Chapter 14

Extracting Rich Information from Images

Anne E. Carpenter

Summary

Now that automated image-acquisition instruments (high-throughput microscopes) are commercially available and becoming more widespread, hundreds of thousands of cellular images are routinely generated in a matter of days. Each cellular image generated in a high-throughput screening experiment contains a tremendous amount of information; in fact, the name *high-content screening* (HCS) refers to the high information content inherently present in cell images (J Biomol Screen 2:249–259, 1997). Historically, most of this information is ignored and the visual information present in images for a particular sample is often reduced to a single numerical output per well, usually by calculating the mean per-cell measurement for a particular feature. Here, we provide a detailed protocol for the use of open-source cell image analysis software, *CellProfiler*, to measure hundreds of features of each individual cell, including the size and shape of each compartment or organelle, and the intensity and texture of each type of staining in each subcompartment. We use as an example publicly available images from a cytoplasm-to-nucleus translocation assay.

Key words: Complex phenotypes, Data visualization, High-content screening, High-throughput data analysis, Microscopy, Visual phenotypes.

1. Introduction

Each cellular image generated in a high-throughput screening experiment contains a tremendous amount of information. Historically, most of this information is ignored and the visual information present in images for a particular sample is often reduced to a single numerical output per well, usually by calculating the mean per-cell measurement for a particular feature. Researchers often want access to more than this data for a number of reasons:

P.A. Clemons et al. (eds.), *Cell-Based Assays for High-Throughput Screening, Methods in Molecular Biology, Vol. 486*

DOI: 10.1007/978-1-60327-545-3_1

- Even if a single measured feature is sufficient to score an assay, it may be unknown in advance *which* feature will be most useful for scoring the phenotype of interest, especially during the prototyping/assay development phase of a project.
- Even if a single measured feature is used to select hits, it is often useful to use other phenotypes measured from the same images as a virtual secondary screen, providing a richer characterization of each hit.
- The phenotype of interest might not be distinguishable using a single measured feature, but rather several per-image features inform the decision of whether to call a sample a hit. As an example, a minimum of N cells must be alive in the sample, in addition to the average staining brightness per cell being above a particular threshold. Obtaining several desired features can be challenging within some software packages. Frequently, prepackaged software designed for a particular assay requires that the cells be prepared with a particular set of stains, and is not flexible to certain kinds of multiplexing. In only some cases can this hurdle be worked around by combining the results of reprocessing the images with several separate software routines.
- The phenotype of interest might not be distinguishable using a single measured feature, but rather several *per-cell* features inform the decision of whether to call a sample a hit. As an example, a cell might need to be within a particular size range in addition to meeting a particular staining brightness threshold. This might especially be the case if the cell population being analyzed is mixed, and only cells of a particular type are to be used to make the primary measurement of interest. In some cases, tens to hundreds of features may be needed to properly classify cells as exhibiting a particular phenotype (for example, a complex morphology), using machine-learning techniques.
- A population-averaging technique other than mean might be required to distinguish samples of interest (e.g., median, percentage of cells above a threshold, Kolmogorov-Smirnov statistic).
- Thresholds for determining cells of interest might not be precisely known prior to the experiment. Most software processes images "on-the-fly," that is, concurrent with image acquisition. Thresholds determined in advance may not be suitable, in retrospect, for the entire experimental image set.
- Per-cell data itself might be of interest for downstream analysis, for example, for clustering or machine-learning techniques.
- Certain measured features may be useful to exclude artifactual samples or cells from contaminating an analysis.

In many cases, HCS software does not produce, or limits access to, the data necessary to score a particular phenotype, especially the per-cell data that is critical for more complex phenotypes. One alternative is for an expert biologist to score every image by eye, and indeed if the experiment is important enough or the visual phenotypes complex enough, this has been worthwhile *(2)*. Fortunately, automated image analysis has been improving. Many researchers have successfully used *CellProfiler*, a flexible open-source software package we have developed, to score meaningfully complex visual phenotypes for high-content screening. We have recently described and validated *CellProfiler* for many phenotypes in cell-based high-content screening assays *(3)* and for other biological samples *(4)*.

The basis for such screens is hundreds of measured cellular features for each individual cell in each individual image. The basic steps of an image analysis pipeline in *CellProfiler* are as follows:

1. Identify anomalies in the images, for example, uneven illumination of the sample or autofocus errors.
2. Correct those anomalies when possible.
3. Identify individual nuclei in each image (using images of a DNA stain – note that nuclei are typically more uniform in shape and more easily separated from one another than cells, so we first segment nuclei, then use segmented nuclei to seed the segmentation of individual cells).
4. Identify cell boundaries in each image (using images of a cellular stain, if available).
5. Identify subcellular organelles (using images of the organelle/structure stain).
6. Identify cellular subcompartments as desired (for example, cytoplasm and nuclear or plasma membrane).
7. Count the cells, measure the size and shape of each compartment or organelle, and measure the intensity and texture of each type of staining in each subcompartment. These measurements are exported to a database and are then available for downstream analysis using various computational tools.

Here, we explain the practical tips and techniques for applying *CellProfiler* to a cytoplasm-to-nucleus translocation assay. Many signaling pathways involve the translocation of protein from one subcellular compartment to another. In the assay used here, a fluorescently labeled *forkhead* protein (FKHR-EGFP) moves from the cytoplasm to the nucleus in response to positive-control drugs (1-h treatment with either wortmannin or LY294002) in human osteosarcoma cells, U2OS (**Fig. 1**). In this example, we show how to measure hundreds of features other than the primary readout (ratio of staining in the cytoplasm vs. nucleus) for downstream data analysis. This type of analysis revealed a change in nuclear

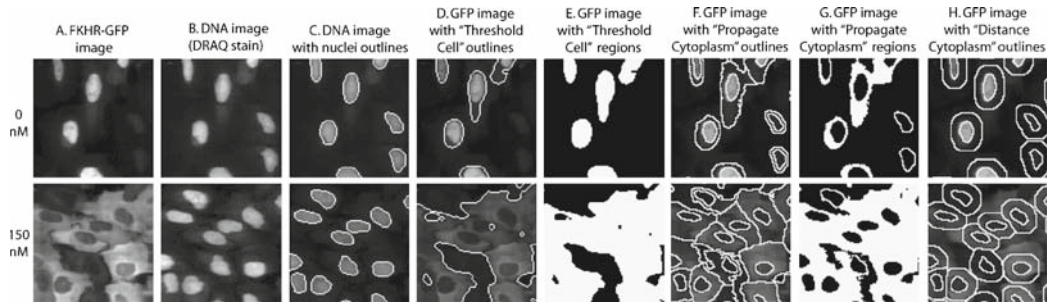

Fig. 1. Example images of the Cytoplasm to Nucleus Translocation assay. Top row = no drug; bottom row = 1-h incubation with 150 nM wortmannin. The actual images are approximately fivefold larger field of view in each dimension (640 pixels square), in 8-bit grayscale BMP format, with four replicates for each of the nine-point dose curve for each drug, plus negative controls. They are further described (*BioImage plate*) and were acquired from http://www.ravkin.net/SBS/CNT_Images.htm (courtesy of Ilya Ravkin). Outlines and regions are as processed by *CellProfiler* and saved using the OverlayOutlines or ConvertToImage module followed by a SaveImages module. Note that all images were contrast-enhanced using AutoLevels in Photoshop for display purposes only.

morphology in addition to the cytoplasm-to-nucleus translocation *(3)*. The steps involved in the protocol here are as follows:

1. Download and install *CellProfiler* software.
2. Download the example pipelines and run them on the example images.
3. Load the illumination correction pipeline, adjust it for your images, and run it.
4. Adjust the main pipeline for your images using test images.
5. Run the main pipeline on all images.
6. Export data.
7. Perform downstream data analysis.

It is much easier to adapt an existing pipeline than to build a new one from scratch, so we suggest using this cytoplasm-to-nucleus translocation pipeline as the basis for any HCS screen. In fact, it can be adapted to most cell types and assays using the point-and-click interface of *CellProfiler*. In some sense, the example pipeline we describe here is roughly similar no matter what the assay/phenotype of interest. The basic concept of identifying cells and measuring everything possible about them is constant from experiment to experiment; the uniqueness of each assay comes in determining which measure(s) are most useful to identify interesting samples, by data mining. A full discussion of appropriate data-mining methods is beyond the scope of this chapter, but we conclude the protocol by briefly reviewing the methods required to identify any particular phenotype of interest based on these measurements.

2. Materials

1. Fluorescence microscopy images to be processed. The setup time for an analysis is the same whether a handful or hundreds of thousands of images are processed. We routinely analyze tens of thousands of images per experiment. The images can be located within subfolders and need not be in a particular order or have a particular naming convention. A variety of file formats are currently readable by *CellProfiler*: *.bmp*, *.cur*, *.fts*, *.fits*, *.gif*, *.hdf*, *.ico*, *.jpg*, *.jpeg*, *.pbm*, *.pcx*, *.pgm*, *.png*, *.pnm*, *.ppm*, *.ras*, *.tif*, *.tiff*, *.xwd*, *.dib*, *.mat*, *.fig*, *.zvi*, *.flex*, *.stk*.
2. Computer (Macintosh, Windows, or Unix/Linux): A desktop or laptop computer with at least 1 GB of RAM (2 GB or more is preferable) and at least a 1-GHz processor is recommended.
3. *CellProfiler* software: The software can be downloaded for free (*see* **Subheading 3.1**).
4. Example images and their corresponding *CellProfiler* pipelines. These can be downloaded for free (*see* **Subheading 3.2**).
5. Computing cluster: This is optional, depending on the number of images to be processed and the processing speed for a particular analysis pipeline. The example image pipeline demonstrated here on 96 sample images requires about an hour on a single computer with a 2.6-GHz processor and 8-GB RAM. Larger image sets will likely require a computing cluster.

3. Methods

3.1. Download and Install CellProfiler Software

1. Choose whether to use the regular or developer's version.
2. Most users will use the regular version (also known as the "compiled"/binary version) suitable for their computing platform (Windows, MacIntosh, or Unix/Linux). This version is free (GPL license) and does not require a MATLAB license.
3. Researchers wishing to implement their own image-analysis algorithms should download the developer's version (that is, the MATLAB source code). The developer's version is free and open-source (GPL license), but does require installing MATLAB software and its license, not detailed here (Mathworks, Natick, MA; www.mathworks.com).

4. Download the version of software you chose from www.cellprofiler.org/download.htm.
5. Follow the installation instructions from the webpage to install *CellProfiler* and start it (*see* **Note 1**). If you encounter difficulties on this step, consult the installation instructions (www.cellprofiler.org/install.htm), or visit the online forum (www.cellprofiler.org/forum) to see if this problem has been encountered and solved before.

3.2. Download the Example Pipelines and Run Them on the Example Images

1. Download the example images and pipeline called *Human cytoplasm-nucleus translocation assay (SBS)* from www.cellprofiler.org/examples.htm. After downloading the file, make sure that it is *unzipped* (that is, uncompressed). Often, this occurs automatically, but if the file you downloaded does not automatically produce an accompanying folder, you should unzip the file manually. Usually, double-clicking the file accomplishes this (using your computer's built-in unzipping functionality); otherwise, use unzipping software. This will produce a folder containing the images described in Fig. 1.
2. Run the example pipeline called ExampleSBSPIPE.mat on the example images you downloaded. To do this, follow the instructions in Help > Getting Started > GettingStarted. This step allows you to see how processing typically proceeds (*see* **Notes 2–4**). If you obtain *Out of memory* errors, *see* **Note 5**.

3.3. Load the Illumination Correction Pipeline, Adjust It for Your Images, and Run It (see Notes 6 and 7)

1. After loading the ExampleSBS-IlluminationPIPE.mat pipeline into *CellProfiler*, make LoadImages module adjustments: Images need not be named or organized a particular way in order to use *CellProfiler*. Setting this module tells *CellProfiler* where to retrieve images and gives each image a meaningful name for the other modules to access. There are two basic methods to tell *CellProfiler* which batches of images are to be analyzed. The Order option is used when images are present in a folder or series of subfolders in repeating order (e.g., DAPI, GFP, DAPI, GFP, …) and the Text option is used when images have a particular piece of text in the name (e.g., all the DNA channel images contain the text "_D"; all the GFP channel images contain the text "_G"). Placing two different LoadImages modules in the pipeline allows you to choose one entire folder of a particular image type and a separate folder with a different image type (if GFP and DAPI images are stored in separate folders). There are a number of ways images can be loaded and identified; see the help for this module for more details (*see* **Note 8**).
2. CorrectIlluminationCalculate module adjustments: The only adjustment might be the two settings related to *width of artifacts*. The overall goal of this module is to produce an image that is

a believable representation of the illumination pattern that is affecting your images. The help for this module describes the various available methods, but Median Filtering is the most common method for HCS image sets. You will note that the example pipeline is set up to override the automatic calculation of artifact width, with a filter size of 150. If this image is too smooth (that is, it is not capturing the full amount of variation you think is present in the illumination pattern), then the filter size (or artifact width) should be decreased. If the image is too rough (that is, there is too much fine detail in the pattern such that it represents the random distribution of cells in your images rather than the real illumination pattern), then the filter size (or artifact width) should be increased, or you have an image set too small to accurately calculate the illumination pattern. If you have more than two channels for your image set, you should add additional CorrectIlluminationCalculate and SaveImages modules to the pipeline.

3. SaveImages module adjustments: no adjustments should be needed.
4. Once the modules are appropriately set, run the adjusted illumination correction pipeline to produce illumination correction images for your image set. Make sure that the default image and output folders are appropriately set, and click Analyze images. After processing completes, the resulting illumination correction images (Channel1ILLUM.mat and Channel2ILLUM.mat) should be produced and deposited in the default output folder (*see* **Notes 9** and **10**).

3.4 Adjust the Main Pipeline for Your Images Using Test Images (see Notes 11 and 12)

1. Using your computer's normal interface, create a test folder and copy several test images into it (*see* **Note 13**).
2. Using your computer's normal interface, create a test output folder.
3. In *CellProfiler*, set the default image and output folder to be your test image folder and test output folder, respectively (*see* **Note 14**).
4. LoadImages module adjustments: *see* **Subheading 3.3**, **step 1**.
5. LoadSingleImage module adjustments: make sure that the illumination-correction images you produced as described in **Subheading 3.3**, **step 4**, are specified to be loaded by this module.
6. LoadText module adjustments: first, you must produce an appropriate text file containing the doses corresponding to your images (*see* **Note 15**). This is necessary for the CalculateStatistics module to tell you how effective the assay is overall. Using the example file 1049_Dose.txt as a template (*see* **Note 16**), adjust it so that each line of the text file corresponds to the dose for each image cycle (each field

of view) that you are processing, in the order that the cycles will be processed. Looking in the window in the bottom left on the main *CellProfiler* window shows you the order that your images will be processed. If you have only positive and negative controls rather than a full dose curve, you can use a dose of 1 for positives and 0 for negatives in the text file. Once you have created the text file and placed it in the project folder, be sure that the LoadText file is set to use the text file you just made (*see* **Note 17**).

7. IdentifyPrimAutomatic module adjustments: there are two IdentifyPrimAutomatic modules in this pipeline – the first identifies nuclei using the DRAQ channel (**Fig. 1C**) and the second simply identifies the portions of the cells that show GFP staining (**Fig. 1D** and **E**). It turns out that for the primary readout of the assay (ratio of GFP staining in cytoplasm vs. nucleus), the second is not useful, but we include it here under the principle of measuring a large number of features of each image and choosing later what is useful. Several methods for object identification are available in a modular fashion within the IdentifyPrimAutomatic modules (*see* **Note 18**). The first IdentifyPrimAutomatic module is probably the most important to adjust in getting the pipeline to work well on your image set, so take time to see the help for the module for a complete description of the available methods and tips for the settings. In general, the settings that might require adjustment will be as follows:
   (a) The *typical diameter of objects*, the *automatic thresholding method*, *threshold correction factor*, and *approximate percentage covered by objects* (if *CellProfiler* appears to be inaccurately deciding between nuclei and background).
   (b) The *lower and upper bounds on threshold* (very important to be set properly to prevent poor results on images that are extremely different from expected, usually images with dramatic artifacts).
   (c) The *size of smoothing filter* and *suppress local maxima* (if *CellProfiler* is inappropriately splitting nuclei in two or merging clumps of cells together).
8. IdentifySecondary module adjustments: there are two IdentifySecondary modules in this pipeline – the first finds the edges of the GFP staining (**Fig. 1F** and **G**). In samples where GFP is primarily nuclear, the *objects* found by this module will simply be the nuclei, but in samples where the GFP staining is primarily cytoplasmic, the edges of the *objects* will extend to cell edges. The second IdentifySecondary module creates a ring around each nucleus, which is used as a proxy for cell edges in the absence of a cellular stain (**Fig. 1H**; *see* **Note 19**). In general, the settings that might require adjustment for the

first IdentifySecondary module (Propagate option) will be as follows:

(a) The *automatic thresholding method*, *threshold correction factor*, and *approximate percentage covered by objects* (if *CellProfiler* appears to be inaccurately deciding between cells and background).

(b) The *lower and upper bounds on threshold* (very important to be set properly to prevent poor results on images that are extremely different from expected, usually images with dramatic artifacts).

(c) The *regularization factor* (this controls the precise dividing line between cells that touch each other). In general, the only setting that might require adjustment for the second IdentifySecondary module (Distance option) will be *number of pixels to expand* (this controls the size of the rings around each nucleus).

9. IdentifyTertiarySubregion module adjustments: no adjustments should be necessary.
10. MeasureCorrelation module adjustments: no adjustments should be necessary, unless channels are available beyond DNA and GFP, or unless new compartments have been defined.
11. MeasureObjectIntensity module adjustments: no adjustments should be necessary, unless channels are available beyond DNA and GFP, or unless new compartments have been defined.
12. MeasureObjectAreaShape module adjustments: no adjustments should be necessary, unless new compartments have been defined.
13. MeasureTexture module adjustments: this module should be adjusted if channels are available beyond DNA and GFP, or if new compartments have been defined. Further, the scale of the texture (in units of pixels) should be adjusted for a particular image set. The scale of the texture is essentially empirical; a higher number measures larger patterns of texture, whereas smaller numbers measure more localized patterns of texture (fine texture). You are essentially asking *CellProfiler* to tell you whether the staining pattern is smooth on a particular scale. You can also add several texture modules, each measuring a different scale of texture, to the pipeline. See the help for this module for more details.
14. CalculateRatios module adjustments: no adjustments should be necessary, unless channels are available beyond DNA and GFP, or unless new compartments have been defined.

15. ClassifyObjects module adjustments: the threshold needs to be set empirically for this module. Of course, as has been stated as the overall approach here, we recommend choosing thresholds postprocessing, using appropriate data exploration/analysis tools, but if the threshold is already close to known, it can be useful to classify cells as positive or negative within the pipeline, so that CalculateStatistics can be used on the resulting data.
16. CalculateStatistics module adjustments: *see* **Subheading 3.4**, **step 6**.

3.5. Run the Main Pipeline on All Images

1. If the number of images is manageable for a single computer, change the image folder from the test images folder to the real images folder; change the output file name (if desired), and click the Analyze images button.
2. If the processing time would be too great on a single computer, then run the processing on a cluster as follows:
   (a) Download and install *CellProfiler* on a computing cluster. See the installation instructions as well as Help > General Help > Batch Processing within *CellProfiler* and the online forum (www.cellprofiler.org/forum). The developer's version may be used on an entire cluster if MATLAB licenses are available. Alternatively, you should use a compiled version. There are a wide variety of computing clusters in existence; one compiled version of *CellProfiler* for cluster computers is available for download at www.cellprofiler.org. If this version is not compatible with a particular cluster, the developer's version (source code) can be downloaded and recompiled (using MATLAB's compiler) on a representative cluster computer. This will require a single MATLAB license, but the resulting compiled code can be run on the entire cluster without MATLAB licenses.
   (b) Add the ExportToDatabase module (optional). When splitting your images into batches, your data will be produced in separate data files for each batch and you have two options:
   - If the resulting data files are not overwhelmingly large, you can merge the output files into a single output file (*see* **Subheading 3.6**, **step 2**). This option does not require adding any additional modules to the pipeline.
   - Most often for large image sets, you will prefer to export the resulting data to a MySQL or Oracle database for further analysis and exploration. In this case, add the ExportToDatabase module to the pipeline and configure it according to the help for the module.

(c) Add the CreateBatchFiles module to the end of the pipeline and configure it appropriately, according to the help for the module.

(d) Run the first image cycle locally. Clicking the Analyze images button will cause *CellProfiler* to process the first batch of images locally, and then produce the necessary files for batch processing.

(e) Submit the batches to your cluster for processing. See the Help > General Help > Batch Processing within *CellProfiler* for details.

(f) Check the progress of processing. For example, if using LSF software to submit jobs to your cluster, *see* **Table 1** for some basic commands to monitor the progress of processing.

3.6. Export Data

Once processing has completed, the data can be exported in three ways:

1. If you ran all images on one computer, the data can be exported to Excel using the data tool in the main window (Data Tools > ExportData) (*see* **Note 20**).

Table 1
Commands to monitor processing of jobs when using LSF software for cluster computing

Goal	Command
List all jobs	bjobs
Count all jobs	bjobs | wc -l
Count running jobs	bjobs | grep RUN | wc -l; echo jobs_running
Count pending jobs	bjobs | grep PEND | wc -l; echo jobs_pending
Check how many jobs are running and pending for a particular sample	bjobs -w | grep samplename | wc -l
Kill all jobs	bkill 0
Switch job to another queue	bswitch normal 189091 (where 189091 is the job number and normal is the name of the desired queue to switch to)
Check how many jobs are running and pending in a particular queue	echo normal; bjobs | grep normal | wc -l (where normal is the name of the queue of interest)
See the output of a job while it is still running	bpeek
To submit an individual job	bsub -B -N -u email@X.edu matlab -nodisplay -r Batch (*Notes*: -B sends email at beginning of the job, -N at the end; -q QUEUENAME allows selecting a queue; BATCH is the name of the job, for example Test3Batch_2_to_2.)

2. If you ran the batches on a cluster, but the sizes of the resulting output files are not overly large, use DataTools > MergeOutputFiles to merge the output files into a single file, and then use DataTools > ExportData to export the data to a tab-delimited file that you can open in Microsoft Excel (*see* **Note 20**).
3. If you ran batches on a cluster and used the ExportToDatabase module, the data can be exported by following the instructions for the ExportToDatabase module.

3.7. Perform Downstream Data Analysis

As mentioned in the introduction, we briefly overview options here:

1. Challenges of image-based data analysis: Typical screening experiments include tens to hundreds of thousands of images and each image contains hundreds or thousands of cells. *CellProfiler* measures hundreds of features about each of these millions of cells. Drawing meaningful conclusions from this data beyond very well-defined, simple analyses can be very challenging, due both to the huge amount of data involved and to the high dimensionality of the measurements. Furthermore, biases in the data can occur due to variation across the physical layout of the multiwell plates (known as plate or edge effects). Bias correction is an active area of research *(5–7)* and beyond the scope of this chapter.
2. Types of image-based data analysis: The introduction lists some of the more complex types of data analysis that might be necessary to identify samples of interest. Here, we group and describe some of these types of analyses. Note that data exploration itself is often a necessary step in the process of choosing an analysis method, and after hits are chosen in the screen, further data exploration may be of interest to characterize hits.
   a. Per-image averaging: Per-cell measurements can be combined to give per-image values by taking means, medians, etc., or otherwise reducing each measurement to a small set of parameters. This approach can work well for highly penetrant phenotypes. Several per-image features may be necessary to identify samples of interest.
   b. Per-image population comparisons: Each sample's cell population can be compared to a control cell population using distribution-based metrics such as Kolmogorov-Smirnov *(8)* or Kuiper *(9)* tests, or compute sample-based information-theoretic estimates, such as the KL divergence between the two distributions *(10)*.
   c. Per-cell classification: Individual cells can be classified as fitting a particular phenotype, based on the cell's measurements. The proportion of cells in each sample fitting the phenotype is thus the primary readout. This type of analysis is able to detect very small changes in the per-

centage of cells showing a particular phenotype (for example, from 1 to 3%), which is highly relevant considering that cell populations are inherently highly variable *(11)*. Moreover, reducing per-cell measurements to per-image measurements ignores the relationships between measured features on a per-cell basis. These relationships can sometimes be important for distinguishing samples of interest. For example, two images may have identical proportions of bright-staining and dark-staining cells in the green and red channels, but in one sample, the bright green cells tend also to be bright red, while in the other, bright green cells tend to be dark red. Such phenotypes would be impossible to distinguish using the per-image approaches described earlier. There are two options for per-cell classification:

- If a small number of measured features are sufficient to identify a cell subpopulation of interest, and if these measured features are known in advance, the subpopulation of interest can be identified in scatter plots of per-cell data, with the axes chosen by the researcher. Successive gating (choosing a subpopulation in a scatter plot, replotting that subpopulation on a new scatter plot with two new axes) can make use of more than just two measured features per cell.
- When more than just a few features are needed to define the cell subpopulation of interest, or if these features are not known in advance, machine-learning methods can be used to train a computer to recognize the phenotype on a per-cell basis. We have had success using an approach similar to content-based image retrieval in the field of computer vision *(12, 13)*. The biologist sorts cells as positive or negative for their phenotype of interest, and the software develops rules that are able to distinguish these examples based on the rich set of per-cell data collected by *CellProfiler*. The biologist sorts cells until the classifier becomes sufficiently accurate, and then all cells in all images are scored automatically by the computer as positive or negative. Images are then scored based on the number or percentage of phenotype-positive cells.

3. Tools for image-based data analysis: For data exploration and analysis from high-content screens, interactivity is needed between per-image data, per-cell data metadata for each sample, and raw images. The amount of data, (in terms of samples, cells, and features) pushes the limits of most software packages. We have recently released open-source data exploration software called *CellProfiler Analyst* (www.cellprofiler.org) readily enable the data analysis approaches

described above, especially that in **Subheading 15.3.7**, **step 2c**. At the time of this writing, other software is available for some of the analyses described above, including Spotfire (www.spotfire.com), CellMine (www.bioimagene.com), GeneData Screener (www.genedata.com), in addition to software sold in conjunction with high-throughput microscopes. Skilled researchers can also query databases directly to analyze data as needed.

4. Notes

1. If some of the text in the main window is not easy to see, go to File > Set Preferences to permanently set a more appropriate font size (*see* **Note 21**).
2. You will note that one display window opens for each module in the pipeline, and that as each field of view (*cycle*) is processed, the window refreshes to display the current cycle. The Status window remains open during processing and indicates how many cycles will be performed, how many are completed, and the amount of time elapsed. It also gives you the option to pause or cancel processing. The Details button indicates how much time is consumed on a per-module basis.
3. The measurement data acquired during each image cycle is stored at the end of every cycle in an output file, which grows larger and larger as the cycles complete. You can name the output file using the box next to *Analyze images* in the main window. The default output file name is DefaultOUT.mat, and it is impossible to overwrite an existing output file – if you attempt to do this, *CellProfiler* will ask if it is acceptable to add a number to the file name so that it is a new, unique file name.
4. Processing is much faster if display windows are closed. They may be closed during processing, but it is best to wait until the first cycle is finished to maintain the ability to reopen them properly later, using the Open figure and Close figure buttons in the main window. You can also go to File > Set Preferences to change the Display Mode if you would like to choose which windows to display each time you process an image set, or if you would like to *not* display windows, by default (*see* **Note 21**).
5. Image processing, especially for large image sets, is quite memory-intensive, and at some point during processing as measurements accumulate, you may run out of memory on your computer. If so, try the following:

(a) Close all other programs on the computer before starting *CellProfiler*.

(b) Run the example images on a small test set of images. You can remove a subset of the images from the folder and process just the remaining ones. You will also need to adjust the Dose text file accordingly – *see* **Subheading 3.4**, **step 6**.

(c) Use a different computer with more memory.

(d) For more tips, *see* Help > General Help > Memory and Speed within *CellProfiler*.

6. The physical limitations of any microscope lead to nonuniformity in the optical path of the sample, microscope, and/or its camera. We use the term *illumination anomalies* to refer to all sources of nonuniformity, including an uneven light source, uneven filters or lenses, and nonuniform camera sensing. Most software controlling the microscopes has the ability to perform a white-referencing step, where an image of a uniformly fluorescent (*flat*) sample is used to calculate and then correct for the illumination anomalies. This option is often ignored or improperly calibrated. Even when images are supposedly corrected by this method, these variations are frequently greater than 1.5-fold across the field of view, and can severely compromise the accuracy of measurements obtained. The standard illumination correction described here is based on the assumption that all the images in the experiment were collected in a single acquisition run, where as many elements as possible are constant; for example, the optical path and filters have not been moved, the lamp has maintained constant intensity, and the multiwell plate bottoms and the location imaged within each well are uniform. It is also based on the assumption that the background staining is negligible relative to the signal, and that the distribution of cells within each image is uniform (that is, there are no consistent biases of cells in one particular location). We have developed methods to detect and correct for nonuniform cell distributions but, while effective, they have not yet been made easily usable and are not discussed here (Thouis R. Jones, AEC, David M. Sabatini, and Polina Golland, unpublished data). For the standard illumination-correction calculation, we estimate the illumination variation as a smoothed per-channel average of all the images in the screen. See our other work for more details on the theoretical basis of this approach *(14)*.

7. To test whether the illumination stayed relatively constant across an experiment, you can calculate the illumination correction images as described here for subsets of images – a

set of images at the beginning and a separate set of images at the end of the experiment. If the overall pattern is relatively similar, then it is experimentally justified to apply the images calculated *from* the entire set *to* the entire set. If the patterns are substantially different from the beginning to the end of the experiment, you will need to develop a more sophisticated method of illumination correction, or make an attempt to find the point in time where the illumination pattern changed by calculating the illumination correction images from several subsets of images across the experiment. Beware, however, that this calculation will not work well for too small a subset of images. *See* **Subheading 3.3**, **step 2**, for a further description of how to tell whether an image set is *too small* for good calculation.

8. Color images can also be processed in *CellProfiler*. After loading the color images with the LoadImages module, use the ColorToGray module to split the images into their component colors.
9. When image sets are large, there are two ways to speed up the calculation of the illumination correction pipeline: you can run a separate pipeline for each wavelength on individual local computers, or on remote cluster computers. To use remote computers, add the CreateBatchFiles module to the end of the pipeline (*see* **Subheading 3.5**, **step 2**). In that module, set the number of images per batch to be larger than the total number of image cycles in your image set, because each wavelength's processing must be performed in a single batch job.
10. The display window for the CorrectIllumination_Calculate modules will show in red the minimum and maximum pixel intensity for the calculated illumination correction images. If the illumination varies more than 5- to 10-fold across the field of view (that is, the minimum is 1 and the maximum is more than 5–10), you should probably consider improving image acquisition to make the illumination more uniform, or cropping the images (using the Crop module, Rectangle option) to remove the worst parts of the images.
11. For help or a technical explanation for any module, click the "?" button in the main *CellProfiler* window below the pipeline, while the module of interest is selected, or go to Help > Modules Help and choose the module of interest.
12. There are multiple modules for measurement in the pipeline, even though not all of them may be useful to score a particular assay (see the introduction for the rationale). This pipeline measures the size and shape of each cell (e.g., area, perimeter, extent, convexity, and several Zernike moments)

and the intensity and texture of the various stains (e.g., mean and standard deviation of intensity, correlation of stains, and Gabor filter response at various scales). Measurements are made for each cellular compartment that has been defined (e.g., nucleus, cytoplasm, and entire cell).

13. Good images to use in your test set include negative and positive controls and images from the beginning and end of an acquisition run. In general, you want the very *worst*-case images to be included in your testing, not the *best* ones. The biggest pitfall in setting up cell image analysis is overtuning the analysis to a small, nonrepresentative set of images.
14. Browsing or typing into the main window will set the default image and output folders for your current session. To set them semipermanently, go to File > Set Preferences to select an image and output folder (*see* **Note 21**).
15. If the image set you are processing is not a dose-response experiment or a positive/negative control experiment, the LoadText module and CalculateStatistics module can be removed from the pipeline altogether.
16. The file included in the example image set, 1049_Dose.txt, contains the dose information for the example images. Note that the first half of the example image set is a dose-response curve for the drug wortmannin, and the second half is a dose-response curve for LY294002. For simplicity, the example pipeline described in this article processes both halves of the experiment together and therefore the CalculateStatistics module calculates the assay statistics across both drugs. For this reason, we have entered the doses in 1049_Dose.txt as 0–10 rather than their actual absolute concentrations in nM, to allow the entire image set to be run together, and to allow the calculation of a single Z factor for both positive controls. Note that, strictly speaking, the V factors and EC_{50} values from the example image set are not exactly interpretable, because we have combined the dose-response curves from two different drugs.
17. You can enter doses as exponents in the text file and then set the CalculateStatistics module to log-transform values. See the CalculateStatistics module for more details on how to set up the text file. In addition, for time-lapse image series, time points can be used as doses.
18. Very unusual cell types might require the development of new algorithms, which can be written and integrated into the open-source code of *CellProfiler*, given its modular and flexible structure. See Help > Developer Info in the main window of *CellProfiler*.

19. The ideal approach would be to collect images of a cellular stain (e.g., actin) and use IdentifySecondary's Propagate option to determine the edges of cells based on the cell stain image. The Propagate option is typically preferred over the Watershed method because Propagate does not rely on the borders of cells being consistently bright or dark, and it is less sensitive to gaps in staining *(15)*.
20. You can also put the module ExportToExcel in the pipeline to automatically export the data to Microsoft Excel files at the end of processing. Note that the files produced are tab-delimited, and can be opened by any spreadsheet program, not just Microsoft Excel. Warning: many versions of Microsoft Excel are limited to 256 columns and 65,536 rows, so your data may be too large to open in this program. Swapping rows and columns when exporting might be a good workaround if you reach these limits in only one dimension.
21. You can save a different set of preferences for each user, or even for each project, by choosing not to save the preferences as default, but rather giving them a specific name, *User1Preferences.mat*, for example.

Acknowledgments

The author is grateful to David M. Sabatini and his laboratory members at the Whitehead Institute for Biomedical Research, and Polina Golland and Thouis R. Jones at the Massachusetts Institute of Technology for their support and intellectual contributions to the methods described in this work. Images from Ilya Ravkin and testing by Adam Papallo are also much appreciated. The methods described in this work were also developed with the support of academic grants from the Society for Biomolecular Sciences and L'Oreal for Women in Science, and a Novartis fellowship from the Life Sciences Research Foundation.

References

1. Giuliano, K. A., DeBiasio, R. L., Dunlay, R. T., Gough, A., Volosky, J. M., Zock, J., et al. (1997) High-content screening: a new approach to easing key bottlenecks in the drug discovery process. *J. Biomol. Screen.* **2**, 249–259.
2. Kiger, A., Baum, B., Jones, S., Jones, M. R., Coulson, A., Echeverri, C., et al. (2003) A functional genomic analysis of cell morphology using RNA interference. *J. Biol.* **2**, 27.
3. Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., et al.

(2006) CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7**, R100.

4. Lamprecht, M. R., Sabatini, D. M., and Carpenter, A. E. (2007) CellProfiler: free, versatile software for automated biological image analysis. *Biotechniques* **42**, 71–75.
5. Root, D. E., Kelley, B. P., and Stockwell, B. R. (2003) Detecting spatial patterns in biological array experiments. *J. Biomol. Screen.* **8**, 393–398.
6. Makarenkov, V., Kevorkov, D., Zentilli, P., Gagarin, A., Malo, N., and Nadon, R. (2006) HTS-Corrector: software for the statistical analysis and correction of experimental high-throughput screening data. *Bioinformatics* **22**, 1408–1409.
7. Malo, N., Hanley, J. A., Cerquozzi, S., Pelletier, J., and Nadon, R. (2006) Statistical practice in high-throughput screening data analysis. *Nat. Biotechnol.* **24**, 167–175.
8. Perlman, Z. E., Slack, M. D., Feng, Y., Mitchison, T. J., Wu, L. F., and Altschuler, S. J. (2004) Multidimensional drug profiling by automated microscopy. *Science* **306**, 1194–1198.
9. Kuiper, N. H. (1962) Tests concerning random points on a circle. *Proc. K. Ned. Akad. Wet. Ser. A* **63**, 38–47.
10. Kullback, S. and Leibler, R. A. (1951) On information and sufficiency. *Ann. Math. Stat.* **22**, 79–86.
11. Levsky, J. M. and Singer, R. H. (2003) Gene expression and the myth of the average cell. *Trends Cell Biol.* **13**, 4–6.
12. Muller, H., Michoux, N., Bandon, D., and Geissbuhler, A. (2004) A review of content-based image retrieval systems in medical applications – clinical benefits and future directions. *Int. J. Med. Inform.* **73**, 1–23.
13. Tieu, K. and Viola, P. (2004) Boosting image retrieval. *Int. J. Comput. Vis.* **56**, 17–36.
14. Jones, T. R., Carpenter, A. E., Sabatini, D. M., and Golland, P. (2006) Methods for high-content, high-throughput image-based cell screening, in *Proceedings of the Workshop on Microscopic Image Analysis with Applications in Biology (MIAAB)* (Metaxas, D. N., Rittcher, J., and Sebastian, T., eds.) Copenhagen, Denmark, pp. 65–72.
15. Jones, T. R., Carpenter, A. E., and Golland, P. (2005) Voronoi-based segmentation of cells on image manifolds, in Proceedings of the ICCV Workshop on Computer Vision for Biomedical Image Applications (CVBIA) (Liu, T. J., Zhang, C., eds.) Springer, Berlin, Germany, pp. 535–543.