For this tutorial, we will be building a pipeline to find individual worms in subregions and then extract measurements.

To begin, start CellProfiler 2.1. The CellProfiler interface consists of the pipeline panel to the left, which is divided into input modules and analysis modules. A space for module notes for annotations and a file list will appear to the right. To provide CellProfiler with images files to be analyzed, drag and drop the input image files into the file list space.

The input module metadata enables extraction of metadata information from the image file names and folders. Click on metadata and a list of input modules and the modules settings for the Metadata module will appear on the right.

Check the box Extract Metadata and a number of options will appear underneath. Select Extract from File/folder name as the extraction method in order to extract the well ids from the file names using regular expressions.

To learn more about regular expressions, click on the question mark next to this option in the module settings. Click “Add another extraction method” and then extract the plate and gene names from the folder names using regular expressions.

Verify your metadata extraction using the update button below the module settings to ensure that each input image from the images module is associated with the plate, gene, and well id metadata.

Next, select the input module names and types. Set the input image type to Color Images and assign the images the name RawData.

The group’s modules is not needed for this assay, so it can be left as its default setting. In order to load the worm model, you will need to adjust the output settings. Click on the View Output settings button in the button left of the CellProfiler interface and then adjust the default output folder to point to the folder where you want to store your output.

Once the input modules are all set, the analysis pipeline is built up by inserting modules into the analysis panel. Modules are added by clicking the plus sign next to Adjust Modules in the bottom left corner of the interface to bring up a list of selectable modules.

Most object detection methods in CellPrpofiler are designed for bright objects on a dark background. Since the input images are color images with a bright background, the first thing we need to do is invert the image pixel intensities. Adding the module called ImageMath located under the module category Imaging Processing.

Add the module by selecting and double clicking it. The module’s now been added to your analysis pipeline. When the module is selected, its settings are shown on the panel to right where they can be adjusted. For this module, select invert as the operation called the output image InvertedRaw and select RawData as the input.

Most images processing algorithms operate on greyscale data, so the next thing we need to do is convert the inverted color image into a greyscale image in two different ways. First, we add the module color to grey, and then combine the red, green, and blue channels of the inverted raw image into a single image set that we will call a OrigGrey. The image will be used to identify the worms.

To see the effect of a module, click on start test mode. It is now possible to step through the analysis pipeline and model the output at each step. Add another color to grey module and this time, split the red, green, and blue channels and called them OrigRed, OrigGreen, and OrigBlue. These images will be used for identification of bubbles and fatty regions.

As it turns out, bubbles have high contrast in the red image channel and appear brighter than the worms in the inverted image. The same is true for the edges of the wells visible in the corners of the image. We can detect both of these structures in the origRed image using the object identification module IdentifyPrimaryObjects. using a manual intensity threshold of .5, and allowing a size range from 10 pixels to 10,000 pixels.

Make sure not to discard objects touching the border of the image, as we do not want to lose clusters of worms were only one or a few worms are partly outside the field of view. In this case, we will not attempt to separate touching objects, and we will call the output Well_n_Bubbles.

We also want to remove shadows surrounding bubbles and well edges. to do this, we will add the module ExpandorShrinkObjects and select expand objects by specified number of pixels, and expend the Well_n_bubbles object by 5 pixels. We will call the output expanded Wells_n_bubbles.

Next, we use the mask image module to mask the origGray image with expanded_well_n_bubbles, to get a new image, MaskedGray, without artifacts. Make sure to invert the mass in order to keep worms and remove artifacts. We will now use IdentifyPrimaryObjects to separate worms and worm clusters from the image background.

With the MaskedGrey image as input, we set the size range from between 20 pixels and 30000 pixels in order to remove small debris and ensure that large worm clusters are kept.

We use automated thresholding with global robust background as the method. Since there's a risk for empty wells, we also set a minimum threshold level at 0.09. In this case, we will not attempt to separate touching objects. Uncheck the fill holes and identify objects setting. Finally, we will give the name WormObjects to the Worm Objects.

We can view the worm identification results by stepping through this module in test mode.

Add the ConvertObjectsToImageModule to convert the worm objects to a binary image, call the output image WormBinary.

Add the module UntangleWorms from the Worm Toolbox. Set it to Untangle, using the WormBinary as input and choose “both” as the overlapping style. This sets the module to work on overlapping mass when measuring size and shape, but non overlapping mass when measuring intensity. Click the box for retaining both kinds of outlines and set the training set file name to the DefaultWormModel.xml file.

Again, we can view the untangling results by stepping through the module in test mode.

As check the results use overlay outlines to overlay the worm outlines on the original RawData image. Select the two outlines and call the output image OrigOverlay. Now the worms have been identified in untangling, we want to begin to extract shape and intensity measurements.

We start by adding MeasureObjectSizeShape to the pipeline and measure from the NonOverlappingWorm objects. Be sure to deselect the Calculate the Zernike features since these measurements are not relevant to this context and add computational time.

Since the true width of the worms will be biased by the width of the worm model, we want to extract the actual average width from the WormObjects. We achieve this by using the Morph module in order to calculate the distance transform of the WormBinary image. Select distance as the operation to perform and call the output image WormWidthsFromBinary.

Add the measure ObjectIntensity module to measure worm intensities. Select OrigRed OrigBlue and WormsWidthFromBinary as the images to measure. Select NonOverlappingWorms as the objects to measure. The measurements on WormWidthsFromBinary will provide information on the worm widths while measurements on OrigRed and OrigBlue will provide information on the stained distribution and color.

The StraightenWorms module from the Worm Toolbox extracts intensity measurements from subregions of the worms after digitally straightening the worms and dividing them into a fixed number of transfer segments in longitudinal stripes. These measurements can provide information on where fat is located inside the worm.

In this module, we select “NonOverlappingWormsToStraightren” and specify to defaultwormmodel.xml as a training set. We then select a single transfer segment and five longitudinal stripes. Note that if a head marker is available, all worms could be automatically aligned. We select OrigBlue as the image for measurements.

Before identifying the fat subregions of the worm, we want to mask away all regions in the image that should not be considered as a worm. We do this by adding the module MaskImage.

Set OrigBlue as the input image and WormsBinary as the masking image. The OrigBlue image is selected to define fatty regions. Since the has its greatest contrast in the blue image channel. The output of this module is called MaskedBlue. Now we use identify primary objects to find the fatty regions.

This time, we let the diameter vary in size from 5 pixels to 10,000 pixels and specify a fixed intensity threshold for whats regarded as fat, set for .4 for this dataset. This fixed threshold was selected by trying a range of thresholds and viewing the result and should be adjusted for a new dataset. We do not attempt to separate clumped objects burt we do retain the outlines.

We will call the output objects FatObjects and the outlines FatOutlines.

To be able to see the detected fat outlines we will use the OverlayOutlines module to add them to the image to which we already outlined the worms, namely OrigOverlay. We do this by adding the module OverlayOutlines and selecting OrigOverlay as the image on which to display the outlines, selecting FatOutlines for the outlines and calling the output image OrigOverlayWithFat.

Run the pipeline in test mode to individually confirm the output.

Now extract the shape in intensity measurements from the FatObjects by adding a module MeasureObjectSizeShape. Uncheck the Run Zernike feature again.

We also add the module MeasureObjectIntensity and measure intensity features from OrigBlue and OrigRed images with FatObejcts as the selected Objects.

In order to assign each detected fatty region to a worm we add the module RelateObjects with FatObjects as the children and NonOverlappingWorms as the parents. We also check the calculate per-parent means for all children measurements box to aggregate the fast statistics for each worm.

Finally, we'll save all measurements segmentations, worm outlines, as well as the images with the overlayed outlines. At the module SaveImages and save the OrigOverlay with FatImage using the RawData file name as the prefix and _res as the appended suffix

Next, add another SaveImages module to save the worm outlines as they will be used for later in the machine learning step in CellProfilerAnalyst. Select the overlap Worm Outlines as the input, the RawData name for the prefixes above and _outlines as the suffix. Also, check the record file and path information to the saved image box.

Add a third SaveImages module to save the segmentation mass called OverlappingWorms and save them as objects with the file name extension _worm_objects and the format as tif which is the only format for saving overlapping objects. These will be used for input for pipeline number four.

Export all measurements to an SQLite database using the ExportToDatabaseModule. Choose SQLite as the database type and name the experiment MyExperiment. Check the box to create a CellProfiler Analyst properties file and choose NonOverlappingWorms as the objects used for location. The plate type should be 96.

Set the plate metadata to plate and a well metadata to well and specify that only selected object will be used for export and select the NonOverlappingWormObjects.

Your pipeline is now complete, so exit test mode by pressing the exit tested button. Since we've finished previewing the results, the display windows can be optionally closed to save time and memory during the analysis run. Select hide all windows on run from the window menu item to close all the display windows and have them remain closed during the analysis run. Then, press the Analyze Images button to run the pipeline on all images.

The output will be in an SQLite database with measurements, a properties file for data exploration in CellProfiler Analyst, a set of images showing outlines of worms in fatty regions on top of the raw data, a set of images without lines only, and a set of segmentation masks describing the overlapping worms.