INSTALLATION NOTES:

- Call script with “python kymotracker_calling_script.py” in terminal
- You will need to install the python package npTDMS
    - “pip install npTDMS” in terminal or however you normally install python packages
- You will need to update the lumicks.pylake package
    - I use “pip install lumicks.pylake --upgrade” but this might require some troubleshooting because this might be different for Macs
    - Need version >= 0.7.1
- Also need numpy, pandas, and matplotlib downloaded

DESCRIPTION:

This script is designed to process TDMS and .h5 kymograph files and perform batch processing of tracked line foci for downstream analysis. Lines are tracked using either of Lumicks' two line tracking algorithms – more information can be found at Lumicks' Kymotracking Tutorial. This script automatically exports the lines tracked for each of the colors requested (time data in second, coordinate data in nm) as well as the kymotracking settings for each call of the algorithm in a .xlsx file. Metadata from the files is stored in a separate .csv file. There are built in options to extract the photon counts around the foci (as determined by the line_width variable) and the distance between a chosen foci color and the other color's foci. All additional data is stored in the same .xlsx file.

This script uses text inputs instead of graphical inputs because it is easier to program text inputs in when doing batch processing of data. Questions will be asked out to the terminal and user input is required to both provide initial information about the batch processing [what folder are the .tdms/.h5 files to analyze in, what tracking method to use, what fluorophores to track, what type of area do you want to analyze (rectangle across the whole trace or a more complex region), do you want to extract photon count intensities] and then confirmation or rejection of the two steps of the program for each file to analyze - [1] manually selecting an area to analyze and [2] calling the line tracking algorithm for each fluorophore color. Below I will try to describe the intended use of the code and give examples of correct inputs, please let me know if you have any questions.

INPUT LIST:

If you want to bypass having to answer the different questions in terminal you can input keywords as a space separated list of variable=value. You can do this for some or all of the inputs. Do not include apostrophes as shown in these definitions ("Ex: tracking_method=greedy line_width=4, colors_to_track=RG"). Example inputs through answering the terminal prompts or pre-defining the inputs in the initial terminal call are shown below, after which all of the terminal input terms are defined.

Example Input Options

Answer Input Questions in Terminal

```
PS C:\Users\johnw\Documents\Liu Lab\Kymotracker Development> python .\kymotracker_calling_script.py
--------------------------------------------------------------------------------
List of files to be analyzed
################################################################################
20200821-134449 Cy3 H2A LD655 ORC ejection #007-002.tdms
20200306_162137_ORC_binding_Kymograph_6.h5
################################################################################
Please sort the files so that only similar experiments are in the same folder:
Input file name to save (do not add any file extension)?
test_file
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Please indicate desired tracking method from the lumicks.pylake options
[1] track_greedy
[2] track_lines
Input integer number of the correct method:
1
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Choose colors to track for all files
Input RGB values, Ex: RG or RGB:
RG
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Would you like to extract the photon counts sum of the lines?
Input yes/no:
yes
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Would you like to extract the distance between tracked lines?
Input yes/no:
yes
What color would you like to use as the base for extracting this distance?
Input R/G/B
G
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Choose option of how to manually input the area of analysis:
[1] Manually define top and bottom positions of a rectangle to analyze
[2] Manually define a more complex region (containing pulls and relaxes)
Input integer number of the correct method
1
--------------------------------------------------------------------------------
```

Manual Area Selection/Kymotracking

Specify Inputs in Terminal

Terminal Command:

```
python kymotracker_calling_script.py tracking_method=greedy
file_name=test_file colors_to_track=RG
opt_to_extract_intensities=yes
opt_to_extract_distance_between_foci=yes
opt_for_area_selection=1 line_width=6
color_to_track_distance=G
```

Manual Area Selection/Kymotracking

Definitions of Terminal Inputs

- tracking_method : "greedy" or "lines"
  This variable determines which of Lumicks' tracking methods the script uses. The "track_greedy" method or the "track_lines" method
- file_name : string
  This variable defines the base file to save the line data/kymotracker setting for file_name + _summary.xlsx and the metadata file as file_name + _metadata.csv
- colors_to_track : string of any combination of R and/or G and/or B
  This variable defines which channels are going to be put through the kymotracker algorithm
- opt_to_extract_intensities : string of "yes" or "no"
  This variable allows the user to control whether or not the script also outputs the sum of the photon counts around each foci. The range over which the sum is measured is based off of the line_width variable num_frames = math.ceil(line_width/2)

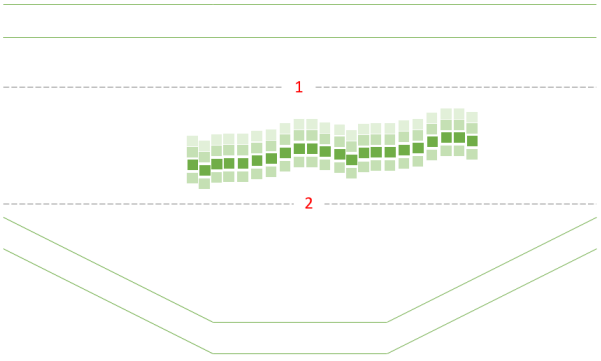
- opt_to_extract_distance_between_foci: string of "yes" or "no"
  The variable goes through each of the "base" lines (as defined by colors_to_track_distance) and finds the closest point in each of the other color traces (ex: the closest red line to a green base line) and reports the distance. Negative distance means the tracked line is below the baseline, and positive distance means the tracked line is above the baseline
- color_to_track_distance
  This variable defines a “base” line for the closest distance search
- def_line_width : integer
  This variable overrides the default line_width value for each of the kymotracker algorithm calls. Since the kymotracker values reset between each of the files being analyzed this can be a way to reduce the number of variables you have to change each file
- opt_for_area_selection : integer, 1 or 2
  This variable determines the method used to select the area fed into the kymotracker algorithm. Method 1 just involves choosing a point above and below the region of interest - and the algorithm uses a rectangle with those as the vertical points Method 2 involves choosing the top left then top right point and then every point of the bottom area where you want to define. The bottom line will be a horizontal line until the first clicked point, then will track linearly to the next point and so on until the end of the kymograph. From the last point, the last defined slope will continue until the end of the kymograph

MANUAL AREA SELECTION:

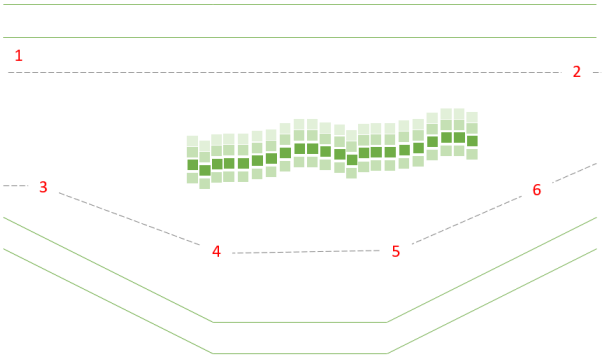
For step one, which is manually selecting an area to analyze, there are two options – method one and method two which are demonstrated below. In method one the plot will automatically close after two points have been chosen, while in method two (since there could be as many inputs as you want) you must wait for the 30 second timer to close the plot.

Examples

Method One



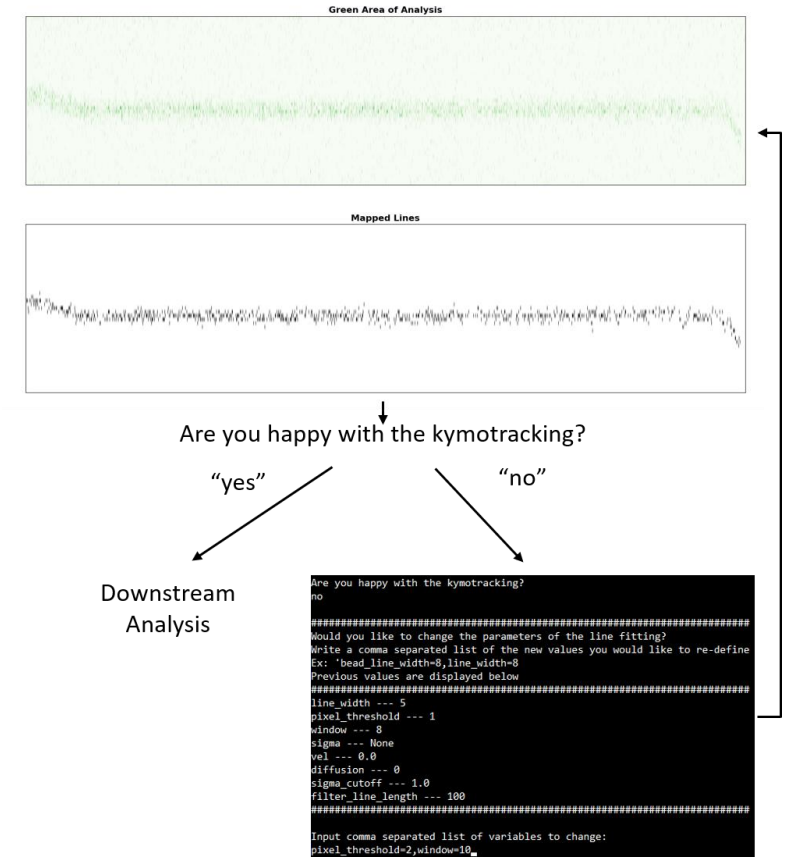
Method Two



Red numbers show order of the clicks
Grey lines outline the area to analyze

KYMOTRACKER REFINEMENT:

Once the area selection has been confirmed, the kymotracker algorithm of choice is called and returns a plot of the channel data from that area as well as a pixelated map of the lines. After closing the plot, the user is then asked if they are happy with the kymotracking. If the user is happy, the program then completes the desired analysis, but if the user is not happy the user is able to change any of the specific algorithm's variables. An example of the kymotracker refinement process is shown below:



Cheat Sheet of Variables to Change:

Algorithm	Problem	Variables to Increase	Variables to Decrease
track_greedy	Too many lines?	pixel_threshold filter_line_length sigma	sigma_cutoff diffusion
	Too few lines?	sigma_cutoff diffusion	pixel_threshold filter_line_length sigma
	Lines not connecting?	window	
track_lines	Too many lines?	starting_threshold continuation_threshold filter_line_length	max_lines
	Too few lines?	max_lines	starting_threshold continuation_threshold filter_line_length
	Lines not connecting?		continuation_threshold max_lines

Cheat Sheet Notes

- You can change any of these default value by going into the python script file, searching for "# set defaults by changing these variable declarations" and then changing the default variable declaractions
- Try to go in separately through CTrapViewer for .tdms files or the CTrap GUI script for .h5 files to get a good approximation of the line width
- I personally have not used track_lines as much because it seems like it has a harder time connecting longer traces

KYMOTRACKER TERMS FULL DEFINITIONS:

track_greedy:

- line_width : float
  Expected line width in pixels.
- pixel_threshold : float
  Intensity threshold for the pixels. Local maxima above this intensity level will be designated as a line origin.
- window : int
  Number of kymograph lines in which the particle is allowed to disappear (and still be part of the same line).
- sigma : float or None
  Uncertainty in the particle position. This parameter will determine whether a peak in the next frame will be linked to this one. Increasing this value will make the algorithm tend to allow more positional variation in the lines. If none, the algorithm will use half the line width.
- vel : float
  Expected velocity of the traces in the image. This can be used for non-static particles that are expected to move at an expected rate (default: 0.0).
- diffusion : float
  Expected diffusion constant (default: 0.0). This parameter will influence whether a peak in the next frame will be connected to this one. Increasing this value will make the algorithm allow more positional variation in.
- sigma_cutoff : float
  Sets at how many standard deviations from the expected trajectory a particle no longer belongs to this trace. Lower values result in traces being more stringent in terms of continuing (default: 2.0).
- filter_line_length : int
  This one is normally tied into a different function in Lumicks' scripts.
  Sets a threshold for a filter where any lines shorter than this threshold will be removed from the data set.

track_lines:

- line_width:float
  Expected line width in pixels.
- max_lines:int
  Maximum number of lines to trace.
- start_threshold:float

Threshold for the value of the derivative.

- continuation_threshold:float
  Derivative threshold for the continuation of a line.
- angle_weight: float
  Factor which determines how the angle between normals needs to be weighted relative to distance. High values push for straighter lines. Weighting occurs according to distance + angle_weight * angle difference.
- filter_line_length : int
  This one is normally tied into a different function in Lumicks' scripts
  Sets a threshold for a filter where any lines shorter than this threshold will be removed from the data set.