Area and Eccentricity of Fibroblasts During Trypsinization

Kyle Thomas
(Dated: May 16, 2015)

Area and eccentricity of mouse-embryo fibroblast cells are both measured as the cells release the top 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. Some 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.

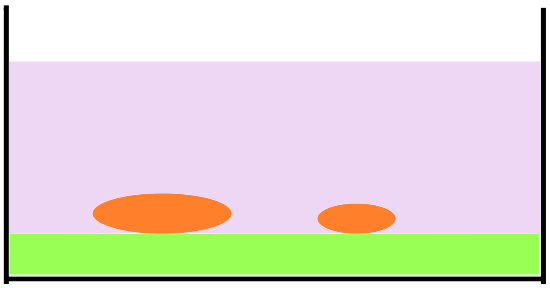
Cells are imaged every regularly beginning 40 minutes before trypsinization via TrypLE Select. The images are collected with a confocal microscope using a 70/30 transmission/reflection beam-splitter. The images are collected in image sets called z-stacks.

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. Orange cells submerged in violet growth medium atop green collagen are diagrammed.



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 produces a folder containing all the z-stacks acquired during the experiment. Each z-stack is comprised of several images, as described before.

The image files for a z-stack are combined into a single TIFF using Matlab. A maximal intensity projection is used to transform all images comprising a z-stack into a single image.

The Matlab function 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) of every image in the image's z-stack. An example MIP-creation process is shown in Figure 2.

FIG. 2. The mip function is illustrated projecting simple example images.

The function maskmaker is used to draw a region loosely enclosing a single cell at its most stretched out. Maskmaker sets pixels outside the region to black. This creates a mask loosely surrounding the cell of interest. Applying this mask leaves the user-selected cell as the largest object in the image so that the segmentation of the selected cell can proceed.

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 filopodia are stretched out to ensure the whole cell is visible in later images as the cell's filopodia recede.

Masking works by ensuring that the cell of interest is the largest object shown. A mask is made loosely enclosing a single cell so that the user can specify a certain cell for software to outline precisely. Cells are precisely outlined by software to save time and to ensure uniformity in drawing outlines.

Filling the objects in the image proceeds as follows. The brightness of each masked image is multiplied by a constant factor. This factor is 15 by default for the trypsinization images. Some z-stacks were brightened by a factor of more than 15 to facilitate masking entire cells.

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.

Several experiments show multiple cells in each image. For example, consider an image that contains cells A and B. Each cell's shape and eccentricity can be measured separately.

A cell is measured by creating a separate binary set of images for that cell. The cell is segmented by ensuring it is the only object visible in its image-set.

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 these images. This process is repeated to measure cell B.

Image segmentation yields multipage-tiff-files, where each page is a 1024 pixel by 1024 pixel image. All images in a file show only the same particular cell. The images are taken 20 minutes apart so that each file watches a different cell as it releases the collagen substrate.

TODO:

1. write a conclusion
2. write acknowledgements
3. discuss binarizing giving noise b/c thresholding not optimal

CONCLUSIONS

ACKNOWLEGEMENTS

REFERENCES

APPENDICES

Appendix A: Experiment Protocol

The protocol for three collagen-samples of concentration 1.5 and gelled at 21°C is as follows:

1. Create the collagen-solution.
   (a) Total volume to be made is V = (90μL per sample)(3 samples)(150%) = 405μL.
   (b) Combine 40.5μL 10X PBS, 14.2μL NaOH, and 276.3μL growth-medium.
   (c) Shake this solution at 400rpm for 20 seconds.
   (d) Store at 6°C for 10 minutes.
   (e) Add 71μL collagen homogeneously.
   (f) Stir collagen into solution with a 100μL pipette tip.
   (g) Pipette 90μL collagen-solution into each of the 3 samples a, b, and c.
2. Wrap each sample in film and let gelate at 21°C for two hours.
3. Place growth-medium on all samples and place cells on sample a.
4. Place cells on sample b four hours later.
5. Place cells on sample c four hours later.
6. Using confocal fluorescence microscopy, capture a 10μm tall z-stack of images of fibers and cells, separately, every 20 minutes for four hours.
   (a) After the first z-stack, replace the growth-medium on the sample with 1X PBS.
   (b) After the second z-stack, replace the PBS with fresh PBS.
   (c) After the third z-stack, replace the PBS with 1X TrypLe^{TM} Select.
7. When sample a is finished imaging, image sample b in the same way as sample a was imaged.
8. When sample b is finished imaging, image sample c in the same way as sample a was imaged.

Appendix B: Code

```
% 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

```
folder = '../../data/3-segmented';
files = ls(fullfile(folder,'*.tif'));
for file = 1:size(files,1)

    % ask for time interval between z-stack acquisition
    stack = stackreader(folder,files(file,:));

    % get time, area, and eccetricity values of object for each page
    disp('############################################################');
    disp(['PROCESSING SEGMENTED OBJECT #' num2str(file) ':']);
    disp( files(file,:) );
    disp('############################################################');
    timeInterval = input('\nTIME BETWEEN Z-STACKS: ');
    timeOffset = input(['NUMBER OF Z-STACKS BEFORE '...
                        'TRYPLE SELECT WAS ADDED: ']);
    time         = zeros(size(stack,3),1);
    area         = zeros(size(stack,3),1);
    eccentricity = zeros(size(stack,3),1);
    for page = 1:size(stack,3)
        time(page) = (page - timeOffset - 1)*timeInterval
        info = regionprops(stack(:,:,page),'Area','Eccentricity');
        for region = 1:size(info,1)
            if info(region).Area ~= 0
                area(page) = info(region).Area
            end
            if info(region).Eccentricity ~= 0
                eccentricity(page) = info(region).Eccentricity
            end
        end
    end
    % store time, area, and eccetricity values of object for each page
    data = cat(2,time,cat(2,area,eccentricity));

    % write time, area, and eccentricity to csv
    csvFilename = fullfile(folder,[files(file,:) '.csv']);
    dlmwrite(csvFilename,data,'newline','pc')
    disp(sprintf(['WROTE DATA TO FILE ' ]));

end
```

Listing 2. regionPropsScript.m

```
% creates separate scatter plots of data from csv

% initialize variables
csvDirectory = '../../data/3-segmented';
csvFiles = ls(fullfile(csvDirectory,'*.csv'));

% initialize figures and set each to hold
eccentricityDotPlot = figure('Name',['Relative Change in'...
                                     'Eccentricity vs. Time'],...
                             'NumberTitle','off');
hold all;
eccentricityPlot = figure('Name','Eccentricity vs. Time',...
                          'NumberTitle','off');
hold all;
areaDotPlot = figure('Name','Relative Change in Area vs. Time',...
                     'NumberTitle','off');
hold all;
areaPlot = figure('Name','Area vs. Time','NumberTitle','off');
hold all;

for csvFileIndex = 1:size(csvFiles,1)
    csvFile         = csvFiles(csvFileIndex,:);
    csvData         = csvread(fullfile(csvDirectory,csvFile));

    time            = csvData(:,1);
    area            = csvData(:,2);
    eccentricity    = csvData(:,3);

    % calculate the time derivative of relative area
    areaDot = zeros(length(area)-1,1);
    for areaDotIndex = 1:length(areaDot)
        areaDot(areaDotIndex) =...
            (area(areaDotIndex+1)-area(areaDotIndex));%/...
%             area(1);
    end

    % calculate the time derivative of relative eccentrivity
    eccentricityDot = zeros(length(eccentricity)-1,1);
    for eccentricityDotIndex = 1:length(eccentricityDot)
        eccentricityDot(eccentricityDotIndex) =...
            (eccentricity(eccentricityDotIndex+1)...
             -eccentricity(eccentricityDotIndex));%/...
%             /eccentricity(1);
    end


    figure(areaPlot);
    scatter(time,area,'+');
    figure(eccentricityPlot);
    scatter(time,eccentricity,'*');
    figure(areaDotPlot);
    scatter(time(1:end-1),areaDot,'.');
    figure(eccentricityDotPlot);
    scatter(time(1:end-1),eccentricityDot,'x');
end
```

Listing 3. plotterScript.m