and/or location. A special syntax called "regular expressions" is used to match text patterns in the file name or path, and then assign this text as metadata for the images you specify. The tag for each metadata is assigned a name that is meaningful to you.

    When would you want to use this option? If you want to take advantage of the fact that acquisition software often automatically assigns a regular nomenclature to the filenames or the containing folders. Alternately, the researcher acquiring the images may also have a specific nomenclature they adhere to for bookkeeping purposes.

- *Import from file*: This option retrieves metadata from a comma-delimited file (known as a CSV file, for comma-separated values) of information; you will be prompted to specify the location of the CSV file. You can create such a file using a spreadsheet program such as Microsoft Excel.

    When would you want to use this option? You have information curated in software that allows for export to a spreadsheet. This is commonly the case for laboratories that use data management systems that track samples and acquisition.

Extract from image file headers: This option retrieves information from the internal structure of the file format itself. Typically, image metadata is embedded in the image file as header information; this information includes the dimensions and color depth among other things. If you select this method, press the "Update metadata" button to extract the metadata. Note that this extraction process can take a while for assays with lots of images since each one needs to read for extraction. Since the metadata is often image-format specific, this option will extract information that is common to most image types:

- *Series:* The series index of the image. This value is set to "None" if not applicable. Some image formats can store more than one stack in a single file; for those, the *Series* value for each stack in the file will be different
- *Frame:* The frame index of the image. This value is set to "None" if not applicable. For stack frames and movies, this is the frame number for an individual 2-D image slice.
- *ColorFormat:* Set to "Monochrome" for grayscale images, "RGB" for color.
- *SizeZ:* The number of image slices. Typically has a value > 1 for confocal stacks and the like.
- *SizeT:* The number of image frames. Typically has a value > 1 for movies.
- *SizeC:* The number of color channels. Typically has a value > 1 for non-grayscale images and for confocal stacks containing channel images acquired using different filters and illumination sources.

  When would you want to use this option? You want to analyze images that are contained as file stacks, i.e., the images that are related to each other in some way, such as by time (temporal), space (spatial), or color (spectral).

Specifics on the metadata extraction options are described below. Any or all of these options may be used at time; press the "Add another extraction method" button to add more.

Metadata source

You can extract the metadata from the image's file name or from its folder name.

Regular expression

(Used only if you want to extract metadata from the file name)
The regular expression to extract the metadata from the file name is entered here. Note that this field is available whether you have selected *Text-Regular expressions* to load the files or not. Please see the general module help for more information on construction of a regular expression.

Clicking the magnifying glass icon to the right will bring up a tool for checking the accuracy of your regular expression. The regular expression syntax can be used to name different parts of your expression. The syntax *(?P<fieldname>expr)* will extract whatever matches *expr* and assign it to the measurement,*fieldname* for the image.

For instance, a researcher uses plate names composed of a string of letters and numbers, followed by an underscore, then the well, followed by another underscore, followed by an "s" and a digit representing the site taken within the well (e.g., *TE12345_A05_s1.tif*). The following regular expression will capture the plate, well, and site in the fields "Plate", "Well", and "Site":

^(?P<Plate>.*)_(?P<Well>[A-P][0-9]{1,2})_s(?P<Site>[0-9])	
^	Start only at beginning of the file name
(?P<Plate>	Name the captured field *Plate*
.*	Capture as many characters as follow

_	Discard the underbar separating plate from well
(?P<Well>	Name the captured field *Well*
[A-P]	Capture exactly one letter between A and P
[0-9]{1,2}	Capture one or two digits that follow
_s	Discard the underbar followed by *s* separating well from site
(?P<Site>	Name the captured field *Site*
[0-9]	Capture one digit following

The regular expression can be typed in the upper text box, with a sample file name given in the lower text box. Provided the syntax is correct, the corresponding fields will be highlighted in the same color in the two boxes. Press *Submit* to enter the typed regular expression.

You can create metadata tags for any portion of the filename or path, but if you are specifying metadata for multiple images, an image cycle can only have one set of values for each metadata tag. This means that you can only specify the metadata tags which have the same value across all images listed in the module. For example, in the example above, you might load two wavelengths of data, one named *TE12345_A05_s1_w1.tif* and the other *TE12345_A05_s1_w2.tif*, where the number following the *w* is the wavelength. In this case, a "Wavelength" tag *should not* be included in the regular expression because while the "Plate", "Well" and "Site" metadata is identical for both images, the wavelength metadata is not.

Note that if you use the special fieldnames <*WellColumn*> and <*WellRow*> together, LoadImages will automatically create a <*Well*> metadata field by joining the two fieldname values together. For example, if <*WellRow*> is "A" and <*WellColumn*> is "01", a field <*Well*> will be "A01". This is useful if your well row and column names are separated from each other in the filename, but you want to retain the standard well nomenclature.

Regular expression

(Used only if you want to extract metadata from the path)
Enter the regular expression for extracting the metadata from the path. Note that this field is available whether you have selected *Text-Regular expressions* to load the files or not.

Clicking the magnifying glass icon to the right will bring up a tool that will allow you to check the accuracy of your regular expression. The regular expression syntax can be used to name different parts of your expression. The syntax *(?<fieldname>expr)* will extract whatever matches *expr* and assign it to the image's *fieldname* measurement.

For instance, a researcher uses folder names with the date and subfolders containing the images with the run ID (e.g., *./2009_10_02/1234/*) The following regular expression will capture the plate, well, and site in the fields *Date* and *Run*:

.*[\\/](?P<Date>.*)[\\/](?P<Run>.*)$	
.*[\\/]	Skip characters at the beginning of the pathname until either a slash (/) or backslash (\) is encountered (depending on the operating system)
(?P<Date>	Name the captured field *Date*
.*	Capture as many characters that follow
[\\/]	Discard the slash/backslash character
(?P<Run>	Name the captured field *Run*

.*	Capture as many characters as follow
$	The *Run* field must be at the end of the path string, i.e., the last folder on the path. This also means that the Date field contains the parent folder of the Date folder.

Extract metadata from

Select whether you want to extract metadata from all of the images chosen by the **Images** module or a subset of the images.

This setting controls how different image types (e.g., an image of the GFP stain and a brightfield image) have different metadata extracted. There are two choices:

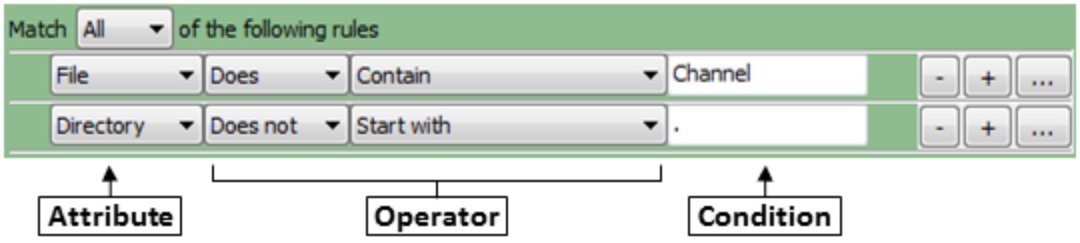
- *All images*: Extract metadata from all images specified in **Images**. This is the simplest choice and the appropriate one if you have only one kind of image (or only one image). CellProfiler will extract metadata from all images using the same method per iteration.
- *Images matching a rule*: Extract metadata depending on specific file attributes. This is the appropriate choice if more than one image was taken of each imaging site. You can specify distinctive criteria for each image subset with matching metadata.

Select the filtering criteria

Select *Yes* to display and use rules to select files for metadata extraction.

Clicking the rule menus shows you all the file *attributes*, *operators* and *conditions* you can specify to narrow down the image list.

1. For each rule, first select the *attribute* that the rule is to be based on. For example, you can select "File" to define a rule that will filter files on the basis of their filename.
2. The *operator* drop-down is then updated with operators applicable to the attribute you selected. For example, if you select "File" as the attribute, the operator menu includes text operators such as *Contain* or *Starts with*. On the other hand, if you select "Extension" as the attribute, you can choose the logical operators "Is" or "Is not" from the menu.
3. In the operator drop-down menu, select the operator you want to use. For example, if you want to match data exactly, you may want the "Exactly match" or the "Is" operator. If you want the condition to be more loose, select an operator such as "Contains".
4. Use the *condition* box to type the condition you want to match. The more you type, the more specific the condition is.
    - As an example, if you create a new filter and select *File* as the attribute, then select "Does" and "Contain" as the operators, and type "Channel" as the condition, the filter finds all files that include the text "Channel", such as "Channel1.tif" "Channel2.jpg", "1-Channel-A01.BMP" and so on.
    - If you select "Does" and "Start with" as the operators and "Channel1" in the Condition box, the rule will includes such files as "Channel1.tif" "Channel1-A01.png", and so on.



You can also create regular expressions (an advanced syntax for pattern matching; see below) in order to select particular files.

To add another rule, click the plus buttons to the right of each rule. Subtract an existing rule by clicking the minus button.

You can also link a set of rules by choosing the logical expression *All* or *Any*. If you use *All* logical expression, all the rules be true for a file to be included in the File list. If you use the *Any* option, only one of the conditions has to be met for a file to be included.

If you want to create more complex rules (e.g, some criteria matching all rules and others matching any), you can create sets of rules, by clicking the ellipsis button (to the right of the plus button). Repeat the above steps to add more rules to the filter until you have all the conditions you want to include.

Details on regular expressions

A *regular expression* is a general term refering to a method of searching for pattern matches in text. There is a high learning curve to using them, but are quite powerful once you understand the basics.

Patterns are specified using combinations of metacharacters and literal characters. There are a few classes of metacharacters, partially listed below. Some helpful links follow:

- A more extensive explanation of regular expressions can be found here
- A helpful quick reference can be found here
- Pythex provides quick way to test your regular expressions. Here is an example to capture information from a common microscope nomenclature.

The following metacharacters match exactly one character from its respective set of characters:

Metacharacter	Meaning
.	Any character
[]	Any character contained within the brackets
[^]	Any character not contained within the brackets
\w	A word character [a-z_A-Z0-9]
\W	Not a word character [^a-z_A-Z0-9]
\d	A digit [0-9]
\D	Not a digit [^0-9]
\s	Whitespace [\t\r\n\f\v]
\S	Not whitespace [^ \t\r\n\f\v]

The following metacharacters are used to logically group subexpressions or to specify context for a position in the match. These metacharacters do not match any characters in the string:

Metacharacter	Meaning
( )	Group subexpression
|	Match subexpression before or after the |
^	Match expression at the start of string
$	Match expression at the end of string
\<	Match expression at the start of a word
\>	Match expression at the end of a word

The following metacharacters specify the number of times the previous metacharacter or grouped subexpression may be matched:

Metacharacter	Meaning
*	Match zero or more occurrences
+	Match one or more occurrences
?	Match zero or one occurrence
{n,m}	Match between n and m occurrences

Characters that are not special metacharacters are all treated literally in a match. To match a character that is a special metacharacter, escape that character with a '\'. For example '.' matches any character, so to match a '.' specifically, use '\.' in your pattern. Examples:

- `[trm]ail` matches 'tail' or 'rail' or 'mail'.
- `[0-9]` matches any digit between 0 to 9.
- `[^Q-S]` matches any character other than 'Q' or 'R' or 'S'.
- `[[]A-Z]` matches any upper case alphabet along with square brackets.
- `[ag-i-9]` matches characters 'a' or 'g' or 'h' or 'i' or '-' or '9'.
- `[a-p]*` matches '' or 'a' or 'aab' or 'p' etc.
- `[a-p]+` matches 'a' or 'abc' or 'p' etc.
- `[^0-9]` matches any string that is not a number.
- `^[0-9]*$` matches either a blank string or a natural number.
- `^-[0-9]+$|^\+?[0-9]+$` matches any integer.

Metadata file location

The file containing the metadata must be a comma-delimited file (CSV). You can create or edit such a file using a spreadsheet program such as Microsoft Excel.

The CSV file needs to conform to the following format:

- Each column describes one type of metadata.
- Each row describes the metadata for one image site.
- The column headers are uniquely named. You can optionally prepend "Metadata_" to the header name in order to insure that it is interpreted correctly.
- The CSV must be plain text, i.e., without hidden file encoding information. If using Excel on a Mac to edit the file, choose to save the file as "Windows CSV" or "Windows Comma Separated".

The file must be saved as plain text, i.e., without hidden file encoding information. If using Excel on a Mac to edit the file, choose to save the file as "Windows CSV" or "Windows Comma Separated".

Match file and image metadata

Match columns in your .csv file to image metadata items. If you are using a CSV in conjunction with the filename/path metadata matching, you might want to capture the metadata in common with both sources. For example, you might be extracting the well tag from the image filename while your CSV contains treatment dosage information paired with each well. Therefore, you would want to let CellProfiler know that the well tag extracted from the image filename and the well tag noted in the CSV are in fact the one and the same.

This setting controls how rows in your CSV file are matched to different images. Set the drop-downs to

pair the metadata tags of the images and the CSV, such that each row contains the corresponding tags. This can be done for as many metadata correspondences as you may have for each source; press + to add more rows.

Use case insensitive matching?

This setting controls whether row matching takes the metadata case into account when matching. If you note that your CSV metadata is not being applied, your choice on this setting may be the culprit.

Select *No* so that metadata entries that only differ by case (for instance, "A01" and "a01") will not match.

Select *Yes* to match metadata entries that only differ by case.

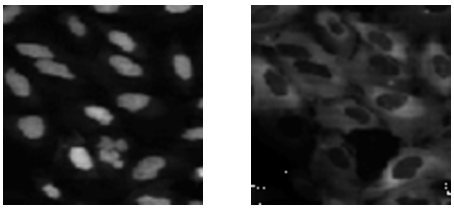
Module: NamesAndTypes

The **NamesAndTypes** module gives images and/or channels a meaningful name to a particular image or channel, as well as defining the relationships between images to create an image set.

Once the relevant images have been identified using the **Images** module (and/or has had metadata associated with the images using the **Metadata** module), the **NamesAndTypes** module gives each image a meaningful name by which modules in the analysis pipeline will refer to it.

What is an "image set"?

An *image set* is the collection of channels that represent a single field of view. For example, a fluorescent assay may have samples using DAPI and GFP to label separate cellular sub-compartments (see figure below), and for each site imaged, one DAPI (left) and one GFP image (right) is acquired by the microscope. Sometimes, the two channels are combined into a single color images and other times they are stored as two separate grayscale images, as in the figure.



For the purposes of analysis, you want the DAPI and GFP image for a given site to be loaded and processed together. Therefore, the DAPI and GFP image for a given site comprise an image set for that site.

What do I need as input?

The **NamesAndTypes** module receives the file list produced by the **Images** module. If you used the **Metadata** module to attach metadata to the images, this information is also received by **NamesAndTypes** and available for its use.

What do the settings mean?

In the above example, the **NamesAndTypes** module allows you to assign each of these channels a unique name, provided by you. All files of a given channel will be referred to by the chosen name within the pipeline, and the data exported by the pipeline will also be labeled according to this name. This simplifies the bookkeeping of your pipeline and results by making the input and output data more intuitive: a large number of images are referred to by a small collection of names, which are hopefully easier for you to recognize.

The most common way to perform this assignment is by specifying the pattern in the filename which the channel(s) of interest have in common. This is done using user-defined rules in a similar manner to that of the **Images** module; other attributes of the file may also be used. If you have multiple channels, you then assign the relationship between channels. For example, in the case mentioned above, the DAPI and GFP images are named in such a way that it is apparent to the researcher which is which, e.g., "_w1" is contained in the file for the DAPI images, and "_w1" in the file name for the GFP images.

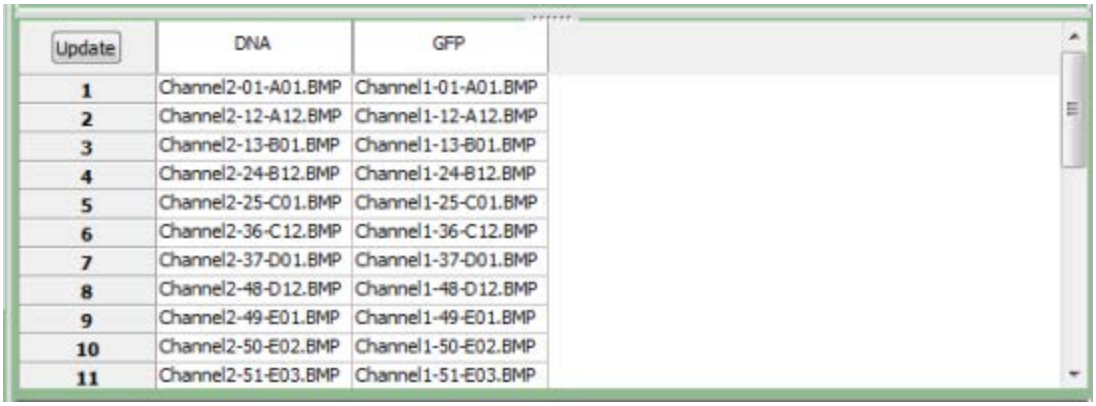
You can also use **NamesAndTypes** to define the relationships between images. For example, if you have acquired multiple wavelengths for your assay, you will need to match the channels to each other for each

field of view so that they are loaded and processed together. This can be done by using their associated metadata. If you would like to use the metadata-specific settings, please see the **Metadata** module or *Help > General help > Using Metadata in CellProfiler* for more details on metadata usage and syntax.

What do I get as output?

The **NamesAndTypes** module is the last of the required input modules. After this module, you can choose any of the names you defined from a drop-down list in any downstream analysis module which requires an image as input. If you defined a set of objects using this module, those names are also available for analysis modules that require an object as input.

In order to see whether the images are matched up correctly to form the image sets you would expect, press the "Update" button below the divider to display a table of results using the current settings. Each row corresponds to a unique image set, and the columns correspond to the name you specified for CellProfiler to identify the channel. You can press this button as many times as needed to display the most current image sets obtained. When you complete your pipeline and perform an analysis run, CellProfiler will process the image sets in the order shown.



Update	DNA	GFP
1	Channel2-01-A01.BMP	Channel1-01-A01.BMP
2	Channel2-12-A12.BMP	Channel1-12-A12.BMP
3	Channel2-13-B01.BMP	Channel1-13-B01.BMP
4	Channel2-24-B12.BMP	Channel1-24-B12.BMP
5	Channel2-25-C01.BMP	Channel1-25-C01.BMP
6	Channel2-36-C12.BMP	Channel1-36-C12.BMP
7	Channel2-37-D01.BMP	Channel1-37-D01.BMP
8	Channel2-48-D12.BMP	Channel1-48-D12.BMP
9	Channel2-49-E01.BMP	Channel1-49-E01.BMP
10	Channel2-50-E02.BMP	Channel1-50-E02.BMP
11	Channel2-51-E03.BMP	Channel1-51-E03.BMP

Available measurements

- *FileName, PathName:* The prefixes of the filename and location, respectively, of each image set written to the per-image table.
- *ObjectFileName, ObjectPathName:* (For used for images loaded as objects) The prefixes of the filename and location, respectively, of each object set written to the per-image table.

Settings:

Assign a name to

This setting allows the user to specify a name to images or subsets of images so they can be treated separately by downstream modules. For example, giving a different name to a GFP stain image and a brightfield image of the same site allows each to be processed independently.

There are three choices:

- *All images*: Give every image the same name. This is the simplest choice and the appropriate one if you have only one kind of image (or only one image). CellProfiler will give each image the same name and the pipeline will load only one of the images per iteration.
- *Images matching rules*: Give images one of several names depending on the file name, directory and metadata. This is the appropriate choice if more than one image was acquired from each

imaging site. You will be asked for distinctive criteria for each image and will be able to assign each category of image a name that can be referred to in downstream modules.

Image set matching method

Select how you want to match the image from one channel with the images from other channels.

This setting controls how CellProfiler picks which images should be matched together when analyzing all of the images from one site.

You can match corresponding channels to each other in one of two ways:

- *Order*: CellProfiler will order the images in each channel alphabetically by their file path name and, for movies or TIF stacks, will order the frames by their order in the file. CellProfiler will then match the first from one channel to the first from another channel.
  This approach is sufficient for most applications, but will match the wrong images if any of the files are missing or misnamed. The image set list will then get truncated according to the channel with the fewer number of files.
- *Metadata*: CellProfiler will match files with the same metadata values. This option is more complex to use than *Order* but is more flexible and less prone to inadvertent errors. Please see the **Metadata** module for more details on metadata collection and usage.

  As an example, an experiment is run on a single multiwell plate with two image channels (OrigBlue, *w1* and OrigGreen, *w2*) containing well and site metadata extracted using the **Metadata** module. A set of images from two sites in well A01 might be described using the following:

File name	Well	Site	Wavelength
P-12345_A01_s1_w1.tif	A01	s1	w1
P-12345_A01_s1_w2.tif	A01	s1	w2
P-12345_A01_s2_w1.tif	A01	s2	w1
P-12345_A01_s2_w2.tif	A01	s2	w2

  We want to match the channels so that each field of view in uniquely represented by the two channels. In this case, to match the *w1* and *w2* channels with their respective well and site metadata, you would select the *Well* metadata for both channels, followed by the *Site* metadata for both channels. In other words:

OrigBlue	OrigGreen
Well	Well
Site	Site

  In this way, CellProfiler will match up files that have the same well and site metadata combination, so that the *w1* channel belonging to well A01 and site 1 will be paired with the *w2* channel belonging to well A01 and site 1. This will occur for all unique well and site pairings, to create an image set similar to the following:

Image set tags		Channels	
Well	**Site**	**OrigBlue (w1)**	**OrigGreen (w2)**
A01	s1	P-12345_A01_s1_w1.tif	P-12345_A01_s1_w2.tif
A01	s2	P-12345_A01_s2_w1.tif	P-12345_A01_s2_w2.tif

  Image sets for which a given metadata value combination (e.g., well, site) is either missing or duplicated for a given channel will simply be omitted.

In addition, CellProfiler can match a single file for one channel against many files from another channel. This is useful, for instance, for applying an illumination correction file for an entire plate against every image file for that plate. In this instance, this would be done by selecting *Plate* as the common metadata tag and *(None)* for the rest:

OrigBlue	**IllumBlue**
Plate	Plate
Well	(None)
Site	(None)

There are two special cases in metadata handling worth mentioning:

- *Missing metadata:* For a particular metadata tag, one image from a given image set has metadata values defined but another image does not. An example is when a microscope aborts acquisition prematurely in the middle of scanning two channels for a site, and captures one channel but not the other. In this case, plate, well and site metadata value exists for one image but not for the other since it was never acquired.
- *Duplicate metadata:* For a particular metadata tag, the same metadata values exist for multiple image sets such that they are not uniquely defined. An example is when a microscope re-scans a site in order to recover from a prior error. In this case, there may be one image from one channel but *two* images for the other channel, for the same site. Therefore, multiple instances of the same plate, well and site metadata values exist for the same image set.

In both of these cases, the exact pairing between channels no longer exists. For missing metadata, the pairing is one-to-none, and for duplicate metadata, the pairing is one-to-two. In these instances where a match cannot be made, **NamesAndTypes** will simply omit the confounding metadata values from consideration. In the above example, an image set will not be created for the plate, well and site combination in question.

Set intensity range from

This option determines how the image intensity should be rescaled from 0.0 – 1.0.

- *Image metadata:* Rescale the image intensity so that saturated values are rescaled to 1.0 by dividing all pixels in the image by the maximum possible intensity value allowed by the imaging hardware. Some image formats save the maximum possible intensity value along with the pixel data. For instance, a microscope might acquire images using a 12-bit A/D converter which outputs intensity values between zero and 4095, but stores the values in a field that can take values up to 65535. Choosing this setting ensures that the intensity scaling value is the maximum allowed by the hardware, and not the maximum allowable by the file format.
- *Image bit-depth:* Ignore the image metadata and rescale the image to 0 – 1 by dividing by 255 or 65535, depending on the number of bits used to store the image.

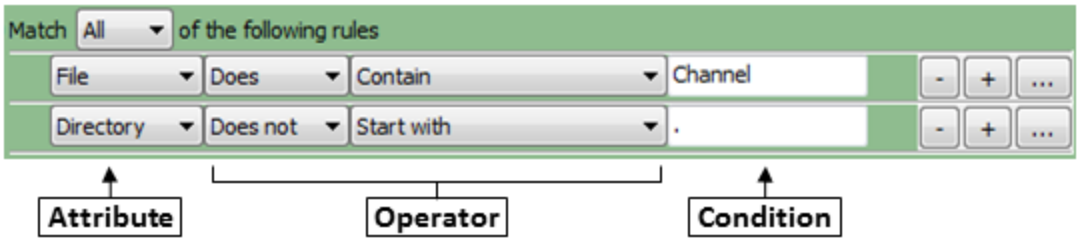
Please note that CellProfiler does not provide the option of loading the image as the raw, unscaled values. If you wish to make measurements on the unscaled image, use the **ImageMath** module to multiply the scaled image by the actual image bit-depth.

Select the rule criteria

Specify a filter using rules to narrow down the files to be analyzed.

Clicking the rule menus shows you all the file *attributes*, *operators* and *conditions* you can specify to narrow down the image list.

1. For each rule, first select the *attribute* that the rule is to be based on. For example, you can select "File" to define a rule that will filter files on the basis of their filename.
2. The *operator* drop-down is then updated with operators applicable to the attribute you selected. For example, if you select "File" as the attribute, the operator menu includes text operators such as *Contain* or *Starts with*. On the other hand, if you select "Extension" as the attribute, you can choose the logical operators "Is" or "Is not" from the menu.
3. In the operator drop-down menu, select the operator you want to use. For example, if you want to match data exactly, you may want the "Exactly match" or the "Is" operator. If you want the condition to be more loose, select an operator such as "Contains".
4. Use the *condition* box to type the condition you want to match. The more you type, the more specific the condition is.
    - As an example, if you create a new filter and select *File* as the attribute, then select "Does" and "Contain" as the operators, and type "Channel" as the condition, the filter finds all files that include the text "Channel", such as "Channel1.tif" "Channel2.jpg", "1-Channel-A01.BMP" and so on.
    - If you select "Does" and "Start with" as the operators and "Channel1" in the Condition box, the rule will includes such files as "Channel1.tif" "Channel1-A01.png", and so on.



    You can also create regular expressions (an advanced syntax for pattern matching; see below) in order to select particular files.

To add another rule, click the plus buttons to the right of each rule. Subtract an existing rule by clicking the minus button.

You can also link a set of rules by choosing the logical expression *All* or *Any*. If you use *All* logical expression, all the rules be true for a file to be included in the File list. If you use the *Any* option, only one of the conditions has to be met for a file to be included.

If you want to create more complex rules (e.g, some criteria matching all rules and others matching any), you can create sets of rules, by clicking the ellipsis button (to the right of the plus button). Repeat the above steps to add more rules to the filter until you have all the conditions you want to include.

Details on regular expressions

A *regular expression* is a general term refering to a method of searching for pattern matches in text. There is a high learning curve to using them, but are quite powerful once you understand the basics.

Patterns are specified using combinations of metacharacters and literal characters. There are a few classes of metacharacters, partially listed below. Some helpful links follow:

- A more extensive explanation of regular expressions can be found here
- A helpful quick reference can be found here
- Pythex provides quick way to test your regular expressions. Here is an example to capture information from a common microscope nomenclature.

The following metacharacters match exactly one character from its respective set of characters:

Metacharacter	Meaning
.	Any character
[]	Any character contained within the brackets
[^]	Any character not contained within the brackets
\w	A word character [a-z_A-Z0-9]
\W	Not a word character [^a-z_A-Z0-9]
\d	A digit [0-9]
\D	Not a digit [^0-9]
\s	Whitespace [\t\r\n\f\v]
\S	Not whitespace [^ \t\r\n\f\v]

The following metacharacters are used to logically group subexpressions or to specify context for a position in the match. These metacharacters do not match any characters in the string:

Metacharacter	Meaning
( )	Group subexpression
|	Match subexpression before or after the |
^	Match expression at the start of string
$	Match expression at the end of string
\<	Match expression at the start of a word
\>	Match expression at the end of a word

The following metacharacters specify the number of times the previous metacharacter or grouped subexpression may be matched:

Metacharacter	Meaning
*	Match zero or more occurrences
+	Match one or more occurrences
?	Match zero or one occurrence
{n,m}	Match between n and m occurrences

Characters that are not special metacharacters are all treated literally in a match. To match a character that is a special metacharacter, escape that character with a '\'. For example '.' matches any character, so to match a '.' specifically, use '\.' in your pattern. Examples:

- `[trm]ail` matches 'tail' or 'rail' or 'mail'.
- `[0-9]` matches any digit between 0 to 9.
- `[^Q-S]` matches any character other than 'Q' or 'R' or 'S'.
- `[[]A-Z]` matches any upper case alphabet along with square brackets.
- `[ag-i-9]` matches characters 'a' or 'g' or 'h' or 'i' or '-' or '9'.
- `[a-p]*` matches '' or 'a' or 'aab' or 'p' etc.
- `[a-p]+` matches 'a' or 'abc' or 'p' etc.
- `[^0-9]` matches any string that is not a number.
- `^[0-9]*$` matches either a blank string or a natural number.
- `^-[0-9]+$|^\+?[0-9]+$` matches any integer.

Name to assign these images

Enter the name that you want to call this image. After this point, this image will be referred to by this name, and can be selected from any drop-down menu that requests an image selection.

Name to assign these objects

Enter the name that you want to call this set of objects. After this point, this object will be referred to by this name, and can be selected from any drop-down menu that requests an object selection.

Select the image type

You can specify how these images should be treated:

- *Grayscale image:* An image in which each pixel represents a single intensity value. Most of the modules in CellProfiler operate on images of this type.
  If this option is applied to a color image, the red, green and blue pixel intensities will be averaged to produce a single intensity value.
- *Color image:* An image in which each pixel repesents a red, green and blue (RGB) triplet of intensity values. Please note that the object detection modules such as **IdentifyPrimaryObjects** expect a grayscale image, so if you want to identify objects, you should use the **ColorToGray** module in the analysis pipeline to split the color image into its component channels.
  You can use the *Grayscale image* option to collapse the color channels to a single grayscale value if you don't need CellProfiler to treat the image as color.
- *Binary mask:* A *mask* is an image where some of the pixel intensity values are zero, and others are non-zero. The most common use for a mask is to exclude particular image regions from consideration. By applying a mask to another image, the portion of the image that overlaps with the non-zero regions of the mask are included. Those that overlap with the zeroed region are "hidden" and not included in downstream calculations. For this option, the input image should be a binary image, i.e, foreground is white, background is black. The module will convert any nonzero values to 1, if needed. You can use this option to load a foreground/background segmentation produced by one of the **Identify** modules.
- *Illumination function:* An *illumination correction function* is an image which has been generated for the purpose of correcting uneven illumination/lighting/shading or to reduce uneven background in images. Typically, is a file in the MATLAB .mat format. See **CorrectIlluminationCalculate** and **CorrectIlluminationApply** for more details.
- *Objects:* Use this option if the input image is a label matrix and you want to obtain the objects that it defines. A label matrix is a grayscale or color image in which the connected regions share the same label, which defines how objects are represented in CellProfiler. The labels are integer values greater than or equal to 0. The elements equal to 0 are the background, whereas the elements equal to 1 make up one object, the elements equal to 2 make up a second object, and so on. This option allows you to use the objects immediately without needing to insert an **Identify** module to extract them first. See **IdentifyPrimaryObjects** for more details.
  This option can load objects created by the **SaveImages** module. These objects can take two forms, with different considerations for each:
  - *Non-overalapping* objects are stored as a label matrix. This matrix should be saved as grayscale, rather than color.
  - *Overlapping objects* are stored in a multi-frame TIF, each frame of whichc consists of a grayscale label matrix. The frames are constructed so that objects that overlap are placed in different frames.

Retain outlines of loaded objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: RenameOrRenumberFiles

Rename or Renumber Files renames or renumbers files on the hard drive.

This file-renaming utility adjusts text within image file names. ***Be very careful with this module because its purpose is to rename (and overwrite) files!*** You will have the opportunity to confirm the name change for the first cycle only. If the folder containing the files contains subfolders, the subfolders and their contents will also be renamed. The module will not rename the file in test mode, so you should use test mode to ensure that the settings are correct.

You can use this module to standardize the number of characters in your file names and to remove unwanted characters from your file names. This is especially useful if you want file names that have numbers in them to appear in numerical order when processed by **NamesAndTypes**.

While this module performs basic renaming operations, if you can extract metadata from your images using the **Metadata** module, you may find using metadata substitution in **SaveImages** to be more flexible. Please refer to metadata handling for those modules and in Help for more details.

Examples

Renumbering can be useful when numbers within image filenames do not have a minimum number of digits and thus appear out of order when listed in some Unix/Mac OSX systems. For example, on some systems, files would appear like this and thus be measured differently from the expected sequence by CellProfiler:

1DrosophilaDAPI_1.tif
1DrosophilaDAPI_10.tif
1DrosophilaDAPI_2.tif
1DrosophilaDAPI_3.tif
1DrosophilaDAPI_4.tif

To renumber the files in the expected order, the numeric digits need to be padded with zeros to the same length. In this case, you would want to:

- Retain 16 characters at the beginning ("1DrosophilaDAPI_").
- Retain 4 characters at the end (".tif").
- "Renumber" as the handling method using 3 numerical digits.
- Set the text addition box to "No".

Original name	*New name*
1DrosophilaDAPI_1.tif	1DrosophilaDAPI_001.tif
1DrosophilaDAPI_10.tif	1DrosophilaDAPI_010.tif
1DrosophilaDAPI_100.tif	1DrosophilaDAPI_100.tif

Renaming can be useful when file names are too long or have characters that interfere with other software or file systems. To accomplish the following, you would want to:

- Retain 5 characters at the beginning ("1Dros")
- Retain 8 characters at the end ("_<3-digit number>.tif", assuming they have already been renumbered)
- Select "Delete" as the handling method

- Check the text addition box and enter "DP" as text to place between the retained start and ending strings

Original name	*New name*
1DrosophilaDAPI_001.tif	1DrosDP_001.tif
1DrosophilaDAPI_010.tif	1DrosDP_010.tif
1DrosophilaDAPI_100.tif	1DrosDP_100.tif

See also: **NamesAndTypes**, **SaveImages**

Settings:

Select the input image

Select the images associated with the files you want to rename. This should be an image loaded by the **Input** modules. Be very careful because you will be renaming these files!

Number of characters to retain at start of file name

Number of characters at the start of the old file name that will be copied over verbatim to the new file name. For instance, if this setting is "6" and the file name is "Image-734.tif", the output file name will also start with "Image-".

Number of characters to retain at the end of file name

Number of characters at the end of the old file name that will be copied over verbatim to the new file name. For instance, if this setting is "4" and the file name is "Image-734.tif", the output file name will also end with ".tif".

Handling of remaining characters

You can either treat the characters between the start and end as numbers or you can delete them. If you treat them as numbers, you will be given the opportunity to pad the numbers with zeros so that all of your file names will have a uniform length. For instance, if you were to renumber the highlighted portion of the file "Image-734.tif" using four digits, the result would be "Image-0734.tif".

Number of digits for numbers

(Used only if Renumber is selected)
Use this setting to pad numbers with zeros so that they all have a uniform number of characters. For instance, padding with four digits has the following result:

Original	**Padded**
1	0001
10	0010
100	0100
1000	1000

Add text to the file name?

Select *Yes* if you want to add text to the file name. If you had chosen *Renumber* above, the module will

add the text after your number. If you had chosen *Delete*, the module will replace the deleted text with the text you enter here.

Replacement text

(Used only if you chose to add text to the file name)
Enter the text that you want to add to each file name.

Replace spaces?

Select *Yes* to replace spaces in the final version of the file name with some other text.

Select *No* if the file name can have spaces or if none of the file names have spaces.

Space replacement

This is the text that will be substituted for spaces in your file name.

Module: SaveImages

Save Images saves image or movie files.

Because CellProfiler usually performs many image analysis steps on many groups of images, it does *not* save any of the resulting images to the hard drive unless you specifically choose to do so with the **SaveImages** module. You can save any of the processed images created by CellProfiler during the analysis using this module.

You can choose from many different image formats for saving your files. This allows you to use the module as a file format converter, by loading files in their original format and then saving them in an alternate format.

Note that saving images in 12-bit format is not supported, and 16-bit format is supported for TIFF only.

See also **NamesAndTypes**, **ConserveMemory**.

Settings:

Select the type of image to save

The following types of images can be saved as a file on the hard drive:

- *Image:* Any of the images produced upstream of **SaveImages** can be selected for saving. Outlines created by **Identify** modules can also be saved with this option, but you must select "Retain outlines..." of identified objects within the **Identify** module. You might also want to use the **OverlayOutlines** module prior to saving images.
- *Mask:* Relevant only if the **Crop** module is used. The **Crop** module creates a mask of the pixels of interest in the image. Saving the mask will produce a binary image in which the pixels of interest are set to 1; all other pixels are set to 0.
- *Cropping:* Relevant only if the **Crop** module is used. The **Crop** module also creates a cropping image which is typically the same size as the original image. However, since the **Crop** permits removal of the rows and columns that are left blank, the cropping can be of a different size than the mask.
- *Movie:* A sequence of images can be saved as a movie file. Currently only AVIs can be written. Each image becomes a frame of the movie.
- *Objects:* Objects can be saved as an image. The image is saved as grayscale unless you select a color map other than gray. Background pixels appear as black and each object is assigned an intensity level corresponding to its object number. The resulting image can be loaded as objects by the **NamesAndTypes** module. Objects are best saved as TIF files. **SaveImages** will use an 8-bit TIF file if there are fewer than 256 objects and will use a 16-bit TIF otherwise. Results may be unpredictable if you save using PNG and there are more than 255 objects or if you save using one of the other file formats.

Select the image to save

(Used only if "Image", "Mask" or "Cropping" are selected to save)
Select the image you want to save.

Select the objects to save

(Used only if saving "Objects")
Select the objects that you want to save.

Select the module display window to save

(Used only if saving "Module window")
Enter the module number/name for which you want to save the module display window.

Select method for constructing file names

(Used only if saving non-movie files)
Several choices are available for constructing the image file name:

- *From image filename:* The filename will be constructed based on the original filename of an input image specified in **NamesAndTypes**. You will have the opportunity to prefix or append additional text.

  If you have metadata associated with your images, you can append an text to the image filename using a metadata tag. This is especially useful if you want your output given a unique label according to the metadata corresponding to an image group. The name of the metadata to substitute can be provided for each image for each cycle using the **Metadata** module. You can insert a previously defined metadata tag by either using:

  - The insert key
  - A right mouse button click inside the control
  - In Windows, the Context menu key, which is between the Windows key and Ctrl key

  The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:
  - Use the up and down arrows to cycle through possible values.
  - Right-click on the tag to display and select the available values.

  Please see the **Metadata** module for more details on metadata collection and usage.

- *Sequential numbers:* Same as above, but in addition, each filename will have a number appended to the end that corresponds to the image cycle number (starting at 1).
- *Single name:* A single name will be given to the file. Since the filename is fixed, this file will be overwritten with each cycle. In this case, you would probably want to save the image on the last cycle (see the *Select how often to save* setting). The exception to this is to use a metadata tag to provide a unique label, as mentioned in the *From image filename* option.

Select image name for file prefix

(Used only when "From image filename" is selected for contructing the filename)
Select an image loaded using **NamesAndTypes**. The original filename will be used as the prefix for the output filename.

Enter single file name

(Used only when "Sequential numbers" or "Single name" are selected for contructing the filename)
Specify the filename text here. If you have metadata associated with your images, enter the filename text with the metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control

- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

Do not enter the file extension in this setting; it will be appended automatically.

Number of digits

(Used only when "Sequential numbers" is selected for contructing the filename)
Specify the number of digits to be used for the sequential numbering. Zeros will be used to left-pad the digits. If the number specified here is less than that needed to contain the number of image sets, the latter will override the value entered.

Append a suffix to the image file name?

Select *Yes* to add a suffix to the image's file name. Select *No* to use the image name as-is.

Text to append to the image name

(Used only when constructing the filename from the image filename)
Enter the text that should be appended to the filename specified above.

Saved file format

(Used only when saving non-movie files)
Select the image or movie format to save the image(s). Most common image formats are available; MAT-files are readable by MATLAB.

Output file location

(Used only when saving non-movie files)
This setting lets you choose the folder for the output files. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder*

sub-folder, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

An additional option is the following:

- *Same folder as image*: Place the output file in the same folder that the source image is located.

For *Elsewhere...*, *Default Input Folder sub-folder* and *Default Output Folder sub-folder*, if you have metadata associated with your images via **Metadata** module, you can name the folder using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

. For instance, if you have a metadata tag named "Plate", you can create a per-plate folder by selecting one the subfolder options and then specifying the subfolder name as "\g<Plate>". The module will substitute the metadata values for the current image set for any metadata tags in the folder name.Please see the **Metadata** module for more details on metadata collection and usage.

If the subfolder does not exist when the pipeline is run, CellProfiler will create it.

If you are creating nested subfolders using the sub-folder options, you can specify the additional folders separated with slashes. For example, "Outlines/Plate1" will create a "Plate1" folder in the "Outlines" folder, which in turn is under the Default Input/Output Folder. The use of a forward slash ("/") as a folder separator will avoid ambiguity between the various operating systems.

Image bit depth

(Used only when saving files in a non-MAT format)
Select the bit-depth at which you want to save the images. **16-bit images are supported only for TIF formats. Currently, saving images in 12-bit is not supported.**

Overwrite existing files without warning?

Select *Yes* to automatically overwrite a file if it already exists. Select *No* to be prompted for confirmation first.

If you are running the pipeline on a computing cluster, select *Yes* since you will not be able to intervene and answer the confirmation prompt.

When to save

(Used only when saving non-movie files)
Specify at what point during pipeline execution to save file(s).

- *Every cycle:* Useful for when the image of interest is created every cycle and is not dependent on results from a prior cycle.
- *First cycle:* Useful for when you are saving an aggregate image created on the first cycle, e.g., **CorrectIlluminationCalculate** with the *All* setting used on images obtained directly from **NamesAndTypes**.
- *Last cycle* Useful for when you are saving an aggregate image completed on the last cycle, e.g., **CorrectIlluminationCalculate** with the *All* setting used on intermediate images generated during each cycle.

Rescale the images?

(Used only when saving non-MAT file images)
Select *Yes* if you want the image to occupy the full dynamic range of the bit depth you have chosen. For example, if you save an image to an 8-bit file, the smallest grayscale value will be mapped to 0 and the largest value will be mapped to 2^8-1 = 255.

This will increase the contrast of the output image but will also effectively stretch the image data, which may not be desirable in some circumstances. See **RescaleIntensity** for other rescaling options.

Save as grayscale or color image?

(Used only when saving "Objects")
You can save objects as a grayscale image or as a color image.

- *Grayscale:* Use the pixel's object number (label) for the grayscale intensity. Background pixels are colored black. Grayscale images are more suitable if you are going to load the image as objects using **NamesAndTypes** or some other program that will be used to relate object measurements to the pixels in the image. You should save grayscale images using the .TIF or .MAT formats if possible; otherwise you may have problems saving files with more than 255 objects.
- *Color:* Assigns different colors to different objects.

Select colormap

(Used only when saving non-MAT file images)
This affects how images color intensities are displayed. All available colormaps can be seen here.

Record the file and path information to the saved image?

Select *Yes*to store filename and pathname data for each of the new files created via this module as a per-image measurement.

Instances in which this information may be useful include:

- Exporting measurements to a database, allowing access to the saved image. If you are using the machine-learning tools or image viewer in CellProfiler Analyst, for example, you will want to enable this setting if you want the saved images to be displayed along with the original images.
- Allowing downstream modules (e.g., **CreateWebPage**) to access the newly saved files.

Create subfolders in the output folder?

Select *Yes* to create subfolders to match the input image folder structure.

Base image folder

Used only if creating subfolders in the output folder In subfolder mode, **SaveImages** determines the folder for an image file by examining the path of the matching input file. The path that SaveImages uses is relative to the image folder chosen using this setting. As an example, input images might be stored in a folder structure of "images*experiment-name*\ *date**plate-name*". If the image folder is "images", **SaveImages** will store images in the subfolder, "*experiment-name**date**plate-name*". If the image folder is "images*experiment-name*", **SaveImages** will store images in the subfolder, *date**plate-name*".

Saved movie format

(Used only when saving movie files)
Select the movie format to use when saving movies. AVI and MOV store images from successive image sets as movie frames. TIF stores each image as an image plane in a TIF stack.

Module: Align

Align aligns images relative to each other, for example, to correct shifts in the optical path of a microscope in each channel of a multi-channel set of images.

For two or more input images, this module determines the optimal alignment among them. Aligning images is useful to obtain proper measurements of the intensities in one channel based on objects identified in another channel, for example. Alignment is often needed when the microscope is not perfectly calibrated. It can also be useful to align images in a time-lapse series of images. The module stores the amount of shift between images as a measurement, which can be useful for quality control purposes.

Note that the second image (and others following) is always aligned with respect to the first image. That is, the X/Y offsets indicate how much the second image needs to be shifted by to match the first.

Available measurements

- *XShift, Yshift:* The pixel shift in X and Y of the aligned image with respect to the original image.

Settings:

Select the alignment method

Two options for the alignment method are available:

- *Mutual Information:* This more general method works well for aligning images from different modalities that contain the same information, but are expressed differently. However, this method performs better than Normalized Cross Correlation, even in the same modality, if the images are not highly correlated. It is iterative, and thus tends to be slower than other methods, but is more likely to be correct. Essentially, alignment is performed by measuring how well one image "explains" the other. For example, a flourescent image can be aligned to a brightfield image by this method since the relevant features are bright in one modality where they are dim in the other.
- *Normalized Cross Correlation:* This is a good means of alignment in the case of images acquired with the same modality (e.g., all images to be aligned are fluorescent). It is fast, however it can be highly influenced by a particular, possibly spurious, feature and in turn generate anomalously large shifts. It allows for a linear relationship between the intensities of the two images, i.e., the relevant features in the images to be aligned all have varying degrees of brightness.

References

- Lewis JP. (1995) "Fast normalized cross-correlation." *Vision Interface*, 1-7.

Crop mode

The crop mode determines how the output images are either cropped or padded after alignment. The alignment phase calculates the areas in each image that are found to be overlapping. In almost all cases, there will be portions of some or all of the images that don't overlap with any other aligned image. These portions have no counterpart and will be excluded from analysis. There are three choices for cropping:

- *Crop to aligned region:* Crop every image to the region that overlaps in all images. This makes downstream analysis simpler because all of the output images have authentic pixel data at all positions, however it discards parts of images. Also, the output images may not be the same size as

the input images which may cause problems if downstream modules use aligned and unaligned images (which may be of differing sizes) in combination.
- *Pad images:* Align every image and pad with masked black pixels to make each image the same size. This results in larger images, but preserves all information in each of the images. This may be the best choice if images undergo an operation such as smoothing that could use the information that would otherwise be cropped.
- *Keep size:* Maintain the sizes of the images but align them, masking the unaligned portions with black pixels. **Align** aligns all images relative to the first. This is a reasonable option for alignments with small displacements since it maintains a consistent image size which may be useful if output images from different image sets will be compared against each other after processing. The reference image can also be used across image sets. For example, the reference image could be loaded for all image sets in a group to align the entire group's images similarly, then the aligned images could be combined in a module such as **MakeProjection**.

Select the first input image

Specify the name of the first image to align.

Name the first output image

Enter the name of the first aligned image.

Select the second input image

Specify the name of the second image to align.

Name the second output image

Enter the name of the second aligned image.

Module: ApplyThreshold

Apply Threshold sets pixel intensities below or above a certain threshold to zero

---

ApplyThreshold produces either a grayscale or binary image based on a threshold which can be pre-selected or calculated automatically using one of many methods.

Settings:

Select the input image

Choose the image to be thresholded.

Name the output image

Enter a name for the thresholded image.

Select the output image type

Two types of output images can be produced:

- *Grayscale:* The pixels that are retained after some pixels are set to zero or shifted (based on your selections for thresholding options) will have their original intensity values.
- *Binary (black and white):* The pixels that are retained after some pixels are set to zero (based on your selections for thresholding options) will be white and all other pixels will be black (zeroes).

Set pixels below or above the threshold to zero?

(Used only when "Grayscale" thresholding is selected)
For grayscale output, the dim pixels below the threshold can be set to zero or the bright pixels above the threshold can be set to zero. Choose *Below threshold* to threshold dim pixels and *Above threshold* to threshold bright pixels.

Subtract the threshold value from the remaining pixel intensities?

(Used only if the output image is Grayscale and pixels below a given intensity are to be set to zero)
Select *Yes* to shift the value of the dim pixels by the threshold value.

Number of pixels by which to expand the thresholding around those excluded bright pixels

(Used only if the output image is grayscale and pixels above a given intensity are to be set to zero)
This setting is useful when attempting to exclude bright artifactual objects: first, set the threshold to exclude these bright objects; it may also be desirable to expand the thresholded region around those bright objects by a certain distance so as to avoid a "halo" effect.

Threshold strategy

The thresholding strategy determines the type of input that is used to calculate the threshold. The image thresholds can be based on:

- The pixel intensities of the input image (this is the most common).
- A single value manually provided by the user.
- A single value produced by a prior module measurement.
- A binary image (called a *mask*) where some of the pixel intensity values are set to 0, and others are set to 1.

These options allow you to calculate a threshold based on the whole image or based on image sub-regions such as user-defined masks or objects supplied by a prior module.
The choices for the threshold strategy are:

- *Automatic:* Use the default settings for thresholding. This strategy calculates the threshold using the MCT method on the whole image (see below for details on this method) and applies the threshold to the image, smoothed with a Gaussian with sigma of 1.

    This approach is fairly robust, but does not allow you to select the threshold algorithm and does not allow you to apply additional corrections to the threshold.

- *Global:* Calculate a single threshold value based on the unmasked pixels of the input image and use that value to classify pixels above the threshold as foreground and below as background.

    This strategy is fast and robust, especially if the background is uniformly illuminated.

- *Adaptive:* Partition the input image into tiles and calculate thresholds for each tile. For each tile, the calculated threshold is applied only to the pixels within that tile.

    This method is slower but can produce better results for non-uniform backgrounds. However, for signifcant illumination variation, using the **CorrectIllumination** modules is preferable.

- *Per object:* Use objects from a prior module such as **IdentifyPrimaryObjects** to define the region of interest to be thresholded. Calculate a separate threshold for each object and then apply that threshold to pixels within the object. The pixels outside the objects are classified as background.

    This method can be useful for identifying sub-cellular particles or single-molecule probes if the background intensity varies from cell to cell (e.g., autofluorescence or other mechanisms).

- *Manual:* Enter a single value between zero and one that applies to all cycles and is independent of the input image.

    This approach is useful if the input image has a stable or negligible background, or if the input image is the probability map output of the **ClassifyPixels** module (in which case, a value of 0.5 should be chosen). If the input image is already binary (i.e., where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

- *Binary image:* Use a binary image to classify pixels as foreground or background. Pixel values other than zero will be foreground and pixel values that are zero will be background. This method can be used to import a ground-truth segmentation created by CellProfiler or another program.

    The most typical approach to produce a binary image is to use the **ApplyThreshold** module (image as input, image as output) or the **ConvertObjectsToImage** module (objects as input, image as output); both have options to produce a binary image. It can also be used to create objects from an image mask produced by other CellProfiler modules, such as **Morph**. Note that even though no algorithm is actually used to find the threshold in this case, the final threshold value is reported as the *Otsu* threshold calculated for the foreground region.

- *Measurement:* Use a prior image measurement as the threshold. The measurement should have values between zero and one. This strategy can be used to apply a pre-calculated threshold imported as per-image metadata via the **Metadata** module.

    Like manual thresholding, this approach can be useful when you are certain what the cutoff should be. The difference in this case is that the desired threshold does vary from image to image in the experiment but can be measured using another module, such as one of the **Measure** modules, **ApplyThreshold** or an **Identify** module.

Thresholding method

The intensity threshold affects the decision of whether each pixel will be considered foreground (region(s) of interest) or background. A higher threshold value will result in only the brightest regions being identified, whereas a lower threshold value will include dim regions. You can have the threshold automatically calculated from a choice of several methods, or you can enter a number manually between 0 and 1 for the threshold.

Both the automatic and manual options have advantages and disadvantages.

An automatically-calculated threshold adapts to changes in lighting/staining conditions between images and is usually more robust/accurate. In the vast majority of cases, an automatic method is sufficient to achieve the desired thresholding, once the proper method is selected.
In contrast, an advantage of a manually-entered number is that it treats every image identically, so use this option when you have a good sense for what the threshold should be across all images. To help determine the choice of threshold manually, you can inspect the pixel intensities in an image of your choice. To view pixel intensities in an open image, use the pixel intensity tool which is available in any open display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window..
The manual method is not robust with regard to slight changes in lighting/staining conditions between images.
The automatic methods may ocasionally produce a poor threshold for unusual or artifactual images. It also takes a small amount of time to calculate, which can add to processing time for analysis runs on a large number of images.

The threshold that is used for each image is recorded as a per-image measurement, so if you are surprised by unusual measurements from one of your images, you might check whether the automatically calculated threshold was unusually high or low compared to the other images. See the **FlagImage** module if you would like to flag an image based on the threshold value.

There are a number of methods for finding thresholds automatically:

- *Otsu:* This approach calculates the threshold separating the two classes of pixels (foreground and background) by minimizing the variance within the each class.

    This method is a good initial approach if you do not know much about the image characteristics of all the images in your experiment, especially if the percentage of the image covered by foreground varies substantially from image to image.

    Our implementation of Otsu's method allows for assigning the threhsold value based on splitting the image into either two classes (foreground and background) or three classes (foreground, mid-level, and background). See the help below for more details.

- *Mixture of Gaussian (MoG):*This function assumes that the pixels in the image belong to either a

background class or a foreground class, using an initial guess of the fraction of the image that is covered by foreground.

If you know that the percentage of each image that is foreground does not vary much from image to image, the MoG method can be better, especially if the foreground percentage is not near 50%.

This method is our own version of a Mixture of Gaussians algorithm (*O. Friman, unpublished*). Essentially, there are two steps:

1. First, a number of Gaussian distributions are estimated to match the distribution of pixel intensities in the image. Currently three Gaussian distributions are fitted, one corresponding to a background class, one corresponding to a foreground class, and one distribution for an intermediate class. The distributions are fitted using the Expectation-Maximization algorithm, a procedure referred to as Mixture of Gaussians modeling.
2. When the three Gaussian distributions have been fitted, a decision is made whether the intermediate class more closely models the background pixels or foreground pixels, based on the estimated fraction provided by the user.

- *Background:* This method simply finds the mode of the histogram of the image, which is assumed to be the background of the image, and chooses a threshold at twice that value (which you can adjust with a Threshold Correction Factor; see below). The calculation includes those pixels between 2% and 98% of the intensity range.

  This thresholding method is appropriate for images in which most of the image is background. It can also be helpful if your images vary in overall brightness, but the objects of interest are consistently *N* times brighter than the background level of the image.

- *RobustBackground:* Much like the Background: method, this method is also simple and assumes that the background distribution approximates a Gaussian by trimming the brightest and dimmest 5% of pixel intensities. It then calculates the mean and standard deviation of the remaining pixels and calculates the threshold as the mean + 2 times the standard deviation.

  This thresholding method can be helpful if the majority of the image is background, and the results are often comparable or better than the *Background* method.

- *RidlerCalvard:* This method is simple and its results are often very similar to *Otsu*. *RidlerCalvard* chooses an initial threshold and then iteratively calculates the next one by taking the mean of the average intensities of the background and foreground pixels determined by the first threshold. The algorithm then repeats this process until the threshold converges to a single value.

  This is an implementation of the method described in Ridler and Calvard, 1978. According to Sezgin and Sankur 2004, Otsu's overall quality on testing 40 nondestructive testing images is slightly better than Ridler's (average error: Otsu, 0.318; Ridler, 0.401).

- *Kapur:* This method computes the threshold of an image by searching for the threshold that maximizes the sum of entropies of the foreground and background pixel values, when treated as separate distributions.

  This is an implementation of the method described in Kapur *et al*, 1985.

- *Maximum correlation thresholding (MCT):* This method computes the maximum correlation between the binary mask created by thresholding and the thresholded image and is somewhat similar mathematically to *Otsu*.

The authors of this method claim superior results when thresholding images of neurites and other images that have sparse foreground densities.

This is an implementation of the method described in Padmanabhan *et al*, 2010.

References

- Sezgin M, Sankur B (2004) "Survey over image thresholding techniques and quantitative performance evaluation." *Journal of Electronic Imaging*, 13(1), 146-165. (link)
- Padmanabhan K, Eddy WF, Crowley JC (2010) "A novel algorithm for optimal image thresholding of biological data" *Journal of Neuroscience Methods* 193, 380-384. (link)
- Ridler T, Calvard S (1978) "Picture thresholding using an iterative selection method", *IEEE Transactions on Systems, Man and Cybernetics*, 8(8), 630-632.
- Kapur JN, Sahoo PK, Wong AKC (1985) "A new method of gray level picture thresholding using the entropy of the histogram." *Computer Vision, Graphics and Image Processing*, 29, 273-285.

Select binary image

(Used only if Binary image selected for thresholding method)
Select the binary image to be used for thresholding.

Manual threshold

(Used only if Manual selected for thresholding method)
Enter the value that will act as an absolute threshold for the images, a value from 0 to 1.

Select the measurement to threshold with

(Used only if Measurement is selected for thresholding method)
Choose the image measurement that will act as an absolute threshold for the images.

Two-class or three-class thresholding?

(Used only for the Otsu thresholding method)

- *Two classes:* Select this option if the grayscale levels are readily distinguishable into only two classes: foreground (i.e., regions of interest) and background.
- *Three classes*: Choose this option if the grayscale levels fall instead into three classes: foreground, background and a middle intensity between the two. You will then be asked whether the middle intensity class should be added to the foreground or background class in order to generate the final two-class output.

Note that whether two- or three-class thresholding is chosen, the image pixels are always finally assigned two classes: foreground and background.

Three-class thresholding may be useful for images in which you have nuclear staining along with low-intensity non-specific cell staining. Where two-class thresholding might incorrectly assign this intermediate staining to the nuclei objects for some cells, three-class thresholding allows you to assign it to the foreground or background as desired.

However, in extreme cases where either there are almost no objects or the entire field of view is covered with objects, three-class thresholding may perform worse than two-class.

Assign pixels in the middle intensity class to the foreground or the background?

(Used only for three-class thresholding)
Choose whether you want the pixels with middle grayscale intensities to be assigned to the foreground class or the background class.

Approximate fraction of image covered by objects?

(Used only when applying the MoG thresholding method)
Enter an estimate of how much of the image is covered with objects, which is used to estimate the distribution of pixel intensities.

Method to calculate adaptive window size

(Used only if an adaptive thresholding method is used)
The adaptive method breaks the image into blocks, computing the threshold for each block. There are two ways to compute the block size:

- *Image size:* The block size is one-tenth of the image dimensions, or 50 × 50 pixels, whichever is bigger.
- *Custom:* The block size is specified by the user.

Size of adaptive window

(Used only if an adaptive thresholding method with a Custom window size are selected)
Enter the window for the adaptive method. For example, you may want to use a multiple of the largest expected object size.

Threshold correction factor

This setting allows you to adjust the threshold as calculated by the above method. The value entered here adjusts the threshold either upwards or downwards, by multiplying it by this value. A value of 1 means no adjustment, 0 to 1 makes the threshold more lenient and > 1 makes the threshold more stringent.

> When the threshold is calculated automatically, you may find that the value is consistently too stringent or too lenient across all images. This setting is helpful for adjusting the threshold to a value that you empirically determine is more suitable. For example, the Otsu automatic thresholding inherently assumes that 50% of the image is covered by objects. If a larger percentage of the image is covered, the Otsu method will give a slightly biased threshold that may have to be corrected using this setting.

Lower and upper bounds on threshold

Enter the minimum and maximum allowable threshold, a value from 0 to 1. This is helpful as a safety precaution when the threshold is calculated automatically, by overriding the automatic threshold.

> For example, if there are no objects in the field of view, the automatic threshold might be calculated as unreasonably low; the algorithm will still attempt to divide the foreground from background (even though there is no foreground), and you may end up with spurious false positive foreground regions. In such cases, you can estimate the background pixel intensity and set the lower bound according to this empirically-determined value.
> To view pixel intensities in an open image, use the pixel intensity tool which is available in any open

display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window.

Select the smoothing method for thresholding

(Only used for strategies other than Automatic and Binary image)
The input image can be optionally smoothed before being thresholded. Smoothing can improve the uniformity of the resulting objects, by removing holes and jagged edges caused by noise in the acquired image. Smoothing is most likely *not* appropriate if the input image is binary, if it has already been smoothed or if it is an output of the *ClassifyPixels* module.
The choices are:

- *Automatic*: Smooth the image with a Gaussian with a sigma of one pixel before thresholding. This is suitable for most analysis applications.
- *Manual*: Smooth the image with a Gaussian with user-controlled scale.
- *No smoothing*: Do not apply any smoothing prior to thresholding.

Threshold smoothing scale

(Only used if smoothing for threshold is Manual)
This setting controls the scale used to smooth the input image before the threshold is applied. The scale should be approximately the size of the artifacts to be eliminated by smoothing. A Gaussian is used with a sigma adjusted so that 1/2 of the Gaussian's distribution falls within the diameter given by the scale (sigma = scale / 0.674)

Module: ClassifyPixels

ClassifyPixels classify image pixels as belonging to different classes using the machine-learning tool, ilastik.

ClassifyPixels performs per-pixel classification using the ilastik application. Ilastik is now bundled with the CellProfiler distribution; it applies supervised machine learning techniques to images to learn their features. A user trains a classifier with Ilastik and then saves the classifier. The user then uses the ClassifyPixels module to classify the pixels in an image.

ClassifyPixels produces an "image" consisting of probabilities that the pixel belongs to the chosen class; this image is similar to an intensity image that would be produced by fluorescence imaging. Provided that the classifier is sufficiently accurate, the image is well-suited for input into one of the **Identify** modules for object detection. More instructions on using the interface may be found here. Please note that you must use the same image format for classification as for the initial learning phase.

Currently, ilastik is only available for Windows, and is accessible from in the CellProfiler folder under the Start Menu. A 64-bit system is recommended for running ilastik.

Settings:

Classifier file location

Select the folder containing the classifier file to be loaded. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

Classfier file name

This is the name of the Classfier file.

Select the class

Select the class you want to use. The class number corresponds to the label-class in ilastik

Module: ColorToGray

Color to Gray converts an image with three color channels to a set of individual grayscale images.

This module converts RGB (Red, Green, Blue) color images to grayscale. All channels can be merged into one grayscale image (*Combine*), or each channel can be extracted into a separate grayscale image (*Split*). If you use *Combine*, the relative weights will adjust the contribution of the colors relative to each other.

*Note:*All **Identify** modules require grayscale images.

See also **GrayToColor**.

Settings:

Conversion method

How do you want to convert the color image?

- *Split:* Splits the three channels (red, green, blue) of a color image into three separate grayscale images.
- *Combine* Converts a color image to a grayscale image by combining the three channels (red, green, blue) together.

Image type

Many images contain color channels other than red, green and blue. For instance, GIF and PNG formats can have an alpha channel that encodes transparency. TIF formats can have an arbitrary number of channels which represent pixel measurements made by different detectors, filters or lighting conditions. This setting provides three options to choose from:

- *RGB:* The RGB (red,green,blue) color space is the typical model in which color images are stored. Choosing this option will split the image into any of the red, green and blue component images.
- *HSV:*The HSV (hue, saturation, value) color space is based on more intuitive color characteristics as tint, shade and tone. Choosing this option will split the image into any of the hue, saturation, and value component images.
- *Channels:*This is a more complex model for images which involve more than three chnnels.

Relative weight of the red channel

(Used only when combining channels)
Relative weights: If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight of the green channel

(Used only when combining channels)
Relative weights: If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight of the blue channel

(Used only when combining channels)
Relative weights: If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Channel number

This setting chooses a channel to be processed. *Red: 1* is the first channel in a .TIF or the red channel in a traditional image file. *Green: 2* and *Blue: 3* are the second and third channels of a TIF or the green and blue channels in other formats. *Alpha: 4* is the transparency channel for image formats that support transparency and is channel # 4 for a .TIF file. **ColorToGray** will fail to process an image if you select a channel that is not supported by that image, for example, "5" for a .PNG file

Relative weight of the channel

(Used only when combining channels)
Relative weights: If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Image name

This is the name of the grayscale image that holds the image data from the chosen channel.

Module: CorrectIlluminationApply

Correct Illumination - Apply applies an illumination function, usually created by **CorrectIlluminationCalculate**, to an image in order to correct for uneven illumination (uneven shading).

This module applies a previously created illumination correction function, either loaded by **LoadSingleImage** or created by **CorrectIlluminationCalculate**. This module corrects each image in the pipeline using the function specified.

See also **CorrectIlluminationCalculate**.

Settings:

Select the input image

Select the image to be corrected.

Name the output image

Enter a name for the corrected image.

Select the illumination function

Select the illumination correction function image that will be used to carry out the correction. This image is usually produced by another module or loaded as a .mat format image using the **Images** module or **LoadSingleImage**.

Select how the illumination function is applied

This choice depends on how the illumination function was calculated and on your physical model of the way illumination variation affects the background of images relative to the objects in images; it is also somewhat empirical.

- *Subtract:* Use this option if the background signal is significant relative to the real signal coming from the cells. If you created the illumination correction function using *Background*, then you will want to choose *Subtract* here.
- *Divide:* Choose this option if the the signal to background ratio is high (the cells are stained very strongly). If you created the illumination correction function using *Regular*, then you will want to choose *Divide* here.

Module: CorrectIlluminationCalculate

Correct Illumination - Calculate calculates an illumination function that is used to correct uneven illumination/lighting/shading or to reduce uneven background in images.

This module calculates an illumination function that can either be saved to the hard drive for later use or immediately applied to images later in the pipeline. This function will correct for the uneven illumination in images. If saving, select *.mat* format in **SaveImages**. Use the **CorrectIlluminationApply** module to apply the function to the image to be corrected.

Illumination correction is a challenge to do properly; please see the examples and tutorials pages on the CellProfiler website for further advice.

See also **CorrectIlluminationApply**, **EnhanceOrSuppressFeatures**.

Settings:

Select the input image

Choose the image to be used to calculate the illumination function.

Name the output image

Enter a name for the resultant illumination function.

Select how the illumination function is calculated

Choose which method you want to use to calculate the illumination function. You may chose from the following options:

- *Regular:* If you have objects that are evenly dispersed across your image(s) and cover most of the image, the *Regular* method might be appropriate. Regular intensities makes the illumination function based on the intensity at each pixel of the image (or group of images if you are in *All* mode) and is most often rescaled (see below) and applied by division using **CorrectIlluminationApply.** Note that if you are in *Each* mode or using a small set of images with few objects, there will be regions in the average image that contain no objects and smoothing by median filtering is unlikely to work well. *Note:* it does not make sense to choose (*Regular* + *No smoothing* + *Each*) because the illumination function would be identical to the original image and applying it will yield a blank image. You either need to smooth each image, or you need to use *All* images.
- *Background:* If you think that the background (dim points) between objects show the same pattern of illumination as your objects of interest, you can choose the *Background* method. Background intensities finds the minimum pixel intensities in blocks across the image (or group of images if you are in *All* mode) and is most often applied by subtraction using the **CorrectIlluminationApply** module. *Note:* if you will be using the *Subtract* option in the **CorrectIlluminationApply** module, you almost certainly do not want to rescale the illumination function.

Please note that if a mask was applied to the input image, the pixels outside of the mask will be excluded from consideration. This is useful, for instance, in cases where you have masked out the well edge in an image from a multi-well plate; the dark well edge would distort the illumination correction function along the interior well edge. Masking the image beforehand solves this problem.

Dilate objects in the final averaged image?

(Used only if the Regular method is selected)
For some applications, the incoming images are binary and each object should be dilated with a Gaussian filter in the final averaged (projection) image. This is for a sophisticated method of illumination correction where model objects are produced. Select *Yes* to dilate objects for this approach.

Dilation radius

(Used only if the "Regular" method and dilation is selected)
This value should be roughly equal to the original radius of the objects

Block size

(Used only if "Background" is selected)
The block size should be large enough that every square block of pixels is likely to contain some background pixels, where no objects are located.

Rescale the illumination function?

The illumination function can be rescaled so that the pixel intensities are all equal to or greater than 1. You have the following options:

- *Yes:* Rescaling is recommended if you plan to use the *Regular* method (and hence, the *Divide* option in **CorrectIlluminationApply**) so that the corrected images are in the range 0 to 1.
- *No:* Rescaling is not recommended if you plan to use the *Background* method, which is paired with the *Subtract* option in **CorrectIlluminationApply**. Note that as a result of the illumination function being rescaled from 1 to infinity, the rescaling of each image might be dramatic if there is substantial variation across the field of view, causing the corrected images to be very dark.
- Median*:* This option chooses the median value in the image to rescale so that division increases some values and decreases others.

Calculate function for each image individually, or based on all images?

Calculate a separate function for each image, or one for all the images? You can calculate the illumination function using just the current image or you can calculate the illumination function using all of the images in each group. The illumination function can be calculated in one of the three ways:

- *Each:* Calculate an illumination function for each image individually.
- *All: First cycle:* Calculate an illumination function based on all of the images in a group, performing the calculation before proceeding to the next module. This means that the illumination function will be created in the first cycle (making the first cycle longer than subsequent cycles), and lets you use the function in a subsequent **CorrectIllumination_Apply** module in the same pipeline, but also means that you will not have the ability to filter out images (e.g., by using **FlagImage**). The input images need to be assembled using the **Input** modules; using images produced by other modules will yield an error.
- *All: Across cycles:* Calculate an illumination function across all cycles in each group. This option takes any image as input; however, the illumination function will not be completed until the end of the last cycle in the group. You can use **SaveImages** to save the illumination function after the last cycle in the group and then use the resulting image in another pipeline. The option is useful if you want to exclude images that are filtered by a prior **FlagImage** module.

Smoothing method

If requested, the resulting image is smoothed. See the **EnhanceOrSuppressFeatures** module help for more details. If you are using *Each* mode, this is almost certainly necessary. If you have few objects in each image or a small image set, you may want to smooth.

You should smooth to the point where the illumination function resembles a believable pattern. For example, if you are trying to correct a lamp illumination problem, apply smoothing until you obtain a fairly smooth pattern without sharp bright or dim regions. Note that smoothing is a time-consuming process, but some methods are faster than others.

- *Fit Polynomial:* This methdod is fastest but does not allow a very tight fit compared to the slower median and Gaussian filtering methods.
- Median Filter, *Gaussian Filter:* Use a median or Gaussian filter, respectively. We typically recommend *Median Filter* vs. *Gaussian Filter* because the median is less sensitive to outliers, although the results are also slightly less smooth and the fact that images are in the range of 0 to 1 means that outliers typically will not dominate too strongly anyway.
- *Smooth to Average:* A less commonly used option is to completely smooth the entire image, which will create a flat, smooth image where every pixel of the image is the average of what the illumination function would otherwise have been.
- *Splines:* This method (*Lindblad and Bengtsson, 2001*) fits a grid of cubic splines to the background while excluding foreground pixels from the calculation. It operates iteratively, classifying pixels as background, computing a best fit spline to this background and then reclassifying pixels as background until the spline converges on its final value.
- *Convex Hull:* This method algorithm proceeds as follows:
    - Choose 256 evenly-spaced intensity levels between the minimum and maximum intensity for the image
    - Set the intensity of the output image to the minimum intensity of the input image
    - Iterate over the intensity levels, from lowest to highest
        - For a given intensity, find all pixels with equal or higher intensities
        - Find the convex hull that encloses those pixels
        - Set the intensity of the output image within the convex hull to the current intensity

  The Convex Hull method can be used on an image whose objects are darker than their background and whose illumination intensity decreases monotonically from the brightest point.

References

- J Lindblad and E Bengtsson (2001) "A comparison of methods for estimation of intensity nonuniformities in 2D and 3D microscope images of fluorescence stained cells.", Proceedings of the 12th Scandinavian Conference on Image Analysis (SCIA), pp. 264-271

Method to calculate smoothing filter size

(Used only if a smoothing method other than Fit Polynomial is selected)
Calculate the smoothing filter size. There are three options:

- *Automatic:* The size is computed as 1/40 the size of the image or 30 pixels, whichever is smaller.
- *Object size:* The size is obtained relative to the width of artifacts to be smoothed.
- *Manually:* Use a manually entered value.

Approximate object size

(Used only if Automatic is selected for smoothing filter size calculation)

Enter the approximate width of the artifacts to be smoothed, in pixels.

Smoothing filter size

(Used only if Manually is selected for smoothing filter size calculation)
Enter the size of the desired smoothing filter, in pixels.

Automatically calculate spline parameters?

(Used only if Splines are selected for the smoothing method)
Select *Yes* to automatically calculate the parameters for spline fitting.

Select *No* to specify the background mode, background threshold, scale, maximum number of iterations and convergence.

Background mode

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This setting determines which pixels are background and which are foreground.

- *auto*: Determine the mode from the image. This will set the mode to dark if most of the pixels are dark, bright if most of the pixels are bright and gray if there are relatively few dark and light pixels relative to the number of mid-level pixels
- *dark*: Fit the spline to the darkest pixels in the image, excluding brighter pixels from consideration. This may be appropriate for a fluorescent image.
- *bright*: Fit the spline to the lightest pixels in the image, excluding the darker pixels. This may be appropriate for a histologically stained image.
- *gray*: Fit the spline to mid-range pixels, excluding both dark and light pixels. This may be appropriate for a brightfield image where the objects of interest have light and dark features.

Number of spline points

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This is the number of control points for the spline. A value of 5 results in a 5x5 grid of splines across the image and is the value suggested by the method's authors. A lower value will give you a more stable background while a higher one will fit variations in the background more closely and take more time to compute.

Background threshold

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This setting determines the cutoff used when excluding foreground pixels from consideration. On each iteration, the method computes the standard deviation of background pixels from the computed background. The number entered in this setting is the number of standard deviations a pixel can be from the computed background on the last pass if it is to be considered as background during the next pass.

You should enter a higher number to converge stabily and slowly on a final background and a lower number to converge more rapidly, but with lower stability. The default for this parameter is two standard deviations; this will provide a fairly stable background estimate.

Image resampling factor

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This setting controls how the image is resampled to make a smaller image. Resampling will speed up processing, but may degrade performance if the resampling factor is larger than the diameter of foreground objects. The image will be downsampled by the factor you enter. For instance, a 500x600 image will be downsampled into a 250x300 image if a factor of 2 is entered.

Maximum number of iterations

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This setting determines the maximum number of iterations of the algorithm to be performed. The algorithm will perform fewer iterations if it converges.

Residual value for convergence

(Used only if Splines are selected for the smoothing method and spline parameters are not calculated automatically)
This setting determines the convergence criterion. The software sets the convergence criterion to the number entered here times the signal intensity; the convergence you enter is the fraction of the signal intensity that indicates convergence. The algorithm derives a standard deviation of the background pixels from the calculated background on each iteration. The algorithm terminates when the difference between the standard deviation for the current iteration and the previous iteration is less than the convergence criterion.

Enter a smaller number for the convergence to calculate a more accurate background. Enter a larger number to calculate the background using fewer iterations, but less accuracy.

Retain the averaged image?

The averaged image is the illumination function prior to dilation or smoothing. It is an image produced during the calculations, not typically needed for downstream modules. It can be helpful to retain it in case you wish to try several different smoothing methods without taking the time to recalculate the averaged image each time.

Select *Yes* to retain this averaged image. Use the **SaveImages** module to save it to your hard drive.

Name the averaged image

(Used only if the averaged image is to be retained for later use in the pipeline)
Enter a name that will allow the averaged image to be selected later in the pipeline.

Retain the dilated image?

The dilated image is the illumination function after dilation but prior to smoothing. It is an image produced during the calculations, and is not typically needed for downstream modules.

Select *Yes* to retain this dilated image. Use the **SaveImages** module to save it to your hard drive.

Name the dilated image

(Used only if the dilated image is to be retained for later use in the pipeline)
Enter a name that will allow the dilated image to be selected later in the pipeline.

Module: Crop

Crop crops or masks an image.

This module crops images into a rectangle, ellipse, an arbitrary shape provided by you, the shape of object(s) identified by an **Identify** module, or a shape created using a previous **Crop** module in the pipeline.

Keep in mind that cropping changes the size of your images, which may have unexpected consequences. For example, identifying objects in a cropped image and then trying to measure their intensity in the *original* image will not work because the two images are not the same size.

Available measurements

- *AreaRetainedAfterCropping:* The area of the image left after cropping.
- *OriginalImageArea:* The area of the original input image.

Special note on saving images: You can save the cropping shape that you have defined in this module (e.g., an ellipse you drew) so that you can use the *Image* option in future analyses. To do this, save either the mask or cropping in **SaveImages**. See the **SaveImages** module help for more information on saving cropping shapes.

Settings:

Select the input image

Choose the image to be cropped.

Name the output image

Enter the name to be given to cropped image.

Select the cropping shape

Select the shape into which you would like to crop:

- *Rectangle:* Self-explanatory.
- *Ellipse:* Self-explanatory.
- *Image:* Cropping will occur based on a binary image you specify. A choice box with available images will appear from which you can select an image. To crop into an arbitrary shape that you define, choose *Image* and use the **LoadSingleImage** module to load a black and white image that you have already prepared from a file. If you have created this image in a program such as Photoshop, this binary image should contain only the values 0 and 255, with zeros (black) for the parts you want to remove and 255 (white) for the parts you want to retain. Alternately, you may have previously generated a binary image using this module (e.g., using the *Ellipse* option) and saved it using the **SaveImages** module.
  In any case, the image must be exactly the same starting size as your image and should contain a contiguous block of white pixels, because the cropping module may remove rows and columns that are completely blank.
- *Objects:* Crop based on labeled objects identified by a previous **Identify** module.

- *Previous cropping:* The cropping generated by a previous cropping module. You will be able to select images that were generated by previous **Crop** modules. This **Crop** module will use the same cropping that was used to generate whichever image you choose.

Select the cropping method

Choose whether you would like to crop by typing in pixel coordinates or clicking with the mouse.

- *Coordinates:* For *Ellipse*, you will be asked to enter the geometric parameters of the ellipse. For *Rectangle*, you will be asked to specify the coordinates of the corners.
- *Mouse:* For *Ellipse*, you will be asked to click five or more points to define an ellipse around the part of the image you want to analyze. Keep in mind that the more points you click, the longer it will take to calculate the ellipse shape. For *Rectangle*, you can click as many points as you like that are in the interior of the region you wish to retain.

Apply which cycle's cropping pattern?

Specify how a given cropping pattern should be applied to other image cycles:

- *First:* The cropping pattern from the first image cycle is applied to all subsequent cyles. This is useful if the first image is intended to function as a template in some fashion.
- *Every:* Every image cycle is cropped individually.

Left and right rectangle positions

(Used only if Rectangle selected as cropping shape, or if using Plate Fix)
Specify the left and right positions for the bounding rectangle by selecting one of the following:

- *Absolute:* Specify these values as absolute pixel coordinates in the original image. For instance, you might enter "25", "225", and "Absolute" to create a 200×200 pixel image that is 25 pixels from the top-left corner.
- *From edge:* Specify the position relative to the image edge. For instance, you might enter "25", "25", and "Edge" to crop 25 pixels from both the left and right edges of the image, irrespective of the image's original size.

Top and bottom rectangle positions

(Used only if Rectangle selected as cropping shape, or if using Plate Fix)
Specify the top and bottom positions for the bounding rectangle by selecting one of the following:

- *Absolute:* Specify these values as absolute pixel coordinates. For instance, you might enter "25", "225", and "Absolute" to create a 200×200 pixel image that's 25 pixels from the top-left corner.
- *From edge:* Specify position relative to the image edge. For instance, you might enter "25", "25", and "Edge" to crop 25 pixels from the edges of your images irrespective of their size.

Coordinates of ellipse center

(Used only if Ellipse selected as cropping shape)
Specify the center pixel position of the ellipse.

Ellipse radius, X direction

(Used only if Ellipse selected as cropping shape)
Specify the radius of the ellipse in the X direction.

Ellipse radius, Y direction

(Used only if Ellipse selected as cropping shape)
Specify the radius of the ellipse in the Y direction.

Use Plate Fix?

(Used only if Image selected as cropping shape)
Select *Yes* to attempt to regularize the edges around a previously-identified plate object.

When attempting to crop based on a previously identified object such as a rectangular plate, the plate may not have precisely straight edges: there might be a tiny, almost unnoticeable "appendage" sticking out. Without Plate Fix, the **Crop** module would not crop the image tightly enough: it would retain the tiny appendage, leaving a lot of blank space around the plate and potentially causing problems with later modules (especially ones involving illumination correction).

Plate Fix takes the identified object and crops to exclude any minor appendages (technically, any horizontal or vertical line where the object covers less than 50% of the image). It also sets pixels around the edge of the object (for regions greater than 50% but less than 100%) that otherwise would be 0 to the background pixel value of your image, thus avoiding problems with other modules.

Important note: Plate Fix uses the coordinates entered in the boxes normally used for rectangle cropping (Top, Left and Bottom, Right) to tighten the edges around your identified plate. This is done because in the majority of plate identifications you do not want to include the sides of the plate. If you would like the entire plate to be shown, you should enter "1:end" for both coordinates. If, for example, you would like to crop 80 pixels from each edge of the plate, you could enter Top, Left and Bottom, Right values of 80 and select *From edge*.

Remove empty rows and columns?

Use this option to choose whether to remove rows and columns that lack objects:

- *No:* Leave the image the same size. The cropped areas will be set to zeroes, and will appear as black.
- *Edges:* Crop the image so that its top, bottom, left and right are at the first non-blank pixel for that edge.
- *All:* Remove any row or column of all-blank pixels, even from the internal portion of the image.

Select the masking image

(Used only if Image selected as cropping shape)
Select the image to be use as a cropping mask.

Select the image with a cropping mask

(Used only if Previous cropping selected as cropping shape)
Select the image associated with the cropping mask that you want to use.

Select the objects

(Used only if Objects selected as cropping shape)
Select the objects that are to be used as a cropping mask.

Module: EnhanceEdges

Enhance Edges enhances or identifies edges in an image, which can improve object identification or other downstream image processing.

This module enhances the edges (gradients) in a grayscale image. All methods other than Canny produce a grayscale image that can be used in an **Identify** module or thresholded using the **ApplyThreshold** module to produce a binary (black/white) mask of edges. The Canny algorithm produces a binary (black/white) mask image consisting of the edge pixels.

Settings:

Select the input image

What did you call the image in which you want to enhance the edges?

Name the output image

What do you want to call the image with edges enhanced?

Automatically calculate the threshold?

(Used only with the Canny option and automatic thresholding)
Select *Yes* to automatically calculate the threshold using a three-category Otsu algorithm performed on the Sobel transform of the image.

Select *No* to manually enter the threshold value.

Absolute threshold

(Used only with the Canny option and manual thresholding)
The upper cutoff for Canny edges. All Sobel-transformed pixels with this value or higher will be marked as an edge. You can enter a threshold between 0 and 1.

Threshold adjustment factor

(Used only with the Canny option and automatic thresholding)
This threshold adjustment factor is a multiplier that is applied to both the lower and upper Canny thresholds if they are calculated automatically. An adjustment factor of 1 indicates no adjustment. The adjustment factor has no effect on any threshhold entered manually entered.

Select an edge-finding method

There are several methods that can be used to enhance edges:

- *Sobel:* Finds edges using the Sobel approximation to the derivative. The Sobel method derives a horizontal and vertical gradient measure and returns the square-root of the sum of the two squared signals.
- *Prewitt:* Finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of the image is maximum.

- *Roberts:* Finds edges using the Roberts approximation to the derivative. The Roberts method looks for gradients in the diagonal and anti-diagonal directions and returns the square-root of the sum of the two squared signals. This method is fast, but it creates diagonal artifacts that may need to be removed by smoothing.
- *LoG:* Applies a Laplacian of Gaussian filter to the image and finds zero crossings.
- *Canny:* Finds edges by looking for local maxima of the gradient of the image. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

Select edge direction to enhance

(Used only with Prewitt and Sobel methods)
The direction of the edges are you are identifying in the image (predominantly horizontal, predominantly vertical, or both).

Calculate value for low threshold automatically?

(Used only with the Canny option and automatic thresholding)
Select *Yes* to automatically calculate the low / soft threshold cutoff for the Canny method.

Select *No* to manually enter the low threshold value.

Low threshold value

(Used only with the Canny option and manual thresholding)
Enter the soft threshold cutoff for the Canny method. The Canny method will mark all Sobel-transformed pixels with values below this threshold as not being edges.

Module: EnhanceOrSuppressFeatures

Enhance Or Suppress Features enhances or suppresses certain image features (such as speckles, ring shapes, and neurites), which can improve subsequent identification of objects.

This module enhances or suppresses the intensity of certain pixels relative to the rest of the image, by applying image processing filters to the image. It produces a grayscale image in which objects can be identified using an **Identify** module.

Settings:

Select the input image

Select the image with features to be enhanced or suppressed.

Name the output image

Enter a name for the feature-enhanced or suppressed image.

Select the operation

Select whether you want to enhance or suppress the features you designated.

- *Enhance:* Produce an image whose intensity is largely composed of the features of interest.
- *Suppress:* Produce an image with the features largely removed.

Feature size

(Used only if circles, speckles or neurites are selected, or if suppressing features)
Enter the diameter of the largest speckle, the width of the circle or the width of the neurites to be enhanced or suppressed, which will be used to calculate an adequate filter size. To view pixel intensities in an open image, use the pixel intensity tool which is available in any open display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window.

Feature type

(Used only if Enhance is selected)
This module can enhance three kinds of image intensity features:

- *Speckles:* A speckle is an area of enhanced intensity relative to its immediate neighborhood. The module enhances speckles using a white tophat filter, which is the image minus the morphological grayscale opening of the image. The opening operation first suppresses the speckles by applying a grayscale erosion to reduce everything within a given radius to the lowest value within that radius, then uses a grayscale dilation to restore objects larger than the radius to an approximation of their former shape. The white tophat filter enhances speckles by subtracting the effects of opening from the original image.
- *Neurites:* Neurites are taken to be long, thin features of enhanced intensity. Choose this option to enhance the intensity of the neurites using the Line structures or Tubeness methods described below.
- *Dark holes:* The module uses morphological reconstruction (the rolling-ball algorithm) to identify dark

holes within brighter areas, or brighter ring shapes. The image is inverted so that the dark holes turn into bright peaks. The image is successively eroded and the eroded image is reconstructed at each step, resulting in an image which is missing the peaks. Finally, the reconstructed image is subtracted from the previous reconstructed image. This leaves circular bright spots with a radius equal to the number of iterations performed.
- *Circles:* The module calculates the circular Hough transform of the image at the diameter given by the feature size. The Hough transform will have the highest intensity at points that are centered within a ring of high intensity pixels where the ring diameter is the feature size. You may want to use the **EnhanceEdges** module to find the edges of your circular object and then process the output by enhancing circles. You can use **IdentifyPrimaryObjects** to find the circle centers and then use these centers as seeds in **IdentifySecondaryObjects** to find whole, circular objects using a watershed.
- *Texture:* **EnanceOrSuppressFeatures** produces an image whose intensity is the variance among nearby pixels. This method weights pixel contributions by distance using a Gaussian to calculate the weighting. You can use this method to separate foreground from background if the foreground is textured and the background is not.
- *DIC:* This method recovers the optical density of a DIC image by integrating in a direction perpendicular to the shear direction of the image.

In addition, this module enables you to suppress certain features (such as speckles) by specifying the feature size.

Range of hole sizes

(Used only if Dark holes is selected)
The range of hole sizes to be enhanced. The algorithm will identify only holes whose diameters fall between these two values.

Smoothing scale

(Used only for the Texture, DIC or Neurites methods)

- *Texture*: This is the scale of the texture features, roughly in pixels. The algorithm uses the smoothing value entered as the sigma of the Gaussian used to weight nearby pixels by distance in the variance calculation.
- *DIC:* Specifies the amount of smoothing of the image in the direction parallel to the shear axis of the image. The line integration method will leave streaks in the image without smoothing as it encounters noisy pixels during the course of the integration. The smoothing takes contributions from nearby pixels which decreases the noise but smooths the resulting image.
- *DIC:* Increase the smoothing to eliminate streakiness and decrease the smoothing to sharpen the image.
- *Neurites:* The *Tubeness* option uses this scale as the sigma of the Gaussian used to smooth the image prior to gradient detection.

 Smoothing can be turned off by entering a value of zero, but this is not recommended.

Shear angle

(Used only for the DIC method)
The shear angle is the direction of constant value for the shadows and highlights in a DIC image. The gradients in a DIC image run in the direction perpendicular to the shear angle. For example, if the shadows run diagonally from lower left to upper right and the highlights appear above the shadows, the

shear angle is 45°. If the shadows appear on top, the shear angle is 180° + 45° = 225°.

Decay

(Used only for the DIC method)
The decay setting applies an exponential decay during the process of integration by multiplying the accumulated sum by the decay at each step. This lets the integration recover from accumulated error during the course of the integration, but it also results in diminished intensities in the middle of large objects. Set the decay to a large value, on the order of 1 - 1/diameter of your objects if the intensities decrease toward the middle. Set the decay to a small value if there appears to be a bias in the integration direction.

Enhancement method

(Used only for the Neurites method)
Two methods can be used to enhance neurites:

- *Tubeness*: This method is an adaptation of the method used by the ImageJ Tubeness plugin. The image is smoothed with a Gaussian. The Hessian is then computed at every point to measure the intensity gradient and the eigenvalues of the Hessian are computed to determine the magnitude of the intensity. The absolute maximum of the two eigenvalues gives a measure of the ratio of the intensity of the gradient in the direction of its most rapid descent versus in the orthogonal direction. The output image is the absolute magnitude of the highest eigenvalue if that eigenvalue is negative (white neurite on dark background), otherwise, zero.
- *Line structures*: The module takes the difference of the white and black tophat filters (a white tophat filtering is the image minus the morphological grayscale opening of the image; a black tophat filtering is the morphological grayscale closing of the image minus the image). The effect is to enhance lines whose width is the "feature size".

Module: FlipAndRotate

Flip and rotate flips (mirror image) and/or rotates an image

Available measurements

- *Rotation:* Angle of rotation for the input image.

Settings:

Select method to flip image

Select how the image is to be flipped.

Select method to rotate image

- *Do not rotate:* Leave the image unrotated. This should be used if you want to flip the image only.
- *Enter angle:* Provide the numerical angle by which the image should be rotated.
- *Enter coordinates:* Provide the X,Y pixel locations of two points in the image that should be aligned horizontally or vertically.
- *Use mouse:* CellProfiler will pause so you can select the rotation interactively. When prompted during the analysis run, grab the image by clicking the left mouse button, rotate the image by dragging with the mouse, then release the mouse button. Press the *Done* button on the image after rotating the image appropriately.

Crop away the rotated edges?

(Used only when rotating images)
When an image is rotated, there will be black space at the corners/edges; select *Yes* to crop away the incomplete rows and columns of the image, or select *No* to leave it as-is.

This cropping will produce an image that is not exactly the same size as the original, which may affect downstream modules.

Calculate rotation

(Used only when using "Use mouse" to rotate images)
Select the cycle(s) at which the calculation is requested and calculated.

- *Individually:* Determine the amount of rotation for each image individually, e.g., for each cycle.
- *Only Once:* Define the rotation only once (on the first image), then then apply it to all images.

Select how the specified points should be aligned

(Used only when using "Enter coordinates" to rotate images)
Specify whether you would like the coordinate points that you entered to be horizontally or vertically aligned after the rotation is complete.

Enter angle of rotation

(Used only when using "Enter angle" to rotate images)
Enter the angle you would like to rotate the image. This setting is in degrees, with positive angles corresponding to counterclockwise and negative as clockwise.

Module: GrayToColor

Gray to Color takes grayscale images and and produces a color image from them.

This module takes grayscale images as input and assigns them to colors in a red, green, blue (RGB) image or a cyan, magenta, yellow, black (CMYK) image. Each color's brightness can be adjusted independently by using relative weights.

See also **ColorToGray**.

Settings:

Select a color scheme

This module can use one of two color schemes to combine images:

- *RGB*: Each input image determines the intensity of one of the color channels: red, green, and blue.
- *CMYK*: Three of the input images are combined to determine the colors (cyan, magenta, and yellow) and a fourth is used only for brightness. The cyan image adds equally to the green and blue intensities. The magenta image adds equally to the red and blue intensities. The yellow image adds equally to the red and green intensities.
- *Stack*: The channels are stacked in order. An arbitrary number of channels is allowed.

Relative weight for the red image

(Used only if RGB is selected as the color scheme)
Enter the relative weight for the red image. If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the green image

(Used only if RGB is selected as the color scheme)
Enter the relative weight for the green image. If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the blue image

(Used only if RGB is selected as the color scheme)
Enter the relative weight for the blue image. If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the cyan image

(Used only if CMYK is selected as the color scheme)
Enter the relative weight for the cyan image. If all relative weights are equal, all colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the magenta image

(Used only if CMYK is selected as the color scheme)
Enter the relative weight for the magenta image. If all relative weights are equal, all colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the yellow image

(Used only if CMYK is selected as the color scheme)
Enter the relative weight for the yellow image. If all relative weights are equal, all colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the brightness image

(Used only if CMYK is selected as the color scheme)
Enter the relative weight for the brightness image. If all relative weights are equal, all colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Module: ImageMath

Image Math performs simple mathematical operations on image intensities.

This module can perform addition, subtraction, multiplication, division, or averaging of two or more image intensities, as well as inversion, log transform, or scaling by a constant for individual image intensities.

Keep in mind that after the requested operations are carried out, the final image may have a substantially different range of pixel intensities than the original. CellProfiler assumes that the image is scaled from 0 – 1 for object identification and display purposes, so additional rescaling may be needed. Please see the **RescaleIntensity** module for more scaling options.

See also **ApplyThreshold**, **RescaleIntensity**, **CorrectIlluminationCalculate**.

Settings:

Operation

Select the operation to perform. Note that if more than two images are chosen, then operations will be performed sequentially from first to last, e.g., for "Divide", (Image1 / Image2) / Image3

- *Add:* Adds the first image to the second, and so on.
- *Subtract:* Subtracts the second image from the first.
- *Absolute Difference:* The absolute value of the difference between the first and second images.
- *Multiply:* Multiplies the first image by the second.
- *Divide:* Divides the first image by the second.
- *Average* Calculates the mean intensity of the images loaded in the module. This is equivalent to the Add option divided by the number of images loaded by this module. If you would like to average all of the images in an entire pipeline, i.e., across cycles, you should instead use the **CorrectIlluminationCalculate** module and choose the *All* (vs. *Each*) option.
- *Maximum:* Returns the element-wise maximum value at each pixel location.
- *Invert:* Subtracts the image intensities from 1. This makes the darkest color the brightest and vice-versa.
- *Log transform (base 2)* Log transforms each pixel's intensity. The actual function is $\log_2(\text{image} + 1)$, transforming values from 0 to 1 into values from 0 to 1.
- *Log transform (legacy)* Log_2 transform for backwards compatibility.
- *None* This option is useful if you simply want to select some of the later options in the module, such as adding, multiplying, or exponentiating your image by a constant.

Note that *Invert*, *Log transform (base 2)*, and *None* operate on only a single image.

Name the output image

Enter a name for the resulting image.

Image or measurement?

You can perform math operations using two images or you can use a measurement for one of the operands. For instance, to divide the intensity of one image by another, choose *Image* for both and pick the respective images. To divide the intensity of an image by its median intensity, use

MeasureImageIntensity prior to this module to calculate the median intensity, then select *Measurement* and use the median intensity measurement as the denominator

Selec the image that you want to use for this operation.

Measurement

This is a measurement made on the image. The value of the measurement is used for the operand for all of the pixels of the other operand's image.

Enter the number that you would like to multiply the above image by. This multiplication is applied before other operations.

Raise the power of the result by

Enter an exponent to raise the result to *after* the chosen operation

Multiply the result by

Enter a factor to multiply the result by *after* the chosen operation

Add to result

Enter a number to add to the result *after* the chosen operation

Set values less than 0 equal to 0?

Values outside the range 0 to 1 might not be handled well by other modules. Select *Yes* to set negative values to 0.

Set values greater than 1 equal to 1?

Values outside the range 0 to 1 might not be handled well by other modules. Select *Yes* to set values greater than 1 to a maximum value of 1.

Ignore the image masks?

Usually, the smallest mask of all image operands is applied after image math has been completed. Select *Yes* to set equal to zero all previously masked pixels and operate on the masked images as if no mask had been applied.

Module: InvertForPrinting

Invert For Printing inverts fluorescent images into brightfield-looking images for printing.

This module turns a single or multi-channel immunofluorescent-stained image into an image that resembles a brightfield image stained with similarly colored stains, which generally prints better.

You can operate on up to three grayscale images (representing the red, green, and blue channels of a color image) or on an image that is already a color image. The module can produce either three grayscale images or one color image as output.

If you want to invert the grayscale intensities of an image, use **ImageMath**.

Settings:

Input image type

Specify whether you are combining several grayscale images or loading a single color image.

Use a red image?

Select *Yes* to specify an image to use for the red channel.

Use a green image?

Select *Yes* to specify an image to use for the green channel.

Use a blue image?

Select *Yes* to specify an image to use for the blue channel.

Select the color image

Select the color image to use.

Output image type

Specify whether you want to produce several grayscale images or one color image.

Name the inverted color image

(Used only when producing a color output image)
Enter a name for the inverted color image.

Module: MakeProjection

Make Projection combines several two-dimensional images of the same field of view together, either by performing a mathematical operation upon the pixel values at each pixel position.

This module combines a set of images by performing a mathematic operation of your choice at each pixel position; please refer to the settings help for more information on the available operations. The process of averaging or summing a Z-stack (3D image stack) is known as making a projection.

This module will create a projection of all images specified in the Input modules. For more information on loading image stacks and movies, see *Help > Creating a Project > Loading Image Stacks and Movies*. To achieve per-folder projections i.e., creating a projection for each set of images in a folder, for all input folders, make the following setting specifications:

1. In the **Images** module, drag-and-drop the parent folder containing the sub-folders.
2. In the **Metadata** module, enable metadata extraction and extract metadata from the folder name by using a regular expression to capture the subfolder name, e.g., `.*[\\/](?P<Subfolder>.*)$`
3. In the **NamesAndTypes** module, specify the appropriate names for any desired channels.
4. In the **Groups** module, enable image grouping, and select the metadata tag representing the sub-folder name as the metadata category.

Keep in mind that the projection image is not immediately available in subsequent modules because the output of this module is not complete until all image processing cycles have completed. Therefore, the projection should be created with a dedicated pipeline.

See also the help for the **Input** modules.

Settings:

Select the input image

Select the image to be made into a projection.

Type of projection

The final projection image can be created by the following methods:

- *Average:* Use the average pixel intensity at each pixel position.
- *Maximum:* Use the maximum pixel value at each pixel position.
- *Minimum:* Use the minimum pixel value at each pixel position.
- *Sum:* Add the pixel values at each pixel position.
- *Variance:* Compute the variance at each pixel position.
  The variance method is described in Selinummi et al (2009). The method is designed to operate on a z-stack of brightfield images taken at different focus planes. Background pixels will have relatively uniform illumination whereas cytoplasm pixels will have higher variance across the z-stack.
- *Power:* Compute the power at a given frequency at each pixel position.
  The power method is experimental. The method computes the power at a given frequency through the z-stack. It might be used with a phase contrast image where the signal at a given pixel will vary sinusoidally with depth. The frequency is measured in z-stack steps and pixels that vary with the given frequency will have a higher score than other pixels with similar variance, but different frequencies.

- *Brightfield:* Perform the brightfield projection at each pixel position.
  Artifacts such as dust appear as black spots which are most strongly resolved at their focal plane with gradually increasing signals below. The brightfield method scores these as zero since the dark appears in the early z-stacks. These pixels have a high score for the variance method but have a reduced score when using the brightfield method.
- *Mask:* Compute a binary image of the pixels that are masked in any of the input images.
  The mask method operates on any masks that might have been applied to the images in a group. The output is a binary image where the "1" pixels are those that are not masked in all of the images and the "0" pixels are those that are masked in one or more of the images.
  You can use the output of the mask method to mask or crop all of the images in a group similarly. Use the mask method to combine all of the masks in a group, save the image and then use **Crop**, **MaskImage** or **MaskObjects** in another pipeline to mask all images or objects in the group similarly.

References

- Selinummi J, Ruusuvuori P, Podolsky I, Ozinsky A, Gold E, et al. (2009) "Bright field microscopy as an alternative to whole cell fluorescence in automated analysis of macrophage images", *PLoS ONE* 4(10): e7497 (link).

Name the output image

Enter the name for the projected image.

Frequency

(Used only if Power is selected as the projection method)
This setting controls the frequency at which the power is measured. A frequency of 2 will respond most strongly to pixels that alternate between dark and light in successive z-stack slices. A frequency of N will respond most strongly to pixels whose brightness cycle every N slices.

Module: MaskImage

Mask Image hides certain portions of an image (based on previously identified objects or a binary image) so they are ignored by subsequent mask-respecting modules in the pipeline.

This module masks an image and saves it in the handles structure for future use. The masked image is based on the original image and the masking object or image that is selected. If using a masking image, the mask is composed of the foreground (white portions); if using a masking object, the mask is composed of the area within the object.

Note that the image created by this module for further processing downstream is grayscale. If a binary mask is desired in subsequent modules, use the **ApplyThreshold** module instead of **MaskImage**.

See also **ApplyThreshold**, **IdentifyPrimaryObjects**, **IdentifyObjectsManually**.

Settings:

Select the input image

Select the image that you want to mask.

Name the output image

Enter the name for the output masked image.

Use objects or an image as a mask?

You can mask an image in two ways:

- *Objects*: Using objects created by another module (for instance **IdentifyPrimaryObjects**). The module will mask out all parts of the image that are not within one of the objects (unless you invert the mask).
- *5(IO_IMAGE)s*: Using a binary image as the mask, where black portions of the image (false or zero-value pixels) will be masked out. If the image is not binary, the module will use all pixels whose intensity is greater than 0.5 as the mask's foreground (white area). You can use **ApplyThreshold** instead to create a binary image and have finer control over the intensity choice.

Select object for mask

(Used only if mask is to be made from objects)
Select the objects you would like to use to mask the input image.

Select image for mask

(Used only if mask is to be made from an image)
Select the image that you like to use to mask the input image.

Invert the mask?

This option reverses the foreground/background relationship of the mask.

- Select *No* to produce the mask from the foregound (white portion) of the masking image or the area within the masking objects.
- Select *Yes*to instead produce the mask from the *background* (black portions) of the masking image or the area *outside* the masking objects.

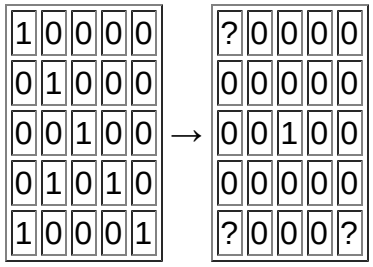
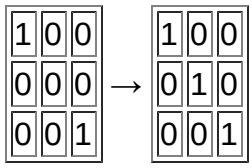
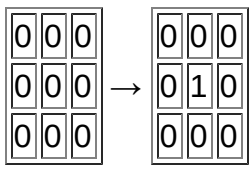
Module: Morph

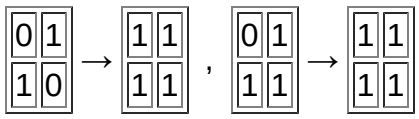
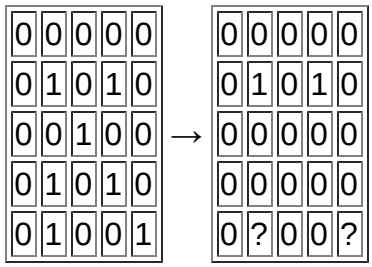
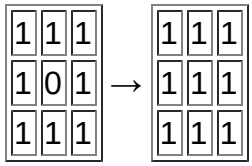
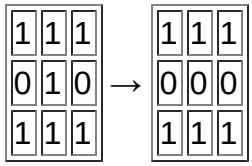
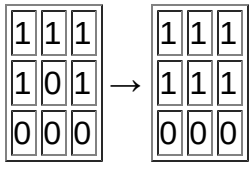
Morph performs low-level morphological operations on binary or grayscale images

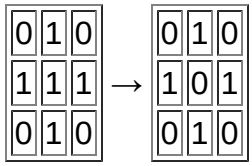
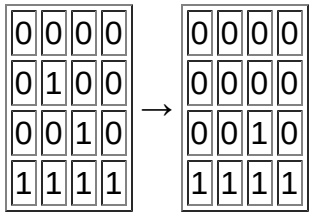
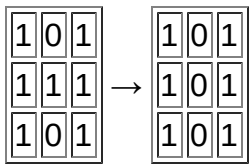
---

This module performs a series of morphological operations on a binary image or grayscale image, resulting in an image of the same type. Many require some image processing knowledge to understand how best to use these morphological filters in order to achieve the desired result. Note that the algorithms minimize the interference of masked pixels; for instance, the dilate operation will only consider unmasked pixels in the neighborhood of a pixel when determining the maximum within that neighborhood.

The following operations are available:

Operation	**Description**	**Input image type allowed**
Bothat	Bottom-hat filter: A bottom-hat filter enhances black spots in a white background. It subtracts the morphological *Close* of the image from the image. See below for a description of *Close*.	Binary, grayscale
Branchpoints	Removes all pixels except those that are the branchpoints of a skeleton. This operation should be applied to an image after skeletonizing. It leaves only those pixels that are at the intersection of branches. 	Binary
Bridge	Sets a pixel to 1 if it has two non-zero neighbors that are on opposite sides of this pixel: 	Binary
Clean	Removes isolated pixels: 	Binary
Close	Performs a dilation followed by an erosion. The effect is to fill holes and join nearby objects.	Binary, grayscale
Convex hull	Finds the convex hull of a binary image. The convex hull is the smallest convex polygon that fits around all foreground pixels of the image: it is the shape that a rubber band would take if stretched around the foreground pixels. The convex hull can be used to regularize the boundary of a large, single object in an image, for instance, the edge of a well.	Binary

Diag	Fills in pixels whose neighbors are diagnonally connected to 4-connect pixels that are 8-connected: 	Binary
Dilate	For binary, replaces any 0 pixel by 1 if any of its neighbors is 1. For grayscale, each pixel is replaced by the maximum of its neighbors and itself.	Binary, grayscale
Distance	Computes the distance transform of a binary image. The distance of each foreground pixel is computed to the nearest background pixel. The resulting image is then scaled so that the largest distance is 1.	Binary
Erode	For binary, replaces any 1 pixel by 0 if any of its neighbors is 0. For grayscale, each pixel is replaced by the minimum of its neighbors and itself.	Binary, grayscale
Endpoints	Removes all pixels except the ones that are at the end of a skeleton: 	Binary
Fill	Sets a pixel to 1 if all of its neighbors are 1: 	Binary
Fill small holes	Sets background pixels surrounded by foreground pixels to 1. This operation fills in small holes in a binary image. You can set the maximum area of a hole in order to restrict the operation to holes of a given size or smaller.	Binary
Hbreak	Removes pixels that form vertical bridges between horizontal lines: 	Binary
Invert	For a binary image, transforms background to foreground and vice-versa. For a grayscale image, invert its intensity.	Binary, Grayscale
Majority	Each pixel takes on the value of the majority that surround it (keep pixel value to break ties): 	Binary
Life	Applies the interaction rules from the Game of Life, an example of a cellular automaton.	Binary
Open	Performs an erosion followed by a dilation. The effect is to break bridges between objects and remove single pixels.	Binary, grayscale
	Removes pixels that are otherwise surrounded by others (4 connected). The	

Remove	effect is to leave the perimeter of a solid object: 	Binary
Shrink	Performs a thinning operation that erodes unless that operation would change the image's Euler number. This means that blobs are reduced to single points and blobs with holes are reduced to rings if shrunken indefinitely.	Binary
Skel	Performs a skeletonizing operation (medial axis transform). Preserves the points at the edges of objects but erodes everything else to lines that connect those edges. See here for a description.	Binary
SkelPE	Performs a skeletonizing operation using the metric, PE * D to control the erosion order. PE is the Poisson Equation (see Gorelick, "Shape representation and classification using the Poisson Equation", IEEE Transactions on Pattern Analysis and Machine Intelligence V28, # 12, 2006) evaluated within the foreground with the boundary condition that the background is zero. D is the distance transform (distance of a pixel to the nearest edge). The resulting skeleton has fewer spurs but some bit of erosion at the endpoints in the binary image.	Binary
Spur	Removes spur pixels, i.e., pixels that have exactly one 8-connected neighbor. This operation essentially removes the endpoints of lines. 	Binary
Thicken	Dilates the exteriors of objects where that dilation does not 8-connect the object with another. The image is labeled and the labeled objects are filled. Unlabeled points adjacent to uniquely labeled points change from background to foreground.	Binary
Thin	Thin lines preserving the Euler number using the thinning algorithm # 1 described in Guo, "Parallel Thinning with Two Subiteration Algorithms", *Communications of the ACM,* Vol 32 #3, page 359. The result generally preserves the lines in an image while eroding their thickness.	Binary
Tophat	Subtracts the morphological opening of the image from the image. This enhances white spots in a black background.	Binary, grayscale
Vbreak	Removes pixels that form horizontal bridges between vertical lines: 	Binary

Settings:

Select the input image

Select the image that you want to perform a morphological operation on. A grayscale image can be

converted to binary using the **ApplyThreshold** module. Objects can be converted to binary using the **ConvertToImage** module.

Name the output image

Enter the name for the output image It will be of the same type as the input image.

Select the operation to perform

Choose one of the operations described in this module's help.

Number of times to repeat operation

This setting controls the number of times that the same operation is applied successively to the image.

- *Once:* Perform the operation once on the image.
- *Forever:* Perform the operation on the image until successive iterations yield the same image.
- *Custom:* Perform the operation a custom number of times.

Repetition number

(Used only if Custom selected)
Enter the number of times to repeat the operation

Scale

Morphological open, close, erode and dialate are performed with structuring elements which determine the diameter of the circle enclosing the pixels to consider when applying the operation. This setting controls the diameter of the structuring element.

Structuring element

(Used only for bothat, close, dilate, erode, open and tophat)
The structuring element controls which neighboring pixels participate in the operation. For instance, for the erode operation, all pixels in the neighborhood of the pixel must be in the foreground for the pixel to be in the foreground in the output image. If a circular structuring element is used, then a pixel will be in the foreground only if all neighborhood pixels within a circle surrounding the pixel are in the foreground in the input image.

The structuring elements are:

- *Disk*: A disk centered on the pixel. The scale setting determines the circle's diameter and all pixels that are at or closer than that diameter will be in the neighborhood.
- *Custom*: A structuring element which lets the user choose the exact neighborhood pixels to use.
- *Diamond*: A diamond centered on the pixel. The scale setting determines the distance between the top and bottom and left and right corners of the diamond.
- *Line*: A line centered on the pixel. The line has two settings. The angle setting gives the rotation of the line in the counter-clockwise direction in degrees, with a horizontal line having an angle of zero. The length of the line is determined by the scale setting - only pixels at or closer than 1/2 of the scale are included in the neighborhood. The line is drawn using the Bresenham algorithm.
- *Octagon*: An octagon centered on the pixel. The octagon is inscribed inside a square. The scale setting controls the length of the square's side. The scale is rounded to the nearest integer in the

series, n * 6 + 1 so a perfect octagon can be drawn.
- *Pair*: The neighborhood of the pixel is composed of the pixel itself and the pixel at the x and y offsets given by the settings.
- *Periodic line*: The points along a line described by an offset, centered on the pixel. The periodic line has three settings. The neighborhood pixels are all within a circle whose diameter is the scale setting. Within the circle, pixels are chosen at N times the x and y offset from the center for positive and negative values of N.
- *Rectangle*: A rectangle centered on the pixel. The rectangle's height and width are given by two settings.
- *Square*: a square centered on the pixel. The scale setting determines the length of the square's side.

X offset

(Used only for the Pair and Periodic line settings). The X offset to the first neighborhood pixel in the structuring element.

Y offset

(Used only for the Pair and Periodic line structuring elements). The Y offset to the first neighborhood pixel in the structuring element.

Angle

(Used only for the Line structuring element). The angle, in degrees counter-clockwise from the horizontal, of the line.

Width

(Used only for the Rectangle structuring element). The width of the rectangle in pixels.

Height

(Used only for the Rectangle structuring element). The height of the rectangle in pixels.

Custom

(Used only for the Custom structuring element). This control lets you specify a custom structuring element.

Rescale values from 0 to 1?

(Used only for the distance operation).

Select *Yes* to rescale the transformed values to lie between 0 and 1. This is the option to use if the distance transformed image is to be used for thresholding by an **Identify** module or the like, which assumes a 0-1 scaling.

Select *No* to leave the values in absolute pixel units. This useful in cases where the actual pixel distances are to be used downstream as input for a measurement module.

Module: OverlayOutlines

Overlay Outlines places outlines produced by an **Identify** module over a desired image.

This module places outlines (in a special format produced by an **Identify** module) on any desired image (grayscale, color, or blank). The resulting image can be saved using the **SaveImages** module.

See also **IdentifyPrimaryObjects, IdentifySecondaryObjects, IdentifyTertiaryObjects**.

Settings:

Display outlines on a blank image?

Select *Yes* to produce an image of the outlines on a black background.

Select *No*, the module will overlay the outlines on an image of your choosing.

Select image on which to display outlines

(Used only when a blank image has not been selected)
Choose the image to serve as the background for the outlines. You can choose from images that were loaded or created by modules previous to this one.

Name the output image

Enter the name of the output image with the outlines overlaid. This image can be selected in later modules (for instance, **SaveImages**).

Outline display mode

Specify how to display the outline contours around your objects. Color outlines produce a clearer display for images where the cell borders have a high intensity, but take up more space in memory. Grayscale outlines are displayed with either the highest possible intensity or the same intensity as the brightest pixel in the image.

Select method to determine brightness of outlines

(Used only when outline display mode is grayscale)
The following options are possible for setting the intensity (brightness) of the outlines:

- *Max of image:* Set the brighness to the the same as the brightest point in the image.
- *Max possible:* Set to the maximum possible value for this image format.

If your image is quite dim, then putting bright white lines onto it may not be useful. It may be preferable to make the outlines equal to the maximal brightness already occurring in the image.

Width of outlines

Enter the width, in pixels, of the outlines to be displayed on the image.

Select outlines to display

Choose outlines to display, from a previous **Identify** module. Each of the **Identify** modules has a checkbox that determines whether the outlines are saved. If you have checked this, you were asked to supply a name for the outline; you can then select that name here.

Load outlines from an image or objects?

Prior versions of **OverlayOutlines** would only display outline images which were optional outputs of the identify modules. For legacy pipelines or to continue using the outline images instead of objects, choose *Image*. Choose *Objects* to create the image directly from the objects. This option will improve the functionality of the contrast options for this module's interactive display and will save memory.

Select objects to display

Choose the objects whose outlines you would like to display.

Module: RescaleIntensity

RescaleIntensity changes the intensity range of an image to your desired specifications.

This module lets you rescale the intensity of the input images by any of several methods. You should use caution when interpreting intensity and texture measurements derived from images that have been rescaled because certain options for this module do not preserve the relative intensities from image to image.

Settings:

Select the input image

Select the image to be rescaled.

Name the output image

Enter the name of output rescaled image.

Rescaling method

There are a number of options for rescaling the input image:

- *Stretch each image to use the full intensity range:* Find the minimum and maximum values within the unmasked part of the image (or the whole image if there is no mask) and rescale every pixel so that the minimum has an intensity of zero and the maximum has an intensity of one.
- *Choose specific values to be reset to the full intensity range:* Pixels are scaled from their user-specified original range to the range 0 to 1. Options are available to handle values outside of the original range.
  To convert 12-bit images saved in 16-bit format to the correct range, use the range 0 to 0.0625. The value 0.0625 is equivalent to 2^{12} divided by 2^{16}, so it will convert a 16 bit image containing only 12 bits of data to the proper range.
- *Choose specific values to be reset to a custom range:* Pixels are scaled from their original range to the new target range. Options are available to handle values outside of the original range.
- *Divide by the image's minimum:* Divide the intensity value of each pixel by the image's minimum intensity value so that all pixel intensities are equal to or greater than 1. The rescaled image can serve as an illumination correction function in **CorrectIlluminationApply**.
- *Divide by the image's maximum:* Divide the intensity value of each pixel by the image's maximum intensity value so that all pixel intensities are less than or equal to 1.
- *Divide each image by the same value:* Divide the intensity value of each pixel by the value entered.
- *Divide each image by a previously calculated value:* The intensity value of each pixel is divided by some previously calculated measurement. This measurement can be the output of some other module or can be a value loaded by the **Metadata** module.
- *Match the image's maximum to another image's maximum:* Scale an image so that its maximum value is the same as the maximum value within the reference image.
- *Convert to 8 bit:* Images in CellProfiler are normally stored as a floating point number in the range of 0 to 1. This option converts these images to class uint8, meaning an 8 bit integer in the range of 0 to 255, reducing the amount of memory required to store the image. *Warning:* Most CellProfiler modules require the incoming image to be in the standard 0 to 1 range, so this conversion may cause downstream modules to behave in unexpected ways.

Method to calculate the minimum intensity

(Used only if "Choose specific values to be reset to a custom range" is selected)
This setting controls how the minimum intensity is determined.

- *Custom:* Enter the minimum intensity manually below.
- *Minimum for each image*: use the lowest intensity in this image as the minimum intensity for rescaling
- *Minimum of all images*: use the lowest intensity from all images in the image group or the experiment if grouping is not being used. **Note:** Choosing this option may have undesirable results for a large ungrouped experiment split into a number of batches. Each batch will open all images from the chosen channel at the start of the run. This sort of synchronized action may have a severe impact on your network file system.

Method to calculate the maximum intensity

(Used only if "Choose specific values to be reset to a custom range" is selected)
This setting controls how the maximum intensity is determined.

- *Custom*: Enter the maximum intensity manually below.
- *Maximum for each image*: Use the highest intensity in this image as the maximum intensity for rescaling
- *Maximum of all images*: Use the highest intensity from all images in the image group or the experiment if grouping is not being used. **Note:** Choosing this option may have undesirable results for a large ungrouped experiment split into a number of batches. Each batch will open all images from the chosen channel at the start of the run. This sort of synchronized action may have a severe impact on your network file system.

Method to rescale pixels below the lower limit

(Used only if "Choose specific values to be reset to a custom range" is selected)
There are several ways to handle values less than the lower limit of the intensity range:

- *Mask pixels:* Creates a mask for the output image. All pixels below the lower limit will be masked out.
- *Set to zero:* Sets all pixels below the lower limit to zero.
- *Set to custom value:* Sets all pixels below the lower limit to a custom value.
- *Scale similarly to others:* Scales pixels with values below the lower limit using the same offset and divisor as other pixels. The results will be less than zero.

Custom value for pixels below lower limit

(Used only if "Choose specific values to be reset to a custom range" and "Set to custom value are selected)
enter the custom value to be assigned to pixels with values below the lower limit.

Method to rescale pixels above the upper limit

(Used only if "Choose specific values to be reset to a custom range" is selected)
There are several ways to handle values greater than the upper limit of the intensity range; Options are described in the Help for the equivalent lower limit question.

Custom value for pixels above upper limit

(Used only if "Choose specific values to be reset to a custom range" and "Set to custom value are selected)
Enter the custom value to be assigned to pixels with values above the upper limit.

Select image to match in maximum intensity

(Used only if "Match the image's maximum to another image's maximum" is selected)
Select the image whose maximum you want the rescaled image to match.

Divisor value

(Used only if "Divide each image by the same value" is selected)
Enter the value to use as the divisor for the final image.

Divisor measurement

(Used only if "Divide each image by a previously calculated value" is selected)
Select the measurement value to use as the divisor for the final image.

Module: Resize

Resize resizes images (changes their resolution).

Images are resized (made smaller or larger) based on user input. You can resize an image by applying a resizing factor or by specifying the desired dimensions, in pixels. You can also select which interpolation method to use.

Settings:

Select the input image

Select the image to be resized.

Name the output image

Enter the name of the resized image.

Resizing method

The following options are available:

- *Resize by a fraction or multiple of the original size:* Enter a single value which specifies the scaling.
- *Resize by specifying desired final dimensions:*
  Enter the new height and width of the resized image.

Resizing factor

(Used only if resizing by a fraction or multiple of the original size)
Numbers less than one (that is, fractions) will shrink the image; numbers greater than one (that is, multiples) will enlarge the image.

Width of the final image

(Used only if resizing by specifying desired final dimensions)
Enter the desired width of the final image, in pixels.

Height of the final image

(Used only if resizing by specifying desired final dimensions)
Enter the desired height of the final image, in pixels.

Interpolation method

- *Nearest Neighbor:* Each output pixel is given the intensity of the nearest corresponding pixel in the input image.
- *Bilinear:* Each output pixel is given the intensity of the weighted average of the 2x2 neighborhood at the corresponding position in the input image.
- *Bicubic:* Each output pixel is given the intensity of the weighted average of the 4x4 neighborhood at

the corresponding position in the input image.

Method to specify the dimensions

(Used only if resizing by specifying the dimensions)
You have two options on how to resize your image:

- *Manual:* Specify the height and width of the output image.
- *>Image::* Specify an image and the input image will be resized to the same dimensions.

Select the image with the desired dimensions

" *(Used only if resizing by specifying desired final dimensions using an image)*
The input image will be resized to the dimensions of the specified image.

Module: RunImageJ

Run ImageJ runs an ImageJ command.

ImageJ is an image processing and analysis program. It operates by processing commands that operate on one or more images, possibly modifying the images. ImageJ has a macro language which can be used to program its operation and customize its operation, similar to CellProfiler pipelines. ImageJ maintains a current image and most commands operate on this image, but it's possible to load multiple images into ImageJ and operate on them together.

The **RunImageJ** module runs one ImageJ command or macro per cycle. It first loads the images you want to process into ImageJ, then runs the command, and, if desired, retrieves images you want to process further in CellProfiler.

Technical notes

ImageJ runs using Java, and as such, relies on proper handling of the Java memory requirements. When ImageJ starts, the Java Virtual Machine (JVM) allocates a portion of memory for its own use from the operating system; this memory is called the *java heap memory*. If you encounter JVM memory errors, you can tell CellProfiler to increase the size of the Java heap memory on startup.

To do this, run CellProfiler from the command line with the following argument: `--jvm-heap-size=JVM_HEAP_SIZE`
where `JVM_HEAP_SIZE` is the amount of memory to be reserved for the JVM. Example formats for `JVM_HEAP_SIZE` include *512000k*, *512m*, *1g*, etc. For example, to increase the JVM heap memory to 2GB, use `--jvm-heap-size=2g`

Settings:

Run an ImageJ command or macro?

This setting determines whether **RunImageJ** runs either a:

- *Command:* Select from a list of available ImageJ commands (those items contained in the ImageJ menus); or
- *Script:* A script written in one of ImageJ 2.0's supported scripting languages.
- *Macro:* An ImageJ 1.x macro, written in the ImageJ 1.x macro language. **Run_ImageJ** runs ImageJ in 1.x compatability mode.

Command

(Used only if running a Command)
The command to execute when the module runs.

Macro

(Used only if running a Macro)
This is the ImageJ macro to be executed. The syntax for ImageJ macros depends on the scripting language engine chosen. We suggest that you use the Beanshell scripting language (Beanshell documentation).

Macro language

This setting chooses the scripting language used to execute any macros in this module

Input the currently active image in ImageJ?

Select *Yes* if you want to set the currently active ImageJ image using an image from a prior CellProfiler module.

Select *No* to use the currently active image in ImageJ. You may want to do this if you have an output image from a prior **RunImageJ** that you want to perform further operations upon before retrieving the final result back to CellProfiler.

Select the input image

(Used only if setting the currently active image)
This is the CellProfiler image that will become ImageJ's currently active image. The ImageJ commands and macros in this module will perform their operations on this image. You may choose any image produced by a prior CellProfiler module.

Retrieve the currently active image from ImageJ?

Select *Yes* if you want to retrieve ImageJ's currently active image after running the command or macro.

Select *No* if the pipeline does not need to access the current ImageJ image. For example, you might want to run further ImageJ operations with additional **RunImageJ** upon the current image prior to retrieving the final image back to CellProfiler.

Name the current output image

(Used only if retrieving the currently active image from ImageJ)
This is the CellProfiler name for ImageJ's current image after processing by the command or macro. The image will be a snapshot of the current image after the command has run, and will be available for processing by subsequent CellProfiler modules.

Wait for ImageJ before continuing?

Some ImageJ commands and macros are interactive; you may want to adjust the image in ImageJ before continuing. Select *Yes* to stop CellProfiler while you adjust the image in ImageJ. Select *No* to immediately use the image.

This command will not wait if CellProfiler is executed in batch mode. See *Help* > *Other Features* > *Batch Processing* for more details on batch processing.

Function to run before each group of images?

You can run an ImageJ 2.0 script, an ImageJ 1.x macro or a command *before* each group of images. This can be useful in order to set up ImageJ before processing a stack of images. Choose *Nothing* if you do not want to run a command or macro, *Command* to choose a command to run, *Script* to run an ImageJ 2.0 script or *Macro* to run an ImageJ 1.x macro in ImageJ 1.x compatibility mode.

Command

(Used only if running a command before an image group)
Select the command to execute before processing a group of images.

Macro

(Used only if running a macro before an image group)
This is the ImageJ macro to be executed before processing a group of images. For help on writing macros, see here.

Function to run after each group of images?

You can run an ImageJ 2.0 script, an ImageJ macro or a command *after* each group of images. This can be used to do some sort of operation on a whole stack of images that have been accumulated by the group operation. Choose *Nothing* if you do not want to run a command or macro, *Command* to choose a command to run, *Script* to run an ImageJ 2.0 script or *Macro* to run an ImageJ 1.x macro in ImageJ 1.x compatibility mode.

Command

(Used only if running a command after an image group)
The command to execute after processing a group of images.

Macro

(Used only if running a macro after an image group)
This is the ImageJ macro to be executed after processing a group of images. For help on writing macros, see here.

Retrieve the image output by the group operation?

You can retrieve the image that is currently active in ImageJ at the end of macro processing and use it later in CellProfiler. The image will only be available during the last cycle of the image group.

Select *Yes* to retrieve the active image for use in CellProfiler. Select *No* if you do not want to retrieve the active image.

Name the group output image

(Used only if retrieving an image after an image group operation)
This setting names the output image produced by the ImageJ command or macro that CellProfiler runs after processing all images in the group. The image is only available at the last cycle in the group

Module: Smooth

Smooth smooths (i.e., blurs) images.

This module allows you to smooth (blur) images, which can be helpful to remove artifacts of a particular size. Note that smoothing can be a time-consuming process.

Settings:

Select smoothing method

This module smooths images using one of several filters. Fitting a polynomial is fastest but does not allow a very tight fit compared to the other methods:

- *Fit Polynomial:* This method treats the intensity of the image pixels as a polynomial function of the x and y position of each pixel. It fits the intensity to the polynomial, $A\,x^2 + B\,y^2 + C\,xy + D\,x + E\,y + F$. This will produce a smoothed image with a single peak or trough of intensity that tapers off elsewhere in the image. For many microscopy images (where the illumination of the lamp is brightest in the center of field of view), this method will produce an image with a bright central region and dimmer edges. But, in some cases the peak/trough of the polynomial may actually occur outside of the image itself.
- *Gaussian Filter:* This method convolves the image with a Gaussian whose full width at half maximum is the artifact diameter entered. Its effect is to blur and obscure features smaller than the artifact diameter and spread bright or dim features larger than the artifact diameter.
- *Median Filter:* This method finds the median pixel value within the artifact diameter you specify. It removes bright or dim features that are much smaller than the artifact diameter.
- *Smooth Keeping Edges:* This method uses a bilateral filter which limits Gaussian smoothing across an edge while applying smoothing perpendicular to an edge. The effect is to respect edges in an image while smoothing other features. *Smooth Keeping Edges* will filter an image with reasonable speed for artifact diameters greater than 10 and for intensity differences greater than 0.1. The algorithm will consume more memory and operate more slowly as you lower these numbers.
- *Circular Average Filter:* This method convolves the image with a uniform circular averaging filter whose size is the artifact diameter entered. This filter is useful for re-creating an out-of-focus blur to an image.
- *Smooth to Average:* Creates a flat, smooth image where every pixel of the image equals the average value of the original image.

Calculate artifact diameter automatically?

(Used only if "Gaussian Filter", "Median Filter", "Smooth Keeping Edges" or "Circular Average Filter" is selected)
Select *Yes* to choose an artifact diameter based on the size of the image. The minimum size it will choose is 30 pixels, otherwise the size is 1/40 of the size of the image.

Select *Yes* to manually enter an artifact diameter.

Typical artifact diameter

(Used only if choosing the artifact diameter automatically is set to "No")
Enter the approximate diameter (in pixels) of the features to be blurred by the smoothing algorithm. This

value is used to calculate the size of the spatial filter. To measure distances in an open image, use the "Measure length" tool under *Tools* in the display window menu bar. If you click on an image and drag, a line will appear between the two endpoints, and the distance between them shown at the right-most portion of the bottom panel. For most smoothing methods, selecting a diameter over ~50 will take substantial amounts of time to process.

Edge intensity difference

(Used only if "Smooth Keeping Edges" is selected)
Enter the intensity step (which indicates an edge in an image) that you want to preserve. Edges are locations where the intensity changes precipitously, so this setting is used to adjust the rough magnitude of these changes. A lower number will preserve weaker edges. A higher number will preserve only stronger edges. Values should be between zero and one. To view pixel intensities in an open image, use the pixel intensity tool which is available in any open display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window.

Clip intensities to 0 and 1?

(Used only if Fit Polynomial is selected)
The *Fit Polynomial* method is the only smoothing option that can yield an output image whose values are outside of the values of the input image. This setting controls whether to limit the image intensity to the 0 - 1 range used by CellProfiler.

Select *Yes* to set all output image pixels less than zero to zero and all pixels greater than one to one.

Select *No* to allow values less than zero and greater than one in the output image.

Module: Tile

Tile tiles images together to form large montage images.

This module allows more than one image to be placed next to each other in a grid layout you specify. It might be helpful, for example, to place images adjacent to each other when multiple fields of view have been imaged for the same sample. Images can be tiled either across cycles (multiple fields of view, for example) or within a cycle (multiple channels of the same field of view, for example).

Tiling images to create a montage with this module generates an image that is roughly the size of all the images' sizes added together. For large numbers of images, this may cause memory errors, which might be avoided by the following suggestions:

- Resize the images to a fraction of their original size, using the **Resize** module prior to this module in the pipeline.
- Rescale the images to 8-bit using the **RescaleIntensity** module, which diminished image quality by decreasing the number of graylevels in the image (that is, bit depth) but also decreases the size of the image.
- Use the **ConserveMemory** module just before this module to clear out images created previously in the pipeline that are stored in memory but no longer needed. Place this module prior to the **Tile** module (and maybe also afterwards) and set it to retain only those images that are needed for downstream modules.

Please also note that this module does not perform *image stitching* (i.e., intelligent adjustment of the alignment between adjacent images). For image stitching, you may find the following list of software packages useful:

- Photomerge Feature in Photoshop CS
- PTGui
- Autostitch
- ImageJ with the MosaicJ plugin

Other packages are referenced here

Settings:

Select an input image

Select the image to be tiled. Additional images within the cycle can be added later by choosing the "*Across cycles*"option below.

Name the output image

Enter a name for the final tiled image.

Tile assembly method

This setting controls the method by which the final tiled image is asembled:

- *Within cycles:* If you have loaded more than one image for each cycle using modules upstream in the pipeline, the images can be tiled. For example, you may tile three different channels (OrigRed,

OrigBlue, and OrigGreen), and a new tiled image will be created for every image cycle.
- *Across cycles:* If you want to tile images from multiple cycles together, select this option. For example, you may tile all the images of the same type (e.g., OrigBlue) across all fields of view in your experiment, which will result in one final tiled image when processing is complete.

Final number of rows

Specify the number of rows would you like to have in the tiled image. For example, if you want to show your images in a 96-well format, enter 8.

Special cases: Let *M* be the total number of slots for images (i.e, number of rows x number of columns) and *N* be the number of actual images.

- If *M* > *N*, blanks will be used for the empty slots.
- If the *M* < *N*, an error will occur since there are not enough image slots. Check "Automatically calculate number of rows?" to avoid this error.

Final number of columns

Specify the number of columns you like to have in the tiled image. For example, if you want to show your images in a 96-well format, enter 12.

Special cases: Let *M* be the total number of slots for images (i.e, number of rows x number of columns) and *N* be the number of actual images.

- If *M* > *N*, blanks will be used for the empty slots.
- If the *M* < *N*, an error will occur since there are not enough image slots. Check "Automatically calculate number of columns?" to avoid this error.

Image corner to begin tiling

Where do you want the first image to be placed? Begin in the upper left-hand corner for a typical multi-well plate format where the first image is A01.

Direction to begin tiling

This setting specifies the order that the images are to be arranged. If your images are named A01, A02, etc, enter *row*".

Use meander mode?

Select *Yes* to tile adjacent images in one direction, then the next row/column is tiled in the opposite direction. Some microscopes capture images in this fashion. The default mode is "comb", or "typewriter" mode; in this mode, when one row is completely tiled in one direction, the next row starts near where the first row started and tiles again in the same direction.

Automatically calculate number of rows?

Tile can automatically calculate the number of rows in the grid based on the number of image cycles that will be processed. Select *Yes* to create a grid that has the number of columns that you entered and enough rows to display all of your images. Select *No* to specify the number of rows.

If you check both automatic rows and automatic columns, **Tile** will create a grid that has roughly the same number of rows and columns.

Automatically calculate number of columns?

Tile can automatically calculate the number of columns in the grid from the number of image cycles that will be processed. Select *Yes* to create a grid that has the number of rows that you entered and enough columns to display all of your images. Select *No* to specify the number of rows.

If you check both automatic rows and automatic columns, **Tile** will create a grid that has roughly the same number of rows and columns.

Module: UnmixColors

Unmix Colors creates separate images per dye stain for histologically stained images.

This module creates separate grayscale images from a color image stained with light-absorbing dyes. Dyes are assumed to absorb an amount of light in the red, green and blue channels that increases proportionally in each channel with increasing amounts of stain; the hue does not shift with increasing staining.

The module separates two or more stains from a background, producing grayscale images. There are several pre-set dye combinations as well as a custom mode that allows a user to calibrate using two images stained with a single dye each.

Some commonly known stains must be specified by the individual dye components. For example:

- Azan-Mallory: Anilline Blue + Azocarmine + Orange-G
- Giemsa: Methylene Blue or Eosin
- Masson Trichrome: Methyl blue + Ponceau-Fuchsin

If there are non-stained cells/components that you also want to separate by color, choose the stain that closest resembles the color you want, or enter a custom value.

Please note that if you are looking to simply split a color image into red, green and blue components, use the **ColorToGray** module rather than **UnmixColors**.

Technical notes

This code is adapted from the ImageJ plugin, *Colour_Deconvolution.java* (described here) written by A.C. Ruifrok, whose paper forms the basis for this code.

References

- Ruifrok AC, Johnston DA. (2001) "Quantification of histochemical staining by color deconvolution." *Analytical & Quantitative Cytology & Histology*, 23: 291-299.

See also **ColorToGray**.

Settings:

Select the input color image

Choose the name of the histologically stained color image loaded or created by some prior module.

Name the output name

Use this setting to name one of the images produced by the module for a particular stain. The image can be used in subsequent modules in the pipeline.

Stain

Use this setting to choose the absorbance values for a particular stain. The stains are:

Stain	Color	Specific to
AEC (3-Amino-9-ethylcarbazole)		Peroxidase
Alican blue		Mucopolysaccharides
Aniline blue		Pollen tubes
Azocarmine		Plasma
DAB		Peroxisomes, mitochondria
Eosin		Elastic, collagen and reticular fibers
Fast red		Nuclei
Fast blue		Myelin fibers
Feulgen		DNA
Hematoxylin		Nucleic acids, endoplasmic reticulum
Hematoxylin and PAS		Nucleus (stained with both Hematoxylin and PAS)
Methyl blue		Collagen
Methyl green		Chromatin
Methylene blue		Nuclei
Orange-G		Erythrocytes, pancreas, pituitary
PAS		Glycogen, carbohydrates
Ponceau-fuchsin		Red counterstain for Masson's trichrome

(Information taken from here, here, and here.)
You can choose *Custom* and enter your custom values for the absorbance (or use the estimator to determine values from a single-stain image).

Red absorbance

(Used only if Custom is selected for the stain)
The red absorbance setting estimates the dye's absorbance of light in the red channel.You should enter a value between 0 and 1 where 0 is no absorbance and 1 is complete absorbance. You can use the estimator to calculate this value automatically.

Green absorbance

(Used only if Custom is selected for the stain)
The green absorbance setting estimates the dye's absorbance of light in the green channel. You should enter a value between 0 and 1 where 0 is no absorbance and 1 is complete absorbance. You can use the estimator to calculate this value automatically.

Blue absorbance

(Used only if Custom is selected for the stain)
The blue absorbance setting estimates the dye's absorbance of light in the blue channel. You should enter a value between 0 and 1 where 0 is no absorbance and 1 is complete absorbance. You can use the estimator to calculate this value automatically.

Module: CalculateImageOverlap

Calculate Image Overlap calculates how much overlap occurs between the white portions of two black and white images

---

This module calculates overlap by determining a set of statistics that measure the closeness of an image or object to its' true value. One image/object is considered the "ground truth" (possibly the result of hand-segmentation) and the other is the "test" image/object; the images are determined to overlap most completely when the test image matches the ground truth perfectly. If using images, the module requires binary (black and white) input, where the foreground is white and the background is black. If you segment your images in CellProfiler using **IdentifyPrimaryObjects**, you can create such an image using **ConvertObjectsToImage** by selecting *Binary* as the color type.

If your images have been segmented using other image processing software, or you have hand-segmented them in software such as Photoshop, you may need to use one or more of the following to prepare the images for this module:

- **ImageMath**: If the objects are black and the background is white, you must invert the intensity using this module.
- **ApplyThreshold**: If the image is grayscale, you must make it binary using this module, or alternately use an **Identify** module followed by **ConvertObjectsToImage** as described above.
- **ColorToGray**: If the image is in color, you must first convert it to grayscale using this module, and then use **ApplyThreshold** to generate a binary image.

In the test image, any foreground (white) pixels that overlap with the foreground of the ground truth will be considered "true positives", since they are correctly labeled as foreground. Background (black) pixels that overlap with the background of the ground truth image are considered "true negatives", since they are correctly labeled as background. A foreground pixel in the test image that overlaps with the background in the ground truth image will be considered a "false positive" (since it should have been labeled as part of the background), while a background pixel in the test image that overlaps with foreground in the ground truth will be considered a "false negative" (since it was labeled as part of the background, but should not be).

Available measurements

- **For images and objects:**
  - *True positive rate:* Total number of true positive pixels / total number of actual positive pixels.
  - *False positive rate:* Total number of false positive pixels / total number of actual negative pixels
  - *True negative rate:* Total number of true negative pixels / total number of actual negative pixels.
  - *False negative rate:* Total number of false negative pixels / total number of actual postive pixels
  - *Precision:* Number of true positive pixels / (number of true positive pixels + number of false positive pixels)
  - *Recall:* Number of true positive pixels/ (number of true positive pixels + number of false negative pixels)
  - *F-factor:* 2 × (precision × recall)/(precision + recall). Also known as F_1 score, F-score or F-measure.
- **For objects:**
  - *Rand index:* A measure of the similarity between two data clusterings. Perfectly random

clustering returns the minimum score of 0, perfect clustering returns the maximum score of 1.
    - *Adjusted Rand index:* A variation of the Rand index which takes into account the fact that random chance will cause some objects to occupy the same clusters, so the Rand Index will never actually be zero. Can return a value between -1 and +1.

References

- Collins LM, Dent CW (1998) "Omega: A general formulation of the Rand Index of cluster recovery suitable for non-disjoint solutions", *Multivariate Behavioral Research*, 23, 231-242 (link)

Settings:

Select the image to be used as the ground truth basis for calculating the amount of overlap

(Used only when comparing foreground/background)
This binary (black and white) image is known as the "ground truth" image. It can be the product of segmentation performed by hand, or the result of another segmentation algorithm whose results you would like to compare.

Select the image to be used to test for overlap

(Used only when comparing foreground/background)
This binary (black and white) image is what you will compare with the ground truth image. It is known as the "test image".

Select the objects to be used as the ground truth basis for calculating the amount of overlap

(Used only when comparing segmented objects)
Choose which set of objects will used as the "ground truth" objects. It can be the product of segmentation performed by hand, or the result of another segmentation algorithm whose results you would like to compare. See the **Load** modules for more details on loading objects.

Select the objects to be tested for overlap against the ground truth

(Used only when comparing segmented objects)
This set of objects is what you will compare with the ground truth objects. It is known as the "test object."

Module: MeasureCorrelation

Measure Correlation measures the correlation between intensities in different images (e.g., different color channels) on a pixel-by-pixel basis, within identified objects or across an entire image.

Given two or more images, this module calculates the correlation between the pixel intensities. The correlation can be measured for entire images, or a correlation measurement can be made within each individual object.

Correlations will be calculated between all pairs of images that are selected in the module, as well as between selected objects. For example, if correlations are to be measured for a set of red, green, and blue images containing identified nuclei, measurements will be made between the following:

- The blue and green, red and green, and red and blue images.
- The nuclei in each of the above image pairs.

Available measurements

- *Correlation coefficient:* The correlation between a pair of images *I* and *J*. Calculated as Pearson's correlation coefficient, for which the formula is covariance(*I* ,*J*)/[std(*I*) × std(*J*)].
- *Slope:* The slope of the least-squares regression between a pair of images I and J. Calculated using the model $A \times I + B = J$, w here *A* is the slope.

Settings:

Select an image to measure

Select an image to measure the correlation from.

Select where to measure correlation

You can measure the correlation in several ways:

- *Within objects:* Measure correlation only in those pixels previously identified as an object. You will be asked to specify which object to measure from.
- *Across entire image:* Measure the correlation across all pixels in the images.
- *Both:* Calculate both measurements above.

All methods measure correlation on a pixel by pixel basis.

Select an object to measure

Select the objects to be measured.

Module: MeasureGranularity

Measure Granularity outputs spectra of size measurements of the textures in the image.

Image granularity is a texture measurement that tries a series of structure elements of increasing size and outputs a spectrum of measures of how well these structure elements fit in the texture of the image. Granularity is measured as described by Ilya Ravkin (references below). The size of the starting structure element as well as the range of the spectrum is given as input.

Available measurements

- *Granularity:* The module returns one measurement for each instance of the granularity spectrum.

References

- Serra J. (1989) *Image Analysis and Mathematical Morphology*, Vol. 1. Academic Press, London
- Maragos P. "Pattern spectrum and multiscale shape representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, N 7, pp. 701-716, 1989
- Vincent L. (2000) "Granulometries and Opening Trees", *Fundamenta Informaticae*, 41, No. 1-2, pp. 57-90, IOS Press, 2000.
- Vincent L. (1992) "Morphological Area Opening and Closing for Grayscale Images", *Proc. NATO Shape in Picture Workshop*, Driebergen, The Netherlands, pp. 197-208.
- Ravkin I, Temov V. (1988) "Bit representation techniques and image processing", *Applied Informatics*, v.14, pp. 41-90, Finances and Statistics, Moskow, (in Russian)

Settings:

Select an image to measure

Select the grayscale images whose granularity you want to measure.

Subsampling factor for granularity measurements

If the textures of interest are larger than a few pixels, we recommend you subsample the image with a factor <1 to speed up the processing. Down sampling the image will let you detect larger structures with a smaller sized structure element. A factor >1 might increase the accuracy but also require more processing time. Images are typically of higher resolution than is required for granularity measurements, so the default value is 0.25. For low-resolution images, increase the subsampling fraction; for high-resolution images, decrease the subsampling fraction. Subsampling by 1/4 reduces computation time by $(1/4)^3$ because the size of the image is $(1/4)^2$ of original and the range of granular spectrum can be 1/4 of original. Moreover, the results are sometimes actually a little better with subsampling, which is probably because with subsampling the individual granular spectrum components can be used as features, whereas without subsampling a feature should be a sum of several adjacent granular spectrum components. The recommendation on the numerical value cannot be determined in advance; an analysis as in this reference may be required before running the whole set. See this pdf, slides 27-31, 49-50.

Subsampling factor for background reduction

It is important to remove low frequency image background variations as they will affect the final granularity

measurement. Any method can be used as a pre-processing step prior to this module; we have chosen to simply subtract a highly open image. To do it quickly, we subsample the image first. The subsampling factor for background reduction is usually [0.125 – 0.25]. This is highly empirical, but a small factor should be used if the structures of interest are large. The significance of background removal in the context of granulometry is that image volume at certain granular size is normalized by total image volume, which depends on how the background was removed.

Radius of structuring element

This radius should correspond to the radius of the textures of interest *after* subsampling; i.e., if textures in the original image scale have a radius of 40 pixels, and a subsampling factor of 0.25 is used, the structuring element size should be 10 or slightly smaller, and the range of the spectrum defined below will cover more sizes.

Range of the granular spectrum

You may need a trial run to see which granular spectrum range yields informative measurements. Start by using a wide spectrum and narrow it down to the informative range to save time.

Module: MeasureImageAreaOccupied

Measure Image Area Occupied measures the total area in an image that is occupied by objects.

This module reports the sum of the areas and perimeters of the objects defined by one of the **Identify** modules, or the area of the foreground in a binary image. If the input image has a mask (for example, created by the **MaskImage** module), the measurements made by this module will take the mask into account by ignoring the pixels outside the mask.

You can use this module to measure the number of pixels above a given threshold if you precede it with thresholding performed by **ApplyThreshold**, and then select the binary image output by **ApplyThreshold** to be measured by this module.

Available measurements

- *AreaOccupied:* The total area occupied by the input objects or binary image.
- *Perimeter:* The total length of the perimeter of the input objects/binary image.
- *TotalImageArea:* The total pixel area of the image.

See also **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, **IdentifyTertiaryObjects**

Settings:

Measure the area occupied in a binary image, or in objects?

The area can be measured in two ways:

- *Binary Image:* The area occupied by the foreground in a binary (black and white) image.
- *Objects:* The area occupied by previously-identified objects.

Select objects to measure

(Used only if 'Objects' are to be measured)
Select the previously identified objects you would like to measure.

Retain a binary image of the object regions?

(Used only if 'Objects' are to be measured)
Select *Yes* if you would like to use a binary image later in the pipeline, for example in **SaveImages**. The image will display the object area that you have measured as the foreground in white and the background in black.

Name the output binary image

(Used only if the binary image of the objects is to be retained for later use in the pipeline)
Specify a name that will allow the binary image of the objects to be selected later in the pipeline.

Select a binary image to measure

(Used only if 'Binary Image' is to be measured)

This is a binary image created earlier in the pipeline, where you would like to measure the area occupied by the foreground in the image.

Module: MeasureImageIntensity

Measure Image Intensity measures the total intensity in an image by summing all of the pixel intensities (excluding masked pixels).

This module will sum all pixel values to measure the total image intensity. The user can measure all pixels in the image or can restrict the measurement to pixels within objects. If the image has a mask, only unmasked pixels will be measured.

Note that for publication purposes, the units of intensity from microscopy images are usually described as "Intensity units" or "Arbitrary intensity units" since microscopes are not calibrated to an absolute scale. Also, it is important to note whether you are reporting either the mean or the integrated intensity, so specify "Mean intensity units" or "Integrated intensity units" accordingly.

Keep in mind that the default behavior in CellProfiler is to rescale the image intensity from 0 to 1 by dividing all pixels in the image by the maximum possible intensity value. This "maxmimum possible" value is defined by the "Set intensity range from" setting in **NamesAndTypes**; see the help for that setting for more details.

Available measurements

- *TotalIntensity:* Sum of all pixel intensity values.
- *MeanIntensity, MedianIntensity:* Mean and median of pixel intensity values.
- *StdIntensity, MADIntensity:* Standard deviation and median absolute deviation (MAD) of pixel intensity values. The MAD is defined as the median($|x_i$ - median(x)|).
- *MinIntensity, MaxIntensity:* Minimum and maximum of pixel intensity values.
- *LowerQuartileIntensity:* The intensity value of the pixel for which 25% of the pixels in the object have lower values.
- *UpperQuartileIntensity:* The intensity value of the pixel for which 75% of the pixels in the object have lower values.
- *TotalArea:* Number of pixels measured, e.g., the area of the image.

See also **MeasureObjectIntensity**, **MaskImage**.

Settings:

Select the image to measure

Choose an image name from the drop-down menu to calculate intensity for that image. Use the *Add another image* button below to add additional images which will be measured. You can add the same image multiple times if you want to measure the intensity within several different objects.

Measure the intensity only from areas enclosed by objects?

Select *Yes* to measure only those pixels within an object of choice.

Select the input objects

(Used only when measuring intensity from area enclosed by objects)
Select the objects that the intensity will be aggregated within. The intensity measurement will be restricted

to the pixels within these objects.

Module: MeasureImageQuality

Measure Image Quality measures features that indicate image quality.

This module can collect measurements indicative of possible image abberations, e.g. blur (poor focus), intensity, saturation (i.e., the percentage of pixels in the image that are minimal and maximal). Details and guidance for each of these measures is provided in the settings help.

Please note that for best results, this module should be applied to the original raw images, as opposed to images that have already been corrected for illumination.

Available measurements

- **Blur metrics**
  - *FocusScore:* A measure of the intensity variance across the image.
  - *LocalFocusScore:* A measure of the intensity variance between image sub-regions.
  - *Correlation:* A measure of the correlation of the image for a given spatial scale.
  - *PowerLogLogSlope:* The slope of the image log-log power spectrum.

- **Saturation metrics**
  - *PercentMaximal:* Percent of pixels at the maximum intensity value of the image.
  - *PercentMinimal:* Percent of pixels at the minimum intensity value of the image.

- **Intensity metrics**
  - *TotalIntensity:* Sum of all pixel intensity values.
  - *MeanIntensity, MedianIntensity:* Mean and median of pixel intensity values.
  - *StdIntensity, MADIntensity:* Standard deviation and median absolute deviation (MAD) of pixel intensity values.
  - *MinIntensity, MaxIntensity:* Minimum and maximum of pixel intensity values.
  - *TotalArea:* Number of pixels measured.

- **Threshold metrics:**
  - *Threshold:* The automatically calculated threshold for each image for the thresholding method of choice.

    Please note that these thresholds are recorded individually for each image and as an aggregate statistic for all images. The mean, median and standard deviation of the threshold values are computed for each of the threshold methods selected and recorded as a measurement in the per-experiment table.

References

- Bray MA, Fraser AN, Hasaka TP, Carpenter AE (2012) "Workflow and metrics for image quality control in large-scale high-content screens." *J Biomol Screen* 17(2):266-74. (link)

Settings:

Calculate metrics for which images?

This option lets you choose which images will have quality metrics calculated.

- *All loaded images:* Use all images loaded with the **Input** modules. The quality metrics selected below will be applied to all loaded images.
- *Select...:* Select the desired images from a list. The quality metric settings selected will be applied to all these images. Additional lists can be added with separate settings.

Select the images to measure

(Used only if "Select..." is chosen for selecting images)
Choose one or more images from this list. You can select multiple images by clicking using the shift or command keys. In addition to loaded images, the list includes the images that were created by prior modules.

Include the image rescaling value?

Select *Yes* to add the image's rescaling value as a quality control metric. This value is set only for images that loaded using the **Input** modules. This is useful in confirming that all images are rescaled by the same value, since some acquisition device vendors may output this value differently. See **NamesAndTypes** for more information.

Calculate blur metrics?

Select *Yes* to compute a series of blur metrics. The blur metrics are the following, along with recomendations on their use:

- *PowerLogLogSlope:* The power spectrum contains the frequency information of the image, and the slope gives a measure of image blur. A higher slope indicates more lower frequency components, and hence more blur (*Field, 1997*). This metric is recommended for blur detection in most cases.
- *Correlation:* This is a measure of the image spatial intensity distribution computed across sub-regions of an image for a given spatial scale (*Haralick, 1973*). If an image is blurred, the correlation between neighboring pixels becomes high, producing a high correlation value. A similar approach was found to give optimal performance for fluorescence microscopy applications (*Vollath, 1987*). Some care is required in selecting an appropriate spatial scale because differences in the spatial scale capture various features: moderate scales capture the blurring of intracellular features better than small scales and larger scales are more likely to reflect intercellular confluence than focal blur. A spatial scale no bigger than the feature of interest is recommended, although you can select as many scales as desired.
- *FocusScore:* This score is calculated using a normalized variance, which was the best-ranking algorithm for brightfield, phase contrast, and DIC images (*Sun, 2004*). Higher focus scores correspond to lower bluriness.
  More specifically, the focus score computes the intensity variance of the entire image divided by mean image intensity. Since it is tailored for autofocusing applications (difference focus for the same field of view), it assumes that the overall intensity and the number of objects in the image is constant, making it less useful for comparision images of different fields of view. For distinguishing extremely blurry images, however, it performs well.
- *LocalFocusScore:* A local version of the Focus Score, it subdivides the image into non-overlapping tiles, computes the normalized variance for each, and takes the mean of these values as the final metric. It is potentially more useful for comparing focus between images of different fields of view, but is subject to the same caveats as the Focus Score. It can be useful in differentiating good versus badly segmented images in the cases when badly segmented images usually contain no cell objects with high background noise.

References

- Field DJ (1997) "Relations between the statistics of natural images and the response properties of cortical cells" *Journal of the Optical Society of America. A, Optics, image science, and vision*, 4(12):2379-94. <(pdf)
- Haralick RM (1979) "Statistical and structural approaches to texture" Proc. IEEE, 67(5):786-804. (link)
- Vollath D (1987) "Automatic focusing by correlative methods" *Journal of Microscopy* 147(3):279-288. (link)
- Sun Y, Duthaler S, Nelson B (2004) "Autofocusing in computer microscopy: Selecting the optimal focus algorithm" *Microscopy Research and Technique*, 65:139-149 (link)

Spatial scale for blur measurements

(Used only if blur measurements are to be calculated)
The *LocalFocusScore* is measured within an $N \times N$ pixel window applied to the image, whereas the *Correlation* of a pixel is measured with repsect to its neighbors N pixels away.

A higher number for the window size measures larger patterns of image blur whereas smaller numbers measure more localized patterns of blur. We suggest selecting a window size that is on the order of the feature of interest (e.g., the object diameter). You can measure these metrics for multiple window sizes by selecting additional scales for each image.

Calculate saturation metrics?

Select *Yes* to calculate the saturation metrics *PercentMaximal* and *PercentMinimal*, i.e., the percentage of pixels at the upper or lower limit of each individual image.

For this calculation, the hard limits of 0 and 1 are not used because images often have undergone some kind of transformation such that no pixels ever reach the absolute maximum or minimum of the image format. Given the noise typical in images, both these measures should be a low percentage but if the images were saturated during imaging, a higher than usual *PercentMaximal* will be observed, and if there are no objects, the *PercentMinimal* value will increase.

Calculate intensity metrics?

Select *Yes* to calculate image-based intensity measures, namely the mean, maximum, minimum, standard deviation and median absolute deviation of pixel intensities. These measures are identical to those calculated by **MeasureImageIntensity**.

Calculate thresholds?

Automatically calculate a suggested threshold for each image. One indicator of image quality is that these threshold values lie within a typical range. Outlier images with high or low thresholds often contain artifacts.

Use all thresholding methods?

(Used only if image thresholds are calculcated)
Select *Yes* to calculate thresholds using all the available methods. Only the global methods are used. While most methods are straightfoward, some methods have additional parameters that require special handling:

- *Otsu:* Thresholds for all combinations of class number, minimzation parameter and middle class

assignment are computed.
- *Mixture of Gaussians (MoG):* Thresholds for image coverage fractions of 0.05, 0.25, 0.75 and 0.95 are computed.

See the **IdentifyPrimaryObjects** module for more information on thresholding methods.

Select a thresholding method

(Used only if particular thresholds are to be calculated)
This setting allows you to apply automatic thresholding methods used in the **Identify** modules. Only the global methods are applied. For more help on thresholding, see the **Identify** modules.

Typical fraction of the image covered by objects

(Used only if threshold are calculated and MoG thresholding is chosen)
Enter the approximate fraction of the typical image in the set that is covered by objects.

Two-class or three-class thresholding?

(Used only if thresholds are calculcated and the Otsu thresholding method is used)
Select *Two classes* if the grayscale levels are readily distinguishable into foregound (i.e., objects) and background. Select *Three classes* if there is a middle set of grayscale levels that belongs to neither the foreground nor background.

For example, three-class thresholding may be useful for images in which you have nuclear staining along with a low-intensity non-specific cell staining. Where two-class thresholding might incorrectly assign this intemediate staining to the nuclei objects, three-class thresholding allows you to assign it to the foreground or background as desired. However, in extreme cases where either there are almost no objects or the entire field of view is covered with objects, three-class thresholding may perform worse than two-class.

Assign pixels in the middle intensity class to the foreground or the background?

(Used only if thresholds are calculcated and the Otsu thresholding method with Three classes is used)
Choose whether you want the middle grayscale intensities to be assigned to the foreground pixels or the background pixels.

Module: MeasureNeurons

Measure Neurons measures branching information for neurons or any skeleton objects with seed points.

This module measures the number of trunks and branches for each neuron in an image. The module takes a skeletonized image of the neuron plus previously identified seed objects (for instance, the neuron soma) and finds the number of axon or dendrite trunks that emerge from the soma and the number of branches along the axons and dendrites. Note that the seed objects must be both smaller than, and touching the skeleton in order to be counted.

The typical approach for this module is the following:

- Identify a seed object. This object is typically a nucleus, identified with a module such as **IdentifyPrimaryObjects**.
- Identify a larger object that touches or encloses this seed object. For example, the neuron cell can be grown outwards from the initial seed nuclei using **IdentifySecondaryObjects**.
- Use the **Morph** module to skeletonize the secondary objects.
- Finally, the primary objects and the skeleton objects are used as inputs to **MeasureNeurons**.

The module determines distances from the seed objects along the axons and dendrites and assigns branchpoints based on distance to the closest seed object when two seed objects appear to be attached to the same dendrite or axon.

Available measurements

- *NumberTrunks:* The number of trunks. Trunks are branchpoints that lie within the seed objects
- *NumberNonTrunkBranches:* The number of non-trunk branches. Branches are the branchpoints that lie outside the seed objects.
- *NumberBranchEnds*: The number of branch end-points, i.e, termini.

Settings:

Select the seed objects

Select the previously identified objects that you want to use as the seeds for measuring branches and distances. Branches and trunks are assigned per seed object. Seed objects are typically not single points/pixels but instead are usually objects of varying sizes.

Select the skeletonized image

Select the skeletonized image of the dendrites and/or axons as produced by the **Morph** module's *Skel* operation.

Retain the branchpoint image?

Select *Yes* if you want to save the color image of branchpoints and trunks. This is the image that is displayed in the output window for this module.

Name the branchpoint image

(Used only if a branchpoint image is to be retained)
Enter a name for the branchpoint image here. You can then use this image in a later module, such as **SaveImages**.

Maximum hole size:

(Used only when filling small holes)
This is the area of the largest hole to fill, measured in pixels. The algorithm will fill in any hole whose area is this size or smaller.

Export the neuron graph relationships?

Select *Yes* to produce an edge file and a vertex file that give the relationships between trunks, branchpoints and vertices.

Intensity image

Select the image to be used to calculate the total intensity along the edges between the vertices.

Vertex file name

Enter the name of the file that will hold the edge information. You can use metadata tags in the file name.

Each line of the file is a row of comma-separated values. The first row is the header; this names the file's columns. Each subsequent row represents a vertex in the neuron skeleton graph: either a trunk, a branchpoint or an endpoint. The file has the following columns:

- *image_number:* The image number of the associated image
- *vertex_number:* The number of the vertex within the image
- *i:* The I coordinate of the vertex.
- *j:* The J coordinate of the vertex.
- *label:* The label of the seed object associated with the vertex.
- *kind:* The vertex type, with the following choices:
    - **T:** Trunk
    - **B:** Branchpoint
    - **E:** Endpoint

Edge file name

Enter the name of the file that will hold the edge information. You can use metadata tags in the file name. Each line of the file is a row of comma-separated values. The first row is the header; this names the file's columns. Each subsequent row represents an edge or connection between two vertices (including between a vertex and itself for certain loops).

The file has the following columns:

- *image_number:* The image number of the associated image
- *v1:* The zero-based index into the vertex table of the first vertex in the edge.
- *v2:* The zero-based index into the vertex table of the second vertex in the edge.
- *length:* The number of pixels in the path connecting the two vertices, including both vertex pixels.
- *total_intensity:* The sum of the intensities of the pixels in the edge, including both vertex pixel intensities.

Module: MeasureObjectIntensity

Measure Object Intensity measures several intensity features for identified objects.

Given an image with objects identified (e.g. nuclei or cells), this module extracts intensity features for each object based on one or more corresponding grayscale images. Measurements are recorded for each object.

Intensity measurements are made for all combinations of the images and objects entered. If you want only specific image/object measurements, you can use multiple MeasureObjectIntensity modules for each group of measurements desired.

Note that for publication purposes, the units of intensity from microscopy images are usually described as "Intensity units" or "Arbitrary intensity units" since microscopes are not calibrated to an absolute scale. Also, it is important to note whether you are reporting either the mean or the integrated intensity, so specify "Mean intensity units" or "Integrated intensity units" accordingly.

Keep in mind that the default behavior in CellProfiler is to rescale the image intensity from 0 to 1 by dividing all pixels in the image by the maximum possible intensity value. This "maxmimum possible" value is defined by the "Set intensity range from" setting in **NamesAndTypes**; see the help for that setting for more details.

Available measurements

- *IntegratedIntensity:* The sum of the pixel intensities within an object.
- *MeanIntensity:* The average pixel intensity within an object.
- *StdIntensity:* The standard deviation of the pixel intensities within an object.
- *MaxIntensity:* The maximal pixel intensity within an object.
- *MinIntensity:* The minimal pixel intensity within an object.
- *IntegratedIntensityEdge:* The sum of the edge pixel intensities of an object.
- *MeanIntensityEdge:* The average edge pixel intensity of an object.
- *StdIntensityEdge:* The standard deviation of the edge pixel intensities of an object.
- *MaxIntensityEdge:* The maximal edge pixel intensity of an object.
- *MinIntensityEdge:* The minimal edge pixel intensity of an object.
- *MassDisplacement:* The distance between the centers of gravity in the gray-level representation of the object and the binary representation of the object.
- *LowerQuartileIntensity:* The intensity value of the pixel for which 25% of the pixels in the object have lower values.
- *MedianIntensity:* The median intensity value within the object
- *MADIntensity:* The median absolute deviation (MAD) value of the intensities within the object. The MAD is defined as the median($|x_i$ - median(x)|).
- *UpperQuartileIntensity:* The intensity value of the pixel for which 75% of the pixels in the object have lower values.
- *Location_CenterMassIntensity_X, Location_CenterMassIntensity_Y:* The pixel (X,Y) coordinates of the intensity weighted centroid (= center of mass = first moment) of all pixels within the object.
- *Location_MaxIntensity_X, Location_MaxIntensity_Y:* The pixel (X,Y) coordinates of the pixel with the maximum intensity within the object.

See also **NamesAndTypes**, **MeasureImageIntensity**.

Settings:

Select an image to measure

Select the grayscale images whose intensity you want to measure.

Select objects to measure

Select the objects whose intensities you want to measure.

Module: MeasureObjectNeighbors

Measure Object Neighbors calculates how many neighbors each object has and records various properties about the neighbors' relationships, including the percentage of an object's edge pixels that touch a neighbor.

Given an image with objects identified (e.g., nuclei or cells), this module determines how many neighbors each object has. You can specify the distance within which objects should be considered neighbors, or that objects are only considered neighbors if they are directly touching.

Available measurements

Object measurements

- *NumberOfNeighbors:* Number of neighbor objects.
- *PercentTouching:* Percent of the object's boundary pixels that touch neighbors, after the objects have been expanded to the specified distance. Note: This measurement is only available if you use the same set of objects for both objects and neighbors.
- *FirstClosestObjectNumber:* The index of the closest object.
- *FirstClosestDistance:* The distance to the closest object.
- *SecondClosestObjectNumber:* The index of the second closest object.
- *SecondClosestDistance:* The distance to the second closest object.
- *AngleBetweenNeighbors:* The angle formed with the object center as the vertex and the first and second closest object centers along the vectors.

Object relationships: The identity of the neighboring objects, for each object. Since per-object output is one-to-one and neighbors relationships are often many-to-one, they may be saved as a separate file in **ExportToSpreadsheet** by selecting *Object relationships* from the list of objects to export.

Technical notes

Objects discarded via modules such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects** will still register as a neighbors for the purposes of accurate measurement. For instance, if an object touches a single object and that object had been discarded, *NumberOfNeighbors* will be positive, but there will not be a corresponding *ClosestObjectNumber*.

See also the **Identify** modules.

Settings:

Select objects to measure

Select the objects whose neighbors you want to measure.

Select neighboring objects to measure

This is the name of the objects that are potential neighbors of the above objects. You can find the neighbors within the same set of objects by selecting the same objects as above.

Method to determine neighbors

There are several methods by which to determine whether objects are neighbors:

- *Adjacent:* In this mode, two objects must have adjacent boundary pixels to be neighbors.
- *Expand until adjacent:* The objects are expanded until all pixels on the object boundaries are touching another. Two objects are neighbors if their any of their boundary pixels are adjacent after expansion.
- *Within a specified distance:* Each object is expanded by the number of pixels you specify. Two objects are neighbors if they have adjacent pixels after expansion.

For *Adjacent* and *Expand until adjacent*, the *PercentTouching* measurement is the percentage of pixels on the boundary of an object that touch adjacent objects. For *Within a specified distance*, two objects are touching if their any of their boundary pixels are adjacent after expansion and *PercentTouching* measures the percentage of boundary pixels of an *expanded* object that touch adjacent objects.

Neighbor distance

(Used only when "Within a specified distance" is selected)
The Neighbor distance is the number of pixels that each object is expanded for the neighbor calculation. Expanded objects that touch are considered neighbors.

Retain the image of objects colored by numbers of neighbors?

An output image showing the input objects colored by numbers of neighbors may be retained. A colormap of your choice shows how many neighbors each object has. The background is set to -1. Objects are colored with an increasing color value corresponding to the number of neighbors, such that objects with no neighbors are given a color corresponding to 0. Use the **SaveImages** module to save this image to a file.

Name the output image

(Used only if the image of objects colored by numbers of neighbors is to be retained for later use in the pipeline)
Specify a name that will allow the the image of objects colored by numbers of neighbors to be selected later in the pipeline.

Select colormap

(Used only if the image of objects colored by numbers of neighbors is to be retained for later use in the pipeline)
Select the colormap to use to color the neighbor number image. All available colormaps can be seen here.

Retain the image of objects colored by percent of touching pixels?

Select *Yes* to keep an image of the input objects colored by the percentage of the boundary touching their neighbors. A colormap of your choice is used to show the touching percentage of each object. Use the **SaveImages** module to save this image to a file.

Name the output image

(Used only if the image of objects colored by percent touching is to be retained for later use in the

pipeline)
Specify a name that will allow the the image of objects colored by percent of touching pixels to be selected later in the pipeline.

Select a colormap

(Used only if the image of objects colored by percent touching is to be retained for later use in the pipeline)
Select the colormap to use to color the percent touching image. All available colormaps can be seen here.

Module: MeasureObjectRadialDistribution

Measure Object Radial Distribution measures the radial distribution of intensities within each object.

Given an image with objects identified, this module measures the intensity distribution from each object's center to its boundary within a user-controlled number of bins, i.e. rings.

The distribution is measured from the center of the object, where the center is defined as the point farthest from any edge. The numbering is from 1 (innermost) to *N* (outermost), where *N* is the number of bins specified by the user. Alternatively, if primary objects exist within the object of interest (e.g. nuclei within cells), you can choose the center of the primary objects as the center from which to measure the radial distribution. This might be useful in cytoplasm-to-nucleus translocation experiments, for example. Note that the ring widths are normalized per-object, i.e., not necessarily a constant width across objects.

Available measurements

- *FracAtD:* Fraction of total stain in an object at a given radius.
- *MeanFrac:* Mean fractional intensity at a given radius; calculated as fraction of total intensity normalized by fraction of pixels at a given radius.
- *RadialCV:* Coefficient of variation of intensity within a ring, calculated over 8 slices.

See also **MeasureObjectIntensity**.

Settings:

Select an image to measure

Select the image that you want to measure the intensity from.

Select objects to measure

Select the objects that you want to measure the intensity from.

Object to use as center?

There are three ways to specify the center of the radial measurement:

- *These objects:* Use the centers of these objects for the radial measurement.
- *Centers of other objects:* Use the centers of other objects for the radial measurement.
- *Edges of other objects:* Measure distances from the edge of the other object to each pixel outside of the centering object. Do not include pixels within the centering object in the radial measurement calculations.

For example, if measuring the radial distribution in a Cell object, you can use the center of the Cell objects (*These objects*) or you can use previously identified Nuclei objects as the centers (*Centers of other objects*).

Select objects to use as centers

(Used only if "Centers of other objects" are selected for centers)

Select the object to use as the center, or select *None* to use the input object centers (which is the same as selecting *These objects* for the object centers).

Scale the bins?

Select *Yes* to divide the object radially into the number of bins that you specify.

Select *No* to create the number of bins you specify based on distance. For this option, the user will be asked to specify a maximum distance so that each object will have the same measurements (which might be zero for small objects) and so that the measurements can be taken without knowing the maximum object radius before the run starts.

Number of bins

Specify the number of bins that you want to use to measure the distribution. Radial distribution is measured with respect to a series of concentric rings starting from the object center (or more generally, between contours at a normalized distance from the object center). This number specifies the number of rings into which the distribution is to be divided. Additional ring counts can be specified by clicking the *Add another set of bins* button.

Maximum radius

Specify the maximum radius for the unscaled bins. The unscaled binning method creates the number of bins that you specify and creates equally spaced bin boundaries up to the maximum radius. Parts of the object that are beyond this radius will be counted in an overflow bin. The radius is measured in pixels.

Module: MeasureObjectSizeShape

Measure Object Size Shape measures several area and shape features of identified objects.

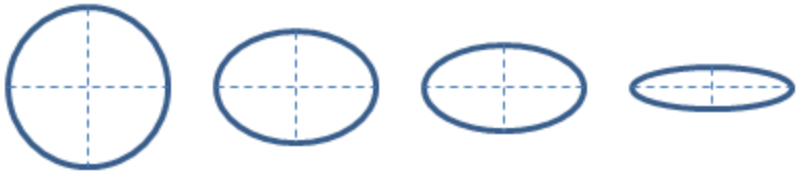
Given an image with identified objects (e.g. nuclei or cells), this module extracts area and shape features of each one. Note that these features are only reliable for objects that are completely inside the image borders, so you may wish to exclude objects touching the edge of the image using **IdentifyPrimaryObjects**.

Please note that the display window for this module shows per-image aggregates for the per-object measurements. If you want to view the per-object measurements themselves, you will need to use **ExportToSpreadsheet** to export them, or use **DisplayDataOnImage** to display the object measurements of choice overlaid on an image of choice.

Available measurements

See the *Technical Notes* below for an explanation of creating an ellipse with the same second-moments as an object region.

- *Area:* The actual number of pixels in the region.
- *Perimeter:* The total number of pixels around the boundary of each region in the image.
- *FormFactor:* Calculated as $4*\pi*\text{Area}/\text{Perimeter}^2$. Equals 1 for a perfectly circular object.
- *Solidity:* The proportion of the pixels in the convex hull that are also in the object, i.e. *ObjectArea/ConvexHullArea*. Equals 1 for a solid object (i.e., one with no holes or has a concave boundary), or <1 for an object with holes or possessing a convex/irregular boundary.
- *Extent:* The proportion of the pixels in the bounding box that are also in the region. Computed as the Area divided by the area of the bounding box.
- *EulerNumber:* The number of objects in the region minus the number of holes in those objects, assuming 8-connectivity.
- *Center_X, Center_Y:* The *x*- and *y*-coordinates of the point farthest away from any object edge. Note that this is not the same as the *Location-X* and *-Y* measurements produced by the **Identify** modules.
- *Eccentricity:* The eccentricity of the ellipse that has the same second-moments as the region. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases; an ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)



- *MajorAxisLength:* The length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region.
- *MinorAxisLength:* The length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region.
- *Orientation:* The angle (in degrees ranging from -90 to 90 degrees) between the x-axis and the major axis of the ellipse that has the same second-moments as the region.
- *Compactness:* The variance of the radial distance of the object's pixels from the centroid divided by the area.
- *MaximumRadius:* The maximum distance of any pixel in the object to the closest pixel outside of the object. For skinny objects, this is 1/2 of the maximum width of the object.

- *MedianRadius:* The median distance of any pixel in the object to the closest pixel outside of the object.
- *MeanRadius:* The mean distance of any pixel in the object to the closest pixel outside of the object.
- *MinFeretDiameter, MaxFeretDiameter:* The Feret diameter is the distance between two parallel lines tangent on either side of the object (imagine taking a caliper and measuring the object at various angles). The minimum and maximum Feret diameters are the smallest and largest possible diameters, rotating the calipers along all possible angles.
- *Zernike shape features:* Measure shape by describing a binary object (or more precisely, a patch with background and an object in the center) in a basis of Zernike polynomials, using the coefficients as features (*Boland et al., 1998*). Currently, Zernike polynomials from order 0 to order 9 are calculated, giving in total 30 measurements. While there is no limit to the order which can be calculated (and indeed users could add more by adjusting the code), the higher order polynomials carry less information.

Technical notes

A number of the object measurements are generated by creating an ellipse with the same second-moments as the original object region. This is essentially the best-fitting ellipse for a given object with the same statistical properties. Furthermore, they are not affected by the translation or uniform scaling of a region.

The Zernike features are computed within the minimum enclosing circle of the object, i.e., the circle of the smallest diameter that contains all of the object's pixels.

References

- Rocha L, Velho L, Carvalho PCP, "Image moments-based structuring and tracking of objects", Proceedings from XV Brazilian Symposium on Computer Graphics and Image Processing, 2002. (pdf)
- Principles of Digital Image Processing: Core Algorithms (Undergraduate Topics in Computer Science): Section 2.4.3 - Statistical shape properties
- Chrystal P (1885), "On the problem to construct the minimum circle enclosing n given points in a plane", *Proceedings of the Edinburgh Mathematical Society*, vol 3, p. 30

See also **MeasureImageAreaOccupied**.

Settings:

Select objects to measure

Select the objects that you want to measure.

Calculate the Zernike features?

Select *Yes* to calculate the Zernike shape features. Since the first 10 Zernike polynomials (from order 0 to order 9) are calculated, this operation can be time consuming if the image contains a lot of objects.

Module: MeasureTexture

Measure Texture measures the degree and nature of textures within objects (versus smoothness).

This module measures the variations in grayscale images. An object (or entire image) without much texture has a smooth appearance; an object or image with a lot of texture will appear rough and show a wide variety of pixel intensities.

This module can also measure textures of objects against grayscale images. Any input objects specified will have their texture measured against *all* input images specfied, which may lead to image-object texture combinations that are unneccesary. If you do not want this behavior, use multiple **MeasureTexture** modules to specify the particular image-object measures that you want.

Available measurements

- *Haralick Features:* Haralick texture features are derived from the co-occurrence matrix, which contains information about how image intensities in pixels with a certain position in relation to each other occur together. **MeasureTexture** can measure textures at different scales; the scale you choose determines how the co-occurrence matrix is constructed. For example, if you choose a scale of 2, each pixel in the image (excluding some border pixels) will be compared against the one that is two pixels to the right. **MeasureTexture** quantizes the image into eight intensity levels. There are then 8x8 possible ways to categorize a pixel with its scale-neighbor. **MeasureTexture** forms the 8x8 co-occurrence matrix by counting how many pixels and neighbors have each of the 8x8 intensity combinations.

  Thirteen measurements are then calculated for the image by performing mathematical operations on the co-occurrence matrix (the formulas can be found here):

  - *AngularSecondMoment*
  - *Contrast*
  - *Correlation*
  - *Variance*
  - *InverseDifferenceMoment*
  - *SumAverage*
  - *SumVariance*
  - *SumEntropy*
  - *Entropy*
  - *DifferenceVariance*
  - *DifferenceEntropy*
  - *InfoMeas1*
  - *InfoMeas2*

  Each measurement is suffixed with the direction of the offset used between pixels in the co-occurrence matrix:
  - *0:* Horizontal
  - *90:* Vertical
  - *45:* Diagonal
  - *135:* Anti-diagonal

- *Gabor "wavelet" features:* These features are similar to wavelet features, and they are obtained by applying so-called Gabor filters to the image. The Gabor filters measure the frequency content in different orientations. They are very similar to wavelets, and in the current context they work exactly

as wavelets, but they are not wavelets by a strict mathematical definition. The Gabor features detect correlated bands of intensities, for instance, images of Venetian blinds would have high scores in the horizontal orientation.

Technical notes

To calculate the Haralick features, **MeasureTexture** normalizes the co-occurence matrix at the per-object level by basing the intensity levels of the matrix on the maximum and minimum intensity observed within each object. This is beneficial for images in which the maximum intensities of the objects vary substantially because each object will have the full complement of levels.

MeasureTexture performs a vectorized calculation of the Gabor filter, properly scaled to the size of the object being measured and covering all pixels in the object. The Gabor filter can be calculated at a user-selected number of angles by using the following algorithm to compute a score at each scale using the Gabor filter:

- Divide the half-circle from 0 to 180° by the number of desired angles. For instance, if the user chooses two angles, **MeasureTexture** uses 0 and 90 ° (horizontal and vertical) for the filter orientations. This is the θ value from the reference paper.
- For each angle, compute the Gabor filter for each object in the image at two phases separated by 90° in order to account for texture features whose peaks fall on even or odd quarter-wavelengths.
- Multiply the image times each Gabor filter and sum over the pixels in each object.
- Take the square root of the sum of the squares of the two filter scores. This results in one score per θ.
- Save the maximum score over all θ as the score at the desired scale.

References

- Haralick RM, Shanmugam K, Dinstein I. (1973), "Textural Features for Image Classification" *IEEE Transaction on Systems Man, Cybernetics*, SMC-3(6):610-621. (link)
- Gabor D. (1946). "Theory of communication" *Journal of the Institute of Electrical Engineers* 93:429-441. (link)

Settings:

Select an image to measure

Select the grayscale images whose texture you want to measure.

Select objects to measure

Select the objects whose texture you want to measure. If you only want to measure the texture for the image overall, you can remove all objects using the "Remove this object" button.

Objects specified here will have their texture measured against *all* images specfied above, which may lead to image-object combinations that are unneccesary. If you do not want this behavior, use multiple **MeasureTexture** modules to specify the particular image-object measures that you want.

Texture scale to measure

You can specify the scale of texture to be measured, in pixel units; the texture scale is the distance between correlated intensities in the image. A higher number for the scale of texture measures larger

patterns of texture whereas smaller numbers measure more localized patterns of texture. It is best to measure texture on a scale smaller than your objects' sizes, so be sure that the value entered for scale of texture is smaller than most of your objects. For very small objects (smaller than the scale of texture you are measuring), the texture cannot be measured and will result in a undefined value in the output file.

Angles to measure

The Haralick texture measurements are based on the correlation between pixels offset by the scale in one of four directions:

- *Horizontal* - the correlated pixel is "scale" pixels to the right of the pixel of interest.
- *Vertical* - the correlated pixel is "scale" pixels below the pixel of interest.
- *Diagonal* - the correlated pixel is "scale" pixels to the right and "scale" pixels below the pixel of interest.
- *Anti-diagonal* - the correlated pixel is "scale" pixels to the left and "scale" pixels below the pixel of interest.

Choose one or more directions to measure.

Measure Gabor features?

The Gabor features measure striped texture in an object, and can take a substantial time to calculate.

Select *Yes* to measure the Gabor features. Select *No* to skip the Gabor feature calculation if it is not informative for your images.

Number of angles to compute for Gabor

(Used only if Gabor features are measured)
Enter the number of angles to use for each Gabor texture measurement. The default value is 4 which detects bands in the horizontal, vertical and diagonal orientations.

Module: ClassifyObjects

Classify Objects classifies objects into different classes according to the value of measurements you choose.

This module classifies objects into a number of different bins according to the value of a measurement (e.g., by size, intensity, shape). It reports how many objects fall into each class as well as the percentage of objects that fall into each class. The module asks you to select the measurement feature to be used to classify your objects and specify the bins to use. It also requires you to have run a measurement or **CalculateMath** previous to this module in the pipeline so that the measurement values can be used to classify the objects.

There are two flavors of classification:

- The first classifies each object according to the measurements you choose and assigns each object to one class per measurement. You may specify more than two classification bins per measurement.
- The second classifies each object according to two measurements and two threshold values. The module classifies each object once per measurement resulting in four possible object classes. The module then stores one measurement per object, based on the object's class.

Note that objects without a measurement are not counted as belonging in a classification bin and will not show up in the output image (shown in the module display window); in the object classification they will have a value of False for all bins. However, they are still counted in the total number of objects and hence are reflected in the classification percentages.

Available measurements

- **Image measurements:**
  - *NumObjectsPerBin:* The number of objects that are classified into each bin.
  - *PctObjectsPerBin:* The percentage of total objects that are classified into each bin.
- **Object measurements:**
  - Single measurement: Classification (true/false) of the N^{th} bin for the M^{th} measurement.
  - Two measurement: Classification (true/false) of the 1^{st} measurement versus the 2^{nd} measurement binned into bins above ("high") and below ("low") the cutoff.

See also **CalculateMath** and any of the modules in the **Measure** category.

Settings:

Make each classification decision on how many measurements?

This setting controls how many measurements are used to make a classifications decision for each object:

- *Single measurement:* Classifies each object based on a single measurement.
- *Pair of measurements:* Classifies each object based on a pair of measurements taken together (that is, an object must meet two criteria to belong to a class).

Select the object to be classified

The name of the objects to be classified. You can choose from objects created by any previous module. See **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, or **IdentifyTertiaryObjects**.

Select the measurement to classify by

Select a measurement made by a previous module. The objects will be classified according to their values for this measurement.

Select bin spacing

You can either specify bins of equal size, bounded by upper and lower limits, or you can specify custom values that define the edges of each bin with a threshold. *Note:* If you would like two bins, choose *Custom-defined bins* and then provide a single threshold when asked. *Evenly spaced bins* creates the indicated number of bins at evenly spaced intervals between the low and high threshold. You also have the option to create bins for objects that fall below or above the low and high threhsold

Number of bins

This is the number of bins that will be created between the low and high threshold

Lower threshold

(Used only if Evenly spaced bins selected)
This is the threshold that separates the lowest bin from the others. The lower threshold, upper threshold, and number of bins define the thresholds of bins between the lowest and highest.

Use a bin for objects below the threshold?

Select *Yes* if you want to create a bin for objects whose values fall below the low threshold. Select *No* if you do not want a bin for these objects.

Upper threshold

(Used only if Evenly spaced bins selected)
This is the threshold that separates the last bin from the others. *Note:* If you would like two bins, choose *Custom-defined bins*.

Use a bin for objects above the threshold?

Select *Yes* if you want to create a bin for objects whose values are above the high threshold.
Select *No* if you do not want a bin for these objects.

Enter the custom thresholds separating the values between bins

(Used only if Custom thresholds selected)
This setting establishes the threshold values for the bins. You should enter one threshold between each bin, separating thresholds with commas (for example, *0.3, 1.5, 2.1* for four bins). The module will create one more bin than there are thresholds.

Give each bin a name?

Select *Yes* to assign custom names to bins you have specified.

Select *No* for the module to automatically assign names based on the measurements and the bin number.

Enter the bin names separated by commas

(Used only if Give each bin a name? is checked)
Enter names for each of the bins, separated by commas. An example including three bins might be *First,Second,Third*.

Retain an image of the classified objects?

Select *Yes* to keep an image of the objects which is color-coded according to their classification, for use later in the pipeline (for example, to be saved by a **SaveImages** module).

Name the output image

Enter the name to be given to the classified object image.

Select the object name

Choose the object that you want to measure from the list. This should be an object created by a previous module such as **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, or **IdentifyTertiaryObjects**.

Select the first measurement

Choose a measurement made on the above object. This is the first of two measurements that will be contrasted together. The measurement should be one made on the object in a prior module.

Method to select the cutoff

Objects are classified as being above or below a cutoff value for a measurement. You can set this cutoff threshold in one of three ways:

- *Mean*: At the mean of the measurement's value for all objects in the image cycle.
- *Median*: At the median of the measurement's value for all objects in the image set.
- *Custom*: You specify a custom threshold value.

Enter the cutoff value

This is the cutoff value separating objects in the two classes.

Select the second measurement

Select a measurement made on the above object. This is the second of two measurements that will be contrasted together. The measurement should be one made on the object in a prior module.

Method to select the cutoff

Objects are classified as being above or below a cutoff value for a measurement. You can set this cutoff threshold in one of three ways:

- *Mean:* At the mean of the measurement's value for all objects in the image cycle.
- *Median:* At the median of the measurement's value for all objects in the image set.
- *Custom:* You specify a custom threshold value.

Use custom names for the bins?

Select *Yes* if you want to specify the names of each bin measurement.
Select *No* to create names based on the measurements. For instance, for "Intensity_MeanIntensity_Green" and "Intensity_TotalIntensity_Blue", the module generates measurements such as "Classify_Intensity_MeanIntensity_Green_High_Intensity_TotalIntensity_Low".

Enter the low-low bin name

(Used only if using a pair of measurements)
Name of the measurement for objects that fall below the threshold for both measurements.

Enter the low-high bin name

(Used only if using a pair of measurements)
Name of the measurement for objects whose first measurement is below threshold and whose second measurement is above threshold.

Enter the high-low bin name

(Used only if using a pair of measurements)
Name of the measurement for objects whose first measurement is above threshold and whose second measurement is below threshold.

Enter the high-high bin name

(Used only if using a pair of measurements)
Name of the measurement for objects that are above the threshold for both measurements.

Retain an image of the classified objects?

Select *Yes* to retain the image of the objects color-coded according to their classification, for use later in the pipeline (for example, to be saved by a **SaveImages** module).

Enter the image name

(Used only if the classified object image is to be retained for later use in the pipeline)
Enter the name to be given to the classified object image.

Module: ConvertObjectsToImage

Convert Objects To Image converts objects you have identified into an image.

This module allows you to take previously identified objects and convert them into an image according to a colormap you select, which can then be saved with the **SaveImages** modules.

If you would like to save your objects but do not need a colormap, you can by bypass this module and use the **SaveImages** module directly by specifying "Objects" as the type of image to save.

Settings:

Select the input objects

Choose the name of the objects you want to convert to an image.

Name the output image

Enter the name of the resulting image.

Select the color format

Select which colors the resulting image should use. You have the following options:

- *Color:* Allows you to choose a colormap that will produce jumbled colors for your objects.
- *Binary (black & white):* All object pixels will be assigned 1 and all background pixels will be assigned 0, creating a binary image.
- *Grayscale:* Gives each object a graylevel pixel intensity value corresponding to its number (also called label), so it usually results in objects on the left side of the image being very dark, progressing toward white on the right side of the image.
- *uint16:* Assigns each object a different number, from 1 to 65535 (the numbers that you can put in a 16-bit integer) and numbers all pixels in each object with the object's number. This format can be written out as a .mat or .tiff file if you want to process the label matrix image using another program.

You can choose *Color* with a *Gray* colormap to produce jumbled gray objects.

Select the colormap

(Used only if Color output image selected)
Choose the colormap to be used, which affects how the objects are colored. You can look up your default colormap under *File > Preferences*.

Module: EditObjectsManually

Edit Objects Manually allows you create, remove and edit objects previously defined.

The interface will show the image that you selected as the guiding image, overlaid with colored outlines of the selected objects (or filled objects if you choose). This module allows you to remove or edit specific objects by pointing and clicking to select objects for removal or editing. Once editing is complete, the module displays the objects as originally identified (left) and the objects that remain after this module (right).

More detailed Help is provided in the editing window via the '?' button.

The pipeline pauses once per processed image when it reaches this module. You must press the *Done* button to accept the selected objects and continue the pipeline.

Available measurements

Image measurements:

- *Count:* The number of edited objects in the image.

Object measurements:

- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the edited objects.

See also **FilterObjects**, **MaskObject**, **OverlayOutlines**, **ConvertToImage**.

Settings:

Select the objects to be edited

Choose a set of previously identified objects for editing, such as those produced by one of the **Identify** modules.

Name the edited objects

Enter the name for the objects that remain after editing. These objects will be available for use by subsequent modules.

Retain outlines of the edited objects?

Select *Yes* if you want to keep images of the outlines of the objects that remain after editing. This image can be saved by downstream modules or overlayed on other images using the **OverlayOutlines** module.

Name the outline image

(Used only if you have selected to retain outlines of edited objects)
Enter a name for the outline image.

Numbering of the edited objects

Choose how to number the objects that remain after editing, which controls how edited objects are associated with their predecessors:

- *Renumber:* The module will number the objects that remain using consecutive numbers. This is a good choice if you do not plan to use measurements from the original objects and you only want to use the edited objects in downstream modules; the objects that remain after editing will not have gaps in numbering where removed objects are missing.
- *Retain:* This option will retain each object's original number so that the edited object's number matches its original number. This allows any measurements you make from the edited objects to be directly aligned with measurements you might have made of the original, unedited objects (or objects directly associated with them).

Display a guiding image?

Select *Yes* to display an image and outlines of the objects.
Select *No* if you do not want a guide image while editing

Select the guiding image

(Used only if a guiding image is desired)
This is the image that will appear when editing objects. Choose an image supplied by a previous module.

Allow overlapping objects?

EditObjectsManually can allow you to edit an object so that it overlaps another or it can prevent you from overlapping one object with another. Objects such as worms or the neurites of neurons may cross each other and might need to be edited with overlapping allowed, whereas a monolayer of cells might be best edited with overlapping off.
Select *Yes* to allow overlaps or select *No* to prevent them.

Module: ExpandOrShrinkObjects

Expand Or Shrink Objects expands or shrinks objects by a defined distance.

The module expands or shrinks objects by adding or removing border pixels. You can specify a certain number of border pixels to be added or removed, expand objects until they are almost touching or shrink objects down to a point. Objects are never lost using this module (shrinking stops when an object becomes a single pixel). The module can separate touching objects without otherwise shrinking the objects.

ExpandOrShrinkObjects can perform some specialized morphological operations that remove pixels without completely removing an object. See the Settings help (below) for more detail.

Special note on saving images: You can use the settings in this module to pass object outlines along to the module **OverlayOutlines** and then save them with the **SaveImages** module. You can also pass the identified objects themselves along to the object processing module **ConvertToImage** and then save them with the **SaveImages** module.

Available measurements

Image measurements:

- *Count:* Number of expanded/shrunken objects in the image.

Object measurements:

- *Location_X, Location_Y:* Pixel (*X,Y*) coordinates of the center of mass of the expanded/shrunken objects.

See also **Identify** modules.

Settings:

Select the input objects

Select the objects that you want to expand or shrink.

Name the output objects

Enter a name for the resulting objects.

Select the operation

Select the operation that you want to perform:

- *Shrink objects to a point:* Remove all pixels but one from filled objects. Thin objects with holes to loops unless the "fill" option is checked.
- *Expand objects until touching:* Expand objects, assigning every pixel in the image to an object. Background pixels are assigned to the nearest object.
- *Add partial dividing lines between objects:* Remove pixels from an object that are adjacent to another object's pixels unless doing so would change the object's Euler number (break an object in

two, remove the object completely or open a hole in an object).
- *Shrink objects by a specified number of pixels:* Remove pixels around the perimeter of an object unless doing so would change the object's Euler number (break the object in two, remove the object completely or open a hole in the object). You can specify the number of times perimeter pixels should be removed. Processing stops automatically when there are no more pixels to remove.
- *Expand objects by a specified number of pixels:* Expand each object by adding background pixels adjacent to the image. You can choose the number of times to expand. Processing stops automatically if there are no more background pixels.
- *Skeletonize each object:* Erode each object to its skeleton.
- *Remove spurs:* Remove or reduce the length of spurs in a skeletonized image. The algorithm reduces spur size by the number of pixels indicated in the setting *Number of pixels by which to expand or shrink*.

Fill holes in objects so that all objects shrink to a single point?

(Used only if one of the "shrink" options selected)
Select *Yes* to ensure that each object will shrink to a single point, by filling the holes in each object.

Select *No* to preserve the Euler number. in this case, the shrink algorithm preserves each object's Euler number, which means that it will erode an object with a hole to a ring in order to keep the hole. An object with two holes will be shrunk to two rings connected by a line in order to keep from breaking up the object or breaking the hole.

Retain the outlines of the identified objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: FilterObjects

Filter Objects eliminates objects based on their measurements (e.g., area, shape, texture, intensity).

This module removes selected objects based on measurements produced by another module (e.g., **MeasureObjectSizeShape**, **MeasureObjectIntensity**, **MeasureTexture**, etc). All objects that do not satisfy the specified parameters will be discarded.

This module also may remove objects touching the image border or edges of a mask. This is useful if you would like to unify images via **ReassignObjectNumbers** before deciding to discard these objects.

Please note that the objects that pass the filtering step comprise a new object set, and hence do not inherit the measurements associated with the original objects. Any measurements on the new object set will need to be made post-filtering by the desired measurement modules.

Available measurements

Image measurements:

- *Count:* The number of objects remaining after filtering.

Object measurements:

- *Parent:* The identity of the input object associated with each filtered object.
- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the remaining objects.

See also any of the **MeasureObject** modules, **MeasureTexture**, **MeasureCorrelation**, and **CalculateRatios**.

Settings:

Name the output objects

Enter a name for the collection of objects that are retained after applying the filter(s).

Select the object to filter

Select the set of objects that you want to filter. This setting also controls which measurement choices appear for filtering: you can only filter based on measurements made on the object you select. If you intend to use a measurement calculated by the **CalculateMath** module to to filter objects, select the first operand's object here, because **CalculateMath** measurements are stored with the first operand's object.

Select the filtering mode

You can choose from the following options:

- *Measurements*: Specify a per-object measurement made by an upstream module in the pipeline.
- *Rules*: Use a file containing rules generated by CellProfiler Analyst. You will need to ensure that the measurements specified by the rules file are produced by upstream modules in the pipeline.
- *Image or mask border*: Remove objects touching the border of the image and/or the edges of an image mask.

Select the filtering method

(Used only if filtering using measurements)
There are five different ways to filter objects:

- *Limits:* Keep an object if its measurement value falls within a range you specify.
- *Maximal:* Keep the object with the maximum value for the measurement of interest. If multiple objects share a maximal value, retain one object selected arbitrarily per image.
- *Minimal:* Keep the object with the minimum value for the measurement of interest. If multiple objects share a minimal value, retain one object selected arbitrarily per image.
- *Maximal per object:* This option requires you to choose a parent object. The parent object might contain several child objects of choice (for instance, mitotic spindles within a cell or FISH probe spots within a nucleus). Only the child object whose measurements equal the maximum child-measurement value among that set of child objects will be kept (for example, the longest spindle in each cell). You do not have to explicitly relate objects before using this module.
- *Minimal per object:* Same as *Maximal per object*, except filtering is based on the minimum value.

Assign overlapping child to

(Used only if filtering per object)
A child object can overlap two parent objects and can have the maximal/minimal measurement of all child objects in both parents. This option controls how an overlapping maximal/minimal child affects filtering of other children of its parents and to which parent the maximal child is assigned. The choices are:

- *Both parents*: The child will be assigned to both parents and all other children of both parents will be filtered. Only the maximal child per parent will be left, but if **RelateObjects** is used to relate the maximal child to its parent, one or the other of the overlapping parents will not have a child even though the excluded parent may have other child objects. The maximal child can still be assigned to both parents using a database join via the relationships table if you are using **ExportToDatabase** and separate object tables.
- *Parent with most overlap*: The child will be assigned to the parent with the most overlap and a child with a less maximal/minimal measurement, if available, will be assigned to other parents. Use this option to ensure that parents with an alternate non-overlapping child object are assigned some child object by a subseequent **RelateObjects** module.

Rules file location

(Used only when filtering using Rules)
Select the location of the rules file that will be used for filtering. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

Rules file name

(Used only when filtering using Rules)
The name of the rules file. This file should be a plain text file containing the complete set of rules.

Each line of this file should be a rule naming a measurement to be made on the object you selected, for instance:

```
IF (Nuclei_AreaShape_Area < 351.3, [0.79, -0.79], [-0.94, 0.94])
```

The above rule will score +0.79 for the positive category and -0.94 for the negative category for nuclei whose area is less than 351.3 pixels and will score the opposite for nuclei whose area is larger. The filter adds positive and negative and keeps only objects whose positive score is higher than the negative score.

Class number

(Used only when filtering using Rules)
Select which of the classes to keep when filtering. The CellProfiler Analyst classifier user interface lists the names of the classes in left-to-right order. **FilterObjects** uses the first class from CellProfiler Analyst if you choose "1", etc.

Please note the following:

- The object is retained if the object falls into the selected class.
- You can make multiple class selections. If you do so, the module will retain the object if the object falls into any of the selected classes.

Select the objects that contain the filtered objects

(Used only if a per-object filtering method is selected)
This setting selects the container (i.e., parent) objects for the *Maximal per object* and *Minimal per object* filtering choices.

Retain outlines of the identified objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
(Used only if the outline image is to be retained for later use in the pipeline)

Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: IdentifyObjectsInGrid

Identify Objects In Grid identifies objects within each section of a grid that has been defined by the **DefineGrid** module.

This module identifies objects that are contained within in a grid pattern, allowing you to measure the objects using **Measure** modules. It requires you to have defined a grid earlier in the pipeline, using the **DefineGrid** module.

For several of the automatic options, you will need to enter the names of previously identified objects. Typically, this module is used to refine locations and/or shapes of objects of interest that you roughly identified in a previous **Identify** module. Within this module, objects are re-numbered according to the grid definitions rather than their original numbering from the earlier **Identify** module.

If placing the objects within the grid is impossible for some reason (the grid compartments are too close together to fit the proper sized circles, for example) the grid will fail and processing will be canceled unless you choose to re-use a grid from a previous successful image cycle.

Special note on saving images: You can use the settings in this module to pass object outlines along to the **OverlayOutlines**module and then save them with the **SaveImages** module. You can also pass along objects themselves to the object processing module **ConvertToImage** and then save them with the **SaveImages** module.

Available measurements

Image measurements:

- *Count:* The number of objects identified.

Object measurements:

- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the identified objects.
- *Number:* The numeric label assigned to each identified object according to the arrangement order specified by the user.

See also **DefineGrid**.

Settings:

Select the defined grid

Select the name of a grid created by a previous **DefineGrid** module.

Name the objects to be identified

Enter the name of the grid objects identified by this module. These objects will be available for further measurement and processing in subsequent modules.

Select object shapes and locations

Use this setting to choose the method to be used to determine the grid objects' shapes and locations:

- *Rectangle Forced Location:* Each object will be created as a rectangle, completely occupying the entire grid compartment (rectangle). This option creates the rectangular objects based solely on the grid's specifications, not on any previously identified guiding objects.
- *Circle Forced Location:* Each object will be created as a circle, centered in the middle of each grid compartment. This option places the circular objects' locations based solely on the grid's specifications, not on any previously identified guiding objects. The radius of all circles in a grid will be constant for the entire grid in each image cycle, and can be determined automatically for each image cycle based on the average radius of previously identified guiding objects for that image cycle, or instead it can be specified as a single radius for all circles in all grids in the entire analysis run.
- *Circle Natural Location:* Each object will be created as a circle, and each circle's location within its grid compartment will be determined based on the location of any previously identified guiding objects within that grid compartment. Thus, if a guiding object lies within a particular grid compartment, that object's center will be the center of the created circular object. If no guiding objects lie within a particular grid compartment, the circular object is placed within the center of that grid compartment. If more than one guiding object lies within the grid compartment, they will be combined and the centroid of this combined object will be the location of the created circular object. Note that guiding objects whose centers are close to the grid edge are ignored.
- *Natural Shape and Location:* Within each grid compartment, the object will be identified based on combining all of the parts of guiding objects, if any, that fall within the grid compartment. Note that guiding objects whose centers are close to the grid edge are ignored. If a guiding object does not exist within a grid compartment, an object consisting of one single pixel in the middle of the grid compartment will be created.

Specify the circle diameter automatically?

(Used only if Circle is selected as object shape)
There are two methods for selecting the circle diameter:

- *Automatic:* Uses the average diameter of previously identified guiding objects as the diameter.
- *Manual:* Lets you specify the diameter directly, as a number.

Circle diameter

(Used only if Circle is selected as object shape and diameter is specified manually)
Enter the diameter to be used for each grid circle, in pixels. To measure distances in an open image, use the "Measure length" tool under *Tools* in the display window menu bar. If you click on an image and drag, a line will appear between the two endpoints, and the distance between them shown at the right-most portion of the bottom panel.

Select the guiding objects

(Used only if Circle is selected as object shape and diameter is specified automatically, or if Natural Location is selected as the object shape)
Select the names of previously identified objects that will be used to guide the shape and/or location of the objects created by this module, depending on the method chosen.

Retain outlines of the identified objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines**

module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: IdentifyObjectsManually

Identify Objects Manually allows you to identify objects in an image by hand rather than automatically.

This module lets you outline the objects in an image using the mouse. The user interface has several mouse tools:

- *Outline:* Lets you draw an outline around an object. Press the left mouse button at the start of the outline and draw the outline around your object. The tool will close your outline when you release the left mouse button.
- *Zoom in:* Lets you draw a rectangle and zoom the display to within that rectangle.
- *Zoom out:* Reverses the effect of the last zoom-in.
- *Erase:* Erases an object if you click on it.

Settings:

Select the input image

Choose the name of the image to display in the object selection user interface.

Name the objects to be identified

What do you want to call the objects that you identify using this module? You can use this name to refer to your objects in subsequent modules.

Retain outlines of the identified objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outlines

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: IdentifyPrimaryObjects

Identify Primary Objects identifies biological components of interest in grayscale images containing bright objects on a dark background.

What is a primary object?

In CellProfiler, we use the term *object* as a generic term to refer to an identifed feature in an image, usually a cellular subcompartment of some kind (for example, nuclei, cells, colonies, worms). We define an object as *primary* when it can be found in an image without needing the assistance of another cellular feature as a reference. For example:

- The nuclei of cells are usually more easily identifiable due to their more uniform morphology, high contrast relative to the background when stained, and good separation between adjacent nuclei. These qualities typically make them appropriate candidates for primary object identification.
- In contrast, cells often have irregular intensity patterns and are lower-contrast with more diffuse staining, making them more challenging to identify than nuclei. In addition, cells often touch their neighbors making it harder to delineate the cell borders. For these reasons, cell bodies are better suited for *secondary object* identification, since they are best identified by using a previously-identified primary object (i.e, the nuclei) as a reference. See the **IdentifySecondaryObjects** module for details on how to do this.

What do I need as input?

To use this module, you will need to make sure that your input image has the following qualities:

- The image should be grayscale.
- The foreground (i.e, regions of interest) are lighter than the background.

If this is not the case, other modules can be used to pre-process the images to ensure they are in the proper form:

- If the objects in your images are dark on a light background, you should invert the images using the Invert operation in the **ImageMath** module.
- If you are working with color images, they must first be converted to grayscale using the **ColorToGray** module.

If you have images in which the foreground and background cannot be distinguished by intensity alone (e.g, brightfield or DIC images), you can use the ilastik package bundled with CellProfiler to perform pixel-based classification (Windows only). You first train a classifier by identifying areas of images that fall into one of several classes, such as cell body, nucleus, background, etc. Then, the **ClassifyPixels** module takes the classifier and applies it to each image to identify areas that correspond to the trained classes. The result of **ClassifyPixels** is an image in which the region that falls into the class of interest is light on a dark background. Since this new image satisfies the constraints above, it can be used as input in **IdentifyPrimaryObjects**. See the **ClassifyPixels** module for more information.

What do the settings mean?

See below for help on the individual settings. The following icons are used to call attention to key items:

- Our recommendation or example use case for which a particular setting is best used.

- Indicates a condition under which a particular setting may not work well.
- Technical note. Provides more detailed information on the setting.

What do I get as output?

A set of primary objects are produced by this module, which can be used in downstream modules for measurement purposes or other operations. See the section "Available measurements" below for the measurements that are produced by this module.

Once the module has finished processing, the module display window will show the following panels:

- *Upper left:* The raw, original image.
- *Upper right:* The identified objects shown as a color image where connected pixels that belong to the same object are assigned the same color (*label image*). It is important to note that assigned colors are arbitrary; they are used simply to help you distingush the various objects.
- *Lower left:* The raw image overlaid with the colored outlines of the identified objects. Each object is assigned one of three (default) colors:
  - Green: Acceptable; passed all criteria
  - Magenta: Discarded based on size
  - Yellow: Discarded due to touching the border

  If you need to change the color defaults, you can make adjustments in *File > Preferences*.
- *Lower right:* A table showing some of the settings selected by the user, as well as those calculated by the module in order to produce the objects shown.

Available measurements

Image measurements:

- *Count:* The number of primary objects identified.
- *OriginalThreshold:* The global threshold for the image.
- *FinalThreshold:* For the global threshold methods, this value is the same as *OriginalThreshold*. For the adaptive or per-object methods, this value is the mean of the local thresholds.
- *WeightedVariance:* The sum of the log-transformed variances of the foreground and background pixels, weighted by the number of pixels in each distribution.
- *SumOfEntropies:* The sum of entropies computed from the foreground and background distributions.

Object measurements:

- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the primary object centroids. The centroid is calculated as the center of mass of the binary representation of the object.

Technical notes

CellProfiler contains a modular three-step strategy to identify objects even if they touch each other. It is based on previously published algorithms (*Malpica et al., 1997; Meyer and Beucher, 1990; Ortiz de Solorzano et al., 1999; Wahlby, 2003; Wahlby et al., 2004*). Choosing different options for each of these three steps allows CellProfiler to flexibly analyze a variety of different types of objects. The module has many options, which vary in terms of speed and sophistication. More detail can be found in the Settings section below. Here are the three steps, using an example where nuclei are the primary objects:

1. CellProfiler determines whether a foreground region is an individual nucleus or two or more clumped nuclei.
2. The edges of nuclei are identified, using thresholding if the object is a single, isolated nucleus, and

using more advanced options if the object is actually two or more nuclei that touch each other.
3. Some identified objects are discarded or merged together if they fail to meet certain your specified criteria. For example, partial objects at the border of the image can be discarded, and small objects can be discarded or merged with nearby larger ones. A separate module, **FilterObjects**, can further refine the identified nuclei, if desired, by excluding objects that are a particular size, shape, intensity, or texture.

References

- Malpica N, de Solorzano CO, Vaquero JJ, Santos, A, Vallcorba I, Garcia-Sagredo JM, del Pozo F (1997) "Applying watershed algorithms to the segmentation of clustered nuclei." *Cytometry* 28, 289-297. (link)
- Meyer F, Beucher S (1990) "Morphological segmentation." *J Visual Communication and Image Representation* 1, 21-46. (link)
- Ortiz de Solorzano C, Rodriguez EG, Jones A, Pinkel D, Gray JW, Sudar D, Lockett SJ. (1999) "Segmentation of confocal microscope images of cell nuclei in thick tissue sections." *Journal of Microscopy-Oxford* 193, 212-226. (link)
- Wählby C (2003) *Algorithms for applied digital image cytometry*, Ph.D., Uppsala University, Uppsala.
- Wählby C, Sintorn IM, Erlandsson F, Borgefors G, Bengtsson E. (2004) "Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections." *J Microsc* 215, 67-76. (link)

See also **IdentifySecondaryObjects**, **IdentifyTertiaryObjects**, **IdentifyObjectsManually** and **ClassifyPixels**

Settings:

Select the input image

Select the image that you want to use to identify objects.

Name the primary objects to be identified

Enter the name that you want to call the objects identified by this module.

Typical diameter of objects, in pixel units (Min,Max)

This setting allows the user to make a distinction on the basis of size, which can be used in conjunction with the *Discard objects outside the diameter range?* setting below to remove objects that fail this criteria.

 The units used here are pixels so that it is easy to zoom in on objects and determine typical diameters. To measure distances in an open image, use the "Measure length" tool under *Tools* in the display window menu bar. If you click on an image and drag, a line will appear between the two endpoints, and the distance between them shown at the right-most portion of the bottom panel.

A few important notes:

- Several other settings make use of the minimum object size entered here, whether the *Discard objects outside the diameter range?* setting is used or not:
  - *Automatically calculate size of smoothing filter for declumping?*
  - *Automatically calculate minimum allowed distance between local maxima?*
  - *Automatically calculate the size of objects for the Laplacian of Gaussian filter?* (shown only if

Laplacian of Gaussian is selected as the declumping method)

- For non-round objects, the diameter here is actually the "equivalent diameter", i.e., the diameter of a circle with the same area as the object.

Discard objects outside the diameter range?

Select *Yes* to discard objects outside the range you specified in the *Typical diameter of objects, in pixel units (Min,Max)* setting. Select *No* to ignore this criterion.

Objects discarded based on size are outlined in magenta in the module's display. See also the **FilterObjects** module to further discard objects based on some other measurement.

Select *Yes* allows you to exclude small objects (e.g., dust, noise, and debris) or large objects (e.g., large clumps) if desired.

Try to merge too small objects with nearby larger objects?

Select *Yes* to cause objects that are smaller than the specified minimum diameter to be merged, if possible, with other surrounding objects.

This is helpful in cases when an object was incorrectly split into two objects, one of which is actually just a tiny piece of the larger object. However, this could be problematic if the other settings in the module are set poorly, producing many tiny objects; the module will take a very long time trying to merge the tiny objects back together again; you may not notice that this is the case, since it may successfully piece together the objects again. It is therefore a good idea to run the module first without merging objects to make sure the settings are reasonably effective.

Discard objects touching the border of the image?

Select *Yes* to discard objects that touch the border of the image. Select *No* to ignore this criterion.

Removing objects that touch the image border is useful when you do not want to make downstream measurements of objects that are not fully within the field of view. For example, morphological measurements obtained from a portion of an object would not be accurate.

Objects discarded due to border touching are outlined in yellow in the module's display. Note that if a per-object thresholding method is used or if the image has been previously cropped or masked, objects that touch the border of the cropped or masked region may also discarded.

Threshold strategy

The thresholding strategy determines the type of input that is used to calculate the threshold. The image thresholds can be based on:

- The pixel intensities of the input image (this is the most common).
- A single value manually provided by the user.
- A single value produced by a prior module measurement.
- A binary image (called a *mask*) where some of the pixel intensity values are set to 0, and others are set to 1.

These options allow you to calculate a threshold based on the whole image or based on image sub-regions such as user-defined masks or objects supplied by a prior module.
The choices for the threshold strategy are:

- *Automatic:* Use the default settings for thresholding. This strategy calculates the threshold using the MCT method on the whole image (see below for details on this method) and applies the threshold to the image, smoothed with a Gaussian with sigma of 1.

    This approach is fairly robust, but does not allow you to select the threshold algorithm and does not allow you to apply additional corrections to the threshold.

- *Global:* Calculate a single threshold value based on the unmasked pixels of the input image and use that value to classify pixels above the threshold as foreground and below as background.

    This strategy is fast and robust, especially if the background is uniformly illuminated.

- *Adaptive:* Partition the input image into tiles and calculate thresholds for each tile. For each tile, the calculated threshold is applied only to the pixels within that tile.

    This method is slower but can produce better results for non-uniform backgrounds. However, for signifcant illumination variation, using the **CorrectIllumination** modules is preferable.

- *Per object:* Use objects from a prior module such as **IdentifyPrimaryObjects** to define the region of interest to be thresholded. Calculate a separate threshold for each object and then apply that threshold to pixels within the object. The pixels outside the objects are classified as background.

    This method can be useful for identifying sub-cellular particles or single-molecule probes if the background intensity varies from cell to cell (e.g., autofluorescence or other mechanisms).

- *Manual:* Enter a single value between zero and one that applies to all cycles and is independent of the input image.

    This approach is useful if the input image has a stable or negligible background, or if the input image is the probability map output of the **ClassifyPixels** module (in which case, a value of 0.5 should be chosen). If the input image is already binary (i.e., where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

- *Binary image:* Use a binary image to classify pixels as foreground or background. Pixel values other than zero will be foreground and pixel values that are zero will be background. This method can be used to import a ground-truth segmentation created by CellProfiler or another program.

    The most typical approach to produce a binary image is to use the **ApplyThreshold** module (image as input, image as output) or the **ConvertObjectsToImage** module (objects as input, image as output); both have options to produce a binary image. It can also be used to create objects from an image mask produced by other CellProfiler modules, such as **Morph**. Note that even though no algorithm is actually used to find the threshold in this case, the final threshold value is reported as the *Otsu* threshold calculated for the foreground region.

- *Measurement:* Use a prior image measurement as the threshold. The measurement should have values between zero and one. This strategy can be used to apply a pre-calculated threshold imported as per-image metadata via the **Metadata** module.

    Like manual thresholding, this approach can be useful when you are certain what the cutoff should be. The difference in this case is that the desired threshold does vary from image to image in the experiment but can be measured using another module, such as one of the **Measure** modules, **ApplyThreshold** or an **Identify** module.

Thresholding method

The intensity threshold affects the decision of whether each pixel will be considered foreground (region(s) of interest) or background. A higher threshold value will result in only the brightest regions being identified, whereas a lower threshold value will include dim regions. You can have the threshold automatically calculated from a choice of several methods, or you can enter a number manually between 0 and 1 for the threshold.

Both the automatic and manual options have advantages and disadvantages.

 An automatically-calculated threshold adapts to changes in lighting/staining conditions between images and is usually more robust/accurate. In the vast majority of cases, an automatic method is sufficient to achieve the desired thresholding, once the proper method is selected.
In contrast, an advantage of a manually-entered number is that it treats every image identically, so use this option when you have a good sense for what the threshold should be across all images. To help determine the choice of threshold manually, you can inspect the pixel intensities in an image of your choice. To view pixel intensities in an open image, use the pixel intensity tool which is available in any open display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window..
 The manual method is not robust with regard to slight changes in lighting/staining conditions between images.
The automatic methods may ocasionally produce a poor threshold for unusual or artifactual images. It also takes a small amount of time to calculate, which can add to processing time for analysis runs on a large number of images.

The threshold that is used for each image is recorded as a per-image measurement, so if you are surprised by unusual measurements from one of your images, you might check whether the automatically calculated threshold was unusually high or low compared to the other images. See the **FlagImage** module if you would like to flag an image based on the threshold value.

There are a number of methods for finding thresholds automatically:

- *Otsu:* This approach calculates the threshold separating the two classes of pixels (foreground and background) by minimizing the variance within the each class.

   This method is a good initial approach if you do not know much about the image characteristics of all the images in your experiment, especially if the percentage of the image covered by foreground varies substantially from image to image.

   Our implementation of Otsu's method allows for assigning the threhsold value based on splitting the image into either two classes (foreground and background) or three classes (foreground, mid-level, and background). See the help below for more details.

- *Mixture of Gaussian (MoG):*This function assumes that the pixels in the image belong to either a background class or a foreground class, using an initial guess of the fraction of the image that is covered by foreground.

   If you know that the percentage of each image that is foreground does not vary much from image to image, the MoG method can be better, especially if the foreground percentage is not near 50%.

   This method is our own version of a Mixture of Gaussians algorithm (*O. Friman, unpublished*). Essentially, there are two steps:
  1. First, a number of Gaussian distributions are estimated to match the distribution of pixel

intensities in the image. Currently three Gaussian distributions are fitted, one corresponding to a background class, one corresponding to a foreground class, and one distribution for an intermediate class. The distributions are fitted using the Expectation-Maximization algorithm, a procedure referred to as Mixture of Gaussians modeling.
2. When the three Gaussian distributions have been fitted, a decision is made whether the intermediate class more closely models the background pixels or foreground pixels, based on the estimated fraction provided by the user.

- *Background:* This method simply finds the mode of the histogram of the image, which is assumed to be the background of the image, and chooses a threshold at twice that value (which you can adjust with a Threshold Correction Factor; see below). The calculation includes those pixels between 2% and 98% of the intensity range.

    This thresholding method is appropriate for images in which most of the image is background. It can also be helpful if your images vary in overall brightness, but the objects of interest are consistently *N* times brighter than the background level of the image.

- *RobustBackground:* Much like the Background: method, this method is also simple and assumes that the background distribution approximates a Gaussian by trimming the brightest and dimmest 5% of pixel intensities. It then calculates the mean and standard deviation of the remaining pixels and calculates the threshold as the mean + 2 times the standard deviation.

    This thresholding method can be helpful if the majority of the image is background, and the results are often comparable or better than the *Background* method.

- *RidlerCalvard:* This method is simple and its results are often very similar to *Otsu*. *RidlerCalvard* chooses an initial threshold and then iteratively calculates the next one by taking the mean of the average intensities of the background and foreground pixels determined by the first threshold. The algorithm then repeats this process until the threshold converges to a single value.

    This is an implementation of the method described in Ridler and Calvard, 1978. According to Sezgin and Sankur 2004, Otsu's overall quality on testing 40 nondestructive testing images is slightly better than Ridler's (average error: Otsu, 0.318; Ridler, 0.401).

- *Kapur:* This method computes the threshold of an image by searching for the threshold that maximizes the sum of entropies of the foreground and background pixel values, when treated as separate distributions.

    This is an implementation of the method described in Kapur *et al*, 1985.

- *Maximum correlation thresholding (MCT):* This method computes the maximum correlation between the binary mask created by thresholding and the thresholded image and is somewhat similar mathematically to *Otsu*.

    The authors of this method claim superior results when thresholding images of neurites and other images that have sparse foreground densities.

    This is an implementation of the method described in Padmanabhan *et al*, 2010.

References

- Sezgin M, Sankur B (2004) "Survey over image thresholding techniques and quantitative performance evaluation." *Journal of Electronic Imaging*, 13(1), 146-165. (link)
- Padmanabhan K, Eddy WF, Crowley JC (2010) "A novel algorithm for optimal image thresholding of

biological data" *Journal of Neuroscience Methods* 193, 380-384. (link)
- Ridler T, Calvard S (1978) "Picture thresholding using an iterative selection method", *IEEE Transactions on Systems, Man and Cybernetics*, 8(8), 630-632.
- Kapur JN, Sahoo PK, Wong AKC (1985) "A new method of gray level picture thresholding using the entropy of the histogram." *Computer Vision, Graphics and Image Processing*, 29, 273-285.

Select binary image

(Used only if Binary image selected for thresholding method)
Select the binary image to be used for thresholding.

Manual threshold

(Used only if Manual selected for thresholding method)
Enter the value that will act as an absolute threshold for the images, a value from 0 to 1.

Select the measurement to threshold with

(Used only if Measurement is selected for thresholding method)
Choose the image measurement that will act as an absolute threshold for the images.

Two-class or three-class thresholding?

(Used only for the Otsu thresholding method)

- *Two classes:* Select this option if the grayscale levels are readily distinguishable into only two classes: foreground (i.e., regions of interest) and background.
- *Three classes*: Choose this option if the grayscale levels fall instead into three classes: foreground, background and a middle intensity between the two. You will then be asked whether the middle intensity class should be added to the foreground or background class in order to generate the final two-class output.

Note that whether two- or three-class thresholding is chosen, the image pixels are always finally assigned two classes: foreground and background.

> Three-class thresholding may be useful for images in which you have nuclear staining along with low-intensity non-specific cell staining. Where two-class thresholding might incorrectly assign this intermediate staining to the nuclei objects for some cells, three-class thresholding allows you to assign it to the foreground or background as desired.
>
> However, in extreme cases where either there are almost no objects or the entire field of view is covered with objects, three-class thresholding may perform worse than two-class.

Assign pixels in the middle intensity class to the foreground or the background?

(Used only for three-class thresholding)
Choose whether you want the pixels with middle grayscale intensities to be assigned to the foreground class or the background class.

Approximate fraction of image covered by objects?

(Used only when applying the MoG thresholding method)

Enter an estimate of how much of the image is covered with objects, which is used to estimate the distribution of pixel intensities.

Method to calculate adaptive window size

(Used only if an adaptive thresholding method is used)
The adaptive method breaks the image into blocks, computing the threshold for each block. There are two ways to compute the block size:

- *Image size:* The block size is one-tenth of the image dimensions, or 50 × 50 pixels, whichever is bigger.
- *Custom:* The block size is specified by the user.

Size of adaptive window

(Used only if an adaptive thresholding method with a Custom window size are selected)
Enter the window for the adaptive method. For example, you may want to use a multiple of the largest expected object size.

Threshold correction factor

This setting allows you to adjust the threshold as calculated by the above method. The value entered here adjusts the threshold either upwards or downwards, by multiplying it by this value. A value of 1 means no adjustment, 0 to 1 makes the threshold more lenient and > 1 makes the threshold more stringent.

> When the threshold is calculated automatically, you may find that the value is consistently too stringent or too lenient across all images. This setting is helpful for adjusting the threshold to a value that you empirically determine is more suitable. For example, the Otsu automatic thresholding inherently assumes that 50% of the image is covered by objects. If a larger percentage of the image is covered, the Otsu method will give a slightly biased threshold that may have to be corrected using this setting.

Lower and upper bounds on threshold

Enter the minimum and maximum allowable threshold, a value from 0 to 1. This is helpful as a safety precaution when the threshold is calculated automatically, by overriding the automatic threshold.

> For example, if there are no objects in the field of view, the automatic threshold might be calculated as unreasonably low; the algorithm will still attempt to divide the foreground from background (even though there is no foreground), and you may end up with spurious false positive foreground regions. In such cases, you can estimate the background pixel intensity and set the lower bound according to this empirically-determined value.
> To view pixel intensities in an open image, use the pixel intensity tool which is available in any open display window. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the display window.

Select the smoothing method for thresholding

(Only used for strategies other than Automatic and Binary image)
The input image can be optionally smoothed before being thresholded. Smoothing can improve the uniformity of the resulting objects, by removing holes and jagged edges caused by noise in the acquired image. Smoothing is most likely *not* appropriate if the input image is binary, if it has already been

smoothed or if it is an output of the *ClassifyPixels* module.
The choices are:

- *Automatic*: Smooth the image with a Gaussian with a sigma of one pixel before thresholding. This is suitable for most analysis applications.
- *Manual*: Smooth the image with a Gaussian with user-controlled scale.
- *No smoothing*: Do not apply any smoothing prior to thresholding.

Threshold smoothing scale

(Only used if smoothing for threshold is Manual)
This setting controls the scale used to smooth the input image before the threshold is applied. The scale should be approximately the size of the artifacts to be eliminated by smoothing. A Gaussian is used with a sigma adjusted so that 1/2 of the Gaussian's distribution falls within the diameter given by the scale (sigma = scale / 0.674)

Automatically calculate the size of objects for the Laplacian of Gaussian filter?

(Used only when applying the LoG thresholding method)

Select *Yes* to use the filtering diameter range above when constructing the LoG filter.

Select *No* in order to manually specify the size. You may want to specify a custom size if you want to filter using loose criteria, but have objects that are generally of similar sizes.

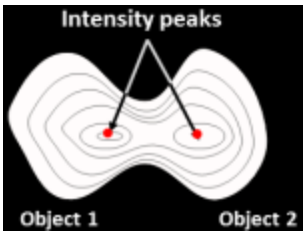
Enter LoG filter diameter

(Used only when applying the LoG thresholding method)
The size to use when calculating the LoG filter. The filter enhances the local maxima of objects whose diameters are roughly the entered number or smaller.

Method to distinguish clumped objects

This setting allows you to choose the method that is used to segment objects, i.e., "declump" a large, merged object into individual objects of interest. To decide between these methods, you can run Test mode to see the results of each.

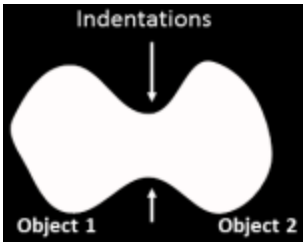
- *Intensity:* For objects that tend to have only a single peak of brightness (e.g. objects that are brighter towards their interiors and dimmer towards their edges), this option counts each intensity peak as a separate object. The objects can be any shape, so they need not be round and uniform in size as would be required for the *Shape* option.



  This choice is more successful when the objects have a smooth texture. By default, the image is automatically blurred to attempt to achieve appropriate smoothness (see *Smoothing filter* options), but overriding the default value can improve the outcome on lumpy-textured objects.

  The object centers are defined as local intensity maxima in the smoothed image.

- *Shape:* For cases when there are definite indentations separating objects. The image is converted to black and white (binary) and the shape determines whether clumped objects will be distinguished. The declumping results of this method are affected by the thresholding method you choose.



  This choice works best for objects that are round. In this case, the intensity patterns in the original image are largely irrelevant. Therefore, the cells need not be brighter towards the interior as is required for the *Intensity* option.

  The binary thresholded image is distance-transformed and object centers are defined as peaks in this image. A distance-transform gives each pixel a value equal to the distance to the nearest pixel below a certain threshold, so it indicates the *Shape* of the object.

- *Laplacian of Gaussian:* For objects that have an increasing intensity gradient toward their center, this option performs a Laplacian of Gaussian (or Mexican hat) transform on the image, which accentuates pixels that are local maxima of a desired size. It thresholds the result and finds pixels that are both local maxima and above threshold. These pixels are used as the seeds for objects in the watershed.
- *None:* If objects are well separated and bright relative to the background, it may be unnecessary to attempt to separate clumped objects. Using the very fast *None* option, a simple threshold will be used to identify objects. This will override any declumping method chosen in the settings below.

Method to draw dividing lines between clumped objects

This setting allows you to choose the method that is used to draw the line bewteen segmented objects, provided that you have chosen to declump the objects. To decide between these methods, you can run Test mode to see the results of each.

- *Intensity:* Works best where the dividing lines between clumped objects are dimmer than the remainder of the objects.

  Technical description: Using the previously identified local maxima as seeds, this method is a watershed (*Vincent and Soille, 1991*) on the intensity image.

- *Shape:* Dividing lines between clumped objects are based on the shape of the clump. For example, when a clump contains two objects, the dividing line will be placed where indentations occur between the two objects. The intensity patterns in the original image are largely irrelevant: the cells need not be dimmer along the lines between clumped objects. Technical description: Using the previously identified local maxima as seeds, this method is a watershed on the distance-transformed thresholded image.
- *Propagate:* This method uses a propagation algorithm instead of a watershed. The image is ignored and the pixels are assigned to the objects by repeatedly adding unassigned pixels to the objects that are immediately adjacent to them. This method is suited in cases such as objects with branching extensions, for instance neurites, where the goal is to trace outward from the cell body along the branch, assigning pixels in the branch along the way. See the help for the **IdentifySecondary** module for more details on this method.
- *None*: If objects are well separated and bright relative to the background, it may be unnecessary to attempt to separate clumped objects. Using the very fast *None* option, a simple threshold will be used to identify objects. This will override any declumping method chosen in the previous question.

Automatically calculate size of smoothing filter for declumping?

(Used only when distinguishing between clumped objects)
Select *Yes* to automatically calculate the amount of smoothing applied to the image to assist in declumping. Select *No* to manually enter the smoothing filter size.

This setting, along with the *Minimum allowed distance between local maxima* setting, affects whether objects close to each other are considered a single object or multiple objects. It does not affect the dividing lines between an object and the background.

Please note that this smoothing setting is applied after thresholding, and is therefore distinct from the threshold smoothing method setting above, which is applied *before* thresholding.

The size of the smoothing filter is automatically calculated based on the *Typical diameter of objects, in pixel units (Min,Max)* setting above. If you see too many objects merged that ought to be separate or too many objects split up that ought to be merged, you may want to override the automatically calculated value.

Size of smoothing filter

(Used only when distinguishing between clumped objects)
If you see too many objects merged that ought to be separated (under-segmentation), this value should be lower. If you see too many objects split up that ought to be merged (over-segmentation), the value should be higher. Enter 0 to prevent any image smoothing in certain cases; for example, for low resolution images with small objects (< ~5 pixels in diameter).

Reducing the texture of objects by increasing the smoothing increases the chance that each real, distinct object has only one peak of intensity but also increases the chance that two distinct objects will be recognized as only one object. Note that increasing the size of the smoothing filter increases the processing time exponentially.

Automatically calculate minimum allowed distance between local maxima?

(Used only when distinguishing between clumped objects)
Select *Yes* to automatically calculate the distance between intensity maxima to assist in declumping. Select *No* to manually enter the permissible maxima distance.

This setting, along with the *Size of smoothing filter* setting, affects whether objects close to each other are considered a single object or multiple objects. It does not affect the dividing lines between an object and the background. Local maxima that are closer together than the minimum allowed distance will be suppressed (the local intensity histogram is smoothed to remove the peaks within that distance). The distance can be automatically calculated based on the minimum entered for the *Typical diameter of objects, in pixel units (Min,Max)* setting above, but if you see too many objects merged that ought to be separate, or too many objects split up that ought to be merged, you may want to override the automatically calculated value.

Suppress local maxima that are closer than this minimum allowed distance

(Used only when distinguishing between clumped objects)
Enter a positive integer, in pixel units. If you see too many objects merged that ought to be separated (under-segmentation), the value should be lower. If you see too many objects split up that ought to be merged (over-segmentation), the value should be higher.

The maxima suppression distance should be set to be roughly equivalent to the minimum radius of a real object of interest. Any distinct "objects" which are found but are within two times this distance from each other will be assumed to be actually two lumpy parts of the same object, and they will be merged.

Speed up by using lower-resolution image to find local maxima?

(Used only when distinguishing between clumped objects)
Select *Yes* to down-sample the image for declumping. This can be helpful for saving processing time on large images.

Note that if you have entered a minimum object diameter of 10 or less, checking this box will have no effect.

Retain outlines of the identified objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Fill holes in identified objects?

Select *After both thresholding and declumping* to fill in background holes that are smaller than the maximum object size prior to declumping and to fill in any holes after declumping. Select *After declumping only to fill in background holes located within identified objects after declumping. Select Never to leave holes within objects.*

Please note that if a foreground object is located within a hole and this option is enabled, the object will be lost when the hole is filled in.

Handling of objects if excessive number of objects identified

This setting deals with images that are segmented into an unreasonable number of objects. This might happen if the module calculates a low threshold or if the image has unusual artifacts.
***IdentifyPrimaryObjects** can handle this condition in one of three ways:*

- *Continue: Don't check for large numbers of objects.*
- *Truncate: Limit the number of objects. Arbitrarily erase objects to limit the number to the maximum allowed.*
- *Erase: Erase all objects if the number of objects exceeds the maximum. This results in an image with no primary objects. This option is a good choice if a large number of objects indicates that the image should not be processed.*

Maximum number of objects

(Used only when handling images with large numbers of objects by truncating)
This setting limits the number of objects in the image. See the documentation for the previous setting for

details.

Module: IdentifySecondaryObjects

Identify Secondary Objects identifies objects (e.g., cell edges) using objects identified by another module (e.g., nuclei) as a starting point.

What is a secondary object?

In CellProfiler, we use the term *object* as a generic term to refer to an identifed feature in an image, usually a cellular subcompartment of some kind (for example, nuclei, cells, colonies, worms). We define an object as *secondary* when it can be found in an image by using another cellular feature as a reference for guiding detection.

For densely-packed cells (such as those in a confluent monolayer), determining the cell borders using a cell body stain can be quite difficult since they often have irregular intensity patterns and are lower-contrast with more diffuse staining. In addition, cells often touch their neighbors making it harder to delineate the cell borders. It is often easier to identify an organelle which is well separated spatially (such as the nucleus) as an object first and then use that object to guide the detection of the cell borders. See the **IdentifyPrimaryObjects** module for details on how to identify a primary object.

In order to identify the edges of secondary objects, this module performs two tasks:

1. Finds the dividing lines between secondary objects which touch each other.
2. Finds the dividing lines between the secondary objects and the background of the image. In most cases, this is done by thresholding the image stained for the secondary objects.

What do I need as input?

This module identifies secondary objects based on two types of input:

1. An *object* (e.g., nuclei) identified from a prior module. These are typically produced by an **IdentifyPrimaryObjects** module, but any object produced by another module may be selected for this purpose.
2. An *image* highlighting the image features defining the cell edges. This is typically a fluorescent stain for the cell body, membrane or cytoskeleton (e.g., phalloidin staining for actin). However, any image which produces these features can be used for this purpose. For example, an image processing module might be used to transform a brightfield image into one which captures the characteristics of a cell body flourescent stain.

What do the settings mean?

See below for help on the individual settings. The following icons are used to call attention to key items:

- 👍 Our recommendation or example use case for which a particular setting is best used.
- 👎 Indicates a condition under which a particular setting may not work well.
- ⚙ Technical note. Provides more detailed information on the setting, if interested.

What do I get as output?

A set of secondary objects are produced by this module, which can be used in downstream modules for measurement purposes or other operations. Because each primary object is used as the starting point for

producing a corresponding secondary object, keep in mind the following points:

- The primary object will always be completely contained within a secondary object. For example, nuclei are completely enclosed within identified cells stained for actin.
- There will always be at most one secondary object for each primary object.

See the section "Available measurements" below for the measurements that are produced by this module.

Once the module has finished processing, the module display window will show the following panels:

- *Upper left:* The raw, original image.
- *Upper right:* The identified objects shown as a color image where connected pixels that belong to the same object are assigned the same color (*label image*). It is important to note that assigned colors are arbitrary; they are used simply to help you distingush the various objects.
- *Lower left:* The raw image overlaid with the colored outlines of the identified secondary objects. The objects are shown with the following colors:
  - Magenta: Secondary objects
  - Green: Primary objects

  If you need to change the color defaults, you can make adjustments in *File* > *Preferences*.
- *Lower right:* A table showing some of the settings selected by the user, as well as those calculated by the module in order to produce the objects shown.

Available measurements

Image measurements:

- *Count:* The number of secondary objects identified.
- *OriginalThreshold:* The global threshold for the image.
- *FinalThreshold:* For the global threshold methods, this value is the same as *OriginalThreshold*. For the adaptive or per-object methods, this value is the mean of the local thresholds.
- *WeightedVariance:* The sum of the log-transformed variances of the foreground and background pixels, weighted by the number of pixels in each distribution.
- *SumOfEntropies:* The sum of entropies computed from the foreground and background distributions.

Object measurements:

- *Parent:* The identity of the primary object associated with each secondary object.
- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the identified secondary objects.

Technical notes

The *Propagation* algorithm is the default approach for secondary object creation, creating each primary object as a "seed" guided by the input image and limited to the foreground region as determined by the chosen thresholding method. λ is a regularization parameter; see the help for the setting for more details. Propagation of secondary object labels is by the shortest path to an adjacent primary object from the starting ("seeding") primary object. The seed-to-pixel distances are calculated as the sum of absolute differences in a 3x3 (8-connected) image neighborhood, combined with λ via sqrt(differences^2 + λ^2).

See also the other **Identify** modules.

Settings:

Select the input objects

What did you call the objects you want to use as "seeds" to identify a secondary object around each one? By definition, each primary object must be associated with exactly one secondary object and completely contained within it.

Name the objects to be identified

Enter the name that you want to call the objects identified by this module.

Select the method to identify the secondary objects

There are several methods available to find the dividing lines between secondary objects which touch each other:

- *Propagation:* This method will find dividing lines between clumped objects where the image stained for secondary objects shows a change in staining (i.e., either a dimmer or a brighter line). Smoother lines work better, but unlike the Watershed method, small gaps are tolerated. This method is considered an improvement on the traditional *Watershed* method. The dividing lines between objects are determined by a combination of the distance to the nearest primary object and intensity gradients. This algorithm uses local image similarity to guide the location of boundaries between cells. Boundaries are preferentially placed where the image's local appearance changes perpendicularly to the boundary (*Jones et al, 2005*).
- *Watershed - Gradient:* This method uses the watershed algorithm (*Vincent and Soille, 1991*) to assign pixels to the primary objects which act as seeds for the watershed. In this variant, the watershed algorithm operates on the Sobel transformed image which computes an intensity gradient. This method works best when the image intensity drops off or increases rapidly near the boundary between cells.
- *Watershed - Image:* This method is similar to the above, but it uses the inverted intensity of the image for the watershed. The areas of lowest intensity will form the boundaries between cells. This method works best when there is a saddle of relatively low intensity at the cell-cell boundary.
- *Distance:* In this method, the edges of the primary objects are expanded a specified distance to create the secondary objects. For example, if nuclei are labeled but there is no stain to help locate cell edges, the nuclei can simply be expanded in order to estimate the cell's location. This is often called the "doughnut" or "annulus" or "ring" approach for identifying the cytoplasm. There are two methods that can be used:
  - *Distance - N*: In this method, the image of the secondary staining is not used at all; the expanded objects are the final secondary objects.
  - *Distance - B*: Thresholding of the secondary staining image is used to eliminate background regions from the secondary objects. This allows the extent of the secondary objects to be limited to a certain distance away from the edge of the primary objects without including regions of background.

References

- Jones TR, Carpenter AE, Golland P (2005) "Voronoi-Based Segmentation of Cells on Image Manifolds", *ICCV Workshop on Computer Vision for Biomedical Image Applications*, 535-543. (link)
- (Vincent L, Soille P (1991) "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations", *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 13(6): 583-598 (link)

Select the input image

The selected image will be used to find the edges of the secondary objects. For *Distance - N* this will not affect object identification, only the final display.

Threshold strategy

The thresholding strategy determines the type of input that is used to calculate the threshold. The image thresholds can be based on:

- The pixel intensities of the input image (this is the most common).
- A single value manually provided by the user.
- A single value produced by a prior module measurement.
- A binary image (called a *mask*) where some of the pixel intensity values are set to 0, and others are set to 1.

These options allow you to calculate a threshold based on the whole image or based on image sub-regions such as user-defined masks or objects supplied by a prior module.
The choices for the threshold strategy are:

- *Automatic:* Use the default settings for thresholding. This strategy calculates the threshold using the MCT method on the whole image (see below for details on this method) and applies the threshold to the image, smoothed with a Gaussian with sigma of 1.

    This approach is fairly robust, but does not allow you to select the threshold algorithm and does not allow you to apply additional corrections to the threshold.

- *Global:* Calculate a single threshold value based on the unmasked pixels of the input image and use that value to classify pixels above the threshold as foreground and below as background.

    This strategy is fast and robust, especially if the background is uniformly illuminated.

- *Adaptive:* Partition the input image into tiles and calculate thresholds for each tile. For each tile, the calculated threshold is applied only to the pixels within that tile.

    This method is slower but can produce better results for non-uniform backgrounds. However, for signifcant illumination variation, using the **CorrectIllumination** modules is preferable.

- *Per object:* Use objects from a prior module such as **IdentifyPrimaryObjects** to define the region of interest to be thresholded. Calculate a separate threshold for each object and then apply that threshold to pixels within the object. The pixels outside the objects are classified as background.

    This method can be useful for identifying sub-cellular particles or single-molecule probes if the background intensity varies from cell to cell (e.g., autofluorescence or other mechanisms).

- *Manual:* Enter a single value between zero and one that applies to all cycles and is independent of the input image.

    This approach is useful if the input image has a stable or negligible background, or if the input image is the probability map output of the **ClassifyPixels** module (in which case, a value of 0.5 should be chosen). If the input image is already binary (i.e., where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

- *Binary image:* Use a binary image to classify pixels as foreground or background. Pixel values other

than zero will be foreground and pixel values that are zero will be background. This method can be used to import a ground-truth segmentation created by CellProfiler or another program.

The most typical approach to produce a binary image is to use the **ApplyThreshold** module (image as input, image as output) or the **ConvertObjectsToImage** module (objects as input, image as output); both have options to produce a binary image. It can also be used to create objects from an image mask produced by other CellProfiler modules, such as **Morph**. Note that even though no algorithm is actually used to find the threshold in this case, the final threshold value is reported as the *Otsu* threshold calculated for the foreground region.

- *Measurement:* Use a prior image measurement as the threshold. The measurement should have values between zero and one. This strategy can be used to apply a pre-calculated threshold imported as per-image metadata via the **Metadata** module.

Like manual thresholding, this approach can be useful when you are certain what the cutoff should be. The difference in this case is that the desired threshold does vary from image to image in the experiment but can be measured using another module, such as one of the **Measure** modules, **ApplyThreshold** or an **Identify** module.

Regularization factor

(Used only if Propagation method is selected)
The regularization factor λ can be anywhere in the range 0 to infinity. This method takes two factors into account when deciding where to draw the dividing line between two touching secondary objects: the distance to the nearest primary object, and the intensity of the secondary object image. The regularization factor controls the balance between these two considerations:

- A λ value of 0 means that the distance to the nearest primary object is ignored and the decision is made entirely on the intensity gradient between the two competing primary objects.
- Larger values of λ put more and more weight on the distance between the two objects. This relationship is such that small changes in λ will have fairly different results (e.,g 0.01 vs 0.001). However, the intensity image is almost completely ignored at λ much greater than 1.
- At infinity, the result will look like Distance - B, masked to the secondary staining image.

Fill holes in identified objects?

Select *Yes* to fill any holes inside objects.

Discard secondary objects touching the border of the image?

Select *Yes* to discard secondary objects which touch the image border. Select *No* to retain objects regardless of whether they touch the image edge or not.

The objects are discarded with respect to downstream measurement modules, but they are retained in memory as "unedited objects"; this allows them to be considered in downstream modules that modify the segmentation.

Discard the associated primary objects?

(Used only if discarding secondary objects touching the image border)
It might be appropriate to discard the primary object for any secondary object that touches the edge of the image.

Select *Yes* to create a new set of objects that are identical to the original primary objects set, minus the objects for which the associated secondary object touches the image edge.

Name the new primary objects

(Used only if associated primary objects are discarded)
You can name the primary objects that remain after the discarding step. These objects will all have secondary objects that do not touch the edge of the image. Note that any primary object whose secondary object touches the edge will be retained in memory as an "unedited object"; this allows them to be considered in downstream modules that modify the segmentation.

Retain outlines of the new primary objects?

(Used only if associated primary objects are discarded)
Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the new primary object outlines

(Used only if associated primary objects are discarded and saving outlines of new primary objects)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Retain outlines of the identified secondary objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: IdentifyTertiaryObjects

Identify Tertiary Objects identifies tertiary objects (e.g., cytoplasm) by removing smaller primary objects (e.g. nuclei) from larger secondary objects (e.g., cells), leaving a ring shape.

What is a tertiary object?

In CellProfiler, we use the term *object* as a generic term to refer to an identifed feature in an image, usually a cellular subcompartment of some kind (for example, nuclei, cells, colonies, worms). We define an object as *tertiary* when it is identified by using a prior primary and secondary objects for reference. A common use case is when nuclei have been found using **IdentifyPrimaryObjects** and the cell body has been found using **IdentifySecondaryObjects** but measurements from the cytoplasm, the region outside the nucleus but within the cell body, are desired. This module may be used to define the cytoplasm as an new object.

What do I need as input?

This module will take the smaller identified objects and remove them from the larger identified objects. For example, "subtracting" the nuclei from the cells will leave just the cytoplasm, the properties of which can then be measured by downstream **Measure** modules. The larger objects should therefore be equal in size or larger than the smaller objects and must completely contain the smaller objects; **IdentifySecondaryObjects** will produce objects that satisfy this constraint. Ideally, both inputs should be objects produced by prior **Identify** modules.

What do I get as output?

A set of tertiary objects are produced by this module, which can be used in downstream modules for measurement purposes or other operations. Because each tertiary object is produced from primary and secondary objects, there will always be at most one secondary object for each primary object. See the section "Available measurements" below for the measurements that are produced by this module.

Note that creating subregions using this module can result in objects with a single label that nonetheless are not contiguous. This may lead to unexpected results when running measurment modules such as **MeasureObjectSizeShape** because calculations of the perimeter, aspect ratio, solidity, etc. typically make sense only for contiguous objects. Other modules, such as **MeasureImageIntensity** and **MeasureTexture** modules, are not affected and will yield expected results.

Available measurements

Image measurements:

- *Count:* The number of tertiary objects identified.

Object measurements:

- *Parent:* The identity of the primary object and secondary object associated with each tertiary object.
- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the identified tertiary objects.

See also **IdentifyPrimaryObject** and **IdentifySecondaryObject** modules.

Settings:

Select the larger identified objects

Select the larger identified objects. This will usually be an object previously identified by a **IdentifySecondaryObjects** module.

Select the smaller identified objects

Select the smaller identified objects. This will usually be an object previously identified by a **IdentifyPrimaryObjects** module.

Name the tertiary objects to be identified

Enter a name for the new tertiary objects. The tertiary objects will consist of the smaller object subtracted from the larger object.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Retain outlines of the tertiary objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Shrink smaller object prior to subtraction?

Select *Yes* to shrink the smaller object by 1 pixel before subtracting the objects. this approach will ensure that there is always a tertiary object produced, even if it is only 1 pixel wide.

Select *No* to subtract the objects directly, which will ensure that no pixels are shared between the primary/secondary/tertiary objects and hence measurements for all three sets of objects will not use the same pixels multiple times. However, this may result in the creation of objects with no area. Measurements can still be made on such objects, but the results will be zero or not-a-number (NaN)

Module: MaskObjects

Mask Objects removes objects outside of a specified region or regions.

This module allows you to delete the objects or portions of objects that are outside of a region (mask) you specify. For example, after identifying nuclei and tissue regions in previous **Identify** modules, you might want to exclude all nuclei that are outside of a tissue region.

If using a masking image, the mask is composed of the foreground (white portions); if using a masking object, the mask is composed of the area within the object. You can choose to remove only the portion of each object that is outside of the region, remove the whole object if it is partially or fully outside of the region, or retain the whole object unless it is fully outside of the region.

Available measurements

Parent object measurements:

- *Count:* The number of new masked objects created from each parent object.

Masked object measurements:

- *Parent:* The label number of the parent object.
- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the masked objects.

Settings:

Select objects to be masked

Select the objects that will be masked (that is, excluded in whole or in part based on the other settings in the module). You can choose from any objects created by a previous object processing module, such as **IdentifyPrimaryObjects**, **IdentifySecondaryObjects** or **IdentifyTertiaryObjects**.

Name the masked objects

Enter a name for the objects that remain after the masking operation. You can refer to the masked objects in subsequent modules by this name.

Mask using a region defined by other objects or by binary image?

You can mask your objects by defining a region using objects you previously identified in your pipeline (*Objects*) or by defining a region based on the white regions in a binary image (*Image*).

Select the masking object

Select the objects that will be used to define the masking region. You can choose from any objects created by a previous object processing module, such as **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, or **IdentifyTertiaryObjects**.

Select the masking image

Select an image that was either loaded or created by a previous module. The image should be a binary image where the white portion of the image is the region(s) you will use for masking. Binary images can be loaded from disk using the **NamesAndTypes** module by selecting "Binary mask" for the image type. You can also create a binary image from a grayscale image using **ApplyThreshold**.

Invert the mask?

This option reverses the foreground/background relationship of the mask.

- Select *No* for the mask to be composed of the foregound (white portion) of the masking image or the area within the masking objects.
- Select *Yes* for the mask to instead be composed of the *background* (black portions) of the masking image or the area *outside* the masking objects.

Handling of objects that are partially masked

An object might partially overlap the mask region, with pixels both inside and outside the region. **MaskObjects** can handle this in one of three ways:

- *Keep overlapping region:* Choosing this option will reduce the size of partially overlapping objects. The part of the object that overlaps the region will be retained. The part of the object that is outside of the region will be removed.
- *Keep:* If you choose this option, **MaskObjects** will keep the whole object if any part of it overlaps the masking region.
- *Remove:* Objects that are partially outside of the masking region will be completely removed if you choose this option.
- *Remove depending on overlap:* Determine whether to remove or keep an object depending on how much of the object overlaps the masking region. **MaskObjects** will keep an object if at least a certain fraction (which you enter below) of the object falls within the masking region. **MaskObjects** completely removes the object if too little of it overlaps the masking region.

Fraction of object that must overlap

(Used only if removing based on a overlap)
Specify the minimum fraction of an object that must overlap the masking region for that object to be retained. For instance, if the fraction is 0.75, then 3/4 of an object must be within the masking region for that object to be retained.

Numbering of resulting objects

Choose how to number the objects that remain after masking, which controls how remaining objects are associated with their predecessors:

- *Renumber:* The objects that remain will be renumbered using consecutive numbers. This is a good choice if you do not plan to use measurements from the original objects; your object measurements for the masked objects will not have gaps (where removed objects are missing).
- *Retain:*: The original labels for the objects will be retained. This allows any measurements you make from the masked objects to be directly aligned with measurements you might have made of the original, unmasked objects (or objects directly associated with them).

Retain outlines of the resulting objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Module: ReassignObjectNumbers

Reassign Object Numbers renumbers previously identified objects.

Objects and their measurements are associated with each other based on their object numbers (also known as *labels*). Typically, each object is assigned a single unique number, such that the exported measurements are ordered by this numbering. This module allows the reassignment of object numbers by either unifying separate objects to share the same label, or splitting portions of separate objects that previously had the same label.

Available measurements

Parent object measurements:

- *Children Count:* The number of relabeled objects created from each parent object.

Reassigned object measurements:

- *Parent:*The label number of the parent object.
- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the center of mass of the reassigned objects.

Technical notes

Reassignment means that the numerical value of every pixel within an object (in the label matrix version of the image) gets changed, as specified by the module settings. In order to ensure that objects are labeled consecutively without gaps in the numbering (which other modules may depend on), **ReassignObjectNumbers** will typically result in most of the objects having their numbers reordered. This reassignment information is stored as a per-object measurement with both the original input and reasigned output objects, in case you need to track the reassignment.

See also **RelateObjects**.

Settings:

Select the input objects

Select the objects whose object numbers you want to reassign. You can use any objects that were created in previous modules, such as **IdentifyPrimaryObjects** or **IdentifySecondaryObjects**.

Name the new objects

Enter a name for the objects whose numbers have been reassigned. You can use this name in subsequent modules that take objects as inputs.

Operation

You can choose one of the following options:

- *Unify:* Assign adjacent or nearby objects the same label based on certain criteria. It can be useful,

for example, to merge together touching objects that were incorrectly split into two pieces by an **Identify** module.
- *Split:* Assign a unique number to separate objects that currently share the same label. This can occur if you applied certain operations with the **Morph** module to objects.

Maximum distance within which to unify objects

(Used only with the Unify option and the Distance method)
Objects that are less than or equal to the distance you enter here, in pixels, will be unified. If you choose zero (the default), only objects that are touching will be unified. Note that *Unify* will not actually connect or bridge the two objects by adding any new pixels; it simply assigns the same object number to the portions of the object. The new, unified object may therefore consist of two or more unconnected components.

Unify using a grayscale image?

(Used only with the Unify option)
Select *Yes* to use the objects' intensity features to determine whether two objects should be unified. If you choose to use a grayscale image, *Unify* will unify two objects only if they are within the distance you have specified *and* certain criteria about the objects within the grayscale image are met.

Select the grayscale image to guide unification

(Used only if a grayscale image is to be used as a guide for unification)
Select the name of an image loaded or created by a previous module.

Minimum intensity fraction

(Used only if a grayscale image is to be used as a guide for unification)
Select the minimum acceptable intensity fraction. This will be used as described for the method you choose in the next setting.

Method to find object intensity

(Used only if a grayscale image is to be used as a guide for unification)
You can use one of two methods to determine whether two objects should unified, assuming they meet the distance criteria (as specified above):

- *Centroids:* When the module considers merging two objects, this method identifies the centroid of each object, records the intensity value of the dimmer of the two centroids, multiplies this value by the *minimum intensity fraction* to generate a threshold, and draws a line between the centroids. The method will unify the two objects only if the intensity of every point along the line is above the threshold. For instance, if the intensity of one centroid is 0.75 and the other is 0.50 and the *minimum intensity fraction* has been chosen to be 0.9, all points along the line would need to have an intensity of min(0.75, 0.50) * 0.9 = 0.50 * 0.9 = 0.45.
  This method works well for round cells whose maximum intensity is in the center of the cell: a single cell that was incorrectly segmented into two objects will typically not have a dim line between the centroids of the two halves and will be correctly unified.
- *Closest point:* This method is useful for unifying irregularly shaped cells which are connected. It starts by assigning background pixels in the vicinity of the objects to the nearest object. Objects are then unified if each object has background pixels that are:
  - Within a distance threshold from each object;
  - Above the minimum intensity fraction of the nearest object pixel;

- Adjacent to background pixels assigned to a neighboring object.

An example of a feature that satisfies the above constraints is a line of pixels that connect two neighboring objects and is roughly the same intensity as the boundary pixels of both (such as an axon connecting two neurons).

Retain outlines of the relabeled objects?

Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the outlines

(Used only if the outline image is to be retained for later use in the pipeline)
Enter a name for the outlines of the identified objects. The outlined image can be selected in downstream modules by selecting them from any drop-down image list.

Unification method

(Used only with the Unify option)
You can unify objects in one of two ways:

- *Distance:* All objects within a certain pixel radius from each other will be unified
- *Per-parent:* All objects which share the same parent relationship to another object will be unified. This is not be confused with using the **RelateObjects** module, in which the related objects remain as individual objects. See **RelateObjects** for more details.

Select the parent object

Select the parent object that will be used to unify the child objects. Please note the following:

- You must have established a parent-child relationship between the objects using a prior **RelateObjects** module.
- Primary objects and their associated secondary objects are already in a one-to-one parent-child relationship, so it makes no sense to unify them here.

Module: RelateObjects

Relate Objects assigns relationships; all objects (e.g. speckles) within a parent object (e.g. nucleus) become its children.

This module allows you to associate *child* objects with *parent* objects. This is useful for counting the number of children associated with each parent, and for calculating mean measurement values for all children that are associated with each parent.

An object will be considered a child even if the edge is the only part touching a parent object. If an child object is touching multiple parent objects, the object will be assigned as a child of all parents that it overlaps with.

Available measurements

Parent object measurements:

- *Count:* The number of child sub-objects for each parent object.
- *Mean measurements:* The mean of the child object measurements, calculated for each parent object.
- *Distances:* The distance of each child object to its respective parent.

Child object measurements:

- *Parent:* The label number of the parent object, as assigned by an **Identify** module.

See also: **ReassignObjectNumbers**.

Settings:

Select the input child objects

Child objects are defined as those objects contained within the parent object. For example, when relating speckles to the nuclei that contains them, the speckles are the children.

Select the input parent objects

Parent objects are defined as those objects which encompass the child object. For example, when relating speckles to the nuclei that contains them, the nuclei are the parents.

Calculate child-parent distances?

Choose the method to calculate distances of each child to its parent.

- *None:* Do not calculate any distances.
- *Minimum:* The distance from the centroid of the child object to the closest perimeter point on the parent object.
- *Centroid:* The distance from the centroid of the child object to the centroid of the parent.
- *Both:* Calculate both the *Minimum* and *Centroid* distances.

Calculate per-parent means for all child measurements?

Select *Yes* to calculate the per-parent mean values of every upstream measurement made with the children objects and stores them as a measurement for the parent; the nomenclature of this new measurements is "Mean_<child>_<category>_<feature>". For this reason, this module should be placed *after* all **Measure** modules that make measurements of the children objects.

Calculate distances to other parents?

(Used only if calculating distances)
Select *Yes* to calculate the distances of the child objects to some other objects. These objects must be either parents or children of your parent object in order for this module to determine the distances. For instance, you might find "Nuclei" using **IdentifyPrimaryObjects**, find "Cells" using **IdentifySecondaryObjects** and find "Cytoplasm" using **IdentifyTertiaryObjects**. You can use **Relate** to relate speckles to cells and then measure distances to nuclei and cytoplasm. You could not use **RelateObjects** to relate speckles to cytoplasm and then measure distances to nuclei, because nuclei is neither a direct parent or child of cytoplasm.

Parent name

(Used only if calculating distances to another parent)
Choose the name of the other parent. The **RelateObjects** module will measure the distance from this parent to the child objects in the same manner as it does to the primary parents. You can only choose the parents or children of the parent object.

Module: StraightenWorms

StraightenWorms straightens untangled worms.

StraightenWorms uses the objects produced by **UntangleWorms** to create images and objects of straight worms from the angles and control points as computed by **UntangleWorms**. The resulting images can then be uniformly analyzed to find features that correlate with position in an ideal representation of the worm, such as the head or gut.

StraightenWorms works by calculating a transform on the image that translates points in the image to points on the ideal worm. **UntangleWorms** idealizes a worm as a series of control points that define the worm's shape and length. The training set contains measurements of the width of an ideal worm at each control point. Together, these can be used to reconstruct the worm's shape and correlate between the worm's location and points on the body of an ideal worm.

StraightenWorms produces objects representing the straight worms and images representing the intensity values of a source image mapped onto the straight worms. The objects and images can then be used to compute measurements using any of the object measurement modules, for instance, **MeasureTexture**.

The module can be configured to make intensity measurements on parts of the worm, dividing the worm up into pieces of equal width and/or height. Measurements are made longitudally in stripes from head to tail and transversely in segments across the width of the worm. Longitudinal stripes are numbered from left to right and transverse segments are numbered from top to bottom. The module will divide the worm into a checkerboard of sections if configured to measure more than one longitudinal stripe and transverse segment. These are numbered by longitudinal stripe number, then transverse segment number. For instance, "Worm_MeanIntensity_GFP_L2of3_T1of4", is a measurement of the mean GFP intensity of the center stripe (second of 3 stripes) of the topmost band (first of four bands). Measurements of longitudinal stripes are designated as "T1of1" indicating that the whole worm is one transverse segment. Likewise measurements of transverse segments are designated as "L1of1" indicating that there is only one longitudinal stripe. Both mean intensity and standard deviation of intensity are measured per worm sub-area.

While **StraightenWorms** can straighten a color image, the module needs a grayscale image to make its intensity measurements. For a color image, the red, green and blue channels are averaged to yield a grayscale image. The intensity measurements are then made on that grayscale image.

Available measurements

Object measurements:

- *Location_X, Location_Y:* The pixel (X,Y) coordinates of the primary object centroids. The centroid is calculated as the center of mass of the binary representation of the object.
- *Worm_MeanIntensity:* The average pixel intensity within a worm.
- *Worm_StdIntensity:* The standard deviation of the pixel intensities within a worm.

References

- Peng H, Long F, Liu X, Kim SK, Myers EW (2008) "Straightening *Caenorhabditis elegans* images." *Bioinformatics*, 24(2):234-42.(link)
- Wählby C, Kamentsky L, Liu ZH, Riklin-Raviv T, Conery AL, O'Rourke EJ, Sokolnicki KL, Visvikis O,

Ljosa V, Irazoqui JE, Golland P, Ruvkun G, Ausubel FM, Carpenter AE (2012). "An image analysis toolbox for high-throughput *C. elegans* assays." *Nature Methods* 9(7): 714-716. (link)

Settings:

Select the input untangled worm objects

This is the name of the objects produced by the **UntangleWorms** module. **StraightenWorms** can use either the overlapping or non-overlapping objects as input. It will use the control point measurements associated with the objects to reconstruct the straight worms. You can also use objects saved from a previous run and loaded via the **Input** modules, objects edited using **EditObjectsManually** or objects from one of the Identify modulues. **StraightenWorms** will recalculate the control points for these images.

Name the output straightened worm objects

This is the name that will be given to the straightened worm objects. These objects can then be used in a subsequent measurement module.

Worm width

This setting determines the width of the image of each worm. The width should be set to at least the maximum width of any untangled worm, but can be set to be larger to include the worm's background in the straightened image.

Training set file location

Select the folder containing the training set to be loaded. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

An additional option is the following:

- *URL*: Use the path part of a URL. For instance, your training set might be hosted at *http://my_institution.edu/server/my_username/TrainingSet.xml* To access this file, you would choose

URL and enter *http://my_institution.edu/server/my_username/* as the path location.

Training set file name

This is the name of the training set file.

Measure intensity distribution?

Select *Yes* to divide a worm into sections and measure the intensities of each section in each of the straightened images. These measurements can help classify phenotypes if the staining pattern across the segments differs between phenotypes.

Number of transverse segments

(*Only used if intensities are measured*)
This setting controls the number of segments measured, dividing the worm longitudally into transverse segments starting at the head and ending at the tail. These measurements might be used to identify a phenotype in which a stain is localized longitudally, for instance, in the head. Set the number of vertical segments to 1 to only measure intensity in the horizontal direction.

Number of longitudinal stripes

(*Only used if intensities are measured*)
This setting controls the number of stripes measured, dividing the worm transversely into areas that run longitudally. These measurements might be used to identify a phenotype in which a stain is localized transversely, for instance in the gut of the worm. Set the number of horizontal stripes to 1 to only measure intensity in the vertical direction.

Align worms?

(*Only used if intensities are measured*)
StraightenWorms can align worms so that the brightest half of the worm (the half with the highest mean intensity) is at the top of the image or at the bottom of the image. This can be used to align all worms similarly if some feature, such as the larynx, is stained and is always at the same end of the worm. Choose *Top brightest* if the brightest part of the worm should be at the top of the image, *Bottom brightest* if the brightest part of the worm should be at the bottom or *Do not align* if the worm should not be aligned. Choose *Flip manually* to bring up an editor for every cycle that allows you to choose the orientation of each worm.

Alignment image

(*Only used if aligning worms*)
This is the image whose intensity will be used to align the worms. You must use one of the straightened images below.

Select an input image to straighten

This is the name of an image that will be straightened similarly to the worm. The straightened image and objects can then be used in subsequent modules such as **MeasureObjectIntensity**.

Name the output straightened image

This is the name that will be given to the image of the straightened worms.

Module: TrackObjects

Track Objects allows tracking objects throughout sequential frames of a series of images, so that from frame to frame each object maintains a unique identity in the output measurements

---

This module must be placed downstream of a module that identifies objects (e.g., **IdentifyPrimaryObjects**). **TrackObjects** will associate each object with the same object in the frames before and after. This allows the study of objects' lineages and the timing and characteristics of dynamic events in movies.

Images in CellProfiler are processed sequentially by frame (whether loaded as a series of images or a movie file). To process a collection of images/movies, you will need to do the following:

- Define each individual movie using metadata either contained within the image file itself or as part of the images nomenclature or folder structure. Please see the **Metadata** module for more details on metadata collection and usage.
- Group the movies to make sure that each image sequence is handled individually. Please see the **Groups** module for more details on the proper use of metadata for grouping.

For complete details, see *Help > Creating a Project > Loading Image Stacks and Movies*.

For an example pipeline using TrackObjects, see the CellProfiler Examples webpage.

Available measurements

Object measurements

- *Label:* Each tracked object is assigned a unique identifier (label). Results of splits or merges are seen as new objects and assigned a new label.
- *ParentImageNumber, ParentObjectNumber:* The *ImageNumber* and *ObjectNumber* of the parent object in the prior frame. For a split, each child object will have the label of the object it split from. For a merge, the child will have the label of the closest parent.
- *TrajectoryX, TrajectoryY:* The direction of motion (in x and y coordinates) of the object from the previous frame to the current frame.
- *DistanceTraveled:* The distance traveled by the object from the previous frame to the current frame (calculated as the magnitude of the trajectory vectors).
- *Displacement:* The shortest distance traveled by the object from its initial starting position to the position in the current frame. That is, it is the straight-line path between the two points.
- *IntegratedDistance:* The total distance traveled by the object during the lifetime of the object.
- *Linearity:* A measure of how linear the object trajectity is during the object lifetime. Calculated as (displacement from initial to final location)/(integrated object distance). Value is in range of [0,1].
- *Lifetime:* The number of frames an objects has existed. The lifetime starts at 1 at the frame when an object appears, and is incremented with each frame that the object persists. At the final frame of the image set/movie, the lifetimes of all remaining objects are output.
- *FinalAge:* Similar to *LifeTime* but is only output at the final frame of the object's life (or the movie ends, whichever comes first). At this point, the final age of the object is output; no values are stored for earlier frames. This is useful if you want to plot a histogram of the object lifetimes; all but the final age can be ignored or filtered out.

Image measurements

- *LostObjectCount:* Number of objects that appear in the previous frame but have no identifiable child

in the current frame.
- *NewObjectCount:* Number of objects that appear in the current frame but have no identifiable parent in the previous frame.
- *SplitObjectCount:* Number of objects in the current frame that resulted from a split from a parent object in the previous frame.
- *MergedObjectCount:* Number of objects in the current frame that resulted from the merging of child objects in the previous frame.

See also: Any of the **Measure** modules, **IdentifyPrimaryObjects**, **Groups**.

Settings:

Choose a tracking method

When trying to track an object in an image, **TrackObjects** will search within a maximum specified distance (see the *distance within which to search* setting) of the object's location in the previous image, looking for a "match". Objects that match are assigned the same number, or label, throughout the entire movie. There are several options for the method used to find a match. Choose among these options based on which is most consistent from frame to frame of your movie.

- *Overlap:* Compares the amount of spatial overlap between identified objects in the previous frame with those in the current frame. The object with the greatest amount of spatial overlap will be assigned the same number (label). Recommended when there is a high degree of overlap of an object from one frame to the next, which is the case for movies with high frame rates relative to object motion.
- *Distance:* Compares the distance between each identified object in the previous frame with that of the current frame. The closest objects to each other will be assigned the same number (label). Distances are measured from the perimeter of each object. Recommended for cases where the objects are not very crowded but where *Overlap* does not work sufficiently well, which is the case for movies with low frame rates relative to object motion.
- *Measurements:* Compares each object in the current frame with objects in the previous frame based on a particular feature you have measured for the objects (for example, a particular intensity or shape measurement that can distinguish nearby objects). The object with the closest-matching measurement will be selected as a match and will be assigned the same number (label). This selection requires that you run the specified **Measure** module previous to this module in the pipeline so that the measurement values can be used to track the objects.
- *LAP:* Uses the linear assignment problem (LAP) framework. The linear assignment problem (LAP) algorithm (*Jaqaman et al., 2008*) addresses the challenges of high object density, motion heterogeneity, temporary disappearances, and object merging and splitting. The algorithm first links objects between consecutive frames and then links the resulting partial trajectories into complete trajectories. Both steps are formulated as global combinatorial optimization problems whose solution identifies the overall most likely set of object trajectories throughout a movie. Tracks are constructed from an image sequence by detecting objects in each frame and linking objects between consecutive frames as a first step. This step alone may result in incompletely tracked objects due to the appearance and disappearance of objects, either in reality or apparently because of noise and imaging limitations. To correct this, you may apply an optional second step which closes temporal gaps between tracked objects and captures merging and splitting events. This step takes place at the end of the analysis run. **References**
    - Jaqaman K, Loerke D, Mettlen M, Kuwata H, Grinstein S, Schmid SL, Danuser G. (2008) "Robust single-particle tracking in live-cell time-lapse sequences." *Nature Methods* 5(8),695-702. (link)
    - Jaqaman K, Danuser G. (2009) "Computational image analysis of cellular dynamics: a case

study based on particle tracking." Cold Spring Harb Protoc. 2009(12):pdb.top65. (link)

Select the objects to track

Select the objects to be tracked by this module.

Select object measurement to use for tracking

(Used only if Measurements is the tracking method)
Select which type of measurement (category) and which specific feature from the **Measure** module will be used for tracking. Select the feature name from the popup box or see each **Measure** module's help for the list of the features measured by that module. If necessary, you will also be asked to specify additional details such as the image from which the measurements originated or the measurement scale.

Maximum pixel distance to consider matches

Objects in the subsequent frame will be considered potential matches if they are within this distance. To determine a suitable pixel distance, you can look at the axis increments on each image (shown in pixel units) or use the distance measurement tool. To measure distances in an open image, use the "Measure length" tool under *Tools* in the display window menu bar. If you click on an image and drag, a line will appear between the two endpoints, and the distance between them shown at the right-most portion of the bottom panel.

Select display option

The output image can be saved as:

- *Color Only:* A color-labeled image, with each tracked object assigned a unique color
- *Color and Number:* Same as above but with the tracked object number superimposed.

Save color-coded image?

Select *Yes* to retain the image showing the tracked objects for later use in the pipeline. For example, a common use is for quality control purposes saving the image with the **SaveImages** module.

Please note that if you are using the second phase of the LAP method, the final labels are not assigned until *after* the pipeline has completed the analysis run. That means that saving the color-coded image will only show the penultimate result and not the final product.

.

Name the output image

(Used only if saving the color-coded image)
Enter a name to give the color-coded image of tracked labels.

Select the motion model

(Used only if the LAP tracking method is applied)
This setting controls how to predict an object's position in the next frame, assuming that each object moves randomly with a frame-to-frame variance in position that follows a Gaussian distribution.

- *Random:* A model in which objects move due to Brownian Motion or a similar process where the variance in position differs between objects. Use this model if the objects move with some random jitter around a stationary location.
- *Velocity:* A model in which the object moves with a velocity. Both velocity and position (after correcting for velocity) vary following a Gaussian distribution. Use this model if the objects move along a spatial trajectory in some direction over time.
- *Both:* **TrackObjects** will predict each object's position using both models and use the model with the lowest penalty to join an object in one frame with one in another. Use this option if both models above are applicable over time.

Number of standard deviations for search radius

(Used only if the LAP tracking method is applied)
TrackObjects will estimate the variance of the error between the observed and predicted positions of an object for each movement model. It will constrain the search for matching objects from one frame to the next to the standard deviation of the error times the number of standard deviations that you enter here.

Search radius limit, in pixel units (Min,Max)

(Used only if the LAP tracking method is applied)
Care must be taken to adjust the upper limit appropriate to the data.
TrackObjects derives a search radius based on the error estimation. Potentially, the module can make an erroneous assignment with a large error, leading to a large estimated error for the object in the next frame. Conversely, the module can arrive at a small estimated error by chance, leading to a maximum radius that does not track the object in a subsequent frame. The radius limit constrains the maximum radius to reasonable values.

The lower limit should be set to a radius (in pixels) that is a reasonable displacement for any object from one frame to the next. The upper limit should be set to the maximum reasonable displacement under any circumstances.

Run the second phase of the LAP algorithm?

(Used only if the LAP tracking method is applied)
Select *Yes* to run the second phase of the LAP algorithm after processing all images. Select *No* to omit the second phase or to perform the second phase when running the module as a data tool.

Since object tracks may start and end not only because of the true appearance and disappearance of objects, but also because of apparent disappearances due to noise and limitations in imaging, you may want to run the second phase which attempts to close temporal gaps between tracked objects and tries to capture merging and splitting events.

For additional details on optimizing the LAP settings, refer to Jaqaman K, Danuser G. "Computational image analysis of cellular dynamics: a case study based on particle tracking." *Cold Spring Harb Protocols* 2009(12) (link), in particular the section "Adjustment of control parameters and diagnostics for track evaluation."

Gap cost

(Used only if the LAP tracking method is applied and the second phase is run)
This setting assigns a cost to keeping a gap caused when an object is missing from one of the frames of a track (the alternative to keeping the gap is to bridge it by connecting the tracks on either side of the

missing frames). The cost of bridging a gap is the distance, in pixels, of the displacement of the object between frames.

Recommendations:

- Set the gap cost higher if tracks from objects in previous frames are being erroneously joined, across a gap, to tracks from objects in subsequent frames.
- Set the cost lower if tracks are not properly joined due to gaps caused by mis-segmentation.

Split alternative cost

(Used only if the LAP tracking method is applied and the second phase is run)
This setting is the cost of keeping two tracks distinct when the alternative is to make them into one track that splits. A split occurs when an object in one frame is assigned to the same track as two objects in a subsequent frame. The split cost takes two components into account:

- The area of the split object relative to the area of the resulting objects.
- The displacement of the resulting objects relative to the position of the original object.

The split cost is roughly measured in pixels. The split alternative cost is (conceptually) subtracted from the cost of making the split.

Recommendations:

- The split cost should be set lower if objects are being split that should not be split.
- The split cost should be set higher if objects that should be split are not.

Merge alternative cost

(Used only if the LAP tracking method is applied and the second phase is run)
This setting is the cost of keeping two tracks distinct when the alternative is to merge them into one. A merge occurs when two objects in one frame are assigned to the same track as a single object in a subsequent frame. The merge score takes two components into account:

- The area of the two objects to be merged relative to the area of the resulting objects.
- The displacement of the original objects relative to the final object.

The merge cost is measured in pixels. The merge alternative cost is (conceptually) subtracted from the cost of making the merge.

Recommendations:

- Set the merge alternative cost lower if objects are being merged when they should otherwise be kept separate.
- Set the merge alternative cost higher if objects that are not merged should be merged.

Maximum gap displacement, in frames

(Used only if the LAP tracking method is applied and the second phase is run)
This setting acts as a filter for unreasonably large displacements during the second phase.

Recommendations:

- The maximum gap displacement should be set to roughly the maximum displacement of an object's

center from frame to frame. An object that makes large frame-to-frame jumps should have a higher value for this setting than one that only moves slightly.
- Be aware that the LAP algorithm will run more slowly with a higher maximum gap displacement value, since the higher this value, the more objects that must be compared at each step.
- Objects that would have been tracked between successive frames for a lower maximum displacement may not be tracked if the value is set higher.

Maximum split score

(Used only if the LAP tracking method is applied and the second phase is run)
This setting acts as a filter for unreasonably large split scores. The split score has two components:

- The area of the initial object relative to the area of the two objects resulting from the split.
- The distances between the original and resulting objects.

Recommendations:

- The LAP algorithm will run more slowly with a maximum split score value.
- Objects that would have been split at a lower maximum split score will not be considered for splitting.

Maximum merge score

(Used only if the LAP tracking method is applied and the second phase is run)
This setting acts as a filter for unreasonably large merge scores. The merge score has two components:

- The area of the resulting merged object relative to the area of the two objects to be merged.
- The distances between the objects to be merged and the resulting object.

Recommendations:

- The LAP algorithm will run more slowly with a higher maximum merge score value.
- Objects that would have been merged at a lower maximum merge score will not be considered for merging.

Maximum gap

(Used only if the LAP tracking method is applied and the second phase is run)
Care must be taken to adjust this setting appropriate to the data.
This setting controls the maximum number of frames that can be skipped when merging a gap caused by an unsegmented object. These gaps occur when an image is mis-segmented and identification fails to find an object in one or more frames.

Recommendations:

- Set the maximum gap higher in order to have more chance of correctly recapturing an object after erroneously losing the original for a few frames.
- Set the maximum gap lower to reduce the chance of erroneously connecting to the wrong object after correctly losing the original object (e.g., if the cell dies or moves off-screen).

Filter objects by lifetime?

Select *Yes* if you want objects to be filtered by their lifetime, i.e., total duration in frames. This is useful for

marking objects which transiently appear and disappear, such as the results of a mis-segmentation.

Recommendations:

- This operation does not actually delete the filtered object, but merely removes its label from the tracked object list; the filtered object's per-object measurements are retained.
- An object can be filtered only if it is tracked as an unique object. Splits continue the lifetime count from their parents, so the minimum lifetime value does not apply to them.

Filter using a minimum lifetime?

(Used only if objects are filtered by lifetime)
Select *Yes* to filter the object on the basis of a minimum number of frames.

Minimum lifetime

Enter the minimum number of frames an object is permitted to persist. Objects which last this number of frames or lower are filtered out.

Filter using a maximum lifetime?

(Used only if objects are filtered by lifetime)
Select *Yes* to filter the object on the basis of a maximum number of frames.

Maximum lifetime

Enter the maximum number of frames an object is permitted to persist. Objects which last this number of frames or more are filtered out.

Module: UntangleWorms

UntangleWorms untangles overlapping worms.

This module either assembles a training set of sample worms in order to create a worm model, or takes a binary image and the results of worm training and labels the worms in the image, untangling them and associating all of a worm's pieces together.

The results of untangling the input image will be an object set that can be used with downstream measurment modules. If using the *overlapping* style of objects, these can be saved as images using **SaveImages** to create a multi-page TIF file by specifying "Objects" as the type of image to save.

Available measurements

Object measurements (for "Untangle" mode only):

- *Length:* The length of the worm skeleton.
- *Angle:* The angle at each of the control points
- *ControlPointX_N, ControlPointY_N:* The X,Y coordinate of a control point *N*. A control point is a sampled location along the worm shape used to construct the model.

Technical notes

Training involves extracting morphological information from the sample objects provided from the previous steps. Using the default training set weights is recommended. Proper creation of the model is dependent on providing a binary image as input consisting of single, separated objects considered to be worms. You can the **Identify** modules to find the tentative objects and then filter these objects to get individual worms, whether by using **FilterObjects**, **EditObjectsManually** or the size criteria in **IdentifyPrimaryObjects**. A binary image can be obtained from an object set by using **ConvertObjectsToImage**.

At the end of the training run, a final display window is shown displaying the following statistical data:

- A boxplot of the direction angle shape costs. The direction angles (which are between -π and π) are the angles between lines joining consective control points. The angle 0 corresponds to the case when two adjacent line segments are parallel (and thus belong to the same line).
- A cumulative boxplot of the worm lengths as determined by the model.
- A cumulative boxplot of the worm angles as determined by the model.
- A heatmap of the covariance matrix of the feature vectors. For *N* control points, the feature vector is of length *N*-1 and contains *N*-2 elements for each of the angles between them, plus an element representing the worm length.

Untangling involves untangles the worms using a provided worm model, built from a large number of samples of single worms. If the result of the untangling is not satisfactory (e.g., it is unable to detect long worms or is too stringent about shape variation) and you do not wish to re-train, you can adjust the provided worm model manually by opening the .xml file in a text editor and changing the values for the fields defining worm length, area etc. You may also want to adjust the "Maximum Complexity" module setting which controls how complex clusters the untangling will handle. Large clusters (> 6 worms) may be slow to process.

References

- Wählby C, Kamentsky L, Liu ZH, Riklin-Raviv T, Conery AL, O'Rourke EJ, Sokolnicki KL, Visvikis O, Ljosa V, Irazoqui JE, Golland P, Ruvkun G, Ausubel FM, Carpenter AE (2012). "An image analysis toolbox for high-throughput *C. elegans* assays." *Nature Methods* 9(7): 714-716. (link)

Settings:

Train or untangle worms?

UntangleWorms has two modes:

- *Train* creates one training set per image group, using all of the worms in the training set as examples. It then writes the training file at the end of each image group.
- *Untangle* uses the training file to untangle images of worms.

Please see the **Groups** module for more details on the proper use of metadata for grouping

Select the input binary image

A binary image where the foreground indicates the worm shapes. The binary image can be produced by the **ApplyThreshold** module.

Overlap style

This setting determines which style objects are output. If two worms overlap, you have a choice of including the overlapping regions in both worms or excluding the overlapping regions from both worms.

- Choose *With overlap* to save objects including overlapping regions.
- Choose *Without overlap* to save only the portions of objects that do not overlap.
- Choose *Both* to save two versions: with and without overlap.

Name the output overlapping worm objects

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
This setting names the objects representing the overlapping worms. When worms cross, they overlap and pixels are shared by both of the overlapping worms. The overlapping worm objects share these pixels and measurements of both overlapping worms will include these pixels in the measurements of both worms.

Name the output non-overlapping worm objects

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
This setting names the objects representing the worms, excluding those regions where the worms overlap. When worms cross, there are pixels that cannot be unambiguously assigned to one worm or the other. These pixels are excluded from both worms in the non-overlapping objects and will not be a part of the measurements of either worm.

Maximum complexity

(Used only if "Untangle" mode is selected)
This setting controls which clusters of worms are rejected as being too time-consuming to process. **UntangleWorms** judges complexity based on the number of segments in a cluster where a segment is the piece of a worm between crossing points or from the head or tail to the first or last crossing point. The

choices are:

- *Medium*: 200 segments (takes up to several minutes to process)
- *High*: 600 segments (takes up to a quarter-hour to process)
- *Very high*: 1000 segments (can take hours to process)
- *Custom*: allows you to enter a custom number of segments.
- *Process all clusters*: Process all worms, regardless of complexity

Custom complexity

(Used only if "Untangle" mode and "Custom" complexity are selected) Enter the maximum number of segments of any cluster that should be processed.

Training set file location

Select the folder containing the training set to be loaded. You can choose among the following options which are common to all file input/output modules:

- *Default Input Folder*: Use the default input folder.
- *Default Output Folder:* Use from the default output folder.
- *Elsewhere...*: Use a particular folder you specify.
- *Default input directory sub-folder*: Enter the name of a subfolder of the default input folder or a path that starts from the default input folder.
- *Default output directory sub-folder*: Enter the name of a subfolder of the default output folder or a path that starts from the default output folder.

Elsewhere and the two sub-folder options all require you to enter an additional path name. You can use an *absolute path* (such as "C:\imagedir\image.tif" on a PC) or a *relative path* to specify the file location relative to a directory):

- Use one period to represent the current directory. For example, if you choose *Default Input Folder sub-folder*, you can enter "./MyFiles" to look in a folder called "MyFiles" that is contained within the Default Input Folder.
- Use two periods ".." to move up one folder level. For example, if you choose *Default Input Folder sub-folder*, you can enter "../MyFolder" to look in a folder called "MyFolder" at the same level as the Default Input Folder.

An additional option is the following:

- *URL*: Use the path part of a URL. For instance, your training set might be hosted at `http://my_institution.edu/server/my_username/TrainingSet.xml` To access this file, you would choose *URL* and enter `http://my_institution.edu/server/my_username/` as the path location.

Training set file name

This is the name of the training set file.

Use training set weights?

Select *Yes* to use the overlap and leftover weights from the training set.

Select *No* to override these weights with user-specified values.

Overlap weight

(Used only if not using training set weights)
This setting controls how much weight is given to overlaps between worms. **UntangleWorms** charges a penalty to a particular putative grouping of worms that overlap equal to the length of the overlapping region times the overlap weight.

- Increase the overlap weight to make **UntangleWorms** avoid overlapping portions of worms.
- Decrease the overlap weight to make **UntangleWorms** ignore overlapping portions of worms.

Leftover weight

(Used only if not using training set weights)
This setting controls how much weight is given to areas not covered by worms. **UntangleWorms** charges a penalty to a particular putative grouping of worms that fail to cover all of the foreground of a binary image. The penalty is equal to the length of the uncovered region times the leftover weight.

- Increase the leftover weight to make **UntangleWorms** cover more foreground with worms.
- Decrease the overlap weight to make **UntangleWorms** ignore uncovered foreground.

Retain outlines of the overlapping objects?

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Outline colormap?

(Used only if "Untangle" mode, "Both" or "With overlap" overlap style and retaining outlines are selected)
This setting controls the colormap used when drawing outlines. The outlines are drawn in color to highlight the shapes of each worm in a group of overlapping worms

Name the overlapped outline image

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
This is the name of the outlines of the overlapped worms.

Retain outlines of the non-overlapping worms?

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
Select *Yes* to retain the outlines of the new objects for later use in the pipeline. For example, a common use is for quality control purposes by overlaying them on your image of choice using the **OverlayOutlines** module and then saving the overlay image with the **SaveImages** module.

Name the non-overlapped outlines image

(Used only if "Untangle" mode and "Both" or "With overlap" overlap style are selected)
This is the name of the of the outlines of the worms with the overlapping sections removed.

Minimum area percentile

(Used only if "Train" mode is selected)
UntangleWorms will discard single worms whose area is less than a certain minimum. It ranks all worms in the training set according to area and then picks the worm at this percentile. It then computes the minimum area allowed as this worm's area times the minimum area factor.

Minimum area factor

(Used only if "Train" mode is selected)
This setting is a multiplier that is applied to the area of the worm, selected as described in the documentation for *Minimum area percentile*.

Maximum area percentile

(Used only if "Train" mode is selected)
UntangleWorms uses a maximum area to distinguish between single worms and clumps of worms. Any blob whose area is less than the maximum area is considered to be a single worm whereas any blob whose area is greater is considered to be two or more worms. **UntangleWorms** orders all worms in the training set by area and picks the worm at the percentile given by this setting. It then multiplies this worm's area by the *Maximum area factor* (see below) to get the maximum area

Maximum area factor

(Used only if "Train" mode is selected)
The *Maximum area factor* setting is used to compute the maximum area as decribed above in *Maximum area percentile*.

Minimum length percentile

(Used only if "Train" mode is selected)
UntangleWorms uses the minimum length to restrict its search for worms in a clump to worms of at least the minimum length. **UntangleWorms** sorts all worms by length and picks the worm at the percentile indicated by this setting. It then multiplies the length of this worm by the *Mininmum length factor* (see below) to get the minimum length.

Minimum length factor

(Used only if "Train" mode is selected)
UntangleWorms uses the *Minimum length factor* to compute the minimum length from the training set as described in the documentation above for *Minimum length percentile*

Maximum length percentile

(Used only if "Train" mode is selected)
UntangleWorms uses the maximum length to restrict its search for worms in a clump to worms of at least the maximum length. It computes this length by sorting all of the training worms by length. It then selects the worm at the *Maximum length percentile* and multiplies that worm's length by the *Maximum length factor* to get the maximum length

Maximum length factor

(Used only if "Train" mode is selected)

UntangleWorms uses this setting to compute the maximum length as described in *Maximum length percentile* above

Maximum cost percentile

(Used only if "Train" mode is selected)
UntangleWorms computes a shape-based cost for each worm it considers. It will restrict the allowed cost to less than the cost threshold. During training, **UntangleWorms** computes the shape cost of every worm in the training set. It then orders them by cost and uses *Maximum cost percentile* to pick the worm at the given percentile. It them multiplies this worm's cost by the *Maximum cost factor* to compute the cost threshold.

Maximum cost factor

(Used only "Train" mode is selected)
UntangleWorms uses this setting to compute the cost threshold as described in *Maximum cost percentile* above.

Number of control points

(Used only if "Train" mode is selected)
This setting controls the number of control points that will be sampled when constructing a worm shape from its skeleton.

Maximum radius percentile

(Used only if "Train" mode is selected)
UntangleWorms uses the maximum worm radius during worm skeletonization. **UntangleWorms** sorts the radii of worms in increasing size and selects the worm at this percentile. It then multiplies this worm's radius by the *Maximum radius factor* (see below) to compute the maximum radius.

Maximum radius factor

(Used only if "Train" mode is selected)
UntangleWorms uses this setting to compute the maximum radius as described in *Maximum radius percentile* above.

Module: ConserveMemory

Conserve Memory speeds up CellProfiler by removing images from memory.

This module removes images from memory, which can speed up processing and prevent out-of-memory errors.

Note: CellProfiler 1.0's **SpeedUpCellProfiler** had an option that let you choose how often the output file of measurements was saved. This option is no longer neccessary since the output file is automatically updated with each image set.

Settings:

Specify which images?

You can select from the following options:

- *Images to remove:* Remove some images from memory and keep the rest.
- *Images to keep:* Keep some images and remove the rest.

Module: CreateWebPage

Create Web Page creates the html file for a webpage to display images (or their thumbnails, if desired).

This module creates an html file that displays the specified images, and optionally a link to a compressed ZIP file of all of the images shown.

Settings:

Select the input images

Select the images to display on the web page.

Use thumbnail images?

Select *Yes* to display thumbnail images (small versions of the images) on the web page that link to the full images.
Select *No* to display the full image directly on the web page.

If you are going to use thumbnails, you will need to load them using the **Input** modules; you can also run a separate pipeline prior to this one to create thumbnails from your originals using the **Resize** and **SaveImages** modules. For some high-content screening systems, thumbnail files are automatically created and have the text "thumb" in the name.

Select the thumbnail images

(Used only if using thumbnails)
Select the name of the images to use for thumbnails.

Webpage file name

Enter the desired file name for the web page. **CreateWebPage** will add the .html extension if no extension is specified. If you have metadata associated with your images, you can name the file using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have metadata tags named "Plate" and "Well", you can create separate per-plate, per-well web pages based on your metadata by inserting the tags "Plate_Well" to specify the name. Please see the **Metadata** module for more details on metadata collection and usage.

Select the folder for the .html file

This setting determines how **CreateWebPage** selects the folder for the .html file(s) it creates.

- *Same as the images*: Place the .html file(s) in the same folder as the files.
- *One level over the images*: Place the .html file(s) in the image files' parent folder.
- *Elsewhere...*: Places the .html file(s) in a folder of your choosing. **CreateWebPage** will use absolute references for your image URLs if you choose this option.
- *Default Input Folder*: Places the .html file(s) in the default input folder.
- *Default Output Folder*: Places the .html file(s) in the default output folder.
- *Default Input Folder sub-folder*: Places the .html file(s) in a subfolder of the default input folder. You will be prompted for the subfolder name after making this choice
- *Default Output Folder sub-folder*: Places the .html file(s) in a subfolder of the default input folder. You will be prompted for the subfolder name after making this choice

Webpage title

This is the title that appears at the top of the browser window. If you have metadata associated with your images, you can name the file using metadata tags. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have a metadata tag named "Plate", you can type "Plate: " and then insert the metadata tag "Plate" to display the plate metadata item. Please see the **Metadata** module for more details on metadata collection and usage.

Webpage background color

This setting controls the background color for the web page.

Number of columns

This setting determines how many images are displayed in each row.

Table border width

The table border width determines the width of the border around the entire grid of displayed images (i.e., the "table" of images) and is measured in pixels. This value can be set to zero, in which case you will not see the table border.

Image spacing

The spacing between images ("table cells"), in pixels.

Image border width

The image border width determines the width of the border around each image and is measured in pixels. This value can be set to zero, in which case you will not see the image border.

Open new window when viewing full image?

This controls the behavior of the thumbnail links.

- *Once only:* Your browser will open a new window when you click on the first thumbnail and will display subsequent images in the newly opened window.
- *For each image:* The browser will open a new window each time you click on a link.
- *No:* The browser will reuse the current window to display the image

Make a ZIP file containing the full-size images?

ZIP files are a common archive and data compression file format, making it convenient to download all of the images represented on the web page with a single click. Select *Yes* to create a ZIP file that contains all your images, compressed to reduce file size.

Enter the ZIP file name

(Used only if creating a ZIP file)
Specify the name for the ZIP file.

Module: DefineGrid

Define Grid produces a grid of desired specifications either manually, or automatically based on previously identified objects.

This module defines the location of a grid that can be used by modules downstream. You can use it in combination with **IdentifyObjectsInGrid** to measure the size, shape, intensity and texture of each object or location in a grid. The grid is defined by the location of marker spots (control spots), which are either indicated manually or found automatically using previous modules in the pipeline. You can then use the grid to make measurements (using **IdentifyObjectsInGrid**). Text annotation of a grid can be shown on top of an image using the **DisplayGridInfo** module (coming soon).

If you are using images of plastic plates, it may be useful to precede this module with an **IdentifyPrimaryObjects** module to find the plastic plate, followed by a **Crop** module to remove the plastic edges of the plate, so that the grid can be defined within the smooth portion of the plate only. If the plates are not centered in exactly the same position from one image to the next, this allows the plates to be identified automatically and then cropped so that the interior of the plates, upon which the grids will be defined, are always in precise alignment with each other.

Available measurements

- *Rows, Columns*: The number of rows and columns in the grid
- *XSpacing, YSpacing:* The spacing in X and Y of the grid elements.
- *XLocationOfLowestXSpot:* The X coordinate location of the lowest spot on the X-axis.
- *YLocationOfLowestYSpot:* The Y coordinate location of the lowest spot on the Y-axis.

See also **IdentifyObjectsInGrid**.

Settings:

Name the grid

This is the name of the grid. You can use this name to retrieve the grid in subsequent modules.

Location of the first spot

Grid cells are numbered consecutively; this option identifies the origin for the numbering system and the direction for numbering. For instance, if you choose *Top left*, the top left cell is cell #1 and cells to the right and bottom are indexed with larger numbers.

Order of the spots

Grid cells can either be numbered by rows, then columns or by columns, then rows. For instance, you might ask to start numbering a 96-well plate at the top left (by specifying the location of the first spot).

- *Rows:* This option will give well A01 the index 1, B01 the index 2, and so on up to H01 which receives the index 8. Well A02 will be assigned the index 9.
- *Columns:* With this option, the well A02 will be assigned 2, well A12 will be assigned 12 and well B01 will be assigned 13.

Define a grid for which cycle?

The setting allows you choose when you want to define a new grid:

- *Once:* If all of your images are perfectly aligned with each other (due to very consistent image acquisition, consistent grid location within the plate, and/or automatic cropping precisely within each plate), you can define the location of the marker spots once for all of the image cycles.
- *Each cycle:* If the location of the grid will vary from one image cycle to the next then you should define the location of the marker spots for each cycle independently.

Select the method to define the grid

Select whether you would like to define the grid automatically (based on objects you have identified in a previous module) or manually. This setting controls how the grid is defined:

- *Manual:* In manual mode, you manually indicate known locations of marker spots in the grid and have the rest of the positions calculated from those marks, no matter what the image itself looks like. You can define the grid either by clicking on the image with a mouse or by entering coordinates.
- *Automatic:* If you would like the grid to be defined automatically, an **IdentifyPrimaryObjects** module must be run prior to this module to identify the objects which will be used to define the grid. The left-most, right-most, top-most, and bottom-most object will be used to define the edges of the grid, and the rows and columns will be evenly spaced between these edges. Note that Automatic mode requires that the incoming objects are nicely defined: for example, if there is an object at the edge of the images that is not really an object that ought to be in the grid, a skewed grid will result. You might wish to use a **FilterObjects** module to clean up badly identified objects prior to defining the grid. If the spots are slightly out of alignment with each other from one image cycle to the next, this allows the identification to be a bit flexible and adapt to the real location of the spots.

Select the previously identified objects

(Used only if you selected Automatic to define the grid)
Select the previously identified objects you want to use to define the grid. Use this setting to specify the name of the objects that will be used to define the grid.

Select the method to define the grid manually

(Used only if you selected Manual to define the grid)
Specify whether you want to define the grid using the mouse or by entering the coordinates of the cells.

- *Mouse:* The user interface displays the image you specify. You will be asked to click in the center of two of the grid cells and specify the row and column for each. The grid coordinates will be computed from this information.
- *Coordinates:* Enter the X and Y coordinates of the grid cells directly. You can display an image of your grid to find the locations of the centers of the cells, then enter the X and Y position and cell coordinates for each of two cells.

Select the image to display

(Used only if you selected Manual + Mouse to define the grid)
Specify the image you want to display when defining the grid. This setting lets you choose the image to display in the grid definition user interface.

Coordinates of the first cell

(Used only if you selected Manual + Coordinates to define the grid)
Enter the coordinates of the first cell on your grid. This setting defines the location of the first of two cells in your grid. You should enter the coordinates of the center of the cell. You can display an image of your grid and use the pixel coordinate display to determine the coordinates of the center of your cell.

Row number of the first cell

(Used only if you selected Manual + Coordinates to define the grid)
Enter the row index for the first cell here. Rows are numbered starting at the origin. For instance, if you chose *Top left* as your origin, well A01 will be row number 1 and H01 will be row number 8. If you chose *Bottom left*, A01 will be row number 8 and H01 will be row number 12.

Column number of the first cell

(Used only if you selected Manual + Coordinates to define the grid)
Enter the column index for the first cell here. Columns are numbered starting at the origin. For instance, if you chose *Top left* as your origin, well A01 will be column number *1* and A12 will be column number *12*. If you chose *Top right*, A01 and A12 will be *12* and *1*, respectively.

Coordinates of the second cell

(Used only if you selected Manual + Coordinates to define the grid)
This setting defines the location of the second of two cells in your grid. You should enter the coordinates of the center of the cell. You can display an image of your grid and use use the pixel coordinate display to determine the coordinates of the center of your cell.

Row number of the second cell

(Used only if you selected Manual + Coordinates to define the grid)
Enter the row index for the second cell here. Rows are numbered starting at the origin. For instance, if you chose *Top left* as your origin, well A01 will be row number 1 and H01 will be row number 8. If you chose *Bottom left*, A01 will be row number 8 and H01 will be row number 12.

Column number of the second cell

(Used only if you selected Manual + Coordinates to define the grid)
Enter the column index for the second cell here. Columns are numbered starting at the origin. For instance, if you chose *Top left* as your origin, well A01 will be column number 1 and A12 will be column number 12. If you chose *Top right*, A01 and A12 will be 12 and 1, respectively.

Retain an image of the grid?

Select *Yes* to retain an image of the grid for use later in the pipeline. This module can create an annotated image of the grid that can be saved using the **SaveImages** module.

Name the output image

(Used only if retaining an image of the grid for use later in the pipeline)
Enter the name you want to use for the output image. You can save this image using the **SaveImages**

module.

Select the image on which to display the grid

(Used only if saving an image of the grid)
Enter the name of the image that should be used as the background for annotations (grid lines and grid indexes). This image will be used for the figure and for the saved image.

Use a previous grid if gridding fails?

If the gridding fails, this setting allows you to control how the module responds to the error:

- *No:* The module will stop the pipeline if gridding fails.
- *Use any previous grid:*: The module will use the the most recent successful gridding.
- *Use the first cycle's grid:* The module will use the first gridding.

Note that the pipeline will stop in all cases if gridding fails on the first image.

Module: IdentifyDeadWorms

IdentifyDeadWorms identifies dead worms by their shape.

Dead *C. elegans* worms most often have a straight shape in an image whereas live worms assume a sinusoidal shape. This module identifies dead worms by fitting a straight shape to a binary image at many different angles to identify the regions where the shape could fit. Each placement point has a x and y location and an angle associated with the fitted shape's placement. Conceptually, these can be visualized in three dimensions with the z direction being the angle (and with the angle, 0, being adjacent to the largest angle as well as the smallest angle greater than zero). The module labels the resulting 3-D volume. It records the X, Y and angle of the centers of each of the found objects and creates objects by collapsing the 3-D volume to 2-D. These objects can then be used as seeds for **IdentifySecondaryObjects**.

IdentifyDeadWorms fits a diamond shape to the image. The shape is defined by its width and length. The length is the distance in pixels along the long axis of the diamond and should be less than the length of the shortest dead worm to be detected. The width is the distance in pixels along the short axis of the diamond and should be less than the width of the worm.

References

- Peng H, Long F, Liu X, Kim SK, Myers EW (2008) "Straightening *Caenorhabditis elegans* images." *Bioinformatics*, 24(2):234-42.(link)
- Wählby C, Kamentsky L, Liu ZH, Riklin-Raviv T, Conery AL, O'Rourke EJ, Sokolnicki KL, Visvikis O, Ljosa V, Irazoqui JE, Golland P, Ruvkun G, Ausubel FM, Carpenter AE (2012). "An image analysis toolbox for high-throughput *C. elegans* assays." *Nature Methods* 9(7): 714-716. (link)

Settings:

Select the input image

The name of a binary image from a previous module. **IdentifyDeadWorms** will use this image to establish the foreground and background for the fitting operation. You can use **ApplyThreshold** to threshold a grayscale image and create the binary mask. You can also use a module such as **IdentifyPrimaryObjects** to label each worm and then use **ConvertObjectsToImage** to make the result a mask.

Name the dead worm objects to be identified

This is the name for the dead worm objects. You can refer to this name in subsequent modules such as **IdentifySecondaryObjects**

Worm width

This is the width (the short axis), measured in pixels, of the diamond used as a template when matching against the worm. It should be less than the width of a worm.

Worm length

This is the length (the long axis), measured in pixels, of the diamond used as a template when matching

against the worm. It should be less than the length of a worm

Number of angles

This is the number of different angles at which the template will be tried. For instance, if there are 12 angles, the template will be rotated by 0°, 15°, 30°, 45° ... 165°. The shape is bilaterally symmetric; that is, you will get the same shape after rotating it by 180°.

Automatically calculate distance parameters?

This setting determines whether or not **IdentifyDeadWorms** automatically calculates the parameters used to determine whether two found-worm centers belong to the same worm.

Select *Yes* to have **IdentifyDeadWorms** automatically calculate the distance from the worm length and width. Select *No* to set the distances manually.

Spatial distance

(Used only if not automatically calculating distance parameters)
Enter the distance for calculating the worm centers, in units of pixels. The worm centers must be at least many pixels apart for the centers to be considered two separate worms.

Angular distance

(Used only if automatically calculating distance parameters)
IdentifyDeadWorms calculates the worm centers at different angles. Two worm centers are considered to represent different worms if their angular distance is larger than this number. The number is measured in degrees.

Module: InputExternal

Input External specifies the image names that will be pulled from external sources (e.g., Java)

InputExternal is a helper module for ImageJ. **Do not add it to a pipeline.**

The **InputExternal** and **OutputExternal** modules are placeholders if CellProfiler is run programatically. For example, another program, e.g., a plugin to ImageJ, is provided with a CellProfiler pipeline. This program should then replace the input modules with **InputExternal** modules and prompt the user what inputs should be supplied to the pipeline through **InputExternal**. The program should also specify which inputs are to be sent back to the source program via **OutputExternal**. The calling program would insert the images into the image set before running the pipeline and remove the images from the image set at the end.

See also **RunImageJ**

Module: LabelImages

LabelImages assigns plate metadata to image sets.

LabelImages assigns a plate number, well and site number to each image set based on the order in which they are processed. You can use **Label Images** to add plate and well metadata for images loaded using *Order* for "Image set matching order" in **NamesAndTypes**. < b>LabelImages assumes the following are true of the image order:

- Each well has the same number of images (i.e., sites) per channel.
- Each plate has the same number of rows and columns, so that the total number of images per plate is the same.

Available measurements

- *Metadata_Plate:* The plate number, starting at 1 for the first plate.
- *Metadata_Well:* The well name, e.g., *A01*.
- *Metadata_Row:* The row name, starting with *A* for the first row.
- *Metadata_Column:* The column number, starting with 1 for the first column.
- *Metadata_Site:* The site number within the well, starting at 1 for the first site

See also the **Metadata** module.

Settings:

Number of image sites per well

This setting controls the number of image sets for each well

Number of columns per plate

Enter the number of columns per plate

Number of rows per plate

The number of rows per plate

Order of image data

This setting specifies how the input data is ordered (assuming that sites within a well are ordered consecutively):

- *Row:*: The data appears by row and then by column. That is, all columns for a given row (e.g. A01, A02, A03...) appear consecutively, for each row in consecutive order.
- *Column:* The data appears by column and then by row. That is, all rows for a given column (e.g. A01, B01, C01...) appear consecutively, for each column in consecutive order.

For instance, the SBS Bioimage example (available here) has files that are named:
Channel1-01-A01.tif
Channel1-02-A02.tif

...
Channel1-12-A12.tif
Channel1-13-B01.tif
...
You would use "Row" to label these because the ordering is by row and then by column.

Module: OutputExternal

Output External specfies which images can be made available to external sources (e.g., Java)

OutputExternal is a helper module for ImageJ. **Do not add it to a pipeline.**

The **InputExternal** and **OutputExternal** modules are placeholders if CellProfiler is run programatically. For example, another program, e.g., a plugin to ImageJ, is provided with a CellProfiler pipeline. This program should then replace the input modules with **InputExternal** modules and prompt the user what inputs should be supplied to the pipeline through **InputExternal**. The program should also specify which inputs are to be sent back to the source program via **OutputExternal**. The calling program would insert the images into the image set before running the pipeline and remove the images from the image set at the end.

See also **RunImageJ**

Module: SendEmail

SendEmail send emails to a specified address at desired stages of the analysis run.

This module sends email about the current progress of the image processing. You can specify how often emails are sent out (for example, after the first cycle, after the last cycle, after every *N* cycles, after *N* cycles). This module should be placed at the point in the pipeline when you want the emails to be sent. If email sending fails for any reason, a warning message will appear but processing will continue regardless.

Settings:

Sender address

Enter the address for the email's "From" field.

Subject line

Enter the text for the email's subject line. If you have metadata associated with your images, you can use metadata tags here. You can insert a previously defined metadata tag by either using:

- The insert key
- A right mouse button click inside the control
- In Windows, the Context menu key, which is between the Windows key and Ctrl key

The inserted metadata tag will appear in green. To change a previously inserted metadata tag, navigate the cursor to just before the tag and either:

- Use the up and down arrows to cycle through possible values.
- Right-click on the tag to display and select the available values.

For instance, if you have plate metadata, you might use the line, "CellProfiler: processing plate " and insert the metadata tag for the plate at the end. Please see the **Metadata** module for more details on metadata collection and usage.

Server name

Enter the address of your SMTP server. You can ask your network administrator for your outgoing mail server which is often made up of part of your email address, e.g., "Something@university.org". You might be able to find this information by checking your settings or preferences in whatever email program you use.

Port

Enter your server's SMTP port. The default (25) is the port used by most SMTP servers. Your network administrator may have set up SMTP to use a different port; also, the connection security settings may require a different port.

Select connection security

Select the connection security. Your network administrator can tell you which setting is appropriate, or you can check the settings on your favorite email program.

Username and password required to login?

Select *Yes* if you need to enter a username and password to authenticate.

Username

Enter your server's SMTP username.

Password

Enter your server's SMTP password.

Recipient address

Enter the address to which the messages will be sent.

When should the email be sent?

Select the kind of event that causes **SendEmail** to send an email. You have the following choices:

- *After first cycle:* Send an email during processing of the first image cycle.
- *After last cycle:* Send an email after all processing is complete.
- *After group start:* Send an email during the first cycle of each group of images.
- *After group end:* Send an email after all processing for a group is complete.
- *Every # of cycles* Send an email each time a certain number of image cycles have been processed. You will be prompted for the number of image cycles if you select this choice.
- *After cycle #:* Send an email after the given number of image cycles have been processed. You will be prompted for the image cycle number if you select this choice. You can add more events if you want emails after more than one image cycle.

Image cycle number

(Used only if sending email after a particular cycle number)
Send an email during processing of the given image cycle. For instance, if you enter 4, then **SendEmail** will send an email during processing of the fourth image cycle.

Image cycle count

(Used only if sending email after every N cycles)
Send an email each time this number of image cycles have been processed. For instance, if you enter 4, then **SendEmail** will send an email during processing of the fourth, eighth, twelfth, etc. image cycles.

Message text

The body of the message sent from CellProfiler. Your message can include metadata values. For instance, if you group by plate and want to send an email after processing each plate, you could use the message "Finished processing plate \g<Plate>".