Logan’s tracking algorithm

# Background:

Logan track is a script designed to efficiently extract regions of interest from live cell imaging experiments and then link objects together frame by frame.

In general, efficiency will not be very high. On average, except to get around 25% of possible complete tracks per movie (so 25 complete tracks out of a possible 100 cells at the beginning of the movie). Efficiency will be higher in shorter experiments or when cell density is lower (The best I get in a 48 hour movie is normally 50%). I generally recommend using 2x2 tiling and making multiple films for each treatment to get the most possible tracks.

Tracking accuracy is very high. The algorithm prioritizes error free complete tracks and does not track cells that come into frame between the beginning and the end of the movie. The linking error rate is roughly 0.1% (1 error every 1000 object linkages). In a 48 hour experiment, this corresponds to about 70-90% of complete tracks that are completely error free. It will be more error prone when cell density is high.

# Usage:

The python portion of the script should be run in the GPU segment 4i environment in anaconda.

If you run the script in Spyder, you’ll need to install the Spyder dependencies to this environment

Most of the hands-on work will involve changing settings. It looks like there are a lot of settings to change but only some need to be changed every time.

(Note: I’ve only run this in Spyder (a python IDE), so I don’t know the best way to set the working directory with jupyter notebooks)

Movies for testing the script are found here:

[\\rc-cifs.its.unc.edu\cook-lab\logan\LoganTrack](file:///\\rc-cifs.its.unc.edu\cook-lab\logan\LoganTrack)

For best results/performance transfer them to your local computer before running.

# Part 1: Segmentation and Measurement

1. Open anaconda and the python script (or notebook)
2. Set the working directory to the folder with the tiffs to be analyzed
   1. ***Right now, the script accepts movies exported into stacked tiffs, with one tiff stack per position***
3. Check the settings (True/False = On/Off):
   1. **gpu:** Whether to use GPU for cellpose segmentation
   2. **shading\_corr:** Corrects the images with the shading correction image, set to false if the images are corrected already (If you need to shading correct you’ll need a shading correction image, ask Logan how to make)
      1. Normally much less trouble to just correct on the microscope…
   3. **foci:** If you want to calculate PCNA or any other variance
   4. **QC image export:** Exports a downscaled image of each frame
   5. **Image overlay:** Overlays mask outlines on each QC image export
   6. **Last frame export:** Useful if matching live movies with fixed images
   7. **Segmentation:** Set to false if you only want to export the last frame, or export QC images
   8. **Track channel:** This is the image that will be segmented, this should match the name on the tiffs (i.e. “C1” or “CFP”)
   9. **channels\_list:** List of the channels you want to quantify, it should match the name on the tiffs (i.e. “C1” or “CFP”)
      1. I generally find it causes fewer problems to export the images using the “C1” etc. channel names
   10. **calculate\_jitter:** Set to true if there are frames where the plate moves significantly (such as after adding drug to the plate, also sometimes between the first and second frame there is a shift in the plate)
   11. **frame\_of\_jitter:** List of frames. These are the frames *before* the plate moves
   12. **test\_mode:** *This is important.* If this is set to true, a section of the script will run that allows you to test cellpose parameters (explained in next section)
4. Run each cell until you get to test mode
   1. Here, test the cellpose parameters on a sample image
   2. **Footprint**: This is the size of the tophat filter used for background subtraction
      1. This should be roughly the pixel diameter of the mitotic cells (you can test this a bit in ImageJ)
      2. You would rather use a number that is too big, rather than too small
   3. **Diameter**: Roughly the average diameter of cells in the movie (Between 25-28 for 2x2 binned movies, 50-60 for 1x1)
   4. **Flow\_threshold**: How close the shape needs to be matched to call something a cell. Higher will include more oddly shaped or different sized objects. 0 includes all objects. (Range is normally 0 – 1, but can go to 5)
   5. **Cellprop\_threshold**: Adjusts the strictness of the threshold. Higher numbers mean *fewer* pixels are classified as objects and reduce either the size or number of objects. (Range is between 0 and 3)
   6. The test image will display with the masks outlined in magenta**, in particular, check to make sure mitotic cells are correctly segmented**
   7. This process is mostly trial and error: If you are see cells that aren’t segmented, then lower the cellprop number and/or increase the footprint. It’s never going to be perfect though so don’t spend too long on this.
   8. Once parameters are set, move onto the main segmentation and measurement of cell
   9. Tip: You will want to set so that it includes more objects/oversegmenting rather than being too strict/excluding objects. A scenario you often run into is generating two objects for one nucleus/cell after mitosis. This isn’t that big of a deal though because objects appearing in later frames aren’t tracked so they just get ignored.
5. Run the next python section
   1. This starts segmentation, then once segmentation is finished, runs the region props measurement (and calculates optical flow/plate movement)
   2. Segmentation is 5-10 minutes per position & 24 hours of frames on GPU
   3. Measurement is also about 5-10 minutes per position & 24 hours of frames
   4. The output is two .csv files for the nuclei and cytosolic measurements

# Part 2: Tracking

1. Open R studio
2. Copy the logantrack\_v04.R file to the directory with the .csv outputs
3. Open the R script
4. If this is your first time running the script, R studio will prompt you to install the required packages. Do this before starting.
5. **The R script is meant to be run line-by-line/section-by-section**
   1. To do this, highlight sections of the code you want to run at once, then click run
6. Import the nuclei.csv results
7. First Config settings:
   1. This part just renames the columns
   2. Channels: The list of channels as in the column names, however the numbers are spelled out (0, 1, 2 = zero, one, two)
   3. Then change the names to want your channels represent (i.e. zero <- pcna)
   4. Cyto: Set to true if you have cyto measurements (will always be true)
   5. Cyto\_var: List of variables that you want cytosolic measurements for
8. When prompted, import the cyto.csv file
9. Run script to the config 2 section
   1. Config 2 matches positions to treatment names, using the filenames
   2. If you only have one treatment/group, set import\_info to false and skip the next step
   3. If you have multiple treatments, you will need a positions .csv file
      1. A list of positions and the corresponding treatment (and an ordering column)
      2. Ask Logan for a template
      3. varNames and varOrder need to match column names
10. Mitosis stats:
    1. Change **mean\_pcna** in:

mutate(mitosis\_score = (perimeter\*area\*length/**mean\_pcna**))

to the name of your nuclear marker (*mean\_mturq* is the other common option)

1. Tracking
   1. Define the track settings
   2. **Ijump**: The search area for the initial cell movement between frame 1 and 2
   3. **Sjump**: The regular search area for tracking subsequent frames (~20-25 works for 2x2 binned images, 40-50 for 1x1 binned images)
   4. **link\_jump:** the search area when linking forward and backward tracks (15 works for 2x2 binned images)
   5. **SpeedModifier**: Adjusts how much the search area is influenced by cell motion (best results between 0.75 and 1, but it doesn’t matter that much)
   6. **Jitter\_Correction**: Set on if you have jitter correction calculated for frames
      1. **Jitter\_frames**: If you have frames with jitters, list the frames *after* the jitters/plate movement
   7. **Frame\_interval**: How many minutes between frames
   8. **Parallel**: If you have more than one position/movie, set this to on to make the tracking go faster
   9. **Test**: If set to true, this will stop the tracking after like 20 frames. *(I normally set this to false, and test the settings using just one movie, then process all of them)*
   10. **Mthresh:** this is threshold for estimating whether a cell is mitotic. Normally the cutoff is set at the 98th percentile value and doesn’t need to be changed.
   11. positions <- unique(cells$n\_position): If you have multiple positions and want to track only one, then use positions <- unique(cells$n\_position)[x], where x is the index of the position you want to track (1, or 2, etc)
2. Run the code until the export file with tracks section
   1. This exports the raw tracking file as a .csv so you don’t need to re-run the tracking part everytime
   2. If the tracking is completed, but you are stepping back into the script after closing R, don’t re-run the tracking part and import the tracking file
3. Identify mitosis and align tracks
   1. For most of our experiments, you’ll want to have the tracks aligned to some point in the cell cycle. My script automatically aligns cells to mitosis. *(Automated alignment to the G1/S transition may also be possible someday).*
   2. LongTrackThreshold: Set the number of frames that defines what minimum track length to include
      1. This is experiment dependent (Do you need tracks that span the entire length of the movie? Do you only need tracks that are the length of one cell cycle/G1 phase?)
   3. Run the rest of the code through to the plotting section
   4. Run the plot to check if mitotic alignment is correct, the result should look something like this:
   5. A graph of a sound wave

      AI-generated content may be incorrect.
   6. The two lines represent the limits for the change in mitosis score for triggering a mitosis call
   7. Make sure that there are no obvious missed mitosis calls (typically they show up at -20 to -24 hours). You can use this graph to set better limits for riseThresh and fallThresh, then re-run mitosis detection. Unfortunately, this is never going to be perfect.
4. Data visualization and track filtering
   1. Export the aligned tracks to a .csv file (write.csv(mtrackedfGens, "tracks\_aligned\_annotated.csv")
   2. You can then graph in your preferred graphing program
      1. Column/variable names are somewhat confusing, so here is a brief dictionary:
      2. Variables with \_cn suffix are cytosolic/nuclear ratios (i.e. dhb\_cn would be the name for CDK activity)
      3. *Track\_id:* This is a label for each track. Cells with the same track\_id belong to the same track
      4. *M\_state:* The output of the mitosis detection algorithm
      5. *"h\_rel\_fm"* and *"h\_rel\_lm":* The time in hours relative to the first mitosis in the track (fm) or last mitosis in the track (lm)
      6. *"time\_in\_cycle"*: The cell age in hours relative to the most recent mitosis. Will be negative if the cell hasn’t undergone mitosis
      7. "generation": The generation of the cell at each frame. Increases by 1 every mitosis
      8. *"track\_id\_gen"*: A unique track\_id for every generation. Allows you to group cells by 1 individual cell cycle
   3. There are plotting tools in the R script, but at the moment require some knowledge of R to use correctly. *(I am working on a solution to this…)*
5. Export labeled track images
   1. This is useful for filtering or quality control of the tracks.
   2. Set the directories for imported the QC images and exporting the labeled track images
   3. The first loop will export each frame with the track\_id written on each cell
   4. The second loop is for exporting a single cell/track
      1. track\_id\_export: Change this to set which track to export

# Common issues:

1. Tracks look terrible/way too few tracks
   1. This can be caused by putting in the wrong frames for jitter correction
   2. Can also be caused by bad cell segmentation
2. If the tracking algorithm is run in parallel you can get a “Failed to establish connection with socket” error. This is rare, but can happen when tracking really big files. If you get this either turn off parallelization or decrease numCores used.

# Technical details:

This part describes the logic of the tracking method, you don’t need to know this stuff to use it, but if you are interested....

* Raw frames are background subtracted using a tophat filter roughly the size of a mitotic cell, enhancing signal to noise ratio
* The image is then slightly smoothed before segmentation using a gaussian filter
* The entire stack is fed into cellpose at once, allowing parallel segmentation on GPU, and reducing run time
* During the measurement step, optical flow is calculated using farnebacks algorithm to estimate cell motion between frames. This is calculated in reverse time (So between frame T-1 and frame T). During testing, I found that this is more error prone at estimating cell movement, so this value is only used in specific circumstances.
* During the measurement loop, when it comes to a jitter frame, phase cross correlation is used to estimate the overall shift in the X and Y direction. This is only done when called to save time.
* If measuring foci, a 2x2 pixel tophat filter is applied to the channel that is measured, which brightens small circles and dims the rest of the image. Region props is then applied to this image (and the other channels)
* The optical flow value for each objects centroid is extracted from the optical flow image.
* Prior to tracking a mitosis score is calculated for each object. This measures the size and brightness of the nucleus. It is used to estimate when a cell is dividing so the tracking logic can be changed. *(In the future a neural network or support vector machine is probably going to work better for identifying mitosis).*
* Tracking logic:
  + There are 2 tracking passes made in the reverse and forward directions (relative to time)
  + Reverse track is prioritized for a few reasons:
    - If matching to a fixed image, we only care about cell tracks that remained in frame at the end of the movie
    - Reverse tracking handles mitotic events more cleanly, both daughters are tracked, and one of the daughters is picked for continuing the track
    - Handles cell death better: Dead cells will not be alive at the end of the movie
    - The downside to this is that partial tracks will be biased towards the end of the movie compared to the beginning
  + Tracking is done by generating a search window that is equal to current object centroid x/y +/- jump distance, then searching the next frame for matches
  + If multiple matches are found, the distance between matches are compared. If one object is clearly closer to the expected position, it is selected for linking.
  + To improve accuracy, the cells prior trajectory is used to influence the search window. This is done by shifting the objects centroid in correspondence to its trajectory: Expected Next Position = (x+x-trajectory, y+y-trajectory)
  + During mitosis, optical flow vectors are used to estimate the cells’ next position. This is done because cell trajectory changes rapidly and unexpectedly during mitosis and this is estimated pretty well by optical flow algorithms.
  + When tracking during a plate jitter, the shift values are used to correct the objects’ expected location in next frame
  + Each object from the beginning or end of the movie is then tracked frame-by-frame before the loop restarts for the next object
    - This is different from other algorithms that track all objects in a frame, then move onto the next frame.
    - Someone should look into if this makes a difference?
  + Track segments present in the forward track but missing in the backwards tracks are then linked together by searching the track endpoint of the reverse track for a corresponding forward track
    - Due to the cell motion estimation, you can get different results when going forwards vs. backwards in time, so the linking step helps get more complete tracks
    - Tracks will not be linked if cell measurements are significantly different
* Mitosis detection
  + Accurate calls for mitosis are needed to correctly align tracks
  + To do this, the frame-to-frame change in the mitosis score (mentioned above) is derived (dMitosis\_score/dTime)
  + Times when the mitosis score rapidly increases, then shortly after, rapidly decrease are identified
  + The time of the rapid decrease is selected as the mitosis point
  + *In the future, neural network probably helps here*