Area and Eccentricity of Fibroblasts During Trypsinization

Kyle Thomas

(Dated: May 19, 2015)

ABSTRACT

Area and eccentricity of mouse-embryo fibroblast cells are both measured as the cells release these cells release the surface of collagen gel. Image sets of the cells are collected at regular time-intervals during and after detachment. Software was written to segment selected cells and measure both the area and eccentricity of each segmented cell. As the main product of this work, the code is included in appendices as well as a link to all of the code.

EXPERIMENTATION METHODS

Mouse embryo fibroblasts exhibiting green fluorescent protein (cell line NIH3T3/GFP) are placed atop a collagen-gel and suspended in growth medium(see the diagram in Figure 1). The fibroblasts attach to the surface of the collagen at 37°C for 15 to 30 hours in a CO2-regulated incubator. The cells were not seen to undergo mitosis.

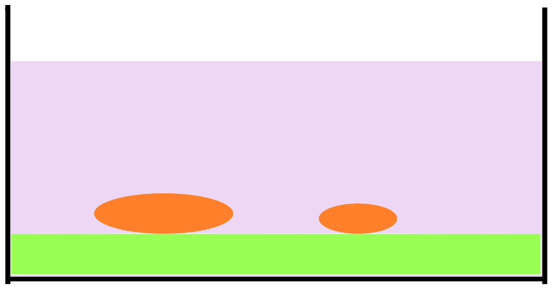
Cells are imaged every regularly beginning 40 minutes before trypsinization via TrypLE Select. The images are collected with a confocal microscope using a 70/30 transmission/reflection beam-splitter. The images are collected in image sets called z-stacks.

A z-stack is a sequence of images taken in quick succession. After the first, lowest image, each image is focused a few microns higher than the preceding image. Imaged in this way, z-stacks essentially provide a "snapshot" of the state of specific cells within the sample.

The stage is adjusted before imaging a sample so that the z-stacks image from ten microns below the surface of the collagen to ten microns above the surface of the collagen. Ten microns is approximately the radius of these cells when not attached to the collagen. Z-stacks are captured every five to fifteen minutes.

Imaging the cells occurs by irradiating the sample with a 488nm laser. The green fluorescent proteins fluoresce at a different wavelength than the 488nm excitation light, thereby distinguishing fluorescence from reflection. Light from 500nm to 550nm is observed for emissions of the GFP.

FIG. 1. A diagram of orange cells submerged in violet growth medium atop green collagen



The fibers were also observed for alignment phenomena which were not seen.To image the fibers, the sample is excited by a 532nm laser and wavelengths in the spectrum 531nm to 536nm are observed for reflection and transmission of the incident light off of the fibers. The observation spectrum imaged for the fibers was found by visually determing the spectrum in which the fibers appear most distinct.

Light detected from the fibers is maximized by observing a spectrum symmetric about the wavelength of the incident light, say 526nm to 536nm. However, such a spectrum has a higher signal to noise ratio because more of the light emitted by the GFP is detected below 531nm.

Both sets of images are acquired with a 30% reflection, 70% transmission beamsplitter. Using the same beamsplitter for both images means time is not spent changing the beamsplitter between images. This time savings between frames allows a z-stack to be captured closer to instantaneously. An instantaneous "snapshot" of the z-stack is ideal.

During imaging, the growth-medium surrounding the cells is rinsed with 1XPBS twice. The PBS is next replaced with TrypLE Select. The TrypLE Select breaks the proteins binding the cells to the collagen-matrix.

The proteins that bind the cells to the collagen are on bulges in the cell-membrane. These protrusions are called filopodia. After the bonds break, these filopodia flatten back into the cells.

This flattening of the philopodia leaves the cells somewhat spherical. The spherical fibroblasts appear as circles when seen through the microscope. While the cells return to a spherical shape, their lack of sphericilaty can be measured as eccentricity, as in the eccentricty of an ellipse.

IMAGE PROCESSING

A single experiment yields a folder containing all z-stacks acquired during that experiment. Each z-stack is comprised of several images, as described before. These images are processed to yeild numerical data.

The first step in processing the images from experiments is to transform each z-stack into single image. There is certainly data lost in the transformation of the z-stacks from several images to a single image. However, this experiment is only concerned with whether a cell is present at each pixel's location.

For this reason, a maximal intensity projection (MIP) is used to transform all images comprising a z-stack into a single image. An MIP creates a maximal intensity projection for each z-stack. This maximal intensity projection is a single image. The maximal intensity projection is made by using the value of the brightest pixel at position (x_1, y_1) chosen from among the values of each image's (x_1, y_1) pixel. An example of the MIP function is shown in Figure 2.

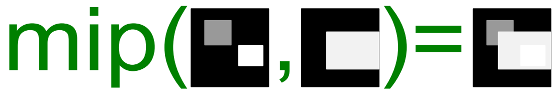


FIG. 2. An illustration of the MIP function creating a projection from two simple example images

The values of all pixels in the MIP-images is multiplied by 50, effectively brighening the images. This is done to facilitate masking and subsequently segmenting entire cells. Seemingly, reducing the brightening factor would reduce the noise at the edges of the segmented cells. If so, this would improve precision accuracy of the numerical data.

The Matlab Image Processing Toolbox function im2bw uses a brightness threshold to create a black and white image from a grayscale image. Pixels brighter than the threshold value are converted to white while other pixels are turned to black. The threshold used to create binary(black and white) images is 0.15.

The custom function maskmaker is used to draw a region loosely enclosing the cell at its most stretched out. Maskmaker sets pixels outside this region to black. This creates a mask loosely surrounding the cell of interest. Multiplying this mask leaves the user-selected cell as the largest object in the resulting masked binary image.

In this way, the function stackmasker applies this mask to all images in the stack. All masked images contain the entire cell of interest. The mask is drawn around the cell when the cells are stretched out to ensure the entire cell is visible in later images.

Masking works by ensuring that the cell of interest is the largest object shown. The reason a mask is made enclosing a cell is so that the user can specify that cell for software to segment precisely. Cells are precisely outlined by software to save time and to ensure uniformity in tracing the cell.

Many experiments show multiple cells in each image. For example, consider a fictitious image that contains cells A and B. Each cell's shape and eccentricity can be measured as follows.

Cell A is loosely masked by the user so that it's the largest object visible. Cell A is segmented in the images. These images showing only cell A are saved. The area and eccentricity of cell A is measured from cell A's segmented images.

A cell is measured by segmenting a set of images for that cell only. The cell is segmented by ensuring it is the only object visible in its image-set. This is why masking needs to specify a single cell as it does.

After cell A is segmented, the user can draw a region containing all of cell B as the largest object in the image-sequence. Subsequently, cell B is masked and segmented. In this way, information about each cell in an image can separately gathered from a single experiment's images.

In figure 3, the two images shown are the initial image and the final image from the same experiment. Figure **??** and figure 4 show the region of interest outlined in figure 3. The image-processing procedure is summarized in figure 4.

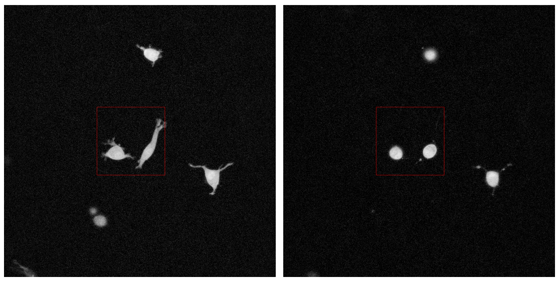


FIG. 3. MIP of first and last z-stacks from an experiment with the region of interest outlined in red

CONCLUSIONS

The plotted numerical data in figure 5 shows an inverse relation between area and time. There also appears to be an inverse relation between eccentricity and time. Analyis of data from further experiments might allow one to model size and eccentricity of cells as they release a substrate.

Acquiring images more frequently would likely increase smoothness of the plotted data. Seemingly, data collected would be more accurate using a taller z-stack. This is because there are single data points that fail to follow the decreasing trend of their data sets.

The accuracy of the numerical data could be improved by minimizing the noise introduced when transforming the greyscale images into binary images. This noise could be reduced by adjusting the brightening factor and adjusting the threshold used to create the binary images. For this, one could maximize some function that increases appropriately with area while decreasing appropriately with roughness.

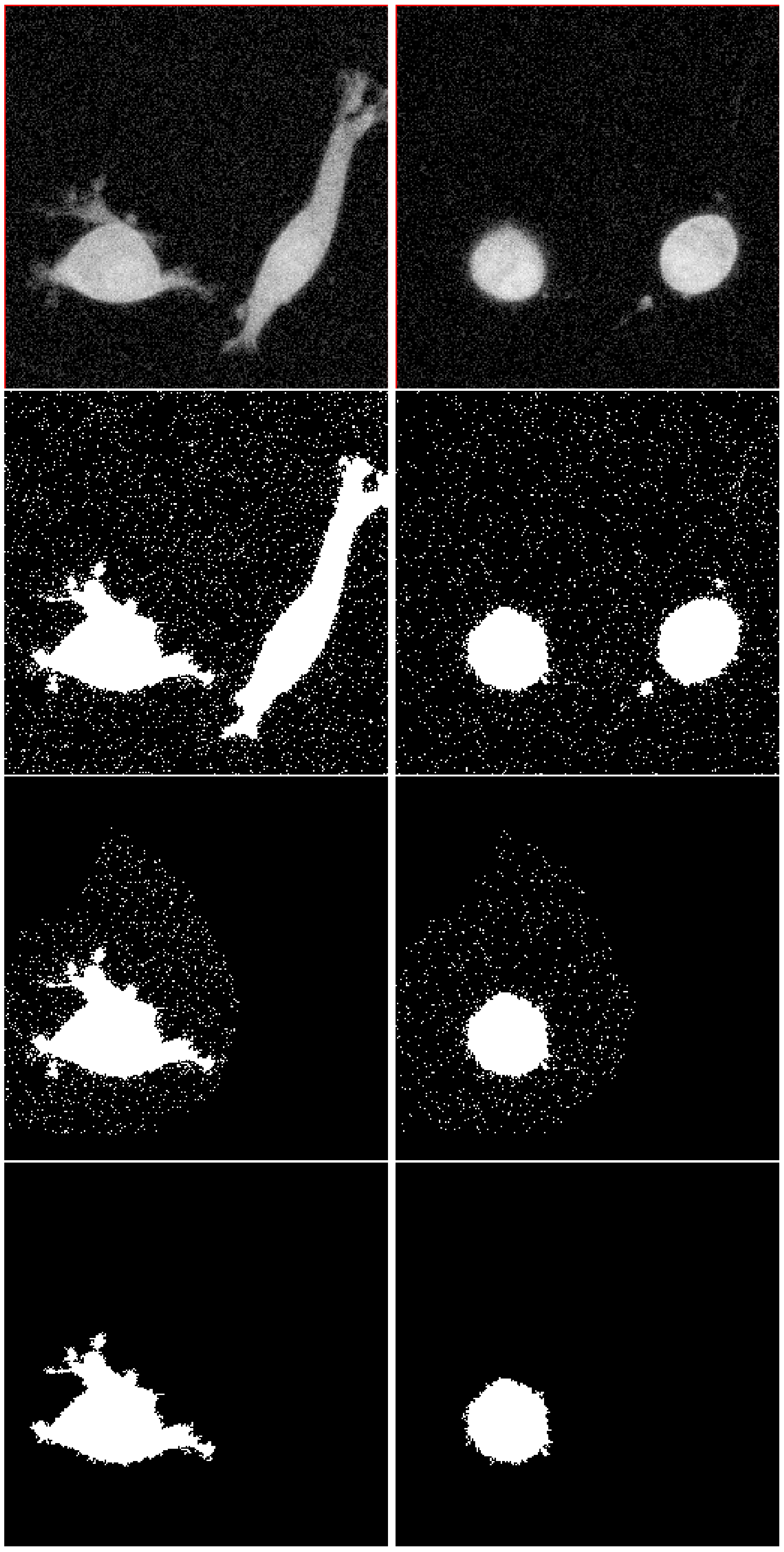
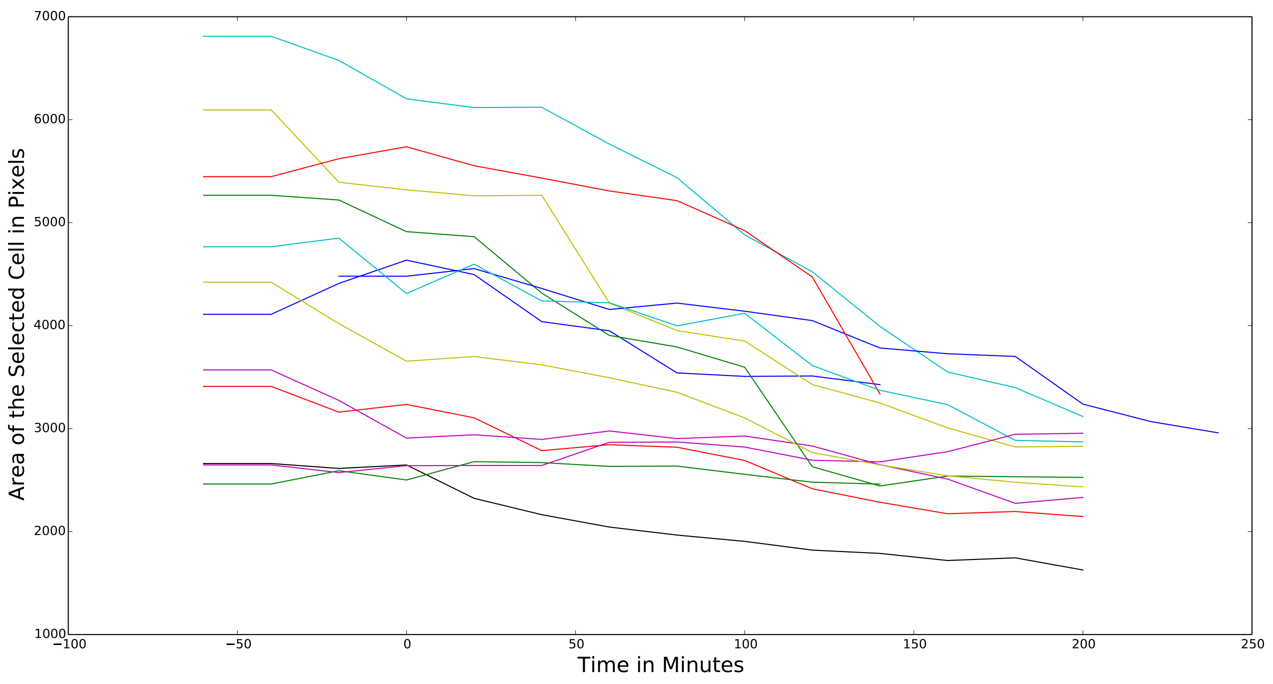


FIG. 4. Region of interest from FIG. 3 at several stages throughout image processing; from top to bottom, the rows show the original images, the result of 'binarization', the result of masking, and the segmented cell from the images.



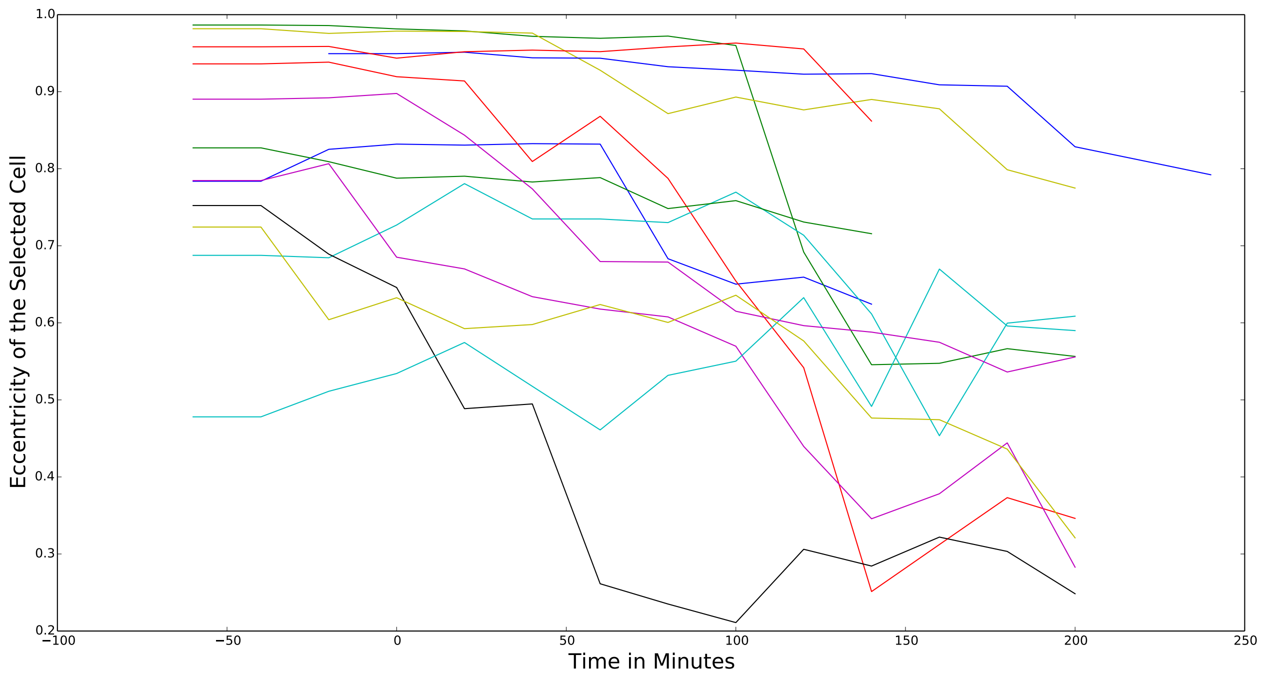


FIG. 5. Plots of experimental data: Area Vs. Time and Eccentricity Vs. Time

CODE TO BE USED AND DISTRIBUTED FREELY

```
% this script segments each object of a tiff-stack into its own stack
% by asking the user to mask a region constaining only one object.
% user decides how much to brighten and threshold the image for
% segmentation. Put all files for each sequence of mip images into
% a separate folder within "directory"

% directory contains folders which each contain an image sequence:
directory = '../../data/0-good-images-no-beads';
% where to put output files:
outputPath = fullfile(directory,'..','3-segmented');
% senesitivity for segmentation:
threshold = 0.15;
% brighten by this factor by default:
defaultBrighteningFactor = 15;
% get folders or file-sets containing stacks:
folders = ls(directory);folders = folders(3:end,:);
% save all intermediate stacks? (true or false)
saveIntermediateStacks = true;
% attempt to resume from the folder that the program left off at
if exist('k','var')
    resume = input(sprintf(['Enter the folder number to start'...
        ' with or press enter to attempt\nresuming where the'...
        ' program left off.\n']));
    if isempty(resume)
        startingfolder = k;
    else
        startingfolder = resume;
    end
else
    startingfolder = 1;
end

for k=startingfolder:length(folders(:,1))
    % read all folders in directory and skip folders with no tiffs
    stackDirectory = fullfile(directory,strtrim(folders(k,:)));
    if size(dir(fullfile(stackDirectory,'*.tif')),1)==0 &&...
            isdir(stackDirectory)
        continue
    end

    % read stack in current folder
    disp(['FOLDER #' num2str(k) ': ' folders(k,:)]);
    stack = stackreader(stackDirectory,'*.tif');

    % loop through binarizing stack until ok
    if exist('brightenedStack','var')
        clear('brightenedStack');
    end
    ok = 0;
    while ~isempty(ok)

        % get brightening factor from user or use default
        if ok ~= 0
            brighteningFactor = ok;
        end
        if ~exist('brighteningFactor','var')
            brighteningFactor = defaultBrighteningFactor;
        end

        % brighten, binarize, and fill the stack
        bwStack = zeros(size(stack));
        for page=1:size(stack,3)
            bwStack(:,:,page) =...
              imfill(im2bw(brighteningFactor*stack(:,:,page)),...
                     'holes');
        end

        % show stack and ask if ok, not ok => redo w/ new parameters
        implay(bwStack);
        ok = input(['PRESS ENTER IF B&W STACK IS OK OR '...
                    'ENTER NEW BRIGHTENING FACTOR: ']);
    end

    % write the binary stack if saveIntermediateStacks is true
    if saveIntermediateStacks
        sprintf(['SAVING BINARIZED STACK IN\n',...
            fullfile(outputPath,'..','1-binarized')])
        stackwriter(bwStack,fullfile(directory,'..','1-binarized'),...
            ['binarized-' folders(k,:)],'','');
    end

    % let user mask region containing mth object and decide whether to
    %  save the selected object's stack

    % ask how many objects in the current stack will be masked
    numobjects = input(sprintf(['How many objects will be'...
        ' segmented in this stack?\n']));

    % for each object, let user create mask
    for m = 1:numobjects
        disp(['MASKING OBJECT #' num2str(m)...
              ' IN STACK ' folders(k,:)]);
        mask = maskmaker(bwStack);
        while 1==1
            appendObjectToFilename = ['object-' num2str(m)];
            maskedstack = stackmasker(bwStack,mask,threshold);
            % write the masked stack if saveIntermediateStacks is true
            if saveIntermediateStacks
                sprintf(['SAVING MASKED STACK IN\n',...
                    fullfile(directory,'..','2-masked')])
                stackwriter(maskedstack,fullfile(directory,...
                    '..','2-masked'),['masked-' folders(k,:)],...
                    '',appendObjectToFilename);
            end
            segmentedstack = maskedstack;
            for page=1:size(maskedstack,3)
                segmentedstack(:,:,page) =...
                    ExtractNLargestBlobs(maskedstack(:,:,page),1);
            end

            % show segmented stack. ask if ok...
            % if ok => save single-object-stack
            implay(segmentedstack);
            ok = input(sprintf(['Either enter 1 to save'...
                ' the segmented stack, press enter to skip,'...
                '\nor enter anything else to tinker.\n']));
            if ok == 1
                sprintf(['WRITING SEGMENTED STACK\n:' folders(k,:) '.tif']);
                stackwriter(segmentedstack,outputPath,...
                    ['segmented-' folders(k,:)],'',...
                    appendObjectToFilename);
                sprintf(['SUCCESSFULLY WROTE THE STACK NAMED\n'...
                    'segmented-', folders(k,:),appendObjectToFilename,...
                    '.tif'])
                break
            elseif isempty(ok)
                break
            else
                tinker
            end
        end
            continue
    end
end
```

Listing 1. segmenter.m