Diameter Mapping of Intestine Over Time.

CALM Facility –Todd Fallesen

Abstract

The idea of this project is to map the diameter of an intestine over time, to see peristaltic movements. The intestines are oriented such that the long axis of the intestine is imaged on the horizontal axis. Under normal peristaltic motion the intestine will expand and contact, giving rise to local increases and decreases in intestine diameter, as measured on the short axis. This program presents the diameter of the intestine over time in what is referred to here on as a *Kymograph*. In our kymograph the y-axis represents time, the x-axis represents distance along the intestine while the gray value represents the diameter of the intestine at that point along the long axis of the intestine. This program is presented both with a GUI and a MATLAB script which can be used for analyzing multiple movies.

Note: Installation notes are at the end of this document

1 Part 1- Understanding the Result

The main result of this program is the *Kymograph*. The kymograph is slightly difficult to conceptualize, though we will attempt here. The intestine of interest is mounted horizontally, with the long axis in the horizontal 1. This long axis of the intestine is the X-axis of the kymograph result.

Now imagine putting a ruler on the very edge of the intestine along the short axis and measuring the diameter of the intestine. If we do this, we get a number that we then write down. We then move one pixel over along the long-axis of the intestine, measure the diameter again, and write that number down. We can keep doing this along the entire length of the intestine we end up with a set of numbers that has the same length as the number of pixels as the length of the intestine. We can visualize these numbers by displaying them as gray scale, the larger the number, the whiter the value. Thus zero width would be black, and the largest width would be white.

In our example above, we now have a single line which represents the diameter of the intestine along the entire length of the intestine for one image. If we repeat this process for every frame in a movie, and line them up underneath each other, we get a kymograph where the x position represents the distance along the intestine, the y

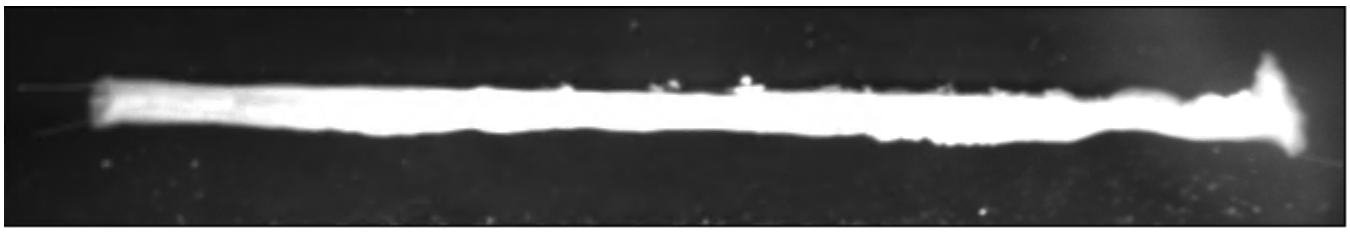

Figure 1: A single frame of an intestine movie. The width of the intestine along the long axis is represented as gray values along a single row in the kymograph.

value corresponds the frame number, and the gray value of the pixel corresponds to the diameter of the pixel at that position in time and space, see figure 2).

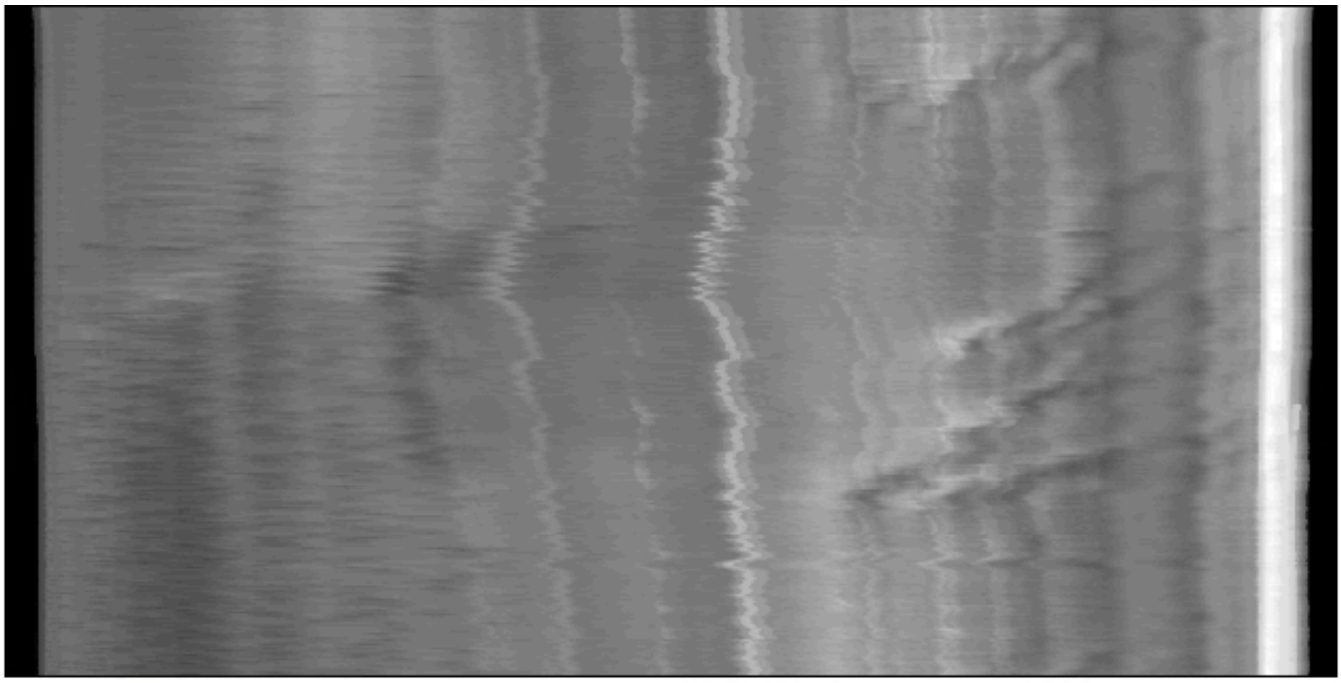

Figure 2: A kymograph representation of the width of an intestine over time. The width of the kymograph corresponds to the width of the intestine, and time progresses forward down the y-axis, that is, the lower on the kymograph you are, the later the time point. In our image, the origin $(0, 0)$ is in the upper left hand corner.

2 Part 2- Running the Graphical User Interface

2.1 Starting the GUI

When you open the GUI, you will be presented with a blank screen with three windows, *Original Image*, *Threshold Image*, and *Kymograph*. The Original image will be a frame of the movie, before thresholding. The thresholded image will be the image after thresholding (which can be adjusted) and finally, the kymograph window will not have anything in it until parameters have been chosen and the button *Make Kymo* is pressed. See figure 3

2.2 Loading the movie

Click on the *Load Image* button to choose an movie to load. This movie must be of *tif* format. A dialog box will appear to ask you what file to chose. Once you have selected the file and clicked *OK* another dialog box will appear while loading the movie. Once the movie is loaded, the first frame of the movie will appear in the window titled **Original Image**. Next to the original image will be a thresholded image of the intestine at that same frame, in the window titled **Threshold Image**. When the movie is first loaded, the default threshold is 0.5. By default, only the largest object is displayed in the **Threshold Image** window, which should be the intestine (see figure 4).

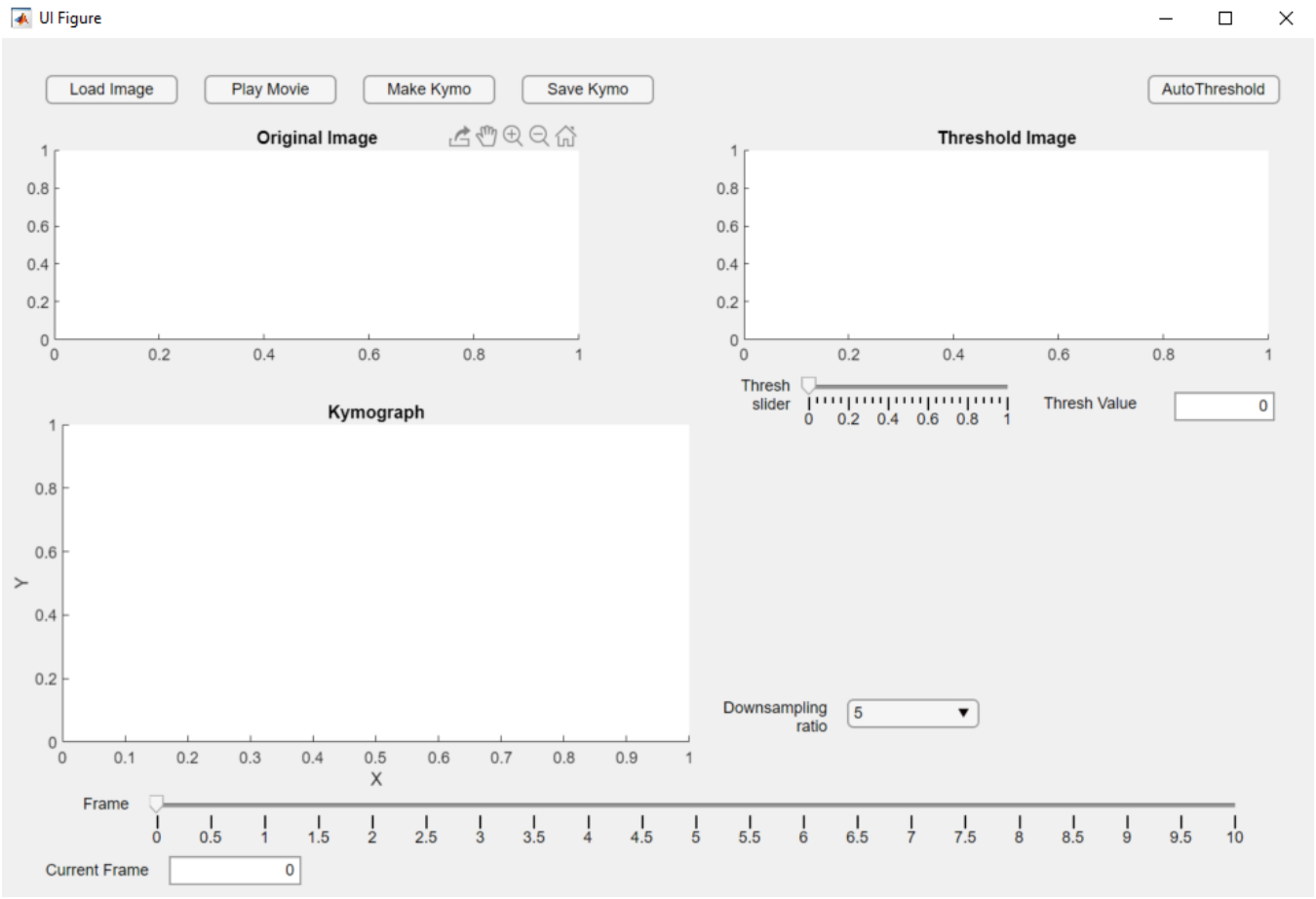

Figure 3: Opening Screen of app before any parameters have been chosen.

2.3 Thresholding

In the **Threshold Window** the intestine appears as a binary image, where the intestine is all white, the background black. Changing the threshold value will change how much of the image is seen as intestine.

There is a green border along the intestine showing the extent of what is detected as intestine. The red box outside the intestine is a *bounding box* which gives a 10 pixel border around the furthest extents of the intestine (see figure 4). In the *kymograph* the width of the intestine is measured along the length of the red box, from the top of the green border to the bottom of the green border. The ten pixels on either side of the intestine between the green border and the red box are considered the 'buffer pixels' which will show as black edges along the length of the kymograph. When the kymograph is generated, the computer will calculate the green border and red bounding box for each frame of the movie, and measure the intestine using the furthest extents of the bounding boxes from the entire movie, i.e., starting at the left-most column ever used in a bounding box and stopping at the right-most column ever used in a bounding box.

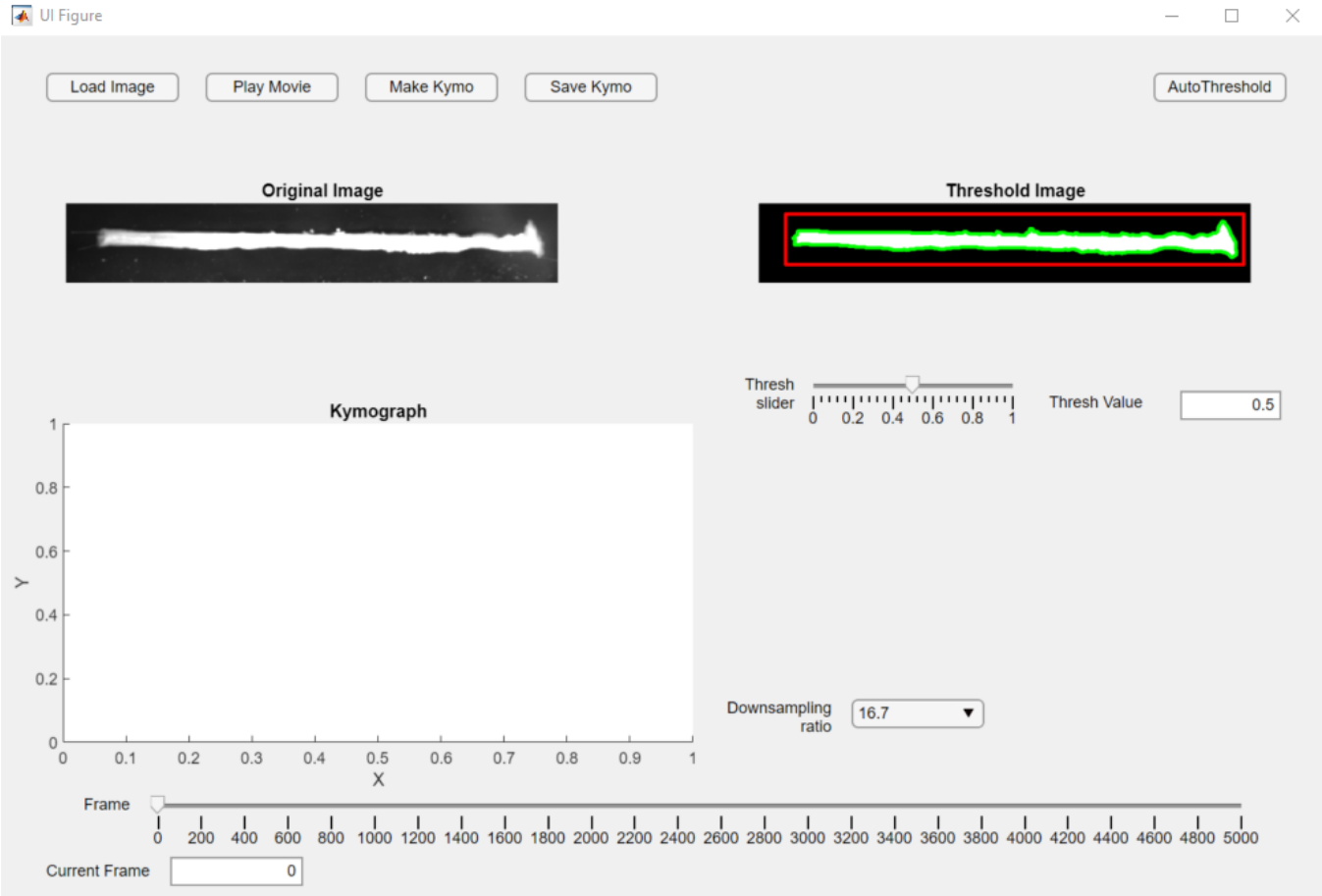

Figure 4: Screen of the app once an image has been loaded. A threshold of 0.5 has been applied automatically to begin, and has not yet been adjusted.

2.3.1 Autothreshold

Clicking the *AutoThreshold* button will find the optimal threshold for the current frame being displayed in the **Threshold Image** window. This calculated value is probably ok for the entire movie. Otsu's method is used for thresholding. For more details, see the section on the *Image_thresh* function.

2.3.2 Thresh slider

Moving the **Thresh Slider** allows you to see what the threshold image is at varying threshold levels. Changing this value will change the value in the **Thresh Value** text window.

2.3.3 Thresh Value Text Window

A value can be entered in this window to quickly evaluate what a specific value of threshold will look like. This may be useful if wishing to replicate previous results.

2.4 Evaluating over frames (Frame Slider)

The **Frame Slider** slider bar allows you to scroll through frames of the movie and see what the current threshold value will look like on those frames. Any threshold chosen will be used on all frames, so it is worth quickly checking to find a threshold value that works well over all frames. The slider bar will automatically click to nearest round number values. The **Current Frame** window shows which frame you are on.

2.5 Downsampling ratio

The full kymograph is as many pixels tall as there are frames in the movie. If there are 5000 frames in the movie, then the kymograph will be 5000 pixels tall, where each row of pixels corresponds to the diameter map for the corresponding frame. For ease of display and analysis, we block average pixels along the height of the kymoograph, that is, average the diameter maps for corresponding positions on the intestine over time. The **Downsamping ratio** is the number of rows that we block average. In the original program, that number was ≈ 16.7.

2.6 Making Kymo

Clicking on the **Make Kymo** button will use the currently selected threshold level and downsampling ratio to generate a Kymograph and display it in the **Kymograph** window. This will take a few moments to work, as the computer is calculating the border and diameter map for each frame of the movie. Changing the **Downsampling ratio** and clicking **Make Kymo** again does not recalculate the borders, and will be fairly quick after the initial kymograph is made. Changing the **Threshold** in any manner, and reclicking the **Make Kymo** button will force the recomputation of all borders and diameter maps, and will take a while.

2.7 Saving Kymo

Clicking the Save Kymo button will bring up a dialog box in which you choose where you want to save the resulting kymograph images. There will be three files saved, where *file_name* is the name you choose:

- file_name_*full.tif* : This is the full kymograph, without any downsampling
- file_name_*downsampled_**XXX**.tif* This image is the reduced kymograph. **XXX** is the integer used for downsampling
- file_name.*csv* This is the diameter map in *csv* form. Each row corresponds to the diameter map for the corresponding frame. The values at each index row i and column j correspond to the diameter at frame i and intestine position j.

As these values will be displayed as images, if the largest value in the diameter map is less than 255, the image is saved as an 8 bit image. If the largest diameter is greater than 255, the image is saved as a 16 bit image. As these animals are not elephants, there is no provision in the code to save as 32 bit images.

3 Part 3- Running the code through the MATLAB script

This kymograph making program can also be run as a simpler script without the GUI functionality. The advantage of running the program this way is that many intestines can be run, in order (concurrently, not simultaneously). The disadvantage is that you have to set the buffer, downsampling ratio before starting. By default, the threshold level is automatically calculated for each movie, but this can be overwritten in the code, to choose a specific threshold.

3.1 Setting file list

The file list is set in the cell variable *movie_list*. The movie_list is contained inside curly brackets, { }. The full path for each movie must be specified, and must be enclosed in single quotes ′ ′. Movies files need to be separated by commas, and if you wish to put the files on separate lines for ease of reading, there needs to be a three period ellipses (**...**) at the end of each line, except the last line. See see figure 5 as an example.

3.2 Setting Save Path

Change the path in the variable **save_directory** to the full path in which you want to save the result files. This needs to be enclosed in single quotes as well, see figure 5.

```
movie_list = {'F:\Projects\Sonia_Spitzer\141220_WT_control_II.tif',...
              'F:\Projects\Sonia_Spitzer\141220_WT_highNaCl_II.tif'};

save_directory = 'C:\path_to_save_in';
```

Figure 5: Example of movie list and save path. Formatting of the movie list must be kept as presented here to load properly.

Note. There will be three files saved per movie. By default, the file name of the movie will be used as the file name for the result, plus a suffix used to identify it. The three files will be:

- movie_file_name_full_kymo.tif: the full kymograph, without any reduction
- movie_file_name_XXX_reduced_kymo.tif: The reduced kymograph, where the XXX represents the downsampling ratio
- movie_file_name_full_table.csv : A csv file of the data from the full kymograph.

These files are the same that are saved from the GUI.

3.3 Setting Variables

The buffer and kymoograph downsampling ratio need to be set. These two variable are set before the main loop

3.3.1 Setting Buffer

The buffer is by default set at 10. Changing the variable **buffer** will change how much blank space is created between the edge of the intestine and the edge of the bounding boxes. This area will appear as black edges on the kymograph.

3.3.2 Setting Kymo Downsample

The kymograph downsample is the number of rows that are block averaged, columnwise, along the height of the kymograph. By default this is set at 16.7, but can be easily changed here. This number can be explored in the GUI to see what works best for your project.

The kymo_reduction number will be concatenated onto the end of the image name when saved, so you have a record of what was done.

3.3.3 Setting Threshold

By default, the threshold is automatically calculated using Otsu's method. This value is recalculated for each movie being processed. To change the threshold to make it constant in all movies, comment out the line.

$$level = graythresh(file(:,:,1));$$

and uncomment the line

$$\#level = 0.52;$$

and set the value to whatever you wish it to be, between $0-1$. A line is commented in MATLAB if it starts with a #. To uncomment a line, remove the # at the beginning of the line. The line will be green in MATLAB if it is commented out.

4 Part 4- Understanding the Underlying Functions

4.1 Movie_import

This function takes one input, *filename* and returns two parameters, *tif_movie* and *numframes*. This function reads in the movie file and generates a four dimensional matrix *tif_file*. The values of the indices of *tif_file(i,j,v,k)* are:

- i = row value
- j = column value
- v = gray value of pixel
- k = frame number of movie

This function will launch a dialog box to for the user to pick the file, and will also throw up a progress bar which iterates over the value $d.Value$. The math containing $d.Value$ is unneeded for the movie, and is only for the progress bar.

The size of the movie file is read at the beginning by the *info* function, which is used to determine the number of frames.

4.2 GUI ONLY: thresh_filt

The function *thresh_filt* is self contained inside the GUI. It is very similar to the other function *image_thresh*. *thresh_filt* uses two functions, *imbinarize* and *bwareafilt*.

imbinarize makes a binary image at whatever threshold level was selected in the program. If autothreshold was pressed, then the threshold selected will be given by the function *graythresh* which will automatically calculate the ideal threshold using Otsu's method.

bwareafilt removes all but the biggest object from the threshold image. This is needed if there are small specks in the image that make it through the thresholding.

As this function is contained inside the GUI app, all variables to and from it are declared as variables attached to the global *app* variable structure, and is beyond the scope of this manual to explain.

4.3 image_thresh

The function *image_thresh* is very similar to the other function *thresh_filt*. *image_thresh* needs two inputs, *image* and *level*, the movie file and threshold level, respectively. *image_thresh* returns on parameter, the thresholded image. *image_thresh* uses two functions, *imbinarize* and *bwareafilt*.

imbinarize makes a binary image at whatever threshold level was selected in the program. If autothreshold was pressed, then the threshold selected will be given by the function *graythresh* which will automatically calculate the ideal threshold using Otsu's method.

bwareafilt removes all but the biggest object from the threshold image. This is needed if there are small specks in the image that make it through the thresholding.

4.4 object_finder

The function *object_finder* is one of the workhorses of this program. It calculates the boundary around the object (green line in the Threshold Window) and the bounding box. This function takes two inputs, *image and buffer*. The image is a single thresholded frame of an image with only one object in it, and the buffer is the distance between the furthest extents of the threshold object and the bounding box. By default the buffer is 10 pixels.

object_finder calls the MATLAB function *bwboundaries* on the thresholded image. We find the furthest extents of the boundary, and pad 10 pixels from the furthest extents to find the bounding box. The bounding box is composed of four numbers, $bounding_box = [h1; w1; h2; w2]$ where h1 is the left short leg, h2 is the right short leg, w1 is the top long leg, w2 is the bottom long leg (h=height, w=width).

The function *object_finder* returns two parameters, the bounding box, which is the coordinates of the four corners of the box, and the boundary, which is the coordinates of all the pixels on the boundary of the thresholded object.

The function *object_finder* is called by the function *get_boundaries* on each frame of the movie, to get biggest boundary box, and make a multi-dimensional matrix of all the boundaries for each frame in the movie.

4.5 get_boundaries

The function *get_boundaries* takes three inputs and gives two output parameters. The input parameters are *movie_tif, buffer, level*, which are the movie from the *movie_import* function, the buffer number of pixels (default 10), and the threshold level, respectively. The two outputs is the set of *all_boxes* which all of the bounding boxes (seen in red in the Threshold Image window) for the entire movie, and *all_boundaries* for the entire movie (seen in green in the Treshhold Image window). This function launches a progress bar.

This function calls the custom function *object_finder* to find the boundaries and bounding boxes.

4.6 biggest_bounding_box

This function takes one input *all_boxes* and gives four outputs: *min_row, max_row, min_col, max_col*. The function finds the furthest extents of the bounding boxes for each frame in the movie, and then finds the furthest extents of the bounding boxes over the entire movie and returns those four values. A bounding box with the corners at the extents found here will contain all bounding boxes.

4.7 find_diameter

The function*find_diameter* is the key to the kymograph making. This custom function takes three inputs, *min_col, max_col,all_boundariesk*, which are the minimum and maximum columns of the biggest bounding box, and the boundary pixels of the thresholded object for the specific frame k we are looking at. The outputs of this custom function is a list of values which correspond to the diameter for each column of the image between the min_col and max_col values.

Since all the values in a thresholded image are 0 or 1, it is easy to find the distance between the top and the bottom of the thresholded object (the intestine). We generate a list of x_positions between the min_col and max_col. These values are the column values in the image. For each x_position we look at the first row in that column that has a non-zero value, which will be the top of the thresholded object. We then find the last row with a non-zero value. The difference between these two values is the diameter of the thresholded object at that x_position. If the column contains only 0 values, as would be the case in the buffer columns, then the value of the diameter is 0.

The output from the *find_diameter* function is the list of diameters for each x_position. These can be appended to each other frame by frame to build up the kymograph.

4.8 GUI ONLY: Make_kymo

The function *Make_kymo* is called by the GUI to build up the kymograph. The same functionality is coded directly into the script program. The function *Make_kymo* uses four inputs, *numframes, min_col, max_col, all_boundaries*, which are the number of frames, the min and max column in the biggest boundary box, and the cell variable *all_boundaries* that contains all the boundaries of the thresholded object for all frames in the movie, each saved as a separate callable cell.

We pre-allocate an empty matrix called IMAGE_DIAMETER that has as many columns as the difference between max_col and min_col, and as many rows as the number of frames. textitMake_kymo calls *find_diameter* on a frame "k" of the movie, and fills in the matrix IMAGE_DIAMETER for row "k" with those values. The function *find_diameter* is called on each frame of the movie to build up the kymograph (or....Make_kymo)

4.9 Save_kymo

The Save_kymo function doesn't exist explicitly in either the GUI or script. In both cases, the kymograph, reduced kymograph and csv file names are constructed, and the files are saved to the save directory. In these code blocks, there is some error checking to make sure the folder to which the files should be saved exists, and if not, to create it. There is a following code block to convert the values of the diameter to uint8 if the max value of the diameter is less than 255, or uint16 if the max value is greater than 255. Following that, the images are saved, as is the csv file of the diameter map.

5 Part 5-Installation

5.1 Matlab Installation

The script and GUI were written using Matlab 2019a, and the app designer. The Image Analysis toolbox is used. These can be installed by IT for Mac, PC or Linux.

5.2 Files needed for Installation

Once MATLAB is installed, the following files are needed, and should all be saved in the same folder.

These files should all be in a single folder. To use them in Matlab, open Matlab, and using the file list on the right hand side, navigate to the folder where the files are. Right click on that folder, and select **Add To Path** → **Selected Folders and Subfolders**. The folder and files should go from being grayed out to being non-grayed-out.

File Name	What it is used for
intestine_all.mlapp	GUI Program file
biggest_bounding_box.m	Determines the greatest extents of the bounding boxes
find_diameter.m	Finds the diameter of the thresholded object across it's length
get_boundaries.m	Calls object_finder on all frames to get boundaries for the entire movie
image_thresh.m	Thresholds image and removes all but the largest object
Make_kymo.m	Runs find_diameter function on all frames, and assembles kymograph
Movie_import.m	Loads the movie into the program
object_finder.m	Finds the boundaries of the thresholded object in a frame
Sonia_Main_For_Movie.m	The main script to run, if not running the GUI

5.3 Gui Installation and Running

To run the GUI, select the intestine_all.mlapp file and right click and click **Run**, or select the file and press F9. This should launch the GUI app.

5.4 Script Installation and Running

To run the script, double click the **Sonia_Main_For_Movie.m** file to open it in the editor. Change the code as needed as explained in section 3. When finished, save the file, and click the big green **Run** button to run the file.