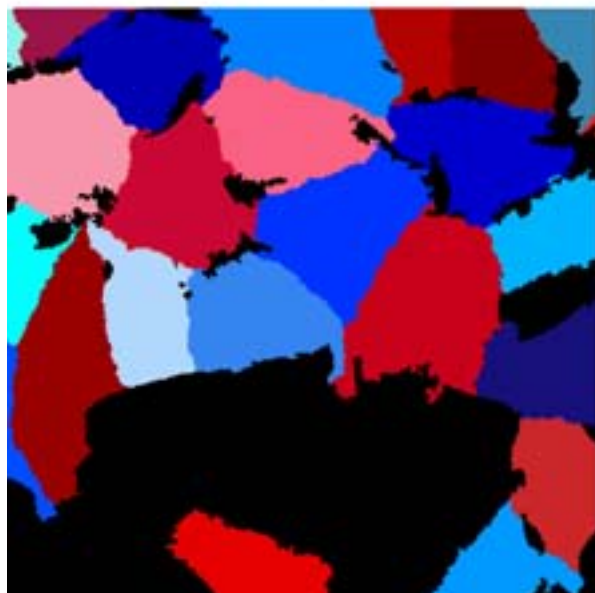
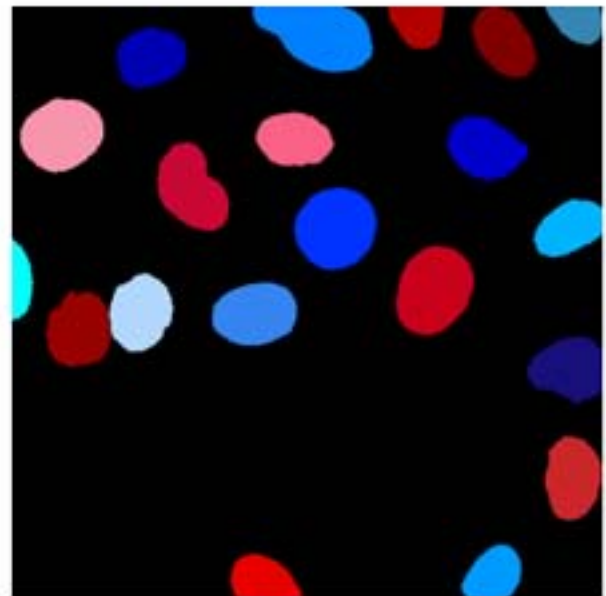
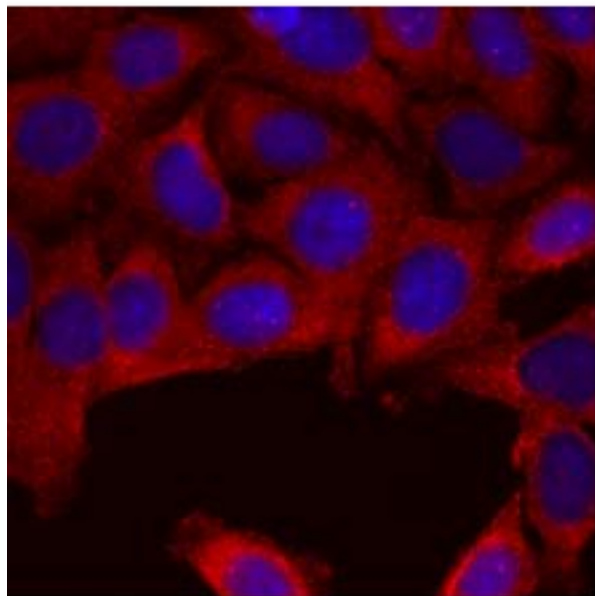




CellProfiler

cell image analysis software



	cell # <u>1</u>	<u>2</u> ...
Cell area	40.1	42.5
Cell perimeter	35.3	38.9
Cell aspect ratio	1.56	2.01
Actin content	4510	4939
Actin texture	16.8	17.2
Cell solidity	0.99	1.03
Cell extent	0.68	0.35
Nuclear area	10.9	11.1
Nuclear perimeter	1.0	1.1
Nuclear aspect ratio	1.56	2.01
...		

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



And now based at



CellProfiler is free and open-source!

If you find it useful, please credit CellProfiler in publications

1. Cite the website (www.cellprofiler.org)
2. Cite the publication (check the website for the citation).
3. Post the reference for your publication on the CellProfiler Forum (accessible from the website) 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.

Table of contents

Using CellProfiler

- **User guide**
 - **Getting started**
 - [When To Use CellProfiler](#)
 - [New Features](#)
 - [How To Build A Pipeline](#)
 - **General help**
 - [Using Metadata In CellProfiler](#)
 - [Memory And Speed](#)
 - [Test Mode](#)
 - [Batch Processing](#)
 - [Run multiple pipelines](#)
 - **Folders and files**
 - [Default Input Folder](#)
 - [Default Output Folder](#)
 - [Output Filename](#)
- **Module figures**
 - [File Save](#)
 - [Zoom](#)
 - [Measure Length Tool](#)
 - [Image Tools](#)
- **Preferences**
 - [Default Input Folder](#)
 - [Default Output Folder](#)
 - [Title font](#)
 - [Table font](#)
 - [Default colormap](#)
 - [Window background](#)
 - [Check for updates](#)
 - [Primary outline color](#)
 - [Secondary outline color](#)
 - [Tertiary outline color](#)

Help for CellProfiler Modules

• Data Tools

- [CalculateMath](#)
- [CalculateStatistics](#)
- [DisplayDataOnImage](#)
- [DisplayDensityPlot](#)
- [DisplayHistogram](#)
- [DisplayPlatemap](#)
- [DisplayScatterPlot](#)
- [ExportToDatabase](#)
- [ExportToSpreadsheet](#)
- [FlagImage](#)

• File Processing

- [CreateBatchFiles](#)
- [LoadData](#)
- [LoadImages](#)
- [LoadSingleImage](#)
- [RenameOrRenumberFiles](#)
- [SaveImages](#)

• Image Processing

- [Align](#)
- [ApplyThreshold](#)
- [CalculateImageOverlap](#)
- [ColorToGray](#)
- [CorrectIlluminationApply](#)
- [CorrectIlluminationCalculate](#)
- [Crop](#)
- [EnhanceEdges](#)
- [EnhanceOrSuppressFeatures](#)
- [FlipAndRotate](#)
- [GrayToColor](#)
- [ImageMath](#)
- [InvertForPrinting](#)
- [MakeProjection](#)
- [MaskImage](#)
- [Morph](#)
- [OverlayOutlines](#)
- [RescaleIntensity](#)
- [Resize](#)

- [RunImageJ](#)
- [Smooth](#)
- [Tile](#)

• **Measurement**

- [MeasureCorrelation](#)
- [MeasureImageAreaOccupied](#)
- [MeasureImageGranularity](#)
- [MeasureImageIntensity](#)
- [MeasureImageQuality](#)
- [MeasureNeurons](#)
- [MeasureObjectIntensity](#)
- [MeasureObjectNeighbors](#)
- [MeasureObjectRadialDistribution](#)
- [MeasureObjectSizeShape](#)
- [MeasureTexture](#)

• **Object Processing**

- [ClassifyObjects](#)
- [ConvertObjectsToImage](#)
- [EditObjectsManually](#)
- [ExpandOrShrinkObjects](#)
- [FilterObjects](#)
- [IdentifyObjectsInGrid](#)
- [IdentifyObjectsManually](#)
- [IdentifyPrimaryObjects](#)
- [IdentifySecondaryObjects](#)
- [IdentifyTertiaryObjects](#)
- [MaskObjects](#)
- [ReassignObjectNumbers](#)
- [RelateObjects](#)
- [TrackObjects](#)

• **Other**

- [ConserveMemory](#)
- [CreateWebPage](#)
- [DefineGrid](#)
- [InputExternal](#)
- [LabelImages](#)
- [OutputExternal](#)
- [PauseCellProfiler](#)
- [SendEmail](#)

When To Use CellProfiler

When should I 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

- 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. PMID: 17076895
- Lamprecht MR, Sabatini DM, Carpenter AE (2007) CellProfiler: free, versatile software for automated biological image analysis. *Biotechniques* 42(1):71-75. PMID: 17269487
- 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/doi: 10.1073/pnas.0808843106. PMID: 19188593 PMCID: PMC2634799
- 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/doi: 10.1186/1471-2105-9-482. PMID: 19014601 PMCID: PMC2614436

New Features

New Features in CellProfiler 2.0

A number of new features have been incorporated into this re-engineered Python version of CellProfiler:

Interface

- *Resizable user interface:* The main CellProfiler interface can now be resized by dragging the window corner.
- *Help for individual module settings:* Every setting in every module now has a help button that you can click to display information and advice for that setting.
- *Settings verification:* CellProfiler constantly checks for setting values that are not allowed, and immediately flags them for you.
- *Context-dependent module settings:* Prior versions of CellProfiler displayed all settings for each module, whether or not the values were necessary, given existing choices for other settings. Now, only those settings you require are displayed, simplifying the interface.
- *Test mode for assay development:* This feature allows you to preview the effect of a module setting on your data. You can step backward or forward in the pipeline as you modify settings, optimizing your results prior to running an actual analysis.
- *Unlimited number of images/objects as module input:* Some modules can accept an arbitrary number of images or objects as input, and you can dynamically add or remove any of these inputs as needed. For example, you can specify any number of single images in LoadSingleImage; previously, the module could accept only three input images at a time. For example, in OverlayOutlines, you can specify that any number of outlines be overlaid on an image; previously, you would have had to string multiple OverlayOutline modules together.
- *Image grouping:* Images which share common metadata tags, whether provided in the filename or in an accompanying text data file, can be processed together. This is useful for any situation in which the images are organized in groups and each group needs to be analyzed as an individual set, such as illumination correction for multiple plates.
- *Module drag and drop:* You can drag and drop selected modules within a pipeline or into another instance of CellProfiler, keeping their associated settings intact.
- *Listing of recent pipelines:* A selectable list of recently used pipelines is available from the menu bar, for easy access.
- *Figure display choice:* Easier access to which module figure display windows are shown. This functionality is now controlled within the pipeline, and is saved as part of the pipeline.


- *Context menus:* The pipeline panel responds to right-clicks, providing easy access to module manipulation or help.
- *Error handling:* This feature sends bug reports (stack traces) to our developers.
- *Better access for developers:* We are providing a developer's guide as a practical introduction for programming in the CellProfiler environment, an email list, and wiki, in addition to the available user forum.

Module improvements

- *Improved Otsu thresholding:* Choose two- or three-class thresholding to handle images where there might be an intermediate intensity level between foreground and background.
- Secondary object identification now permits discarding of objects touching the image border, along with the associated primary objects.
- Filtering objects by measurements now permits a set of objects to be filtered with any number of measurements.
- *Masking of images/objects:* You can create masks for use with both images and objects such that image/object measurements will include only those regions within the masked area.
- *Improved loading of text information:* Previously, you could load only a limited amount of annotation relevant to your images, via a text file. Now you can use comma-delimited files to load tables of metadata, in addition to file lists of input images for analysis.
- *Convex hull* has been included as an image morphological operation.
- A new module, MeasureNeurons, has been added, which measures the number of trunks and branches for each neuron in an image.
- *Detection of new features:* Neurites can be extracted from images of neurons. Branching points of line segments can be found as an image morphological operation. Also, "dark holes" (dark spots surrounded bright rings) can be detected.
- *Improvements to object tracking:* A new tracking algorithm has been added to the TrackObjects module which is capable of bridging temporal gaps in trajectories and accounting for splitting/merging events.
- *Per-object data exporting:* Object data can be exported to a database as a single table containing all user-defined object measurements, or as separate tables, one for each object.
- *SQLite support:* Data can be exported in SQLite, a self-contained database format. Users can create their own local databases and no longer need access to a separate database server. Because CellProfiler Analyst also supports SQLite, any user can access CellProfiler Analyst's suite of data exploration and machine-learning tools without installing a complicated database server.

How To Build A Pipeline

Making 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.

To adjust the CellProfiler source code, see *Help > Developer's Guide*.


Loading an existing pipeline




1. Put the images and pipeline into a folder on your computer.
2. Set the Default Input and Output Folders (lower right of the main window) to be the folder where you put the images.
3. Load the pipeline using *File > Load Pipeline* in the main menu of CellProfiler.
4. Click *Analyze images* 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 > Save Pipeline*.
7. To learn how to use a cluster of computers to process large batches of images, see *Help > General Help > 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 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, 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 up*  and *Move down*  buttons. The *Remove*  button will delete the selected module(s) from the pipeline.

Typically, the first module you must run is **LoadImages**, in which you specify the identity of the images you want to analyze.

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).

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: All measurements will be stored in the CellProfiler-formatted output file, but you can include an **Export** module to automatically export data in a format you prefer.

2. *Adjust the settings in each module.*

In the CellProfiler main window, click a module in the pipeline to see its settings in the main workspace. 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 reference to an image that doesn't exist yet), a  icon will appear next to the module name. Once the errors 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. *Use Test mode 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 > General Help > Test Mode* for more details.

6. Save your pipeline via *File > Save Pipeline*.

Using Metadata In CellProfiler

Using Metadata in CellProfiler

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.

Associating images with metadata

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*:

.*[V](?P<Date>.*)[V](?P<Run>.*)\$	
.*[V]	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 <i>Date</i>
.*	Capture as many characters that follow
[V]	Discard the slash/backslash character
(?P<Run>	Name the captured field <i>Run</i>
.*	Capture as many characters as follow
\$	The <i>Run</i> field must be at the end of the path string, i.e. the last folder on the path. This also means that the <i>Date</i> field contains the parent folder of the <i>Date</i> 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 identifiers. Similarly, if the individual images are loaded using the **LoadImages** module, **LoadImages** should be set to extract the metadata field from the file names. The pipeline will then match the individual images with their corresponding illumination correction functions based on matching "Metadata_Date" fields.

Use of metadata-specific module settings

Once the metadata has been obtained, you can use *metadata tags* to reference them in later modules. A metadata tag has the syntax "`\g<metadata-tag>`" where *<metadata-tag>* is the name of the previously defined metadata field. Several modules are capable of using metadata

tags for various purposes. Examples include:

- You would like to create and apply an illumination correction function to all images from a particular plate. You can use metadata tags to save each illumination correction function with a plate-specific name in **SaveImages**, and then use **LoadSingleImage** to get files with the name associated with your image's plate to be applied to your original images.
- You have a set of experiments for which you would like to produce and save results individually for each experiment but using only one analysis run. You can use metadata tags in **ExportToSpreadsheet** or **ExportToDatabase** to save a spreadsheet for each experiment in a folder named according to the experiment.

In each case, the pre-defined metadata tag is used to name a file or folder. Tags are case-sensitive; the name must match the metadata field defined by **LoadImages** or **LoadData**. The options for the setting will specify whether tags are applicable; see the module setting help for additional information on how to use them in the context of the specific module.

Memory And Speed

Help for memory and speed issues in CellProfiler

CellProfiler includes several options for dealing with out-of-memory errors associated with image analysis:

- *Resize the input images.*



If the image is high-resolution, it may be helpful to determine whether the features of interest can be processed (and accurate data obtained) by using a lower-resolution image. If this is the case, use the **Resize** module (in the *Image Processing* category) to scale down the image to a more manageable size and perform the desired operations on the smaller image.

- *Use the **ConserveMemory** module.*

The **ConserveMemory** module lets you clear the images stored in memory, with the exception of any you specify. Please see the **ConserveMemory** module help for more details.

In addition, there are several options in CellProfiler for speeding up processing:

- *Run without display windows.*

Each module is associated with a display window that takes time to render and/or update. Closing these windows improves speed somewhat. To the left of each module listed in your pipeline an icon  indicates whether the module window will be displayed during the analysis run. You can turn off individual module windows by clicking on the icon; this icon  indicates that the window will not be shown. Select *Window > Hide all windows on run* to prevent display of all module windows.

- *Use care in object identification*

If you have a large image which contains many small objects, a good deal of computer time will be spent processing each individual object, many of which you might not need. To avoid this, make sure that you adjust the diameter options in


IdentifyPrimaryObjects to exclude small objects in which you are not interested, or use a **FilterObjects** module to eliminate such objects.

Test Mode



Test mode for pipeline development

You can test an analysis 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 run* in 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* button. 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 cycle*: Skip ahead to the next image cycle as determined by the image order in **LoadImages/LoadData**. The slider will automatically return to the first module in the pipeline.

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

- *Stop test run*: 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)
- *Choose image / group*: Choose the image or group to jump to. The slider will then automatically return to the first module in the pipeline.
- *Reload modules source*: For developers only. This option will reload the module source code, so any changes to the code will be reflected immediately.

Batch Processing

Batch processing in CellProfiler

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. We often 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 *images* 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. In the CellProfiler folder panel, set the Default Input and Default Output Folders to the *images* and *output* folders created above, respectively.
4. *Create a pipeline for your image set.* You should test it on a few example images from your image set. 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 > General Help > Test Mode* 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 database or into a set of comma-separated files (CSVs) 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. *Analyze your images to create a batch file.* Click the *Analyze images* button and the analysis will begin locally processing the first image set only. Do not be surprised if processing the first image set takes much longer than usual if using **LoadImages** since this module creates a list of all images to be processed which can take a while if there are many of them (this process can be sped up by creating your list of images as a CSV and using the **LoadData** module to load it).

At the end of processing the first cycle locally, the **CreateBatchFiles** module halts execution, creates the proper batch file (a file called *Batch_data.mat*) and saves it in the Default Output Folder (Step 1). 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.mat -c -r -b -f <first_image_set_number> -l  
<last_image_set_number>
```

This command runs the batch by using additional options to CellProfiler that specify the following (type "CellProfiler.py -h" to see a list of available options):

- o -p <Default_Output_Folder_path>/Batch_data.mat: The location of the batch file, where <Default_Output_Folder_path> is the output folder path as seen by the cluster computer.
- o -c: Run "headless", i.e., without the GUI
- o -r: Run the pipeline specified on startup, which is contained in the batch file.
- o -b: Do not build extensions, since by this point, they should already be built.
- o -f <first_image_set_number>: Start processing with the image set specified, <first_image_set_number>
- o -l <last_image_set_number> : Finish processing with the image set specified, <last_image_set_number>

To submit all the batches for a full image set, you will need a script that calls CellProfiler with these options with sequential image set numbers, e.g, 1-50, 51-100, etc and submit

each as an individual job.

Once all the jobs are submitted, the cluster will run each batch individually and output any measurements or images specified in the pipeline. If requested, it will also produce a separate output (i.e., OUT.mat) file containing the data 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, please post your questions on the CellProfiler [forum](#).

Run multiple pipelines

Run multiple pipelines



The **Run multiple pipelines** dialog lets you select several pipelines which will be run consecutively. 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.

Default Input Folder

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.

The contents of the Default Input Folder are shown in the file panel to the left. Double-clicking image file names in this panel opens them in a figure window. If you double-click on .mat pipeline or output files (CellProfiler 1.0) or .cp pipeline files (CellProfiler 2.0), you will be asked if you want to load a pipeline from the file. To refresh the contents of this panel, click the *Refresh* button .

Default Output Folder

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.

Output Filename

Specify the name of the output file where all information about the analysis as well as any measurements will be stored to the hard drive. The output file is a .mat file, which is readable by CellProfiler and by MATLAB. Results in the output file can be accessed or exported using **Data Tools** from the main menu of CellProfiler. The pipeline with its settings can be loaded from an output file using *File > Load Pipeline...*, or by double-clicking the output file in the file list panel (located in the lower left corner of the CellProfiler main window).

The output file will be saved in the Default Output Folder unless you type a full path and file name into the output 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.

File Save

You can save the figure window to a file (currently, Postscript (.PS), PNGs and PDFs are supported). Note that this will save the entire contents of the window, not just the individual subplot(s) or images.

Zoom

- To zoom in, click and drag in the image 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.
- Zoom out is active only when you have zoomed into the field of view. Click any point within the current image window to zoom out to the previous zoom level; that is, each zoom out undoes the previous zoom in.

Measure Length Tool

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**.

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).
- *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 resultant 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.
- *Channels*: For color images only. You can show any combination of the red, green, and blue color channels.

Title font

The *Title Font* preference sets the font used in titles above plots displayed in module figure windows.

Table font

The *Table Font* preference sets the font used in tables displayed in module figure windows.

Default colormap

The *Default Colormap* preference specifies the color map that sets the colors for labels and other elements. See this [page](#) for pictures of available colormaps.

Window background

The *Window Background* preference sets the window background color of the CellProfiler main window.

Check for updates

The *Check for Updates* preference controls how CellProfiler looks for updates on startup.

Primary outline color

The *Primary Outline Color* preference sets the color used for the outline of the object of interest in the *IdentifyPrimaryObjects*, *IdentifySecondaryObjects* and *IdentifyTertiaryObjects* displays.

Secondary outline color

The *Secondary Outline Color* preference sets the color used for objects other than the ones of interest. In *IdentifyPrimaryObjects*, these are the objects that are too small or too large. In *IdentifySecondaryObjects* and *IdentifyTertiaryObjects*, this is the color of the secondary objects' outline.

Tertiary outline color

The *Tertiary Outline Color* preference sets the color used for the objects touching the image border or image mask in *IdentifyPrimaryObjects*.

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 features*: If both input measurements are whole-image measurements, then the result will also be a whole-image measurement.
- *Object features*: 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

What do you want to call the measurement calculated by this module?

Operation

What 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

Is the operand an image or object measurement?

Select the first operand objects

Which objects do 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

By what number would you like to multiply the above operand?

Raise the power of above operand by

To what power would you like to raise the above operand?

Select the second operand measurement type

Is the operand an image or object measurement?

Select the second operand objects

Which objects do 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?

Do you want the log (base 10) of the result?

Multiply the result by

(Used only for operations other than None)

By what number would you like to multiply the result?

Raise the power of result by

(Used only for operations other than None)

To what power would you like to raise the result?

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 $1 - 3 * (\sigma_p + \sigma_n) / |\mu_p - \mu_n|$ 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 $1 - 6 * \text{mean}(\sigma) / |\mu_p - \mu_n|$ 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.
- *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 **LoadData**.

Settings:

Where is information about the positive and negative control status of each image?

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 file outside of CellProfiler and then load that file into the pipeline using **LoadData**. In that case, choose the measurement that matches the column header of the measurement in **LoadData**'s input file. See **LoadData** help for an example text file.

Where is information about the treatment dose for each image?

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 text file outside of

CellProfiler and then load that file into the pipeline using **LoadData**. In that case, choose the measurement that matches the column header of the measurement in **LoadData**'s input file. See **LoadData** help for an example text file.

Log-transform dose values?

This option allows you to log-transform the dose values before fitting a sigmoid curve. Check this box if you have dose-response data. Leave the box unchecked if your data values indicate only positive vs. negative controls.

Create dose/response plots?

Check this box 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 **LoadImages** or **LoadData**, you can name the folder using metadata tags. Tags have the form `"\g<metadata-tag>"` where `<metadata-tag>` is the name of the previously defined metadata field. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

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?

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

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.

Text color

This is the color that will be used when displaying the text. There are several different ways by which you can specify the color:

- *Single letter*. "b"=blue, "g"=green, "r"=red, "c"=cyan, "m"=magenta, "y"=yellow,

"k"=black, "w"=white

- *Name*. You can use any name supported by HTML; a list of colors is shown on this: [page](#).
- *RGB code*. You can specify the color as a combination of the red, green, and blue intensities, for instance, "#FFFF00" for yellow; yellow = red("FF") + green("FF") + blue("00"), where *FF* is hexadecimal for 255, the highest intensity. See [here](#) for a more detailed explanation

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.

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.

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.

Do you wish to specify min/max bounds for the x-axis

Enter the min & max values for the x-axis

DisplayPlatemap

Display Platemap displays a desired measurement in plate map view

A plate map is a...

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 object measurement to plot

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

What type of plate is the data from?

How should the values be aggregated?

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.

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

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 measurement to plot on the X-axis

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

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 **LoadData**, 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.

See also **ExportToSpreadsheet**.

Settings:

Database type

What type of database do you want to use?

- *MySQL* allows the data to be written 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 at [here](#).

Database name

Select a name for the database you want to use

Add a prefix to table names?

Do you want to add a prefix to your table names? This option enables you to prepend text to your table names (Per_Image and Per_Object). CellProfiler will warn you before overwriting an existing table.

Table prefix

(Used if *Add a prefix to table names?* is selected)

What is the table prefix you want to use?

SQL file prefix

(Used if *SQL* is selected as the database type and if *CSV* files are to be written)

What prefix do you want to use to name the SQL file?

Output file location

(Used only when saving csvs, or creating a properties file)

This setting determines where the .csv files are saved if you decide to save measurements to files instead of writing them directly to the database. 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 **LoadImages** or **LoadData**, you can name the folder using metadata tags. Tags have the form *"/g<metadata-tag>"* where *<metadata-tag>* is the name of the previously defined metadata field. 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 last image set processed for any metadata tags in the folder name. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more

details on obtaining, extracting, and using metadata tags from your images.

Create a CellProfiler Analyst properties file?

You can 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 or Oracle is used.

Name the SQLite database file

(Used if SQLite selected as database type)

What is the SQLite database filename to which you want to write?

Calculate the per-image mean values of object measurements?

ExportToDatabase can 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?

ExportToDatabase can 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_Mean", with a column called "Mean_Nuclei_AreaShape_Area".

Note: this option is only available if you have extracted plate and well metadata from the filename or via a **LoadData** module. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images

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 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.

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.

Create one table per object or a single object table?

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 all 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.

Enter an image url prepend if you plan to access your files via http (leave blank if local)

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/"`

Write image thumbnails directly to the database?

Check this option 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.

Select the images you want to save thumbnails of

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.

Metadata tokens

ExportToSpreadsheet can write out separate files for groups of images based on their metadata. 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 *Custom folder with metadata*, enter the directory name "\g<Treatment1>" and enter the file name "\g<Treatment2>". 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

Settings:

Select or enter the column delimiter

What delimiter do you want to use? This is 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).

Prepend the output file name to the data file names?

This can be useful if you want to run a pipeline multiple times without overwriting the old results.

Add image metadata columns to your object data file?

"Image_Metadata_" columns are normally exported in the Image data file, but if you check this box 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, a window will open which allows 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 columns of measurements to export?

Checking this setting will open up a window that allows you to select the columns to export.

Calculate the per-image mean values for object measurements?

ExportToSpreadsheet can 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 **LoadImages** or **LoadData**, you can name the folder using metadata tags. Tags have the form "g<metadata-tag>" where <metadata-tag> is the name of the previously defined metadata field. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

Create a GenePattern GCT file?

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, checking this box 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 **LoadData** or **LoadImages** to input your images, you may use a per-image data measurement (such as metadata) that corresponds to the identifier for this column. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.
- *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 **LoadData**'s input file. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

Export all measurements?

Check this setting to export every measurement. **ExportToSpreadsheet** will create one file per object type, including image and experiment. It will use the object name as the file name, optionally prepending the output file name if specified above. Leave this box unchecked to specify which objects should be exported or to override the automatic names.

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

Data to export

(Used only when Export all measurements? is left unchecked)
Choose *Image*, *Experiment*, or an object name from the list. **ExportToSpreadsheet** will write out a file of measurements for the given category.

Combine these object measurements with those of the previous object?

(Used only when Export all measurements? is left unchecked)

Check this setting to create a file composed of measurements made on this object and the one directly above it. Leave the box unchecked to create separate files for this and the previous object.

File name

(Used only when Export all measurements? is left unchecked)

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. Tags have the form "`\g<metadata-tag>`" where `<metadata-tag>` is the name of the previously defined metadata field. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

Use the object name for the file name?

(Used only when Export all measurements? is left unchecked)

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". You can name the file yourself if you leave this box unchecked.

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 **LoadData** module can then use this flag to put images that pass QC into one group and images that fail into another. If you plan to use a flag in **LoadData**, give it a category of "Metadata" so that it can be used in grouping.

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 in which the flag should reside. Metadata allows you to later group images in the **LoadImages** module based on the flag, if you load the flag data in a future pipeline via the **LoadData** module. Otherwise, you might choose to have the flag stored in the "Image" category or using some other word you prefer. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images

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, 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".

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

- *Any*: 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.
- *All*: 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?

You can skip the remainder of the pipeline for image sets that are flagged by checking this setting. If you check this setting, CellProfiler will not run subsequent modules in the pipeline on the images in any image set that is flagged. CellProfiler will continue to process the pipeline if you leave the setting unchecked.

You may want to check this setting in order to filter out unwanted images during processing. 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.

Select the object whose measurements will be used to flag

(Used only when flag is based on an object measurement)

What did you call the objects whose measurements you want to use for flagging?

Flag images based on low values?

Images with measurements below this cutoff will be flagged.

Flag images based on high values?

Images with measurements above this cutoff will be flagged.

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:\imaging_analysis
```

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

```
/imaging/analysis
```

In this case, the local root path is `Z:\imaging_analysis` and the cluster root path is `/imaging/analysis`.

Settings:

Store batch files in default output folder?

Do you want to store the batch files in the default output folder? Check this box to store batch files in the Default Output folder. Uncheck the box to enter the path to the folder that will be used to store these files.

Output folder path

What is the path to the output folder?

Are the cluster computers running Windows?

Check this box if the cluster computers are running one of the Microsoft Windows operating systems. If you check this box, **CreateBatchFiles** will modify all paths to use the Windows file separator (backslash \). If you leave the box unchecked, **CreateBatchFiles** will modify all

paths to use the Unix or Macintosh file separator (slash,/).

Local root path

What is 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 beginning, 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

What is 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.

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.

The module currently reads files in CSV (comma-separated values) format. These files can be produced by spreadsheet programs and are organized into rows and columns. The lines of the file represent the rows. (Technically, each row is terminated by the newline character ASCII 10.) 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**; using particular names for columns allows 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.* A column whose name begins with "Image_FileName" can be used to supply the file name 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.
- *Columns whose name begins with Image_PathName.* A column whose name begins with "Image_PathName" can be used to supply the path name of an image that you want to load (relative to the base folder). The image's name within CellProfiler appears afterward. For instance, "Image_PathName_CY3" would supply the path names for the CY3-stained images. This is optional; if all image files are in the base folder, this column is not needed.
- *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 identifiers. Similarly, if the individual images are loaded using the **LoadImages** module, **LoadImages** should be set to extract the metadata field 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" fields.

- *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**.

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-specific options in other modules; see the settings help for *Group images by metadata* and, if that setting is selected, *Select metadata fields for grouping* for details.

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.

See also **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?

Check this box 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?

Check this box 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?

Use this option 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 fields for grouping

(Used only if images are to be grouped by metadata)

Select the fields 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 fields.

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.

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-specific 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.

See also **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 movies*: An AVI (Audio Video Interleave) file is a type of movie file. Only uncompressed AVIs are supported.
- *TIF, TIFF, FLEX 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).
- *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.

File selection method

Three options are available:

- *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).
- *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. A more extensive explanation of regular expressions can be found [here](#) and a helpful quick reference can be found [here](#).

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:

- o [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 any string that is a natural number or ''
- ^-[0-9]+\$|^\\+?[0-9]+\$ matches any integer

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 option 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).

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?

If this box is checked, **LoadImages** will search all the subfolders under your specified image folder location for images matching the criteria above.

Check image sets for missing or duplicate files?

Selecting this option will examine the filenames for unmatched or duplicate files based on the filename prefix (such as those generated by HCS systems).

Group images by metadata?

In some instances, you may want to process as a group those images that share a particular metadata tag. 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.

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.

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.

Channel number:

(Used only for multichannel images) 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.

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 <i>Plate</i>

.*	Capture as many characters as follow
_	Discard the underbar separating plate from well
(?P<Well>	Name the captured field <i>Well</i>
[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 <i>s</i> separating well from site
(?P<Site>	Name the captured field <i>Site</i>
[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.

Also 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:

.*[V](?P<Date>.*)[V](?P<Run>.*)\$	
.*[V]	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 <i>Date</i>

.*	Capture as many characters that follow
[\\]	Discard the slash/backslash character
(?P<Run>	Name the captured field <i>Run</i>
.*	Capture as many characters as follow
\$	The <i>Run</i> 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.

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.*

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.

Technical note

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 **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 **LoadImages** or **LoadData**, you can name the folder using metadata tags. Tags have the form "\g<metadata-tag>" where <metadata-tag> is the name of the previously defined metadata field. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

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. Tags have the form "\g<metadata-tag>" where <metadata-tag> is the name of the previously defined metadata field. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images

In either case, the extension, if any, should be included.

Name the image that will be loaded

What do you want to call the image you are loading? You can use this name to select the image in downstream modules.

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 **LoadImages**.

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
- Leave the text addition box unchecked

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

<i>Original name</i>	<i>New name</i>
1DrosophilaDAPI_001.tif	1DrosDP_001.tif
1DrosophilaDAPI_010.tif	1DrosDP_010.tif
1DrosophilaDAPI_100.tif	1DrosDP_100.tif

Settings:

Select the input image

Select the images associated with the files you want to rename. This should be an image loaded by **LoadImages**, **LoadData**, or **LoadSingleImage**. 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?

You can check this setting if you want to add text to the file name. If you chose *Renumber* above, the module will add the text after your number. If you chose *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?

Check this setting to replace spaces in the final version of the file name with some other text. Leave it unchecked 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.

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 **LoadImages**, **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.
- *Crop 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.
- *Image's 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.
- *Module display window*: The window associated with a module can be saved, which will

include all the panels and text within that window. **Currently, this option is not yet available.**

Note that objects cannot be directly saved with the **SaveImages** module. You must first use the **ConvertObjectsToImage** module to convert the objects to an image, followed by **SaveImages**.

Select the image to save

(Used only if saving images, crop masks, and image croppings)

What did you call the images you want to save?

Select the module display window to save

(Used only if saving module display windows)

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)

Four choices are available:

- *From image filename:* The filename will be constructed based on the original filename of an input image specified in **LoadImages** or **LoadData**. You will have the opportunity to prefix or append additional text.
- *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 file name:* A single, fixed name will be given to the file, with no additional text prefixed or appended. Since the filename is fixed, this file will be overwritten with each cycle. Unless you want the file to be updated every cycle during the analysis run, you would probably want to save it on the last cycle (see the [Select how often to save](#) setting)
- *Name with metadata:* The filenames are constructed using the metadata associated with an image cycle 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. Tags have the form "`\g<metadata-tag>`" where `<metadata-tag>` is the name of the previously defined metadata field. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images
- *Image filename with metadata:* This is a combination of *From image filename* and *Name with metadata*. If you have metadata associated with your images, you can append an

extension to the image filename using a metadata tag.

Select image name for file prefix

(Used only when constructing the filename from the image filename, with or without metadata)
Select an image loaded using **LoadImages** or **LoadData**. The original filename will be used as the prefix for the output filename.

Enter single file name

(Used only when constructing the filename from the image filename, a single name or a name with metadata)

If you are constructing the filenames using...

- *Single name:* Enter the filename text here
- *Custom with metadata:* If you have metadata associated with your images, enter the filename text with the metadata tags. Tags have the form "`\g<metadata-tag>`" where `<metadata-tag>` is the name of the previously defined metadata field. For example, if the `plate`, `well_row` and `well_column` tags have the values `XG45`, `A` and `01`, respectively, the string "`Illum_\g<plate>_\g<well_row>_\g<well_column>`" produces the output filename `Illum_XG45_A01`.

Do not enter the file extension in this setting; it will be appended automatically.

Do you want to add a suffix to the image file name?

Check this setting to add a suffix to the image's file name. Leave the setting unchecked 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.

Select file format to use

(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 **LoadImages** or **LoadData**, you can name the folder using metadata tags. Tags have the form *"/g<metadata-tag>"* where *<metadata-tag>* is the name of the previously defined metadata field. 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 **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

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)

What is 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?

Check this box to automatically overwrite a file if it already exists. Otherwise, you will be prompted for confirmation first.

Select how often 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 **LoadImages/LoadData**.
- *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)

Check this box 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.

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](#).

Update file names within CellProfiler?

This setting stores filename and pathname data for each of the new files created via this module, as a per-image measurement.

This setting is useful when 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 check this box if you want the images you are saving via this module to be displayed along with the original images.

This setting also allows downstream modules (e.g., **CreateWebPage**) to look up the newly saved files on the hard drive. Normally, whatever files are present on the hard drive when CellProfiler processing begins (and when the **LoadImages** module processes its first cycle) are the only files recognized within CellProfiler. This setting allows the newly saved files to be accessible to downstream modules. This setting might yield unusual consequences if you are using the **SaveImages** module to save an image directly as loaded (e.g., using the **SaveImages** module to convert file formats), because in some places in the output file it will overwrite the file names of the loaded files with the file names of the saved files. Because this function is rarely needed and might introduce complications, the default setting is unchecked.

Create subfolders in the output folder?

Subfolders will be created to match the input image folder structure.

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.

Available measurements

- *XShift, Yshift*: The pixel shift in X and Y of the Nth aligned image with respect to the first image.

Settings:

Select the alignment method

Two options for the alignment method are available:

- *Mutual Information*: Alignment works whether the images are correlated (bright in one = bright in the other) or anti-correlated (bright in one = dim in the other).
- *Normalized Cross Correlation*: Alignment works only when the images are correlated (bright in one = bright in the other). When using the cross correlation method, the second image should serve as a template and be smaller than the first image selected.

Crop output images to retain just the aligned regions?

If cropping is chosen, all output images are cropped to retain just those regions that exist in all channels after alignment. If cropping is not chosen, the unaligned portions of each image are padded (with zeroes) and appear as black space.

Select the first input image

What is the name of the first image to align? Regardless of the number of input images, they

will all be aligned with respect to the first image.

Name the first output image

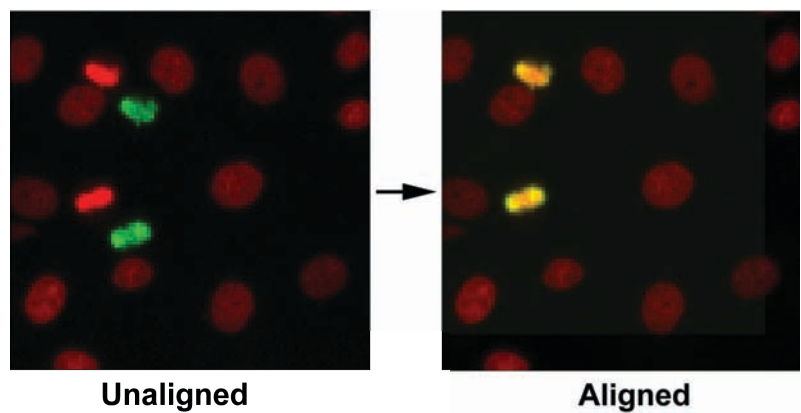
What is the name of the first aligned image?

Select the second input image

What is the name of the second image to align?

Name the second output image

What is the name of the second aligned image?



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

Specify the image to be thresholded.

Name the output image

Give a name to 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*: 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 thresholding a grayscale image) 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 image is grayscale and pixels below a given intensity are to be set to zero) Use this setting if the dim pixels are to be shifted in value by the amount of the threshold.

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.

Select the thresholding method

The intensity threshold affects the decision of whether each pixel will be considered foreground (regions of interest) or background. A stringent threshold will result in only bright regions being identified, with tight lines around them, whereas a lenient threshold will include dim regions and the lines between regions and background will be more loose. You can have the threshold automatically calculated using several methods, or you can enter an absolute number between 0 and 1 for the threshold (to see the pixel intensities for your images in the appropriate range of 0 to 1, use *Tools > Show pixel data* in a window showing your image). Both options have advantages. An absolute number treats every image identically, but is not robust with regard to slight changes in lighting/staining conditions between images. An automatically calculated threshold adapts to changes in lighting/staining conditions between images and is usually more robust/accurate, but it can occasionally produce a poor threshold for unusual/artifactual images. It also takes a small amount of time to calculate.

The threshold that is used for each image is recorded as a measurement in the output file, 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.

There are six methods for finding thresholds automatically:

- *Otsu*: This method is probably best if you are not able to make certain assumptions about every images in your experiment, especially if the percentage of the image covered by regions of interest varies substantially from image to image. Our implementation takes into account the maximum and minimum values in the image and log-transforming the image prior to calculating the threshold. 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%.
- *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. 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 is simple and appropriate for images in which most of the image is background. It 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). Note that the mode is protected from a high number of saturated pixels by counting only pixels < 0.95 . This can be very helpful, for example, if your images vary in overall brightness but the objects of interest are always twice (or actually, any constant) as bright as the background of the image.
 - *Robust background*: This method trims the brightest and dimmest 5% of pixel intensities in the hopes that the remaining pixels represent a Gaussian of intensity values that are mostly background pixels. It then calculates the mean and standard deviation of the remaining pixels and calculates the threshold as the mean + 2 times the standard deviation.
 - *Ridler-Calvard*: This method is simple and its results are often very similar to Otsu's. According to Sezgin and Sankur's paper (*Journal of Electronic Imaging*, 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). Ridler-Calvard 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, repeating this until the threshold converges.
 - *Kapur*: This method computes the threshold of an image by log-transforming its values, then searching for the threshold that maximizes the sum of entropies of the foreground and background pixel values, when treated as separate distributions.

You can also choose between *Global*, *Adaptive*, and *Per-object* thresholding for the automatic methods:

- *Global*: One threshold is calculated for the entire image (fast)
- *Adaptive*: The calculated threshold varies across the image. This method is a bit slower but may be more accurate near edges of regions of interest, or where illumination variation is significant (though in the latter case, using the **CorrectIllumination** modules is preferable).
- *Per-object*: If you are using this module to find child objects located *within* parent objects, the per-object method will calculate a distinct threshold for each parent object.

This is especially helpful, for example, when the background brightness varies substantially among the parent objects.

Important: the per-object method requires that you run an **IdentifyPrimaryObjects** module to identify the parent objects upstream in the pipeline. After the parent objects are identified in the pipeline, you must then also run a **Crop** module with the following inputs:

- The input image is the image containing the sub-objects to be identified.
- Select *Objects* as the shape to crop into.
- Select the parent objects (e.g., *Nuclei*) as the objects to use as a cropping mask.

Finally, in the **IdentifyPrimaryObjects** module, select the cropped image as input image.

Selecting *manual thresholding* allows you to enter a single value between 0 and 1 as the threshold value. This setting can be useful when you are certain what the cutoff should be and it does not vary from image to image in the experiment. If you are using this module to find objects in an image that is already binary (where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

Selecting thresholding via a *binary image* will use the binary image as a mask for the input image. Note that unlike **MaskImage**, the binary image will not be stored permanently as a mask. Also, 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.

Selecting thresholding via *measurement* will use an image measurement previously calculated in order to threshold the image. Like manual thresholding, this setting 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 a Measurement module.

Manual threshold

(Used only if Manual selected for thresholding method)

Enter the value that will act as an absolute threshold for the images, in the range of [0,1].

Lower and upper bounds on threshold

Enter the minimum and maximum allowable threshold, in the range [0,1]. This is helpful as a safety precaution when the threshold is calculated automatically. For example, if there are no objects in the field of view, the automatic threshold might be calculated as unreasonably low. In such cases, the lower bound you enter here will override the automatic threshold.

Threshold correction factor

When the threshold is calculated automatically, it may consistently be too stringent or too lenient. You may need to enter an adjustment factor that you empirically determine is suitable for your images. The number 1 means no adjustment, 0 to 1 makes the threshold more lenient and greater than 1 (e.g., 1.3) makes the threshold more stringent. 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.

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.

Select the input objects

(Used only if a per-object thresholding method is selected)

Select the objects you identified previously and would like to now further identify sub-objects within.

Two-class or three-class thresholding?

(Used only for the Otsu thresholding method)

Select *Two* if the grayscale levels are readily distinguishable into only two classes: foreground (i.e., objects) and background. Select *Three* if the grayscale levels fall instead into three classes. 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.

For example, 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)

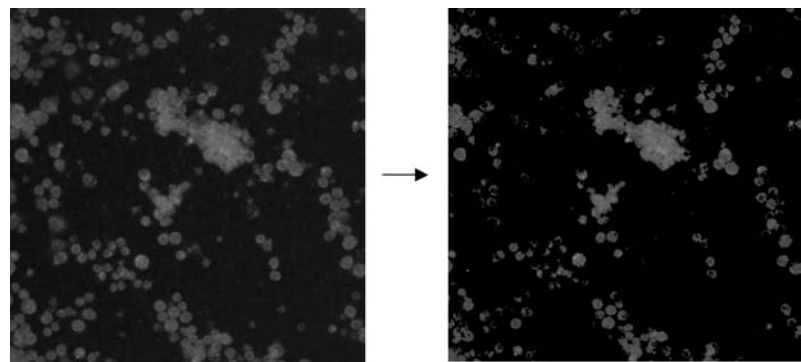
Choose whether you want the pixels with middle grayscale intensities to be assigned to the

foreground class or the background class.

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.



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 precision, recall, F-factor, false positive rate, and false negative rate. One image is considered the "ground truth" (possibly the result of hand-segmentation) and the other is the "test image"; the images are determined to overlap most completely when the test image matches the ground truth perfectly. The module requires binary (black and white) input, where the objects that have been segmented are 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

- *False positive rate*: Total number of false positive pixels / total number of actual negative

pixels

- *False negative rate*: Total number of false negative pixels / total number of actual positive 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*: $2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$. Also known as F_1 score, F-score or F-measure.

Settings:

Select the image to be used as the ground truth basis for calculating the amount of overlap

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

This binary (black and white) image is what you will compare with the ground truth image. It is known as the "test image".

ColorToGray

Color to Gray converts an image with three color channels to one *or* three 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.

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)

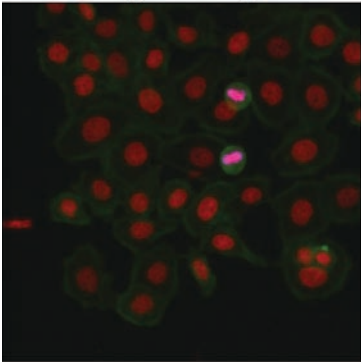
Relative weights: If all relative weights are equal, all three colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight of the blue channel

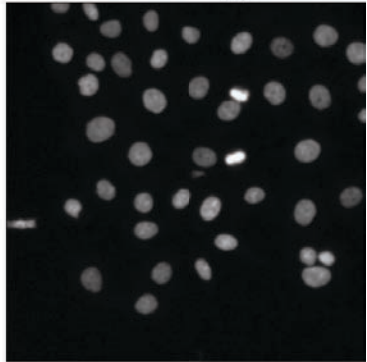
(Used only when combining channels)

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.

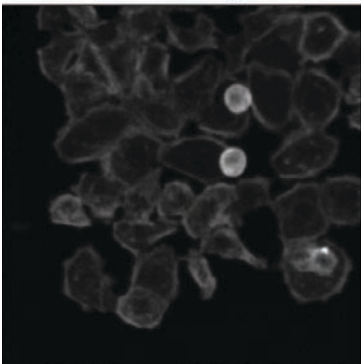
Original image



Blue image



Green image



Red image



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

What did you call the image to be corrected?

Name the output image

What do you want to call the corrected image?

Select the illumination function

What did you call the illumination correction function image that will be used to carry out the correction (produced by another module or loaded as a .mat format image using **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.

- Select *Subtract* 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.
- Select *Divide* 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.

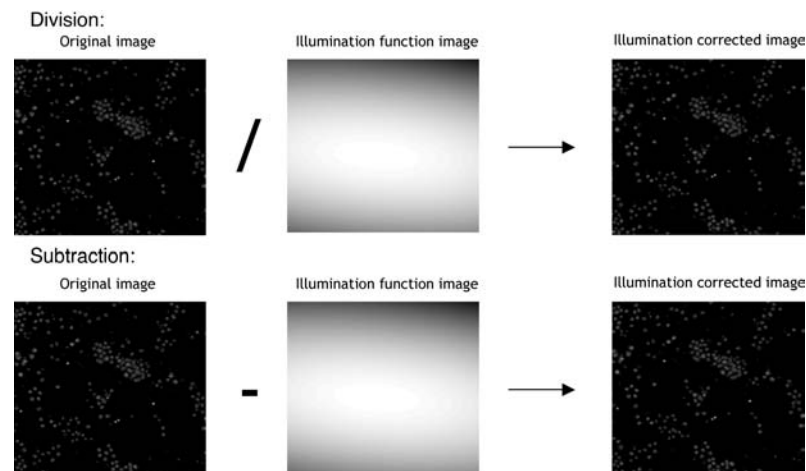
Select the rescaling method

(Used only when applying the illumination correction via divide)

Applying an illumination function can produce an image in a very different range of intensity values relative to the original image. If the illumination correction function is in the range 1 to infinity, *Divide* will usually yield an image in a reasonable range (0 to 1). However, if the image is not in this range, or the intensity gradient within the image is still very great, you may want to rescale the image. There are two methods for rescaling:

- Stretch the image from 0 to 1.
- Match the maximum of the corrected image to the maximum of the original image.

Rescaling is only necessary when using the *Divide* method. (In the *Subtract* method, any resulting negative pixels are set to zero, so no rescaling is necessary.)



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 CellProfiler website for further advice.

See also **CorrectIlluminationApply**, **EnhanceOrSuppressFeatures**.

Settings:

Select the input image

What did you call the images to be used to calculate the illumination function?

Name the output image

What do you want to call the illumination function?

Select how the illumination function is calculated

Do you want to calculate using regular intensities or background intensities?

- *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 intensities*: 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*.

Dilate objects in the final averaged image?

(Used only if the Regular method is selected)

Do you want to dilate objects in the final averaged image? 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.

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. Rescaling is recommended if you plan to use the *Division* option in

CorrectIlluminationApply so that the corrected images are in the range 0 to 1. It is not recommended if you plan to use 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. The *Median* 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 produced by a **LoadImage** or **LoadData** module; 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; fitting a polynomial is fastest but does not allow a very tight fit compared to the slower median and Gaussian filtering methods. We typically recommend median vs. Gaussian because 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. A less commonly used option is to completely smooth the entire image by choosing *Smooth to average*, 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.

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)

What is the approximate width of the artifacts to be smoothed, in pixels?

Smoothing filter size

(Used only if Manual is selected for smoothing filter size calculation)

What is the size of the desired smoothing filter, in pixels?

Retain the averaged image for use later in the pipeline (for example, in SaveImages)?

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.

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 for use later in the pipeline (for example, in SaveImages)?

The dilated image is the illumination function after dilation but prior to smoothing. It is an image produced during the calculations, not typically needed for downstream modules.

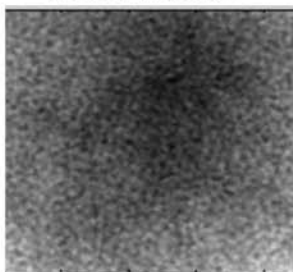
Name the dilated image

(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.

Regular intensities, using All images:

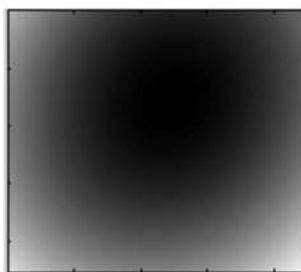
Mean of all images



optional
smoothing



Illumination function image



Regular intensities, using Each image:

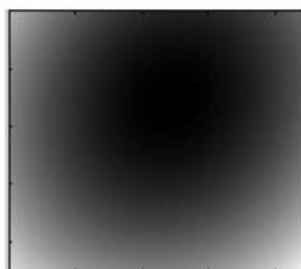
Original image



smoothing



Illumination function image



Background intensities, using all images:

Mean background of all images



optional
smoothing



Illumination function image



Original image



smoothing



Illumination function image



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

What did you call the image to be cropped?

Name the output image

What do you want to call the cropped image?

Select the cropping shape

Into which shape would you 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. A choice box with available images appears if you choose *Cropping*. The images in this box are ones 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

Would you 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. **This functionality has not yet been implemented.**

Apply which cycle's cropping pattern?

Should the cropping pattern in the first image cycle be applied to all subsequent image cycles (*First*) or should every image cycle be cropped individually (*Every*)?

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* to specify these values as absolute pixel coordinates in the original image. For

instance, you might enter "25", "225", and "Absolute" to create a 200x200 pixel image that is 25 pixels from the top-left corner.

- *From edge* to specify position relative to the original image's 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* to specify these values as absolute pixel coordinates. For instance, you might enter "25", "225", and "Absolute" to create a 200x200 pixel image that's 25 pixels from the top-left corner.
- *From edge* to 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)

What is the center pixel position of the ellipse?

Ellipse radius, X direction

(Used only if Ellipse selected as cropping shape)

What is the radius of the ellipse in the X direction?

Ellipse radius, Y direction

(Used only if Ellipse selected as cropping shape)

What is the radius of the ellipse in the Y direction?

Use Plate Fix?

(Used only if Image selected as cropping shape)

Do you want to use Plate Fix? 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

IlluminationCorrection). 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?

Do you want to remove rows and columns that lack objects? Options are:

- *No:* Leave the image the same size. The cropped areas will be turned to black (zeroes)
- *Edges:* Crop the image so that its top, bottom, left and right are at the first nonblank 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)

What is the name of the image to use as a cropping mask?

Select the image with a cropping mask

(Used only if Previous Cropping selected as cropping shape)

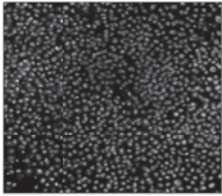
What is the name of the image with the associated cropping mask?

Select the objects

(Used only if Objects selected as cropping shape)

What are the objects to be used as a cropping mask?

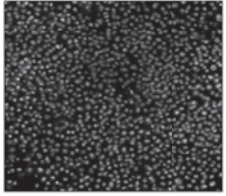
Rectangle: Enter the pixel coordinates for the left and right, and top and bottom corners



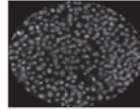
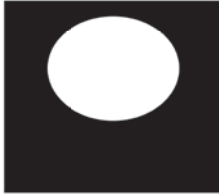
left, right = 50, 130
top, bottom = 115



Image: Provide a prepared black and white image which shows the cropping shape



+



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)

If you select automatic thresholding, it is done using a three-category Otsu algorithm performed on the Sobel transform of the image.

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 threshold entered manually entered.

Select an edge-finding method

There are several methods that can be used to enhance edges:

- *Sobel Method*: 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 Method*: Finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of the image is maximum.
- *Roberts Method*: 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 Method*: Applies a Laplacian of Gaussian filter to the image and finds zero crossings.
- *Canny Method*: 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)

Automatically calculate the low / soft threshold cutoff for the Canny method

Low threshold value

(Used only with the Canny option and manual thresholding)

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.

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

What did you call the image with features to be enhanced or suppressed?

Name the output image

What do you want to call the feature-enhanced or suppressed image?

Select the operation

Do you want to enhance or suppress the features you designated?

- *Enhance* produces an image whose intensity is largely composed of the features of interest.
- Choose *Suppress* to produce an image with the features largely removed.

Feature size

(Used only if speckles or neurites are selected, or if suppressing features)

What is the feature size? The diameter of the largest speckle (or the width of the neurites) to be enhanced or suppressed, which will be used to calculate an adequate filter size. If you don't know the width of your objects, use the *Tools > Show pixel data* image tool in the image window menu to find out.

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. The module takes the difference of the white and black tophat filters (a black tophat filter is the morphological grayscale opening of the image minus the image itself). The effect is to enhance lines whose width is the "feature size".
- *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.

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 hole detection is selected)

The range of hole sizes to be enhanced. The algorithm will identify only holes whose diameters fall between these two values

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

How do you want to flip the image? Left to right, Top to bottom, or both?

Select method to rotate image

- *Angle*: Provide the numerical angle by which the image should be rotated.
- *Coordinates*: Provide the X,Y pixel locations of two points in the image that should be aligned horizontally or vertically.
- *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 unless you choose to crop away the incomplete rows and columns of the image. 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 rotating images with the mouse)

Do you want to determine the amount of rotation for each image individually as you cycle through, or do you want to define it only once (on the first image) and then apply it to all images?

Select how the specified points should be aligned

(Used only when rotating images by entering coordinates)

Should the points you specified be horizontally or vertically aligned after the rotation is complete?

Enter angle of rotation

(Used only when rotating images by entering an angle)

By what angle would you like to rotate the image (in degrees; positive = counterclockwise and negative = clockwise)?

GrayToColor

Gray to Color takes grayscale images 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 (arbitrary number).

Relative weight for the red image

(Used only if RGB is selected)

Enter the 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 for the green image

(Used only if RGB is selected)

Enter the 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 for the blue image

(Used only if RGB is selected)

Enter the 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 for the cyan image

(Used only if CMYK is selected)

Enter the relative weights: 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)

Enter the relative weights: 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)

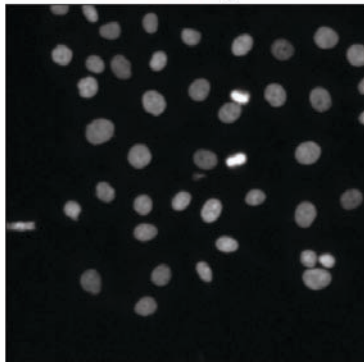
Enter the relative weights: If all relative weights are equal, all colors contribute equally in the final image. To weight colors relative to each other, increase or decrease the relative weights.

Relative weight for the brightness image

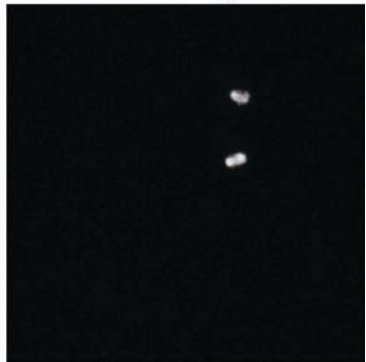
(Used only if CMYK is selected)

Enter the relative weights: 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.

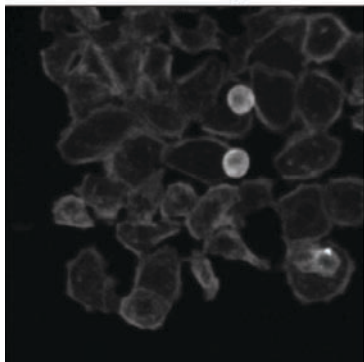
Blue image



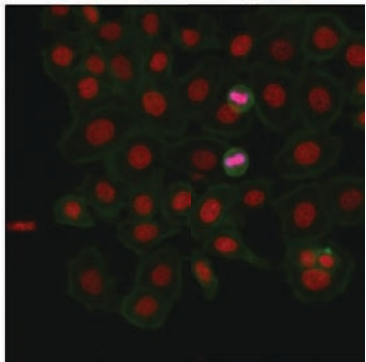
Red image



Green image



Merged image



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.

Multiply factors: The final image may have a substantially different range of pixel intensities than the originals, so each image can be multiplied by a factor prior to the operation. This factor can be any real number. See the **RescaleIntensity** module for more scaling options.

See also **ApplyThreshold**, **RescaleIntensity**, **CorrectIllumination_Calculate**.

Settings:

Operation

What operation would you like performed?

Note: If more than 2 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.
- *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.
- *None* 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*, and *None* operate on only a single image.

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?

Do you want negative values to be set to 0? Values outside the range 0 to 1 might not be handled well by other modules. Here you have the option of setting negative values to 0.

Set values greater than 1 equal to 1?

Do you want values greater than one to be set to 1? Values outside the range 0 to 1 might not be handled well by other modules. Here you have the option of setting values greater than 1 to a maximum value of 1.

Name the output image

What do you want to call the resulting image?

Which image do you want to use for this operation?

By what number would you like to multiply the above image? This multiplication is applied before other operations.

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

Are you combining several grayscale images or loading a single color image?

Use a red image?

Do you want to load an image for the red channel? Leave checked unless there is no red component to your grayscale image.

Select the red image

What did you call the red image?

Use a green image?

Do you want to load an image for the green channel? Leave checked unless there is no red component to your grayscale image.

Select the green image

What did you call the green image?

Use a blue image?

Do you want to load an image for the blue channel? Leave checked unless there is no red component to your grayscale image.

Select the blue image

What did you call the blue image?

Select the color image

What did you call the color image?

Output image type

Do you want to produce several grayscale images or one color image?

Produce a red image?

Do you want to produce an image for the red channel?

Name the red image

(Used only when producing a red image)

What do you want to call the red image?

Produce a green image?

Do you want to produce an image for the green channel?

Name the green image

(Used only when producing a green image)

What do you want to call the green image?

Produce a blue image?

Do you want to produce an image for the blue channel?

Name the blue image

(Used only when producing a blue image)

What do you want to call the blue image?

Name the inverted color image

What do you want to call the inverted color image?

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 either averaging, summing or taking the maximum or minimum pixel intensity at each pixel position. The process of averaging or summing a Z-stack (3D image stack) is known as making a projection.

The image is not immediately available in subsequent modules because the output of this module is not complete until all image processing cycles have completed.

Technical notes:

This module will create a projection of all images specified in **LoadImages**. Previously, the module **LoadImageDirectory** could be used for the same functionality, but on a per-folder basis; i.e., a projection would be created for each set of images in a folder, for all input folders. The functionality of **LoadImageDirectory** can be achieved using image grouping with metadata, with the following setting specifications in **LoadImages**:

1. Specify that all subfolders under the Default input folder are to be analyzed.
2. Extract metadata from the input image path by using a regular expression to capture the subfolder name.
3. Enable grouping of image sets by metadata and specify the subfolder metadata token as the field by which to group.

However, unlike **LoadImageDirectory**, this per-folder projection is also not immediately available in subsequent modules until all image processing cycles for the given subfolder have completed.

See also **LoadImages**.

Settings:

Select the input image

What did you call the images to be made into a projection?

Type of projection

What kind of projection would you like to make? The final 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.

Name the output image

What do you want to call the projected image?

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

Which image do you want to mask?

Name the output image

What do you want to call the 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).
- *Image*: 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 .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)

Which objects would you like to use to mask the input image?

Select image for mask

(Used only if mask is to be made from an image)

Which image would you like to use to mask the input image?

Invert the mask?

This option reverses the foreground/background relationship of the mask.

- If unchecked, the mask will be composed of the foreground (white portion) of the masking image or the area within the masking objects.
- If checked, the mask will instead be composed of the *background* (black portions) of the masking image or the area *outside* the masking objects.

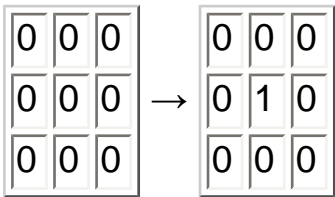
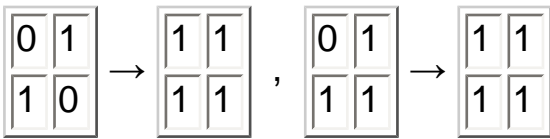
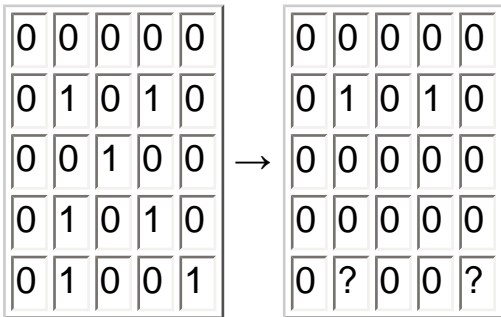
Morph

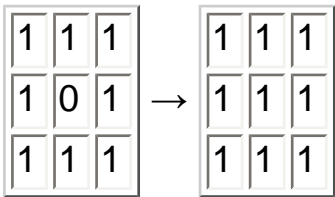
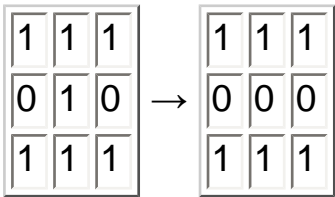
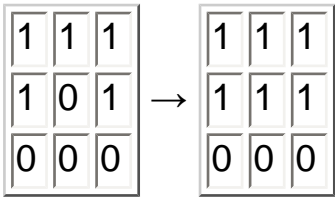
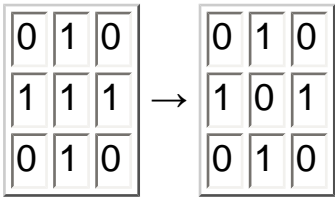
Morph performs low-level morphological operations on binary or grayscale images

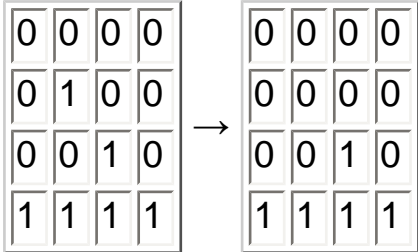
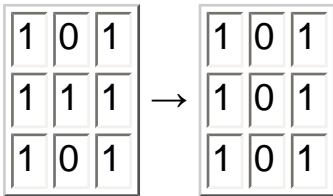
This module performs a series of morphological operations on a binary image or grayscale image, resulting in an image of the same type. Many require some image processing knowledge to understand how best to use these morphological filters in order to achieve the desired result. Note that the algorithms 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																																																		
<i>Bothat</i>	Bottom-hat filter: A bottom-hat filter enhances black spots in a white background. It subtracts the morphological <i>Close</i> of the image from the image. See below for a description of <i>Close</i> .	Binary, grayscale																																																		
<i>Branchpoints</i>	<p>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.</p> <div><table><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table><p>→</p><table><tr><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>?</td><td>0</td><td>0</td><td>0</td><td>?</td></tr></table></div>	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	?	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	?	0	0	0	?	Binary
1	0	0	0	0																																																
0	1	0	0	0																																																
0	0	1	0	0																																																
0	1	0	1	0																																																
1	0	0	0	1																																																
?	0	0	0	0																																																
0	0	0	0	0																																																
0	0	1	0	0																																																
0	0	0	0	0																																																
?	0	0	0	?																																																
<i>Bridge</i>	<p>Sets a pixel to 1 if it has two non-zero neighbors that are on opposite sides of this pixel:</p> <div><table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table><p>→</p><table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table></div>	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	Binary																																
1	0	0																																																		
0	0	0																																																		
0	0	1																																																		
1	0	0																																																		
0	1	0																																																		
0	0	1																																																		

<i>Clean</i>	<p>Removes isolated pixels:</p> 	Binary
<i>Close</i>	Performs a dilation followed by an erosion. The effect is to fill holes and join nearby objects.	Binary, grayscale
<i>Convex hull</i>	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
<i>Diag</i>	<p>Fills in pixels whose neighbors are diagonally connected to 4-connect pixels that are 8-connected:</p> 	Binary
<i>Dilate</i>	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
<i>Distance</i>	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
<i>Erode</i>	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
<i>Endpoints</i>	<p>Removes all pixels except the ones that are at the end of a skeleton:</p> 	Binary

<i>Fill</i>	<p>Sets a pixel to 1 if all of its neighbors are 1:</p> 	Binary
<i>Fill small holes</i>	<p>Sets background pixels surrounded by foreground pixels to 1.</p> <p>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.</p>	Binary
<i>Hbreak</i>	<p>Removes pixels that form vertical bridges between horizontal lines:</p> 	Binary
<i>Invert</i>	<p>For a binary image, transforms background to foreground and vice-versa. For a grayscale image, invert its intensity.</p>	Binary, Grayscale
<i>Majority</i>	<p>Each pixel takes on the value of the majority that surround it (keep pixel value to break ties):</p> 	Binary
<i>Life</i>	<p>Applies the interaction rules from the Game of Life, an example of a cellular automaton.</p>	Binary
<i>Open</i>	<p>Performs an erosion followed by a dilation. The effect is to break bridges between objects and remove single pixels.</p>	Binary, grayscale
<i>Remove</i>	<p>Removes pixels that are otherwise surrounded by others (4 connected). The effect is to leave the perimeter of a solid object:</p> 	Binary

<i>Shrink</i>	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 shrunk indefinitely.	Binary
<i>Skel</i>	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
<i>Spur</i>	Removes spur pixels. These are pixels that are connected only diagonally to other pixels and connected in only one direction: 	Binary
<i>Thicken</i>	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
<i>Thin</i>	Thin lines preserving the Euler number using the thinning algorithm # 1 described in Guo, "Parallel Thinning with Two Subiteration Algorithms", <i>Communications of the ACM</i> , Vol 32 #3, page 359. The result generally preserves the lines in an image while eroding their thickness.	Binary
<i>Tophat</i>	Subtracts the morphological opening of the image from the image. This enhances white spots in a black background.	Binary, grayscale
<i>Vbreak</i>	Removes pixels that form horizontal bridges between vertical lines: 	Binary

Settings:

Select the input image

What image do you want to morph? This is the input image to the module. 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

What do you want to call the resulting image? The output of the module. It will be of the same type as the input image.

Select the operation to perform

What operation do you want to perform? Choose one of the operations described in this module's help.

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.

Custom # of repeats

(Used only if Custom selected)

Enter the number of times to repeat the operation

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?

If you check this setting, the module will produce an image of the outlines on a black background. If the setting is unchecked, 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)

On which image would you like to display the outlines? 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

What do you want to call the image with the outlines displayed? This will be the name of the overlay image, which you can select in later modules (for instance, **SaveImages**).

Select 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)

Would you like the intensity (brightness) of the outlines to be the same as the brightest point in the image, or 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.

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.

Rescaling is especially helpful when converting 12-bit images saved in 16-bit format to the correct range.

Settings:

Select the input image

What did you call the image to be rescaled?

Name the output image

What do you want to call the rescaled image?

Select rescaling method

There are nine 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

CorrectIllumination_Apply.

- *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 **LoadData** 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.

How do you want to calculate the minimum intensity?

(Used only if specific values are to be chosen for a custom range)

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.

How do you want to calculate the maximum intensity?

(Used only if specific values are to be chosen for a custom range)

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.

Select method for rescaling pixels below the lower limit

(Used only if specific values are to be chosen for a custom range)

There are four 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.

Enter custom value for pixels below lower limit

(Used only if a custom rescaling range and a custom value for pixels below the lower limit is selected)

What custom value should be assigned to pixels with values below the lower limit?

Select method for rescaling pixels above the upper limit

(Used only if specific values are to be chosen for a custom range)

How do you want to handle values that are greater than the upper limit of the intensity range? Options are described in the help for the equivalent lower limit question.

Enter custom value for pixels below upper limit

(Used only if a custom rescaling range and a custom value for pixels below the upper limit is selected)

What custom value should be assigned to pixels with values above the upper limit?

Select image to match in maximum intensity

(Used only if the maximum for an image is to be matched to another image)

What did you call the image whose maximum you want the rescaled image to match?

Enter the divisor

(Used only if the same value is used as a divisor)

What value should be used as the divisor for the final image?

Select the measurement to use as a divisor

(Used only if the previously calculated value is used as a divisor)

What measurement value should be used as the divisor for the final image?

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

What did you call the image to be resized?

Name the output image

What do you want to call the resized image?

Select resizing method

How do you want to resize the image?

- *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, in pixels

(Used only if resizing by specifying desired final dimensions)

Enter the desired width of the final image.

Height of the final image, in pixels

(Used only if resizing by specifying desired final dimensions)

Enter the desired height of the final image.

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.

RunImageJ

RunImageJ runs an ImageJ command.

ImageJ is an image processing and analysis program (<http://rsbweb.nih.gov/ij/>). 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, then retrieves images you want to process further in CellProfiler.

Settings:

Command or macro?

This setting determines whether **RunImageJ** runs a command, selected from the list of available commands, or a macro that you write yourself.

Command:

The command to execute when the module runs.

Macro:

This is the ImageJ macro to be executed. For help on writing macros, see <http://rsb.info.nih.gov/ij/developer/macro/macros.html>

Options:

Use this setting to provide options to the command or macro.

Set the current image?

Check this setting if you want to set the current ImageJ image using an image from a previous

module. Leave it unchecked to use ImageJ's current image.

Current image:

This is the image that will become ImageJ's current image. ImageJ commands and macros will perform their operations on this image. Choose an image produced by a previous module.

Get the current image?

Check this setting if you want to retrieve ImageJ's current image after running the command or macro. Leave the setting unchecked if the pipeline does not need to access the current ImageJ image.

Final image:

This is the 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.

Wait for ImageJ?

Some ImageJ commands and macros are interactive; you may want to adjust the image in ImageJ before continuing. Check this box to stop CellProfiler while you adjust the image in ImageJ. Leave the box unchecked to immediately use the image.

This command will not wait if CellProfiler is executed in batch mode.

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, $Ax^2 + By^2 + Cx*y + Dx + Ey + F$. 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, Median, Smooth Keeping Edges or Circular Average Filter is selected)
If this box is checked, the module will 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.

Typical artifact diameter, in pixels

(Used only if choosing the artifact diameter automatically is unchecked)
Enter the approximate diameter 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 easily in an open image, use *Tools > Show pixel data*. Clicking and dragging the mouse will yield a *Length* measurement in the bottom bar of the figure window. 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)
What are the differences in intensity in the edges that you want to preserve? Enter the intensity step that indicates an edge in an image. 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 *Tools > Show pixel data*. When you move your mouse over the image, the pixel intensities will appear in the bottom bar of the figure window.

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](#)

This module replaces the functionality of the obsolete module **PlaceAdjacent**.

Settings:

Select an input image

What did you call the image to be tiled? Additional images within the cycle can be added later by choosing the *Across cycles* option.

Name the output image

What do you want to call the final tiled image?

Tile within cycles or across cycles?

How would you like to tile images? Two options are available:

- *Tile 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. This option takes the place of the obsolete **PlaceAdjacent** module.
- *Tile 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.

Number of rows in final tiled image

How many 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.

Number of columns in final tiled image

How many columns would 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.

Begin tiling in which corner of the final image?

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.

Begin tiling across a row, or down a column?

Are the images arranged in rows or columns? If your images are named A01, A02, etc, enter *row*".

Tile in meander mode?

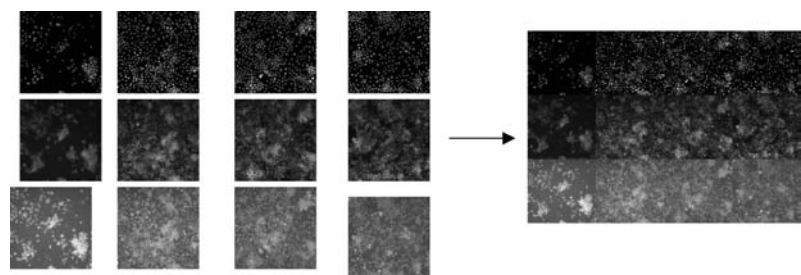
Meander mode tiles adjacent images in one direction, then the next row/column is tiled in the opposite direction. Some microscopes capture images in this fashion. The default mode is "comb", or "typewriter" mode; in this mode, when one row is completely tiled in one direction, the next row starts near 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. Check this box to create a grid that has the number of columns that you entered and enough rows to display all of your images. 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. Check this box to create a grid that has the number of rows that you entered and enough columns to display all of your images. If you check both automatic rows and automatic columns, **Tile** will create a grid that has roughly the same number of rows and columns.



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.

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 $\text{covariance}(I,J)/[\text{std}(I) * \text{std}(J)]$.
- *Slope*: The slope of the least-squares regression between a pair of images I and J. Calculated using the model $A*I + B = J$, where A is the slope.

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.

Settings:

Select an image to measure

What is the name of an image to be measured?

Select where to measure correlation

Do you want to measure the correlation over the whole image, within objects, or both? Both methods measure correlation on a pixel by pixel basis. Selecting *Objects* will measure correlation only in those pixels previously identified as an object (you will be asked to specify which object). Selecting *Images* will measure correlation across all pixels in the images. *Images and objects* will calculate both measurements.

Select an object to measure

What is the name of objects to be measured?

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 of the objects defined by one of the **Identify** modules. Both the area occupied and the total image area will respect the masking, if any, of the primary image used by the Identify module.

You can use this module to measure the number of pixels above a given threshold if you precede it with thresholding performed by **IdentifyPrimaryObjects**.

Available measurements

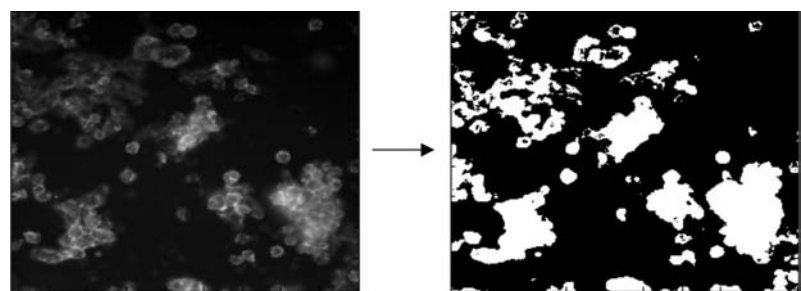
- *AreaOccupied*: The total area occupied by the input objects.
- *TotalImageArea*: The total pixel area of the image.

See also **IdentifyPrimaryObjects**, **IdentifySecondaryObjects**, **IdentifyTertiaryObjects**

Settings:

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.



MeasureImageGranularity

Measure Image 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. Image Analysis and Mathematical Morphology, Vol. 1. Academic Press, London, 1989
- 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. Granulometries and Opening Trees, *Fundamenta Informaticae*, 41, No. 1-2, pp. 57-90, IOS Press, 2000.
- Vincent L. Morphological Area Opening and Closing for Grayscale Images, *Proc. NATO Shape in Picture Workshop*, Driebergen, The Netherlands, pp. 197-208, 1992.
- Ravkin I, Temov V. Bit representation techniques and image processing, *Applied Informatics*, v.14, pp. 41-90, Finances and Statistics, Moskow, 1988 (in Russian)

Settings:

Select an image to measure

What did you call the 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.

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.

Available measurements

- *TotalIntensity*: Sum of all pixel intensity values.
- *MeanIntensity*, *MedianIntensity*: Mean and median of pixel intensity values.
- *MinIntensity*, *MaxIntensity*: Minimum and maximum of pixel intensity values.
- *TotalArea*: Number of pixels measured.

Settings:

Select the image to measure

What did you call the images whose intensity you want 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?

Check this option to restrict the pixels being measured to those within the boundaries of an object.

Select the input objects

(Used only when measuring intensity from area enclosed by objects)

What is the name of the objects to use? The intensity measurement will be restricted to within these objects.

MeasureImageQuality

Measure Image Quality measures features that indicate image quality, including measurements of blur (poor focus) and the percentage of pixels in the image that are minimal and maximal (i.e., saturated)

Available measurements

- *PercentMaximal*: Percent of pixels at the maximum intensity value of the image
- *PercentMinimal*: Percent of pixels at the minimum intensity value of the image
- *FocusScore*: A measure of the intensity variance across image
- *LocalFocusScore*: A measure of the intensity variance between image parts
- *Threshold*: The automatically calculated threshold for each image for the thresholding method of choice.
- *MagnitudeLogLogSlope*, *PowerLogLogSlope*: The slope of the log-log magnitude and power spectra.

Example Output:

Percent of pixels that are at the Maximal Intensity:	RescaledOrig:	0.0002763
Percent of pixels that are at the Minimal Intensity:	RescaledOrig:	0.0000352
Focus Score:	RescaledOrig:	0.016144
Suggested Threshold:	Orig:	0.0022854
Magnitude Spectrum Slope:	RescaledOrig:	-2.2
Power Spectrum Slope:	RescaledOrig:	-2.9

Settings:

Select an image to measure

What did you call the grayscale images whose quality you want to measure?

Check for blur?

Would you like to check for blur? If so, the module will calculate a focus score for each image, indicating how blurry an image is (higher Focus Score = better focus = less blurry). This

calculation is slow, so it is optional. The score is calculated using the normalized variance. We used this algorithm because it was ranked best in this paper: Sun, Y., Duthaler, S., Nelson, B. "Autofocusing in Computer Microscopy: Selecting the optimal focus algorithm," *Microscopy Research and Technique* 65:139-149 (2004).

The calculation of the focus score is as follows:

```
[m,n] = size(Image);  
MeanImageValue = mean(Image(:));  
SquaredNormalizedImage = (Image-MeanImageValue).^2;  
FocusScore{ImageNumber} = ...  
sum(SquaredNormalizedImage(:))/(m*n*MeanImageValue);
```

The above score is designed to determine which image of a particular field of view shows the best focus (assuming the overall intensity and the number of objects in the image is constant); it is not necessarily intended to compare images of different fields of view, although it may be useful for this to some degree. The Local Focus Score is a local version of the Focus Score, which is potentially more useful for comparing focus between images of different fields of view. However, like the Focus Score, the measurement should be used with caution because it may fail to correspond well to the blurriness of images of different fields of view, depending on the conditions.

Window size for blur measurements

(Used only if blur measurements are to be calculated)

The Local Focus Score is measured within an NxN pixel window applied to the image. What value of N would you like to use? We suggest twice the typical object diameter. You can measure the local focus score over multiple window sizes by adding an image to the list more than once and by setting different window sizes for each image.

Check for saturation?

Would you like to check for saturation (maximal and minimal percentages)? The percentage of pixels at the upper or lower limit of each individual image is calculated. 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 noise in images, this should typically 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 will increase.

Calculate threshold?

Would you like to automatically calculate a suggested threshold for each image? One indicator of image quality is that the automatically calculated suggested threshold is within a typical

range. Outlier images with high or low thresholds often contain artifacts.

Select a thresholding method

(Used only if thresholds are to be calculated)

This setting allows you to access automatic thresholding methods used in the **Identify** modules. 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.

Calculate quartiles and sum of radial power spectrum?

Would you like to calculate the quartiles and sum of the radial power spectrum? The Power Spectrum is computed via FFT and the radii of the first three quartiles and the total power are measured.

Two-class or three-class thresholding?

(Used only for the Otsu thresholding method)

Select *Two* if the grayscale levels are readily distinguishable into foreground (i.e., objects) and background. Select *Three* 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 intermediate 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 for the Otsu thresholding method with three-class thresholding)

Choose whether you want the middle grayscale intensities to be assigned to the foreground pixels or the background pixels.

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.

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?

Check this setting 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

Do you want the neuron graph relationship?

Check this setting to produce an edge file and a vertex file that give the relationships between trunks, branchpoints and vertices

Intensity image:

What is the name of 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 kind of vertex it is.
 - **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.

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.

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
- *UpperQuartileIntensity*: The intensity value of the pixel for which 75% of the pixels in the object have lower values.

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.

See also **MeasureImageIntensity**.

Settings:

Select an image to measure

What did you call the grayscale images whose intensity you want to measure?

Select objects to measure

What did you call the objects whose intensities you want to measure?

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

- *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
- *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.

You can retain the image of objects colored by numbers of neighbors or colored by the percentage of pixels that are touching other objects. CellProfiler creates a color image using the color map you choose. Use the **SaveImages** module to save the image to a file. See the settings help for further details on interpreting the output.

Settings:

Method to determine neighbors

How do you want 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)

Within what distance are objects considered neighbors (in pixels)? 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 for use later in the pipeline (for example, in SaveImages)?

An output image showing the input objects colored by numbers of neighbors may be saved. 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.

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)

What colormap do you want to use to color the above image? All available colormaps can be seen [here](#).

Retain the image of objects colored by percent of touching pixels for use later in the pipeline (for example, in SaveImages)?

An output image may be saved of the 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.

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)

What colormap do you want to use to color the above image? All available colormaps can be seen [here](#).

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.

The distribution is measured from the center of the object, where the center is defined as the point farthest from any edge. Alternatively, if primary objects exist within the object of interest (e.g. nuclei within cells), you can choose the center of the 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.

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

What did you call the images you want to process?

Select objects to meaasure

What did you call the objects you want to measure?

Object to use as center?

There are two ways to specify the center of the radial measurement:

- *These objects*: Use the centers of these objects for the radial measurement.
- *Other objects*: Use the centers of other objects for the radial measurement.

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 (*Other objects*).

Select objects to use as centers

(Used only if "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).

Number of bins

How many bins do 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.

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**.

Available measurements

- 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 $4 \cdot \pi \cdot \text{Area} / \text{Perimeter}^2$. Equals 1 for a perfectly circular object.
- 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.) This property is supported only for 2D input label matrices.
- Solidity: The proportion of the pixels in the convex hull that are also in the region. Also known as *convexity*. Computed as $\text{Area} / \text{ConvexArea}$.
- 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.
- 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.
- 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.

See also **MeasureImageAreaOccupied**.

Settings:

Select objects to measure

What did you call the objects you want to measure?

Calculate the Zernike features?

Check this box 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.

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 specified, which may lead to image-object texture combinations that are unnecessary. 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. Fourteen features are then calculated for the image by performing mathematical operations on the co-occurrence matrix:
 - *H1*: Angular Second Moment
 - *H2*: Contrast
 - *H3*: Correlation
 - *H4*: Sum of Squares: Variation
 - *H5*: Inverse Difference Moment
 - *H6*: Sum Average
 - *H7*: Sum Variance
 - *H8*: Sum Entropy
 - *H9*: Entropy
 - *H10*: Difference Variance
 - *H11*: Difference Entropy

- *H12*: Information Measure of Correlation 1
 - *H13*: Information Measure of Correlation 2
- *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

MeasureTexture performs the following algorithm to compute a score at each scale using the Gabor filter:

- Divide the half-circle from 0 to 180 degrees by the number of desired angles. For instance, if the user chooses two angles, MeasureTexture uses 0 degrees and 90 degrees (horizontal and vertical) for the filter orientations. This is the Theta value from the reference paper.
- For each angle, compute the Gabor filter for each object in the image at two phases separated by 90 degrees 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 Theta.
- Save the maximum score over all Theta as the score at the desired scale.

References

- Haralick et al. (1973), "Textural Features for Image Classification," *IEEE Transaction on Systems Man, Cybernetics*, SMC-3(6):610-621.
- Gabor D. (1946). "Theory of communication," *Journal of the Institute of Electrical Engineers* 93:429-441.

Settings:

Select an image to measure

What did you call the grayscale images whose texture you want to measure?

Select objects to measure

What did you call the objects whose texture you want to measure? You can select *None* if you only want to measure the texture for the image overall.

Objects specified here will have their texture measured against *all* images specified above, which may lead to image-object combinations that are unnecessary. 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.

Number of angles to compute for Gabor

How many angles do you want to use for each Gabor texture measurement? The default value is 4 which detects bands in the horizontal, vertical and diagonal orientations.

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.

Available measurements

- Single measurement: Classification (true/false) of the Nth bin for the Mth measurement.
- Two measurement: Classification of the 1st measurement versus the 2nd measurement binned into bins above ("high") and below ("low") the cutoff.

See also **CalculateMath** and any of the modules in the **Measure** category.

Settings:

Should each classification decision be based on a single measurement or on the combination of a pair of measurements?

This setting controls how classifications are recorded:

- *Single measurements*: 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 threshold

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?

Check this box if you want to create a bin for objects whose values fall below the low threshold. Leave the box unchecked 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?

Check this box if you want to create a bin for objects whose values are above the high threshold. Leave the box unchecked 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?

This option lets you assign custom names to bins you have specified. If you leave this unchecked, the module will 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*.

Enter the object name

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 first measurement

Select 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.

Use custom names for the bins?

Check this if you want to specify the names of each bin measurement. If you leave the box unchecked, the module will 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

Name of the measurement for objects that fall below the threshold for both measurements.

Enter the low-high bin name

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

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

Name of the measurement for objects that are above the threshold for both measurements.

Enter the image name

Name that will be associated with the graph image. You can specify this name in a **SaveImages** module if you want to save the image.

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.

Settings:

Select the input objects

What did you call the objects you want to convert to an image?

Name the output image

What do you want to call the resulting image?

Select the color type

What colors should the resulting image use? Choose how you would like the objects to appear:

- *Color*: Allows you to choose a colormap that will produce jumbled colors for your objects.
- *Binary*: 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)

What do you want the colormap to be? This setting affects how the objects are colored. You can look up your default colormap under *File > Preferences*.

EditObjectsManually

Edit Objects Manually allows you to remove specific objects from each image by pointing and clicking

This module allows you to remove specific objects via a user interface where you point and click to select objects for removal. The module displays three images: the objects as originally identified, the objects that have not been removed, and the objects that have been removed.

If you click on an object in the "not removed" image, it moves to the "removed" image and will be removed. If you click on an object in the "removed" image, it moves to the "not removed" image and will not be removed. Clicking on an object in the original image toggles its "removed" state.

The pipeline pauses once per processed image when it reaches this module. You must press the *Continue* button to accept the selected objects and continue the pipeline.

Available measurements

Image features:

- *Count*: The number of edited objects in the image.

Object features:

- *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

What do you want to call the objects that remain after editing? These objects will be available for use by subsequent modules.

Retain outlines of the edited objects?

Check this box 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)

What do you want to call 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:

If you choose *Renumber*, this 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.

If you choose *Retain*, this module 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).

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 features:

- *Count:* Number of expanded/shrunken objects in the image.

Object features:

- *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

What did you call the objects you want to expand or shrink?

Name the output objects

What do you want to call the resulting objects?

Select the operation

What operation do 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?

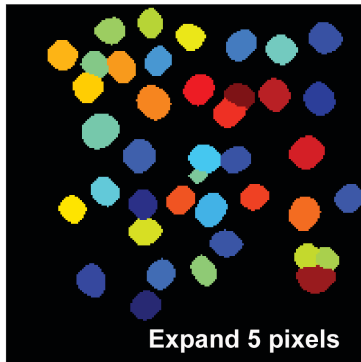
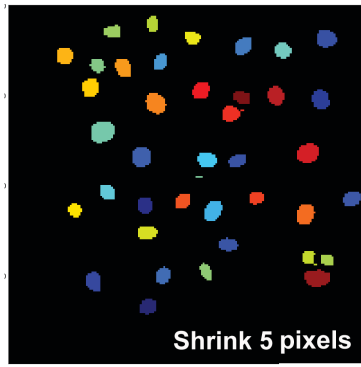
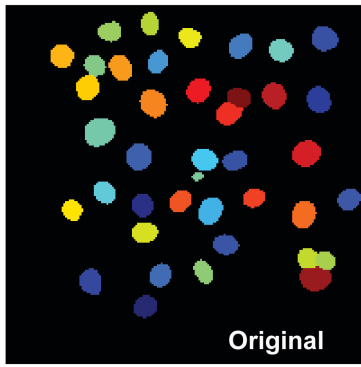
(Used only if one of the "shrink" options selected)

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 and it will erode an object with two holes to two rings connected by a line in order to keep from breaking up the object or breaking the hole. If you fill the holes in each object, then each object will shrink to a single point.

Name the outline image

(Used only if outlines are to be retained for later use in the pipeline)

Choose a name for the outlines of the identified objects that will allow them to be selected as an image later in the pipeline.



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.

Available measurements

Image features:

- *Count:* The number of objects remaining after filtering.

Object features:

- *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

What do you want to call the filtered objects? This will be the name for the collection of objects that are retained after applying the filter(s).

Select the object to filter

What object would you like 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 filter objects, select the first operand's object here, because **CalculateMath** measurements are stored with the first operand's object.

Filter using classifier rules or measurements?

You can choose either a measurement made on the objects or a rules file produced by CellProfiler Analyst. If you choose *Rules*, you will have to ensure that this pipeline makes every measurement in that rules file.

Select the filtering method

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.

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.

Name the outline image

(Used only if the outline image is to be retained for later use in the pipeline)

Choose a name by which the outline image can 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. Also, the identified objects themselves can be passed along to the object processing module **ConvertToImage** and then saved with the **SaveImages** module.

Rules file location

(Used only when filtering by 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 by rules)

The name of the file holding the 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 +.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

Select the measurement to filter by

See the **Measurements** modules help pages for more information on the features measured.

Filter using a minimum measurement value?

(Used only if Limits is selected for filtering method)

Check this box to filter the objects based on a minimum acceptable object measurement value. Objects which are greater than or equal to this value will be retained.

Filter using a maximum measurement value?

(Used only if Limits is selected for filtering method)

Check this box to filter the objects based on a maximum acceptable object measurement value. Objects which are less than or equal to this value will be retained.

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 features:

- *Count:* The number of objects identified.

Object features:

- *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

What do you want to call 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)

The automatic method uses the average diameter of previously identified guiding objects as the diameter. The manual method 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. You can use *Tools > Show Pixel Data* on an open image to measure distances in pixels.

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 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?

The module can create a binary image of the outlines of the objects it creates. You can then use **OverlayOutlines** to overlay the outlines on an image or use **SaveImages** to save them.

Name the outline image

(Used only if outlines are to be saved)

This setting names the outlines of the output objects. You can use this name to refer to the outlines in the **OverlayOutlines** and **SaveImages** modules.

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?

Check this setting to save the outlines around the objects as a binary image.

Name the outlines

(Used only if outlines are to be saved)

What do you want to call the outlines image? You can refer to this image in subsequent modules, such as **SaveImages**.

IdentifyPrimaryObjects

Identify Primary Objects identifies objects in an image

This module identifies primary objects in grayscale images containing bright objects on a dark background. Primary objects (e.g., nuclei) are those that can be found in an image without using any corresponding reference image. By contrast, secondary objects are those that are found using previously-identified primary objects as a reference.

After processing, the window for this module will show objects outlined in three colors:

- Green: Acceptable; passed all criteria
- Red: Discarded based on size
- Yellow: Discarded due to touching the border

The module window will also show another image where the identified objects are displayed with arbitrary colors: the colors themselves do not mean anything but simply help you distinguish the various objects. You can change the colormap in *File > Preferences*.

Steps to prepare images for this module:

- 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.

Available measurements

- *Image features:*
 - *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 features:*

- *Location_X, Location_Y*: The pixel (X,Y) coordinates of the center of mass of the identified primary objects.

Overview of the strategy

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 the primary objects to be identified are nuclei:

1. CellProfiler determines whether an object 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, and del Pozo F (1997). "Applying watershed algorithms to the segmentation of clustered nuclei." *Cytometry* 28, 289-297.
- Meyer F, and Beucher S (1990). "Morphological segmentation." *J Visual Communication and Image Representation* 1, 21-46.
- Ortiz de Solorzano C, Rodriguez EG, Jones A, Pinkel D, Gray JW, Sudar D, and Lockett SJ. (1999). "Segmentation of confocal microscope images of cell nuclei in thick tissue sections." *Journal of Microscopy-Oxford* 193, 212-226.
- Wahlby C (2003) *Algorithms for applied digital image cytometry*, Ph.D., Uppsala University, Uppsala.
- Wahlby C, Sintorn IM, Erlandsson F, Borgefors G, and 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.

Technical note

The initial step of identifying local maxima is performed on the user-controlled heavily smoothed image, the foreground/background is on a hard-coded slightly smoothed image, and the dividing lines between clumped objects (watershed) are done on the non-smoothed image.

Special note on saving images

Using the settings in this module, you can pass object outlines along to the module **OverlayOutlines** and then save them with the **SaveImages** module. You can also pass along the objects themselves to the object processing module **ConvertToImage** and then save them with the **SaveImages module**. This module produces several additional types of objects with names that are automatically passed along with the following naming structure:

- The unedited segmented image, which includes objects on the edge of the image and objects that are outside the size range, can be saved using the name "UneditedSegmented" + whatever you called the objects (e.g., "UneditedSegmentedNuclei").
- The segmented image which excludes objects smaller than your selected size range can be saved using the name "SmallRemovedSegmented" + whatever you called the objects (e.g., "SmallRemovedSegmentedNuclei").

See also **IdentifySecondaryObjects**, **IdentifyTertiaryObjects**, and **IdentifyPrimManual**.

Settings:

Select the input image

What did you call the images you want to use to identify objects?

Name the primary objects to be identified

What do you want to call the objects identified by this module?

Typical diameter of objects, in pixel units (Min,Max)

Most options within this module use this estimate of the size range of the objects in order to distinguish them from noise in the image. For example, for some of the identification methods, the smoothing applied to the image is based on the minimum size of the objects. The units 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 *Tools > Show pixel data*. Once this tool is activated, you can draw a line across objects in your image and the length of the line will be shown in pixel units. Note that 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?

You can choose to discard objects outside the range you specified. This allows you to exclude small objects (e.g., dust, noise, and debris) or large objects (e.g., large clumps) if desired. Objects discarded based on size are outlined in red in the module's display. See also the **FilterObjects** module to further discard objects based on some other measurement.

Try to merge too small objects with nearby larger objects?

Use caution when choosing Yes. Objects that are smaller than the specified minimum diameter will 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?

You can choose to discard objects that touch the border of the image. This is useful in cases when you do not want to make measurements of objects that are not fully within the field of view (because, for example, the morphological measurements 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, objects that touch the border of the cropped region will be discarded with this setting checked.

Select the thresholding method

The intensity threshold affects the decision of whether each pixel will be considered foreground (regions of interest) or background. A stringent threshold will result in only bright regions being identified, with tight lines around them, whereas a lenient threshold will include dim regions and the lines between regions and background will be more loose. You can have the threshold automatically calculated using several methods, or you can enter an absolute number between 0 and 1 for the threshold (to see the pixel intensities for your images in the appropriate range of 0 to 1, use *Tools > Show pixel data* in a window showing your image). Both options have advantages. An absolute number treats every image identically, but is not robust with regard to slight changes in lighting/staining conditions between images. An automatically calculated threshold adapts to changes in lighting/staining conditions between images and is usually more robust/accurate, but it can occasionally produce a poor threshold for unusual/artifactual

images. It also takes a small amount of time to calculate.

The threshold that is used for each image is recorded as a measurement in the output file, 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.

There are six methods for finding thresholds automatically:

- *Otsu*: This method is probably best if you are not able to make certain assumptions about every images in your experiment, especially if the percentage of the image covered by regions of interest varies substantially from image to image. Our implementation takes into account the maximum and minimum values in the image and log-transforming the image prior to calculating the threshold. 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%.
- *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. 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 is simple and appropriate for images in which most of the image is background. It 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). Note that the mode is protected from a high number of saturated pixels by counting only pixels < 0.95 . This can be very helpful, for example, if your images vary in overall brightness but the objects of interest are always twice (or actually, any constant) as bright as the background of the image.
- *Robust background*: This method trims the brightest and dimmest 5% of pixel intensities in the hopes that the remaining pixels represent a Gaussian of intensity values that are mostly background pixels. It then calculates the mean and standard deviation of the remaining pixels and calculates the threshold as the mean + 2 times the standard deviation.
- *Ridler-Calvard*: This method is simple and its results are often very similar to Otsu's.

According to Sezgin and Sankur's paper (*Journal of Electronic Imaging*, 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). Ridler-Calvard 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, repeating this until the threshold converges.

- *Kapur*: This method computes the threshold of an image by log-transforming its values, then searching for the threshold that maximizes the sum of entropies of the foreground and background pixel values, when treated as separate distributions.

You can also choose between *Global*, *Adaptive*, and *Per-object* thresholding for the automatic methods:

- *Global*: One threshold is calculated for the entire image (fast)
- *Adaptive*: The calculated threshold varies across the image. This method is a bit slower but may be more accurate near edges of regions of interest, or where illumination variation is significant (though in the latter case, using the **CorrectIllumination** modules is preferable).
- *Per-object*: If you are using this module to find child objects located *within* parent objects, the per-object method will calculate a distinct threshold for each parent object. This is especially helpful, for example, when the background brightness varies substantially among the parent objects.

Important: the per-object method requires that you run an **IdentifyPrimaryObjects** module to identify the parent objects upstream in the pipeline. After the parent objects are identified in the pipeline, you must then also run a **Crop** module with the following inputs:

- The input image is the image containing the sub-objects to be identified.
- Select *Objects* as the shape to crop into.
- Select the parent objects (e.g., *Nuclei*) as the objects to use as a cropping mask.

Finally, in the **IdentifyPrimaryObjects** module, select the cropped image as input image.

Selecting *manual thresholding* allows you to enter a single value between 0 and 1 as the threshold value. This setting can be useful when you are certain what the cutoff should be and it does not vary from image to image in the experiment. If you are using this module to find objects in an image that is already binary (where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

Selecting thresholding via a *binary image* will use the binary image as a mask for the input image. Note that unlike **MaskImage**, the binary image will not be stored permanently as a mask. Also, 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.

Selecting thresholding via *measurement* will use an image measurement previously calculated in order to threshold the image. Like manual thresholding, this setting 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 a Measurement module.

Select binary image

(Used only if Binary image selected for thresholding method)

What is the binary thresholding image?

Manual threshold

(Used only if Manual selected for thresholding method)

Enter the value that will act as an absolute threshold for the images, in the range of [0,1].

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)

Select *Two* if the grayscale levels are readily distinguishable into only two classes: foreground (i.e., objects) and background. Select *Three* if the grayscale levels fall instead into three classes. 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.

For example, 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.

Automatically calculate the size of objects for the Laplacian of Gaussian filter?

(Used only when applying the LoG thresholding method)

Check this box to use the filtering diameter range above when constructing the LoG filter.

Uncheck the box in order to enter a size that is not related to the filtering 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.

Threshold correction factor

When the threshold is calculated automatically, it may consistently be too stringent or too lenient. You may need to enter an adjustment factor that you empirically determine is suitable for your images. The number 1 means no adjustment, 0 to 1 makes the threshold more lenient and greater than 1 (e.g., 1.3) makes the threshold more stringent. 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, in the range [0,1]. This is helpful as a safety precaution when the threshold is calculated automatically. For example, if there are no objects in the field of view, the automatic threshold might be calculated as unreasonably low. In such cases, the lower bound you enter here will override the automatic threshold.

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 the appropriate number of 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 one peak of brightness per object (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. The module is more successful when the objects have a smooth texture. By default, the image is automatically blurred to attempt to achieve appropriate smoothness (see *Blur* option), but overriding the default value can improve the outcome on lumpy-textured objects. Technical description: Object centers are defined as local intensity maxima in the smoothed image.
- *Shape*: For cases when there are definite indentations separating objects. This works best for objects that are round. The intensity patterns in the original image are largely irrelevant: the image is converted to black and white (binary) and the shape determines whether clumped objects will be distinguished. Therefore, the cells need not be brighter towards the interior as is required for the *Intensity* option. The declumping results of this method are affected by the thresholding method you choose. Technical description: 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 (LoG)*: For objects that have an increasing intensity gradient toward their center, this option performs an LoG (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 next question.

Method to draw dividing lines between clumped objects

This setting allows you to choose the method that is used to draw the line between 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.

- *Distance*: 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.
- *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?

(Used only when distinguishing between clumped objects)

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. The size of the smoothing filter is automatically calculated based on the specified minimum object diameter that you have entered. 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), the *Size of smoothing filter* 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)

This setting, along with the *Size of the 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 object diameter that you have entered, 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)

Note that if you have entered a minimum object diameter of 10 or less, setting this option to Yes will have no effect.

Name the outline image

(Used only if outlines are to be saved)

You can use the outlines of the identified objects in modules downstream, by selecting them from any drop-down image list.

Fill holes in identified objects?

Checking this box will cause holes interior to identified objects to be filled.

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.

IdentifySecondaryObjects

Identify Secondary Objects identifies objects (e.g., cell edges) using "seed" objects identified by an **IdentifyPrimaryObjects** module (e.g., nuclei)

This module identifies secondary objects (e.g., cell edges) based on two inputs:

1. A previous module's identification of primary objects (e.g., nuclei)
2. An image stained for the secondary objects (not required for the *Distance - N* option).

Each primary object is assumed to be completely contained within a secondary object (e.g., nuclei are completely contained within cells stained for actin).

In order to identify the edges of secondary objects, this module performs two tasks:

1. Finding the dividing lines between secondary objects which touch each other.
2. Finding the dividing lines between the secondary objects and the background of the image. This is done by thresholding the image stained for secondary objects, except when using the *Distance - N* option.

Technical notes:

The *Propagation* algorithm labels from LABELS_IN to LABELS_OUT, steered by IMAGE and limited to MASK. MASK should be a logical array. λ is a regularization parameter, larger being closer to Euclidean distance in the image plane, and zero being entirely controlled by IMAGE. Propagation of labels is by shortest path to a nonzero label in LABELS_IN. Distance is the sum of absolute differences in the image in a 3x3 neighborhood, combined with λ via $\sqrt{\text{differences}^2 + \lambda^2}$. Note that there is no separation between adjacent areas with different labels (as there would be using, e.g., watershed). Such boundaries must be added in a postprocess.

Available measurements

- *Image features:*
 - *Count*: The number of secondary objects identified.
- *Object features:*
 - *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.

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.

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 (TR Jones, AE Carpenter, P Golland (2005) *Voronoi-Based Segmentation of Cells on Image Manifolds*, ICCV Workshop on Computer Vision for Biomedical Image Applications, pp. 535-543).
- *Watershed*: This method will find dividing lines between objects by looking for dim lines between objects (Vincent, Luc and Pierre Soille, *Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations*, IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 13, No. 6, June 1991, pp. 583-598).
- *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.

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.

Select the thresholding method

The intensity threshold affects the decision of whether each pixel will be considered foreground (regions of interest) or background. A stringent threshold will result in only bright regions being identified, with tight lines around them, whereas a lenient threshold will include dim regions and the lines between regions and background will be more loose. You can have the threshold automatically calculated using several methods, or you can enter an absolute number between 0 and 1 for the threshold (to see the pixel intensities for your images in the appropriate range of 0 to 1, use *Tools > Show pixel data* in a window showing your image). Both options have advantages. An absolute number treats every image identically, but is not robust with regard to slight changes in lighting/staining conditions between images. An automatically calculated threshold adapts to changes in lighting/staining conditions between images and is usually more robust/accurate, but it can occasionally produce a poor threshold for unusual/artifactual images. It also takes a small amount of time to calculate.

The threshold that is used for each image is recorded as a measurement in the output file, 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.

There are six methods for finding thresholds automatically:

- *Otsu*: This method is probably best if you are not able to make certain assumptions about every images in your experiment, especially if the percentage of the image covered by regions of interest varies substantially from image to image. Our implementation takes into account the maximum and minimum values in the image and log-transforming the image prior to calculating the threshold. 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%.
- *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. 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 is simple and appropriate for images in which most of the image is background. It 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). Note that the mode is protected from a high number of saturated pixels by counting only pixels < 0.95 . This can be very helpful, for example, if your images vary in overall brightness but the objects of interest are always twice (or actually, any constant) as bright as the background of the image.
 - *Robust background*: This method trims the brightest and dimmest 5% of pixel intensities in the hopes that the remaining pixels represent a Gaussian of intensity values that are mostly background pixels. It then calculates the mean and standard deviation of the remaining pixels and calculates the threshold as the mean + 2 times the standard deviation.
 - *Ridler-Calvard*: This method is simple and its results are often very similar to Otsu's. According to Sezgin and Sankur's paper (*Journal of Electronic Imaging*, 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). Ridler-Calvard 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, repeating this until the threshold converges.
 - *Kapur*: This method computes the threshold of an image by log-transforming its values, then searching for the threshold that maximizes the sum of entropies of the foreground and background pixel values, when treated as separate distributions.

You can also choose between *Global*, *Adaptive*, and *Per-object* thresholding for the automatic methods:

- *Global*: One threshold is calculated for the entire image (fast)
 - *Adaptive*: The calculated threshold varies across the image. This method is a bit slower but may be more accurate near edges of regions of interest, or where illumination variation is significant (though in the latter case, using the **CorrectIllumination** modules is preferable).
 - *Per-object*: If you are using this module to find child objects located *within* parent objects, the per-object method will calculate a distinct threshold for each parent object. This is especially helpful, for example, when the background brightness varies substantially among the parent objects.
- Important*: the per-object method requires that you run an **IdentifyPrimaryObjects**

module to identify the parent objects upstream in the pipeline. After the parent objects are identified in the pipeline, you must then also run a **Crop** module with the following inputs:

- The input image is the image containing the sub-objects to be identified.
- Select *Objects* as the shape to crop into.
- Select the parent objects (e.g., *Nuclei*) as the objects to use as a cropping mask.

Finally, in the **IdentifyPrimaryObjects** module, select the cropped image as input image.

Selecting *manual thresholding* allows you to enter a single value between 0 and 1 as the threshold value. This setting can be useful when you are certain what the cutoff should be and it does not vary from image to image in the experiment. If you are using this module to find objects in an image that is already binary (where the foreground is 1 and the background is 0), a manual value of 0.5 will identify the objects.

Selecting thresholding via a *binary image* will use the binary image as a mask for the input image. Note that unlike **MaskImage**, the binary image will not be stored permanently as a mask. Also, 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.

Selecting thresholding via *measurement* will use an image measurement previously calculated in order to threshold the image. Like manual thresholding, this setting 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 a Measurement module.

Threshold correction factor

When the threshold is calculated automatically, it may consistently be too stringent or too lenient. You may need to enter an adjustment factor that you empirically determine is suitable for your images. The number 1 means no adjustment, 0 to 1 makes the threshold more lenient and greater than 1 (e.g., 1.3) makes the threshold more stringent. 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, in the range [0,1]. This is helpful as a safety precaution when the threshold is calculated automatically. For example, if there are no objects in the field of view, the automatic threshold might be calculated as unreasonably low. In such cases, the lower bound you enter here will override the automatic threshold.

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.

Regularization factor

(Used only if Propagation method selected)

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 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 weight the distance between the two values more and more heavily. The regularization factor can be infinitely large, but around 10 or so the intensity image is almost completely ignored and the dividing line will simply be halfway between the two competing primary objects.

Name the outline image

Once the outline image is named here, the outlines of the identified objects may be used by modules downstream, by selecting them from any drop-down image list.

Manual threshold

(Used only if Manual selected for thresholding method)

Enter the value that will act as an absolute threshold for the images, in the range of [0,1].

Select binary image

(Used only if Binary image selected for thresholding method)

What is the binary thresholding image?

Two-class or three-class thresholding?

(Used only for the Otsu thresholding method)

Select *Two* if the grayscale levels are readily distinguishable into only two classes: foreground

(i.e., objects) and background. Select *Three* if the grayscale levels fall instead into three classes. 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.

For example, 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.

Discard secondary objects that touch the edge of the image?

This option will discard objects which have an edge that falls on the border of the image. 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 secondary objects touching the edge are discarded)

It might be appropriate to discard the primary object for any secondary object that touches the edge of the image. The module will create a new set of objects that mirrors your primary objects if you check this setting. The new objects will be identical to the old, except that the new objects will have objects removed if their associated secondary object touches the edge of the image.

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)

Check this setting in order to save images of the outlines of the primary objects after filtering. You can save these images using the **SaveImages** module.

Name the new primary object outlines

(Used only when saving outlines of new primary objects)

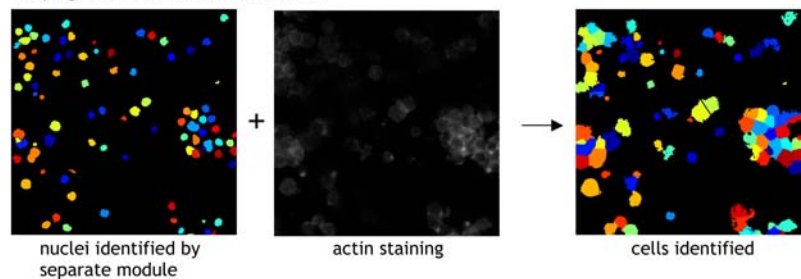
You can name the outline image of the primary objects after filtering. You can refer to this image using this name in subsequent modules such as **SaveImages**.

Select the measurement to threshold with

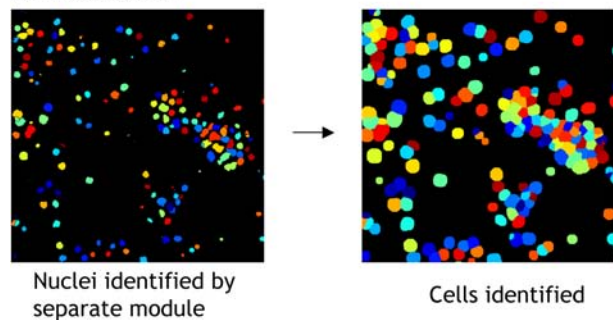
(Used only if Measurement is selected for thresholding method)

Choose the image measurement that will act as an absolute threshold for the images.

Propagate or Watershed method:



Distance method:



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

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 **Measure** modules. The larger objects should therefore be equal in size or larger than the smaller objects and must completely contain the smaller objects. Both inputs should be objects produced by **Identify** modules, not grayscale images.

Note: Creating subregions using this module can result in objects that are not contiguous, which does not cause problems when running the **MeasureImageIntensity** and **MeasureTexture** modules, but does cause problems when running the **MeasureObjectSizeShape** module because calculations of the perimeter, aspect ratio, solidity, etc. cannot be made for noncontiguous objects.

Special note on saving images: You can use the settings in this module to pass object outlines along object outlines can be passed 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 features:*
 - *Count:* The number of tertiary objects identified.
- *Object features:*
 - *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

What did you call the larger identified objects?

Select the smaller identified objects

What did you call the smaller identified objects?

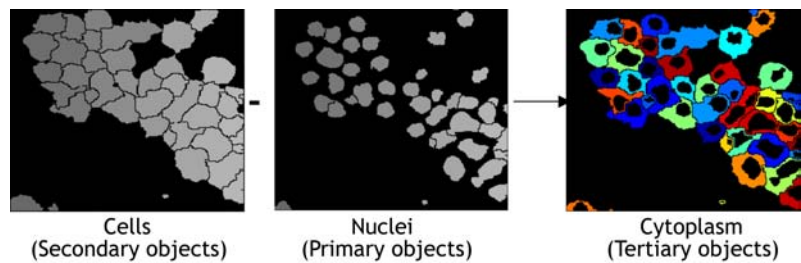
Name the tertiary objects to be identified

What do you want to call the new subregions? The new tertiary subregion will consist of the smaller object subtracted from the larger object.

Name the outline image

(Used only if outlines are to be retained for later use in the pipeline)

Enter a name that will allow the outlines to be selected later in the pipeline.



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.

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

What do you want to call the objects that remain after the masking operation? You can refer to the masked objects in subsequent modules using 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 or by defining a region based on the white regions in a binary image.

Select the masking object(s)

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. Images can be loaded from disk by **LoadImages** or **LoadData**. You can create a binary image from a grayscale image using **ApplyThreshold**.

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:

If you choose *Renumber*, **MaskObjects** 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; your object measurements for the masked objects will not have gaps (where removed objects are missing).

If you choose *Retain*, **MaskObjects** will retain the original numbers. 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?

You can save the outlines of the objects that are produced after masking, then display them over images using the **OverlayOutlines** module.

Name the outline image

(Used only if retaining outline image for later in the pipeline)

What do you want to call the outline image? Subsequent modules can refer to the binary outline image using this name.

Invert the mask?

This option reverses the foreground/background relationship of the mask.

- If unchecked, the mask will be composed of the foreground (white portion) of the masking image or the area within the masking objects.
- If checked, the mask will instead be composed of the *background* (black portions) of the masking image or the area *outside* the masking objects.

ReassignObjectNumbers

Reassign Object Numbers rennumbers objects

Objects in CellProfiler are tracked, and measurements of objects are associated with each other, based on object numbers (also known as object labels). Typically, each object is assigned a single unique number; exported measurements are ordered by this numbering. This module allows the reassignment of object numbers, which may be useful in certain cases. The *Unify* option assigns adjacent or nearby objects the same number 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. The *Split* option assigns unique numbers to portions of separate objects that previously had been using the same number, which might occur if you applied certain operations in the **Morph** module to objects.

Technically, reassignment means that the numerical value of every pixel consisting of an object (in the label matrix version of the image) is changed, according to the module's settings. In order to ensure that objects are numbered consecutively without gaps in the numbering (which other modules may depend on), **ReassignObjectNumbers** will typically result in most of the objects having their numbers reassigned. This reassignment information is stored and can be exported from CellProfiler like any other measurement: each original input object will have its reassigned object number stored as a feature 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

What do you want to call the objects whose numbers have been reassigned? You can use this name in subsequent modules that take objects as inputs.

Operation to perform

Choose *Unify* to assign adjacent or nearby objects the same object number. Choose *Split* to give a unique number to non-adjacent objects that currently share the same object number.

Maximum distance within which to unify objects

(Used only with the Unify option)

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)

Unify can 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 be 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).

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 object is touching two parent objects, the object will be assigned to the parent that shares the largest number of pixels with the child.

Available measurements

- *Parent object features:*
 - *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 features:*
 - *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 distances?

Do you want to calculate the distances of each child to its parent?

- The *minimum distance* is the distance from the centroid of the child object to the closest perimeter point on the parent object.
- The *centroid distance* is the distance from the centroid of the child object to the centroid of the parent.

Calculate per-parent means for all child measurements?

For every measurement that has been made of the children objects upstream in the pipeline, this module calculates the mean value of that measurement over all children and stores it as a measurement for the parent, as "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)

You can 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.

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 group the input using grouping options in **LoadImages** to make sure that each image sequence is handled individually. See the help in that module, and CellProfiler Help > General Help > Using MetaData in CellProfiler for more information. If you are only processing a single movie in each analysis run, you do not need to set up image grouping.

For an example pipeline using TrackObjects, see the CellProfiler [Examples](#) webpage.

Available measurements

- *Object features*
 - **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.
 - **Parent**: The label of the object in the last 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 distance traveled vector).
 - **IntegratedDistance**: The total distance traveled by the object during the lifetime of the object.
 - **Linearity**: A measure of how linear the object trajectory is during the object lifetime. Calculated as (distance from initial to final location)/(integrated object distance). Value is in range of [0,1].
 - **Lifetime**: The duration (in frames) of the object. The lifetime begins at the frame when an object appears and is output as a measurement when the object disappears. At the final frame of the image set/movie, the lifetimes of all remaining

objects are output.

- *Image features*

- **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.
- **DaughterObjectCount**: 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**, **LoadImages**.

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.
- *Measurement*: 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.

Reference:

- 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.

Select the objects to track

What did you call the objects you want to track?

Select object measurement to use for tracking

(Used only if Measurements is the tracking method)

What measurement do you want to use for tracking? Choose 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 using *Tools > Show pixel data* of any CellProfiler figure window.

Select display option

How do you want to display the tracked objects? The output image can be saved as either a color-labeled image, with each tracked object assigned a unique color, or as a color-labeled

image with the tracked object number superimposed.

Save color-coded image?

Do you want to save the image with tracked, color-coded objects? Specify a name to give the image showing the tracked objects. This image can be saved with a **SaveImages** module placed after this module.

Name the output image

(Used only if saving the color-coded image)

What do you want to call the color-coded image, which will be available for downstream modules, such as **SaveImages**?

Cost of being born

(Used only if the LAP tracking method is applied)

What is the cost of an object being born?

Cost of dying

(Used only if the LAP tracking method is applied)

What is the cost of an object dying?

Run the second phase of the LAP algorithm?

(Used only if the LAP tracking method is applied)

Check this box to run the second phase of the LAP algorithm after processing all images. Leave the box unchecked to omit the second phase or to perform the second phase when running as a data tool.

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.

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 score takes into account the area of the split object relative to the area of the resulting objects and the displacement of the resulting objects relative to the position of the original object and is roughly measured in pixels. The split alternative cost is (conceptually) subtracted from the cost of making the split.

The split cost should be set lower if objects are being split that should not be split. It 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 into account the area of the two objects to be merged relative to the area of the resulting objects and 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.

Set the merge alternative cost lower if objects are being merged when they should otherwise be kept separate. Set the cost higher if objects that are not merged should be merged.

Maximum gap displacement

(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. The measurement is roughly the maximum displacement of an object's center from frame to frame. The algorithm will run more slowly with a higher value. The algorithm will not consider objects that would otherwise be tracked between frames if set to a lower value.

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 and the distances between the original and resulting objects. The algorithm will run more slowly with a higher value. The algorithm will exclude objects that would otherwise be split if it is set to a lower value.

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 and the distances between the objects to be merged and the resulting object. The algorithm will run more slowly with a higher value. The algorithm will exclude objects that would otherwise be merged if it is set to a lower value.

Maximum gap

(Used only if the LAP tracking method is applied and the second phase is run)

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.

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 was saved. This option has been moved to the preferences settings (*File > Preferences*).

Settings:

Specify which images?

- Choose *Images to remove* to remove some images from memory and keep the rest.
- Choose *Images to keep* to keep some images and remove the rest.

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?

Check this option to display thumbnail images (small versions of the images) on the web page that link to the full images. Leave it unchecked to display the full image directly on the web page.

If you are going to use thumbnails, you will need to load them using **LoadImages** or **LoadData**; you can 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. Tags have the form "`\g<metadata-tag>`" where `<metadata-tag>` is the name of the previously defined metadata field. 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 using "`\g<Plate>_\g<Well>`" to specify the name. Please see **LoadImages**,

LoadData, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

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.

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. Tags have the form "`\g<metadata-tag>`" where `<metadata-tag>` is the name of the previously defined metadata field. For instance, if you have a metadata tag named "Plate", you can use the metadata substitutions "Plate: `\g<Plate>`", to display the plate metadata item. Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

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. If you select, *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. If you select *For each image*, the browser will open a new window each time you click on a link. If you select, *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. Check this box 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.

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). If you choose *Rows*, then well A01 will be assigned 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. Conversely, if you choose *Columns*, 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?

Would you like to define a new grid for each image cycle, or define a grid once and use it for all images?

- *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

Would you 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 mode*: 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 mode*: 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)

What are 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)

Do 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)

What image do 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 *Tools > Show pixel data* tool 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)

What is this cell's row number? 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 the *Tools > Show pixel data* tool 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)

What is this cell's row number? 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)

What is this cell's column number? 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 for use later in the pipeline (for example, in Savelmages)?

Do you want 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 **Savelmages** module. Check this box if you want to save the annotated image.

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, would you like to use a previous grid that worked? This setting allows you to control how the module responds to errors:

- *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.

The pipeline will stop in all cases if gridding fails on the first image.

InputExternal

InputExternal.py - Let the user create image names that will be pulled from external sources (eg: Java)

LabellImages

LabellImages assigns plate metadata to image sets.

LabellImages 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 **LoadImages**, using the file order instead of the text in the file name.

LabellImages assumes that you have an identical number of image sets per well (the *sites*), and an identical number of rows and columns of wells per plate with complete file lists for all but the last plate and well. Files have to be loaded in one of the two following orders to use **LabellImages**

By row

- All sites for a given well appear consecutively
- All wells for a given row (e.g. A01, A02, A03...) appear consecutively
- All rows for a given column appear consecutively
- All columns for a given plate appear consecutively

or

By column

- All sites for a given well appear consecutively
- All wells for a given column (e.g. A01, B01, C01...) appear consecutively
- All columns for a row column appear consecutively
- All rows for a given plate appear consecutively

LabellImages adds the following measurements to the image table:

Measurement	Description
Metadata_Plate	The plate number, starting at 1 for the first plate
Metadata_Well	The well name, for instance, "A01"
Metadata_Row	The row name, for instance, "A"
Metadata_Column	The column number
Metadata_Site	The site number within the well

Settings:

sites / well:

This setting controls the number of image sets for each well

of columns:

The number of columns per plate

of rows:

The number of rows per plate

Order:

This setting controls whether the data is ordered by row and then by column or by column and then by row. Choose, "Row", if data appears by row and then by column. Choose, "Column", if data appears by column and then by row. For instance, the ExampleSBSImages sample 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.

OutputExternal

OutputExternal.py - Let the user select which images they would like to make available to external sources (eg: Java)

PauseCellProfiler

Pause CellProfiler pauses CellProfiler during analysis

This module allows you to pause CellProfiler's processing at the point where the module resides in the pipeline, which can be helpful if you want to examine the results before proceeding.

Settings:

Pause here, skip subsequent modules or continue without prompting?

There are three options:

- *Pause* will pause CellProfiler at this module's position in the pipeline. The pipeline will stop and a window with a *Resume* button will appear. The pipeline will continue when you hit the *Resume* button or will stop if you hit the *Stop analysis* button on the main window.
- *Skip* will pause as described above, but if you choose to resume, CellProfiler will skip all modules following the **PauseCellProfiler** module and will advance to begin applying the first module in the pipeline to the next image cycle.
- *Continue* will continue pipeline execution without stopping. This enables you to temporarily run the full pipeline without the inconvenience of removing the **PauseCellProfiler** module from the pipeline.

SendEmail

SendEmail send emails to a specified address at desired stages of processing

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. Tags have the form "\g<metadata-tag>" where <metadata-tag> is the name of the previously defined metadata field. For instance, if you have plate metadata, you might use the line, "CellProfiler: processing plate \g<Plate>". Please see **LoadImages**, **LoadData**, or *Help > General help > Using metadata in CellProfiler* for more details on obtaining, extracting, and using metadata tags from your images.

SMTP server

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.

SMTP 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.

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".