Area and Eccentricity of Fibroblasts During Trypsinization

Kyle Thomas

September 21, 2014

Abstract

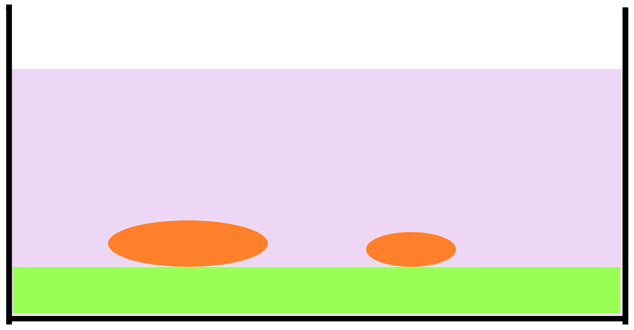
Area and eccentricity of mouse-embryo fibroblast cells are measured as they release the top surface of collagen gel. Image sets of the cells are collected at regular time-intervals as the cells detach. Matlab software was written for measuring the area and eccentricity of each selected cell. The code is included in this paper as well as a link to 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 cells 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 mitos++is.

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

Figure 1: Orange cells submerged in violet growth medium atop green collagen are diagrammed.



Each z-stack is a set of images at depths varying by a few microns from ten microns below the surface of the collagen to ten microns above the surface of the collagen. Note that ten microns is the approximately the radius of the subject cells when not attached. Z-stacks are captured every five to fifteen minutes.

When capturing images of the cells, the sample is excited by a 488nm laser. Light from 500nm to 550nm is observed for emissions of the GFP.

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 protrusions called filopodia. After the bonds break, the filopodia recede back into the cells. This trypsinization leaves the cells in a spherical shape. The spherical fibroblasts appear as circles through the microscope.

Image Processing Methods

Each experiment produces a folder containing an image file for every depth in every z-stack. Image files for each z-stack are combined into a single TIFF file using Matlab. Maximal intensity projection is the method used on z-stacks to map several images into a single image.

The Matlab function mip creates a maximal intensity projection for each z-stack. A 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) from every image in the z-stack.

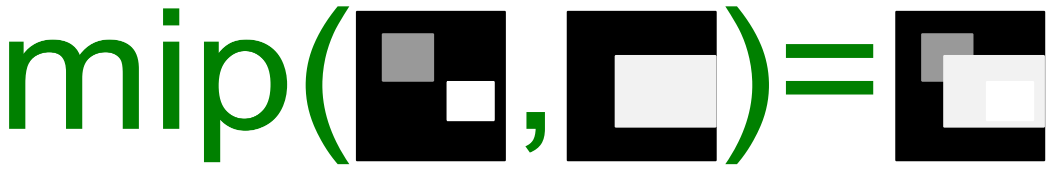


Figure 2: The mip function is illustrated projecting simple example images.

The function maskmaker is used to draw a region loosely enclosing a single cell at its most stretched out. Pixels outside the region are set to black. This creates a mask loosely surrounding the cell of interest.

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 greyscale 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.

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.

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$^{\text{TM}}$ 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

Listing 1: segmenter.m

```
% this script segments each object by letting the user mask a
% region loosely constaining a cell as the largest object. User then
% specifies how much to brighten and threshold image for segmentation.

%###########################################################################
%### CAUTION: CLEAR COMMAND WILL CLEAR VARIABLES UNLESS COMMENTED ###%
%###########################################################################
% comment the clear command below to enable resuming from folder
% clear

%% parameters
% save intermediary images?
saveIntermediaryImages = true;
% start at image-set # resumeAtFolder in alphabetized image-sets
if ~exist('resumeAtFolder','var')
    resumeAtFolder = 1;
end
% data_directory is where folders containing image-sets are
dataPath = ['C:\Users\thomasky\Research\2D-fibertraction\'...
    '_prep-to-mask-and-binarize\good-images-no-beads'];
% string to match all input images (e.g. *.tif)
imageType = 'tif';
% binarization threshold between zero and one
threshold = 0.15;
% brighten images to adjust how images are binarized
brighteningFactor = 40;
% folders is a list of folders containing image-sets
folders = ls(dataPath);

% define output directories
outputPath = fullfile(dataPath, '..','..','segmenter-output');
binaryOutputPath = fullfile( outputPath, '1binary' );
maskOutputPath = fullfile( outputPath, 'masks' );
maskedOutputPath = fullfile( outputPath, '2masked' );
```

```
segmentedOutputPath = fullfile( outputPath,'3segmented');

% create necessary directories for output if they don't exist
if ~isdir( outputPath )
    mkdir( outputPath );
end
if ~isdir( fullfile( outputPath, '1binary'))
    mkdir( fullfile( outputPath, '1binary'));
end
if ~isdir( fullfile( outputPath, 'masks'))
    mkdir( fullfile( outputPath, 'masks'));
end
if ~isdir( fullfile( outputPath, '2masked'))
    mkdir( fullfile( outputPath, '2masked'));
end
segmentedOutputPath = fullfile( outputPath,'3segmented');
if ~isdir( fullfile( outputPath, '3segmented'))
    mkdir( fullfile( outputPath, '3segmented'));
end

% image-sets only found in first-level subdirectories of dataPath
indices = [];
for k = 1:size(folders,1)
    if ~strcmp(strtrim(folders(k,:)),'.') &&...
    ~strcmp(strtrim(folders(k,:)),'..') &&...
    isdir(fullfile(dataPath,folders(k,:)))
        indices = [indices k];
    end
end
folders = folders(indices,:);

% write csv: folder #, data folder name
filename = fullfile( outputPath, 'folder-number-to-folder-name.csv');
for file = 1:length(folders(:,1))
    dlmwrite(filename, [file, strtrim(folders(file,:))], '-append');
end

%#### loop thru image-set folders from where program left off ###%
```

```
for k=resumeAtFolder:length(folders(:,1))


    % read all folders in directory and skip folders with no .tif's
    imageDirectory = fullfile(dataPath,strtrim(folders(k,:)));
    if isdir(imageDirectory) &&...
       size(dir(fullfile(imageDirectory, ['*.' imageType])),1)==0
        disp(['SKIPPING␣SUBDIRECTORY:␣' imageDirectory])
        continue
    else
        disp('PROCESSING␣IMAGE-DIRECTORY:␣')
        disp(imageDirectory)
    end

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

    % 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
    % save binary image-set
    if saveIntermediaryImages
        stackwriter(bwStack,binaryOutputPath,...
                    ['data-set-' num2str(k) '.' imageType])
    end

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

    % let user count objects in image-set to be segmented
    disp('Please␣count␣the␣number␣of␣objects␣to␣be␣segmented.');
    implay(bwStack);
    % ask how many objects in the image-set will be segmented
    numobjects = input(sprintf(['How many objects will be'...
                                 ' segmented in this set of'...
                                 ' images?\n']));

    % let user create mask for each object
    for m = 1:numobjects
        disp(['MASKING OBJECT ' num2str(m)...
              ' IN DATA-SET ' num2str(k) ':' ]);
        disp(folders(k,:));
        maskName = [ 'data-set-' num2str(k) '-mask-for-obj-'...
                     num2str(m) '.' imageType];
        if exist( fullfile(outputPath, 'masks', maskName),'file')
            mask = imread(fullfile(outputPath, 'masks', maskName));
        else
            mask = maskmaker(bwStack);
        end
        % save mask
        imwrite( mask, fullfile(outputPath, 'masks', maskName));
        maskedStack = stackmasker(bwStack,mask,threshold);
        % save masked images
        if saveIntermediaryImages
            stackwriter( maskedStack,maskedOutputPath,...
                         [ 'data-set-' num2str(k) '-masked-obj-'...
                           num2str(m) '.' imageType]);
        end

        % segment largest object in each masked binary image
        segmentedStack = zeros(size(maskedStack));
        for page=1:size(segmentedStack,3)
            segmentedStack(:,:,page) =...
                ExtractNLargestBlobs(maskedStack(:,:,page),1);
        end

        % show segmented images
%           implay(segmentedStack);

        % maybe save segmented images
        if saveIntermediaryImages
```

```
            stackwriter( segmentedStack, segmentedOutputPath,...
                         [ 'data-set-' num2str(k) '-segmented-obj-'...
                           num2str(m) '.' imageType]);
        end
    end
    % increment the folder number so that the process may be resumed
    resumeAtFolder = resumeAtFolder + 1;
end
```

Listing 2: regionPropsScript.m

```
folder = uigetdir(pwd, 'Select␣Folder␣Containing␣.tif␣Files')
outputPath = uigetdir(pwd, 'Select␣outputPath')
files = ls(fullfile(folder,'*.tif'));
defaultTimeBetweenZStacks = 20;
defaultZStacksBeforeTrypleSelect = 2;
% check whether user wants to enter custom time-intervals and -offsets
isUserEnteringCustomTimes = input(['Enter␣zero␣or␣nothing␣to␣'...
                                   'use␣default␣imaging␣times'])
if ~isUserEnteringCustomTimes
    timeInterval = defaultTimeBetweenZStacks
    timeOffset = defaultZStacksBeforeTrypleSelect
end
% loop through images collecting regionprop data into .csv-files
for file = 1:size(files,1)

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

    % get time, area, and eccetricity values of object for each page
    disp('############################################################')
    disp(['PROCESSING␣FOLDER␣#' num2str(file) ':'])
    disp(files(file,:))
    disp('############################################################')
    if isUserEnteringCustomTimes
        timeInterval = input('\n␣ENTER␣TIME␣BETWEEN␣Z-STACKS:␣');
        timeOffset = input(['ENTER␣NUMBER␣OF␣Z-STACKS␣BEFORE␣'...
                            'TRYPLE␣SELECT␣WAS␣ADDED:␣']);
    end
    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(outputPath,[files(file,:) '.csv']);
    dlmwrite(csvFilename,data,'newline','pc')
    disp('WROTE DATA TO FILE:')
    disp(csvFilename)

end
```

Listing 3: plotterScript.m

```
% creates separate scatter plots of data from .csv-files

% initialize variables
csvDirectory = uigetdir
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
```