CellProfiler Tutorial: pixel-based classification

By Kyle Karhohs, PhD

(A version of this document containing animated GIFs in maintained online on our GitHub)

Introducing Ilastik

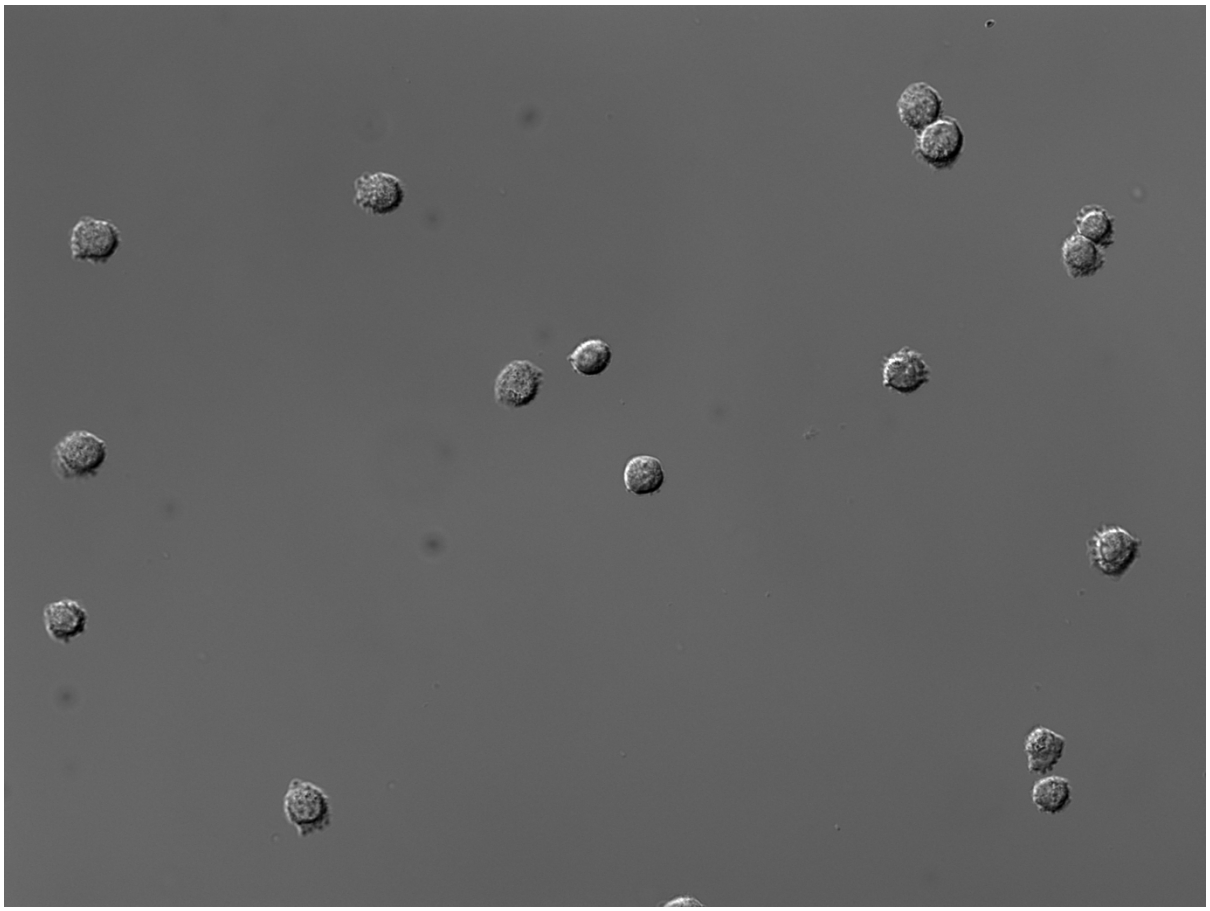
CellProfiler is capable of accurate and reliable segmentation of cells by utilizing a broad collection of classical image processing methods. Peruse the documentation on the IdentifyPrimaryObjects module, for example, to get a sense of these, e.g., thresholding, declumping, and watershed. However, despite the many problems CellProfiler can readily solve, certain types of images are particularly challenging. For instance, when the biologically relevant objects are defined more by texture and context than raw intensity many classical image processing techiques can be foiled; DIC images of cells are a common biological example.

Thankfully, machine learning, particularly pixel-based classification has yielded powerful techniques that can often solve these challenging cases. ilastik is an open-source tool built for pixel-based classification, and, when combined with CellProfiler, the range of biology that can be quantified from images is greatly expanded beyond monocultures of monolayers to include increased complexity such as tissues, organoids, or co-cultures.

Now, let's take a look at how ilastik can be used together with CellProfiler!

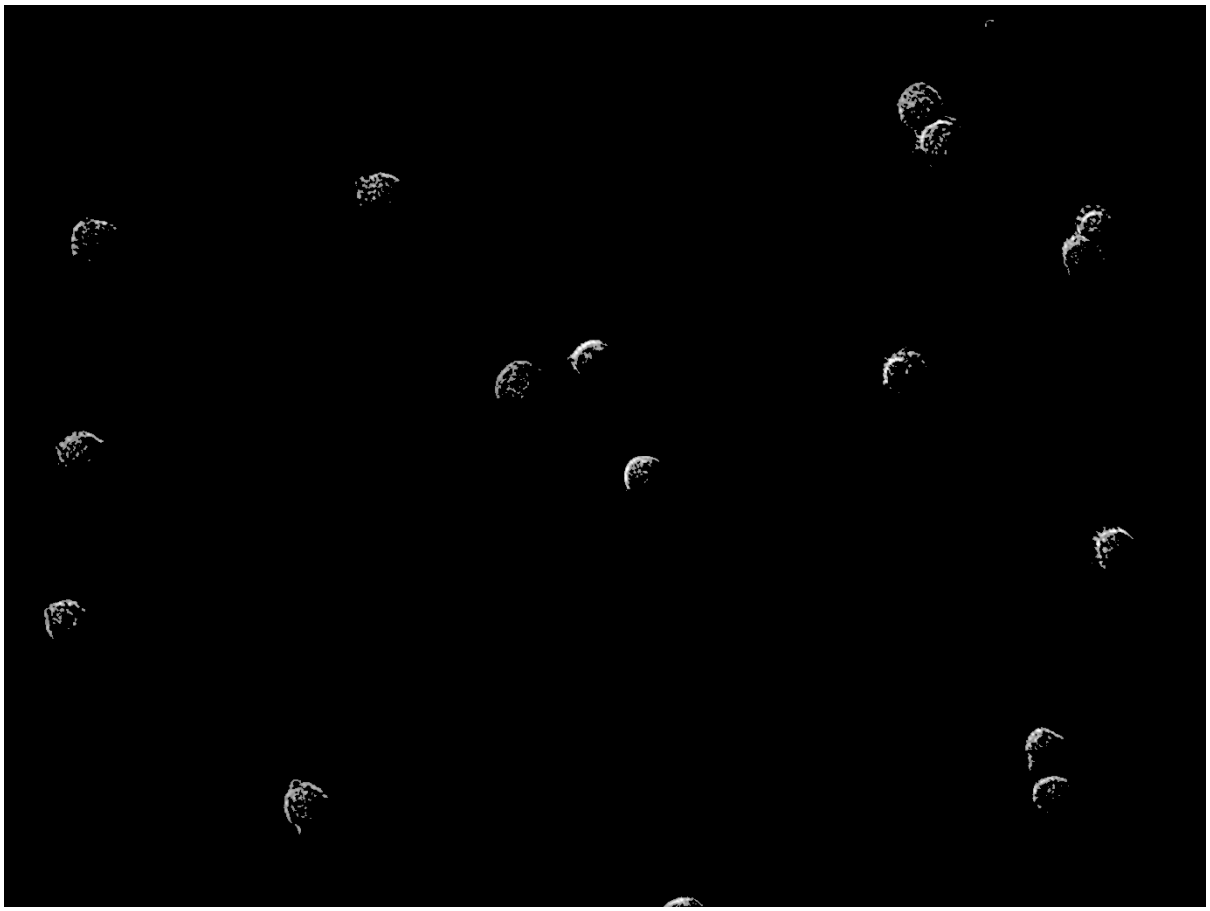
I. DIC conundrum

Consider segmenting DIC images, such as those within the imageset BBBC030. The goal will be to identify individual Chinese Hamster Ovary (CHO) cells and the regions they occupy.



A straightforward thresholding of this image yields poor results, because the cells have almost the same pixel intensity values (and sometimes even darker!) as the background. There is therefore no true foreground for these cells based solely upon an intensity histogram. Thresholding renders the CHO cells into moon-like crescents. While these fragments could be useful for simple cell counting, most metrics of morphology will be inaccurate. Now, note that there is a module, EnhanceOrSuppressFeatures, that is specifically capable of transforming DIC images into something that is readily segmented. But let's pretend for a moment we didn't have that option...

1. Open CellProfiler.
2. Drag-and-drop a BBBC030 image into the **Images** module.
3. Add the **Threshold** module to the pipeline. Select the image name (which should be 'DNA by default').
4. Run the pipeline and take note of the output.



II. Pixel-based classification with ilastik

ilastik employs pixel-based classification and complements CellProfiler. The CHO cells within the DIC image are obvious to the human eye, because we can discern that each cell is defined by a characteristic combination of light and dark patterns. These same patterns can be detected with the machine-learning algorithms within ilastik.

The machine-learning implemented by ilastik requires user annotation about what is background and what is a CHO cell before it can automatically make this determination across a set of images. ilastik provides a user interface for labeling, tagging, and identifying the objects of interest within an image. This annotation creates what is referred to in machine learning as a training set.

Annotation with 2 Labels

1. Open ilastik
2. Start a *Pixel Classification* project.

3. Load at least several BBBC030 images by drag-and-drop into the **Input Data** window.

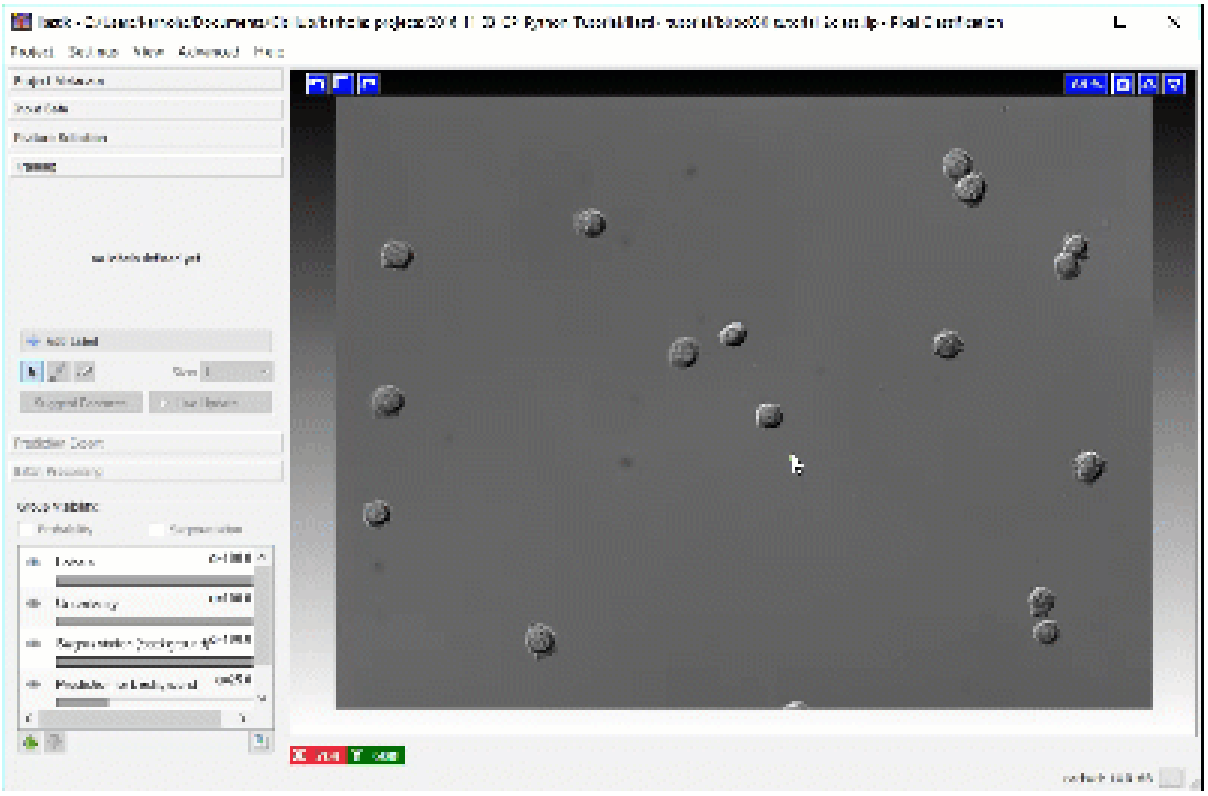
   Now explore the image within the ilastik gui. Here are some shortcuts that may prove useful are:

   - *Ctrl + mouse-wheel* = zoom.
   - The keyboard shortcut *Ctrl-D* will show the grid Ilastik uses to partition the image for processing.
   - Zoom-in far enough that the grid is no longer visible. This will speed up the *Live Update*.

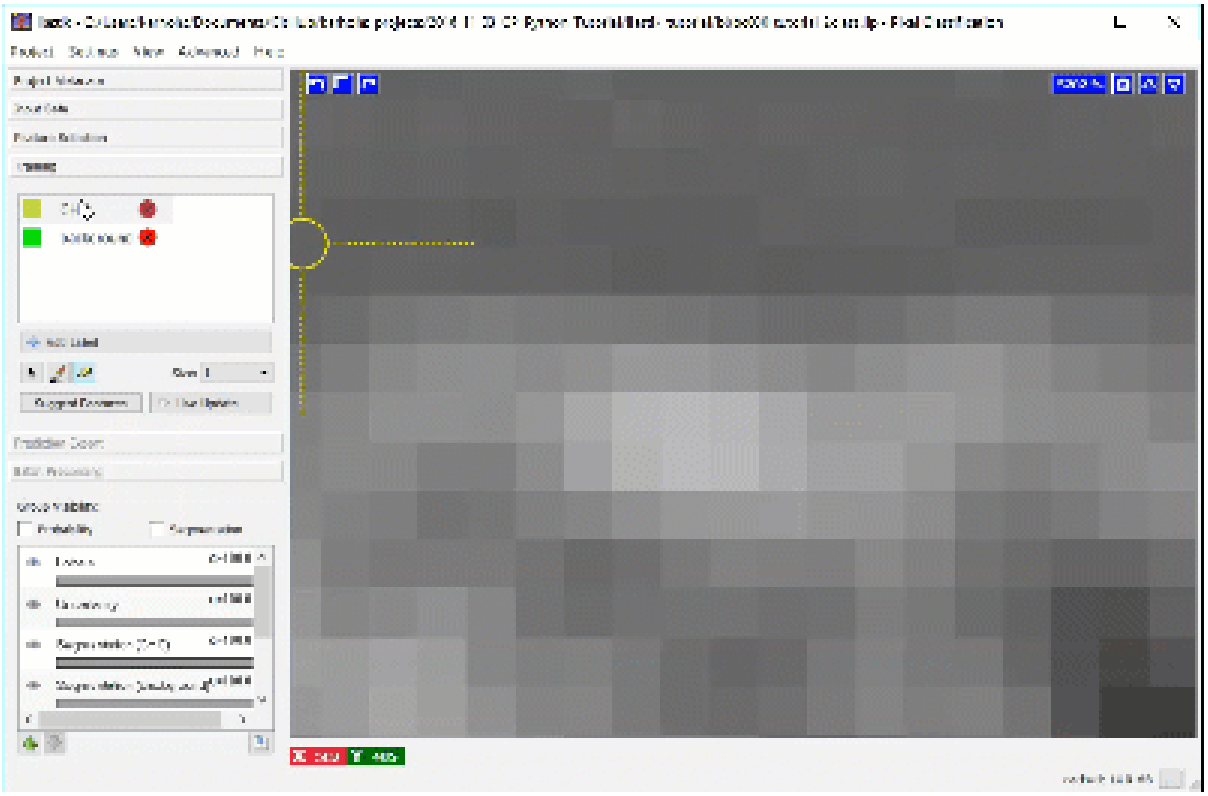
   Begin by labeling pixels for two classes: a background class and a CHO cell class.

4. Open the **Training** window.
5. Click the **+** button of the Training window to add a label. Add two labels named *background* and *CHO*.
6. Using the paint brush tool, label pixels (one at a time) for each class until you are satisfied with the segmentation.

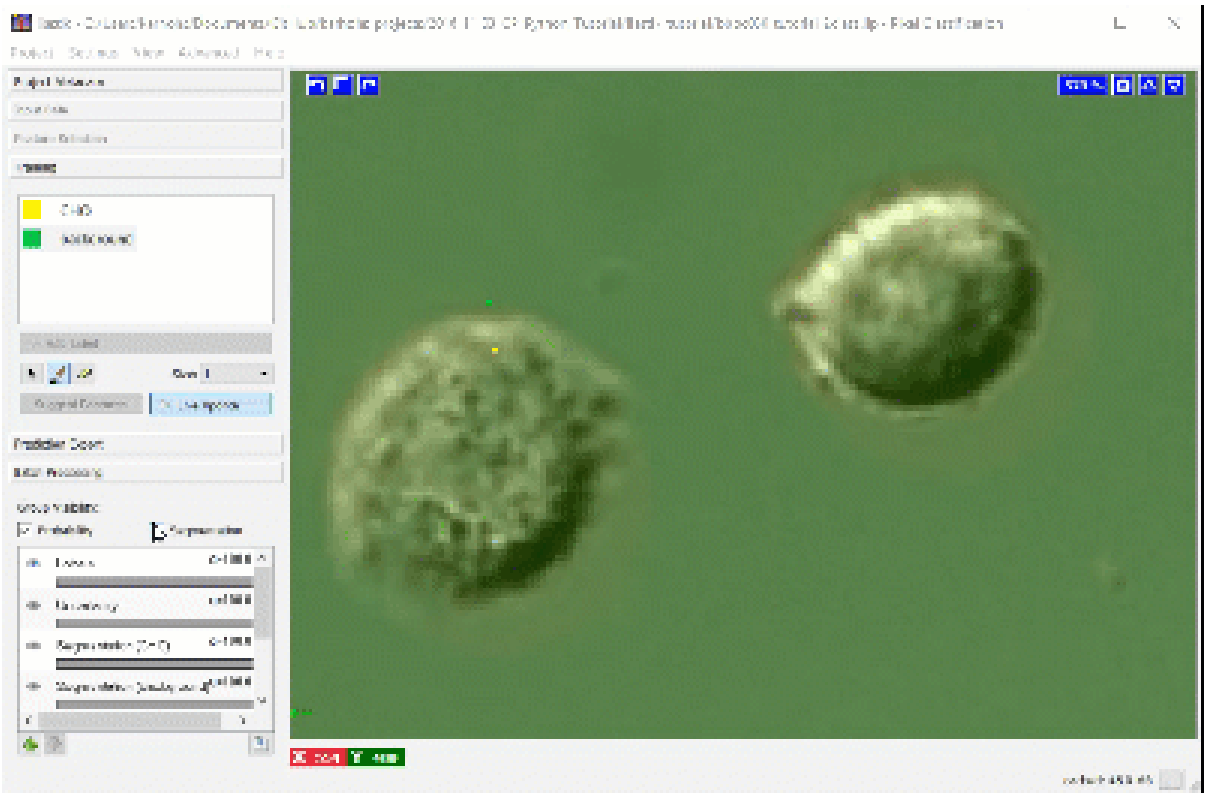
We recommend creating labels for each class one pixel at a time, rather than by making scribbles, to minimize the chance of over-fitting, i.e. too much information about any given area can cause classification to do poorly in other slightly-dissimilar areas. To label one pixel at a time, we'll need to zoom in far enough to resolve the individual pixels in the image. The image below shows how closely we must view individual cells before the pixels of the image become clear.



Using a brush size of 1, we click a single pixel from each class: one within a single CHO cell and the other in the surrounding background. In the next image, the annotation color of the CHO cell is yellow and the annotation color of the background is green. Activating *Live Update* reveals the segmentation looks similar to the results from thresholding. This outcome is promising considering this classification was determined by 1 feature and 1 pixel each for the *CHO* and *background* labels.



Adding more labels, one pixel at a time, we continue to refine the segmentation. Toggling the *Segmentation* and *Uncertainty* views provides real-time feedback that can guide the labeling process. Areas of high uncertainty will be aqua-blue, so annotating those areas will be most beneficial to training the program which pixels belong to which class. You should also view the predicted segmentation, and annotate pixels that are not currently segmented properly.



Continue until it seems that additional labels do not change the results, or a subset of the pixels begin “flipping” between CHO cell and background, or until you've labeled ~20 pixels in your original region. Check and label

other cells in the image, as well as in other images, to make sure the diversity in your experiment is represented in the training set.

Export the probability maps

When satisfied with the results, export the probability maps.

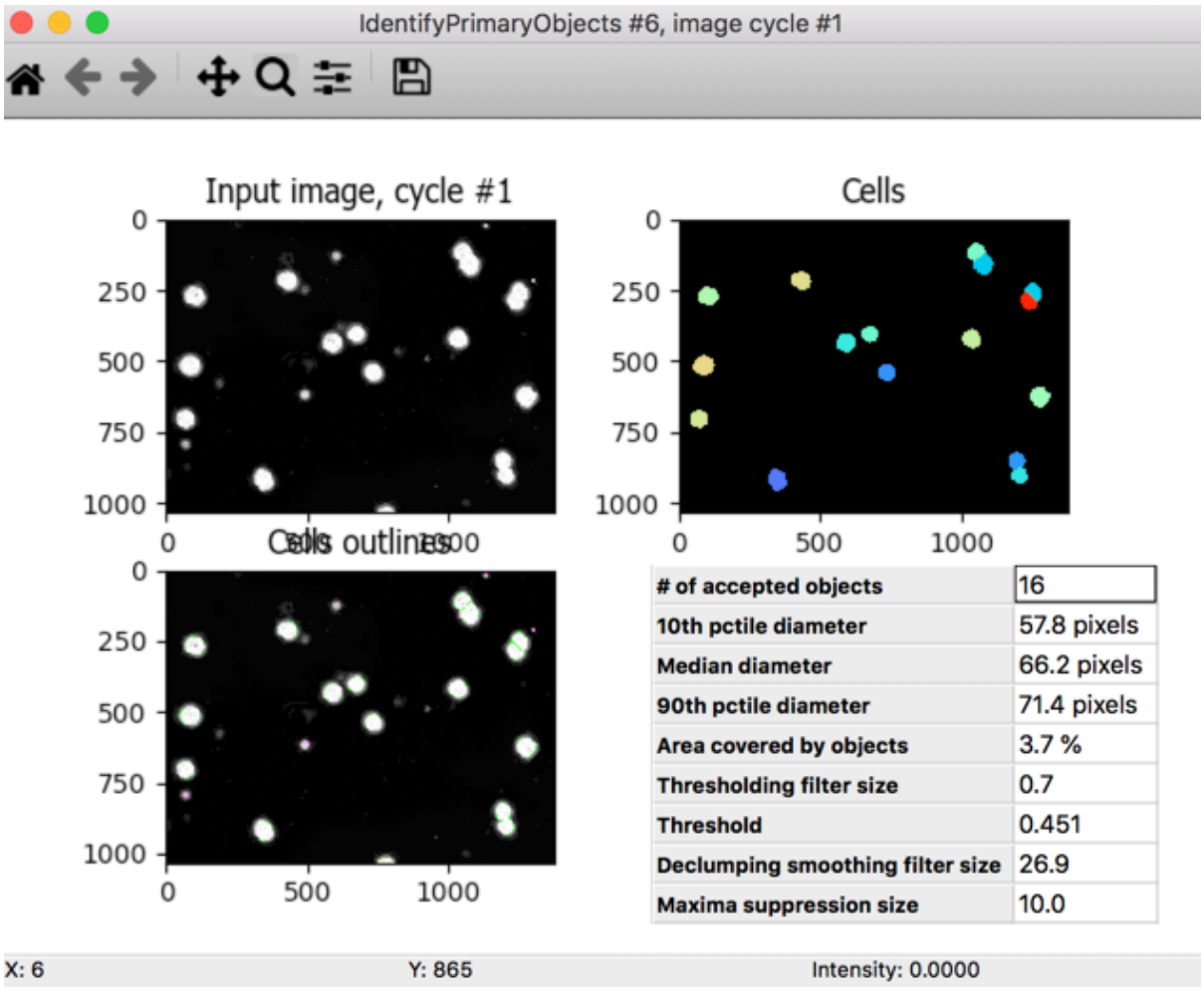
1. Open the **Prediction Export** window.
2. Click the **Choose Export Settings** window.
3. Change **Transpose to Axis Order** to `cyx`.
4. Change **Format** to `tiff sequence` (as opposed to the option that is just `tiff`).
5. Close the export settings dialog box and click the **Export All** button.
6. If you did not initially load all the images into ilastik and wish to create predictions for them all now, go to the **Batch Processing** window, select the remaining unpredicted images and hit **Process all files**. This will take a couple of minutes on most computers.

III. Segmenting probabilities with CellProfiler

The probability map images created with ilastik can then be processed by CellProfiler to identify and measure the CHO objects within the DIC images. The probability map images are grayscale images and can be treated as if they were the result of a “stain” for the cells.

1. Open CellProfiler.
2. Load the *pixel_based_classification.cpppipe* pipeline file.
3. Add the exported probability maps AND their matching original imags to the **Images** module.
4. In the **NamesAndTypes** module, if your first ilastik class was for CHO cells, set the rule criteria for the 'cho' image to `Metadata->Does->Have probnum matching->0`; if you created your background class first and the cell class second, change the final digit to 1.
5. Run the pipeline and review the segmentation. How robustly did it perform on different images?

We have now transformed the patterns and texture of intensity in the DIC image into an image where the intensity reflects the likelihood that a given pixel belongs to a cell. The image below demonstrates how the IdentifyPrimaryObjects module successfully segments all the CHO cells.



# of accepted objects	16
10th pctile diameter	57.8 pixels
Median diameter	66.2 pixels
90th pctile diameter	71.4 pixels
Area covered by objects	3.7 %
Thresholding filter size	0.7
Threshold	0.451
Declumping smoothing filter size	26.9
Maxima suppression size	10.0