USER GUIDE

I. PREPARING THE FILES

- Download the current version of the MATLAB files from this GitHub repository
    - Be sure to download both *MedianTracking.m* and *interpolate_cscvn.m* and put them in the same folder
- Add this add-on to your MATLAB dependencies (*Home > Add-Ons*)
- Download the *.dcm* file you are observing and put it in the same folder as *MedianTracking.m* and *interpolate_cscvn.m*

II. RUNNING THE CODE

NOTE:

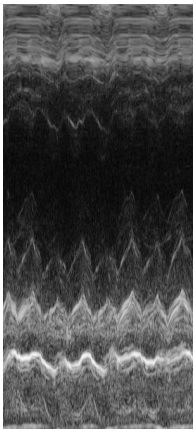
The following steps take you through *MedianTracking.m*. This is the main file used to run the analysis.

Section: `%% Setup & Crop %%`

1. `inf = dicomCollection('<file>.dcm');`
    a. Change the name of the *.dcm* file to the name of whatever file you want to observe
2. `rect = [<x>,<y>,<width>,<name>];`
    a. This crops the view of the file. You may have to change this. If you end up needing to, uncomment the lines below, and run the code.
3. `flip`
    a. This should be set to either 1 or 0. Some of the images appear to be flipped on their vertical axis somehow. The apex of the wall should be on the righthand side. Set this to 1 if the images should be flipped.

Section: `%% Select Cycles %%`

1. This is used to identify the sections we want to analyze. It will output and image that looks like this:



2. `montage = zeros(<slice>, framecount);`
    a. Change `<slice>` to the middle of the cropped frame.
3. Hover over the image to see the x and y values. The x values represent frames. We want to select 3 cycles (from one "trough" to 3 "troughs" over)
    a. Make note of the start and end values for your selected 3 cycles
4. Press *ENTER* to continue the code.
    a. If you are not ready to analyze the frames yet, just hit *Pause* (*Editor > Pause*) and then *Stop* (*Editor > Stop*)

Section: `%% Core Loop %%`

1. Set `fstart` and `fend` to the start and ending frame values you identified earlier.

NOTE:

From this point on, you will be adjusting values to find proper positioning for the scanning of the images. I recommend setting your `fstart` and `fend` values to the same number or close to one another to make it easier to observe changes in your adjustments.

Another general rule of thumb: commenting and uncommenting lines that contain the `plot();` command will display/hide features being generated in a section. Manipulate these in whatever way may help you.

Section: `%% Segment Walls %%`

1. Find these:

```
bin_epi = img > 160;
bin_endo = img < 40;
```

    a. These numbers are the intensity that is mapped to the inner and outer wall for the raw imaging. Changing these will change how many pixels will be observed for the later drawing of the walls. If you don't understand what these do, I would suggest playing around with them a bit.

Section: `%% Epicardium Curve %%`

1. Find:
   ```
   xep = <int>;
   yep = <int>;

   st_epi = <int>;
   end_epi = <int>;
   ep_numpoints = <int>;
   ```
2. At this point, the code rotates a radius clockwise around a central point to find points for the walls.
   a. `xep = <int>; yep = <int>;` represent the center point
   b. `st_epi = <int>; end_epi = <int>;` represent the starting and ending degrees for the rotation
      i. The default values are usually already close and may need a little adjusting to cut off excess tails or complete more
   c. `ep_numpoints = <int>;` is the number of points that will be found. I found that 64 is usually pretty good but change it as you see fit.

Section: `%% Endocardium Curve %%`

1. You will find a similar section to the one above. All the variables have the same function, but are appended with "en" or "endo" instead of "ep" and "epi"
   a. You will find these prefixes throughout. It will follow that same rule of thumb.

Section `%% Generate ROIs %%`

1. `numROIs = 23;`
   a. This integer is the number of ROIs that will be generated on each wall, **plus 1.** Try and make it a number divisible by 3 so that we can observe the anterior, posterior, and apex sections.
2. `hROI = <int>; wROI = <int>;`
   a. These represent the height and width of each individual ROI, respectively.

That should cover about everything you need to change.

III. EXPORTING DATA

The code will **overwrite** into a file named *strain.xlsx* after each complete run (this means, if you want to save the table you generated, rename the file or move it elsewhere before running again). Each row in the excel sheet represents one ROI, and each column is the frame. You can generate a quick graph of the data by highlighting the entire complete table and generating a line plot. Use this line plot to determine how effective the run was.

NOTE:

Overall, you want to generate a sinusoidal shape.

The anterior section, in its current state has difficulty tracking movement because it is too dark. Keep in mind that the ROIs are generated clockwise, so the middle third section of the data will be the anterior.

Overall values are currently looking a little high but try to keep it as low as possible for now (normal strain should not go above 1). If things start looking to high between trials, let me know.