Brief Overview

I wrote these scripts to find images I needed from EVOS scan protocols, which scan many extra images, and to detect clustered lipid droplets in adipocytes. I worked on these two steps separately, creating separate **well finder** and **adipocyte finder** programs that worked together to find images, detect droplets, and store data about percent covered by lipid droplets in an accessible format. **The entire process is shown in the video for Final Method and Final Detection.**

For general use of adipocyteFinder on a directory of pre-selected images, change the directory name in adipocyteFinder.py to your directory with images, and uncomment the code at the top and bottom that is not within functions. Use the return of processImage in recordData if you want to store percent covered.

For general use of wellfinder, you're all set if you're using EVOS scanning on 35 mm dishes. However, you may need to adjust find_regions()'s variables so that the image names are correct. The same goes for if you use a different image matrix (ie, for a 96 well plate).

For use of both programs together, use control_center.py. Change directories and just run. Make sure the directory you're using has the files wellfinder uses (total data images and all scanned individual regions)

These scripts are detailed heavily through their individual steps at the bottom of this document for further usage in adipocyte detection or just making general image finding easier in the lab (ie, finding specific images on more commonly used 96 well plates by clicking on total data images).

Detailed Well finder Information (brief overview, Video of Final Detection Method)

Actions

The program uses a path into a directory of images for one day and one full 35-mm dish. It finds all total data stitched images and uses OpenCV to open each as an array of pixels in the x and y direction. It registers mouse clicks at different x-y points. From a user-interface perspective, these clicks would be at different regions you'd want for future analysis. By taking in the prefix of the total data image and the x-y pixel location on the total data image, the program determines the number of the individual region in scan order, as well as its name. This name can then be sent to be opened in the adipocyte finder.

This program can also be used independently to find images more easily and efficiently, which is helpful especially when dealing with a lot of images that do not store helpful data (i.e. outside of the dish).

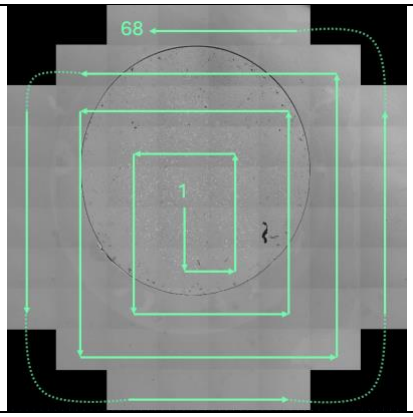
Methods

The well finder program finds individual regions from a total data image, as shown below (Figure 11A). EVOS scans in many individual regions per 6 mm well and the stitches them all in a total data image. Though useful as a visual, this total data image is not at a high enough resolution to detect individual cells and droplets. However, by considering each of the images in it as entries in a matrix, you can associate different regions of pixels on the total data image with spots in the matrix, based on their x-y pixel locations. The matrix below for ordering images was based on video footage of how images appear and save while scanning, for the 35-mm dish setup (Figure 11B). However, it'd be very simple to use another matrix for another dish setup (i.e., the more commonly used 48 and 96 well plate) and change parameters slightly in the program.

To get an individual region's save number, the mod function is used on a point's x-y pixel location / pixels per individual region's axis. This gives the x-y position in the scanning order matrix, at which location is the image number.

Figure 11: Using Total Data Images to Find Individual Regions

3A: Total Data Image and Scan Order



3B: Scanning Order Matrix, in Excel

	A	B	C	D	E	F	G	H
1								
2			68	67	66	65		
3		48	47	46	45	44	43	
4	49	25	24	23	22	21	42	64
5	50	26	9	8	7	20	41	63
6	51	27	10	1	6	19	40	62
7	52	28	11	2	5	18	39	61
8	53	29	12	3	4	17	38	60
9	54	30	13	14	15	16	37	59
10		31	32	33	34	35	36	
11			55	56	57	58		

Detailed Adipocyte finder information (brief overview, Video of Final Detection)

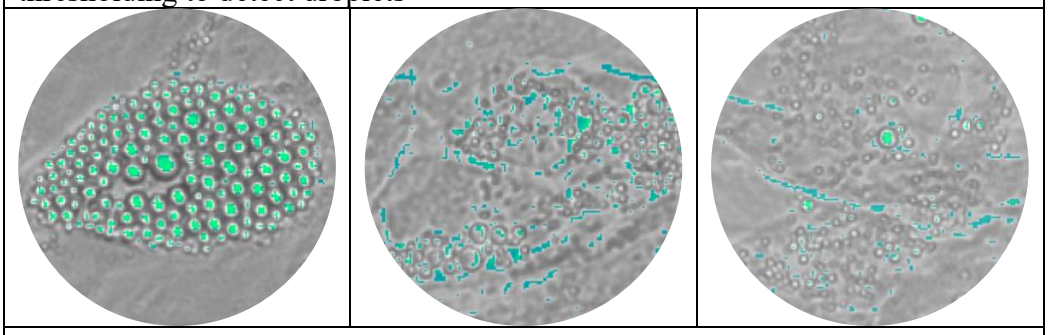
Actions

This program takes in an image name as input and uses several layers of detection filtering to find adipocytes with notable lipid droplets. Lipid droplets can vary significantly in size, shape, and darkness and still be detected well using the steps detailed in *Methods*. The program picks up droplets in clusters—ideal for finding mature adipocytes that are secreting significant amounts of lipid droplets and not finding tiny background droplets. It calculates percent covered by the lipid droplets detected on a given image, and then stores it in a csv file along with the image name, and the day, dish, and well number (all key identifying information) about the sample.

Methods – First Tests

The detection method used utilizes different scikit filtering and blob identification functions. Using these methods and not using machine learning in this case was beneficial, as I could tune the program's parameters with a lot more flexibility than training a machine learning model. I also did not have to make ground truth data. At first, I tried to just use thresholding with grey scale intensity, as the first many lipid droplets I saw were bright and had high contrast. This method worked initially, but after getting more data, I realized this method was not sufficient for also detecting low contrast or dark lipid droplets (Figure 4).

Figure 12. High, medium, and low contrast adipocyte images using thresholding to detect droplets



Description: These adipocytes with high, medium, and low contrast lighting show that thresholding only works well on high contrast images.

Methods – Detection using sci-kit

After testing thresholding alone, I made an accurate detection method using sci-kit blob detection functions—Determinant of Hessian (blob_doh), and Laplacian of Gaussian (blob_log). How these methods work is detailed on sci-kit's website. I tested these methods and found that for lipid droplets in adipocytes, blob_doh was good at finding general areas correctly, without a lot of noise. However, it didn't pick up on all lipid droplets within an adipocyte (low false positives, high false negatives). On the other hand, blob_log was much better at labelling all lipid droplets, but created a lot of noise (low false negatives, high false positives). I decided to apply these methods in succession. First, I removed most noise using blob_doh and to account for missed spots, I plotted areas detected at a bigger radius than found by the function. I then applied blob_log only on the gray scale regions found by blob_doh. This got rid of more noise and amplified regions with many lipid droplets. This approach was significantly better than thresholding alone and with much less noise.

Methods – Removing noise and selecting final area

The next step I took was to blur out the image with just blob_log labelling. This helped to remove any remaining noise and account for all area within an adipocyte that might not have been tagged by blob_doh (especially if it was too dark). I used scikit's gaussian filter to do this. This filter blurs nearby pixels and weighs a given pixel's contribution to another's based on how far apart they are (along a gaussian distribution). This helps create a blur amplifying droplets close together in clusters. Lastly, I thresholded the image to remove all background and just show regions over a certain cutoff. Then, my program calculated the area that was left over at the end of these processes and stored the value along with key image data.

Methods – Parameters for functions

Some of the parameters for these functions through this process were varied based on average greyscale intensity of the image. This was because blob detection functions still work best on high contrast bright blobs but with much more flexibility than thresholding alone, so sensitivity needed to increase on darker images.

Methods – Time and Automation

Since I was running these functions on high resolution images, their runtime was relatively high (20-40s per image, depending on how much is detected). However, the entire process was automated between clicking on images in wellfinder and getting final percent covered data. There was essentially no user interaction needed while the program runs, except at the end where all analyzed images display and the user verifies them for use. For running a 35 mm dish, it takes between 10-20 minutes for 4 individual image regions per well. Considering all analysis is also done in this time, this is much faster than fluorescent imaging and manual analysis in Image J. It also does not require staining which can complicate the health of cells used.