Particle Localization and Tracking GUI: *TrackingGUI_rp.m*

Raghuveer Parthasarathy
Department of Physics
The University of Oregon
raghu@uoregon.edu

Begun April 2012 (based on earlier work). Mostly written 2012-2013.
Last modified **June 23, 2020**

Contents

DESCRIPTION

- **TrackingGUI_rp.m** provides a graphical user interface (GUI) for the localization and tracking of objects in a series of 2D images, applying any of several algorithms for sub-pixel particle localization (radial symmetry fitting[1], Gaussian fitting by least-squares minimization, Gaussian fitting by maximum likelihood estimation, etc.), linking objects into tracks, displaying tracks, culling tracks, and saving the calculated position information. It can be applied to images of bacteria, colloidal particles, single fluorophores from super-resolution microscopy, and more.
- The GUI calls several functions, the most important of which are `fo5_rp.m`, which determines the particle neighborhoods at which to apply the localization algorithms, the localization functions themselves, and `nnlink_rp.m`, which "links" objects in adjacent frames into particle trajectories.
- The GUI allows easy alteration of image processing parameters, e.g. an intensity threshold, the thresholding method, and the neighborhood size for localization.
- The GUI can display calculated positions and tracks, and can call functions to cull tracks based on various properties including track length and spread in position.

[1] R. Parthasarathy, "Rapid, accurate particle tracking by calculation of radial symmetry centers," *Nature Methods* **9:** 724-726 (2012). DOI: 10.1038/nmeth.2071 .

MATLAB Toolboxes Used

- **Image Processing Toolbox** (version 7.2 or higher; not tested on earlier versions).
- **Optimization Toolbox** (v 6.0 or higher) – needed for Gaussian fitting by maximum likelihood estimation and least-squares minimization; not necessary for radial-symmetry-based localization.

Required Files

- **`TrackingGUI_rp.m`** – the GUI program
- **`Particle tracking GUI Manual`** – this file

Misc. functions

- **`bpass.m`** – Bandpass filter (David Grier *et al.*)
- **`calcthreshpts.m`** – determines above-threshold points in images
- **`cullobjects.m`** – culls objects based on a threshold of width and / or gradient-line distance to center
- **`fitline.m`** – fits a line (called by gradientvote.m)
- **`fo5_rp.m`** – calls various localization algorithms to return the center location of a single particle
- **`getnumfilelist.m`** – for determining what image files to load
- **`gradientvote.m`** – for determining particle neighborhoods using a gradient voting scheme

Localization functions

- **`radialcenter.m`** – radial symmetry based localization (R. Parthasarathy, 2011-2012)
- **`radialcenter_stk.m`** – radial symmetry based localization – optimized for images containing many particles (R. Parthasarathy, 2012)
- **`gaussfit2DMLE.m`** – Gaussian fitting by maximum likelihood estimation
- **`gaussfit2Dnonlin.m`** – Gaussian fitting by non-linear least-squared minimization
- **`gaussfit2D.m`** – (Optional) linearized Gaussian fitting. *Avoid this technique*!
- **`gauss2dcirc.m`** – (Optional) weighted linearized Gaussian fitting. *Avoid this technique*!
- **`simpleellipsefit.m`** – (Optional) Determine particle orientation (for rod-like particles) by a simple determination of principal moments

Particle linking and trajectory editing functions

- **`nnlink_rp.m`** – links objects across frames to form trajectories
- **`dedrift_rp.m`** – de-drifts positions in tracks by calculating the mean or median displacement, possibly with averaging or shear.
- **`cullTracks_function.m`** – culls tracks (linked objects) based on various criteria such as standard deviation of positions / sqrt(length), overall track length, and more. Can be applied to the linked object matrix with or without prior de-drifting.
- **`trackveldist.m`** – Simple statistics about the velocities of the trajectories. (Doesn't identify runs, tumbles, or flicks; very simple.)

Instructions (Brief)

- The image data should consist either of a series of consecutively labeled 2D grayscale images (e.g. "*myImage001.tif*," "*myImage002.tif*," "*myImage003.tif*," ...) **or** a multipage (3D) TIFF "stack."

- Call **`TrackingGUI_rp.m`** either with no arguments, in which case you'll be prompted to enter the image file name(s), or enter as the input argument a 3D array consisting of all image frames.
- Set the object size (`bpfiltsize` for spatial filtering and `nsize` for the single-particle-neighborhood size), the threshold option, and the threshold parameter values. (See "screenshot," below.)
- Select the particle localization algorithm.
- Click "Track this frame" to apply the present parameters and fitting method to the presently displayed frame.
- Click "Track all frames" to apply the present parameters and fitting method to the all the 2D images.
- Link objects across frames into "2D + time" tracks. Set the max. step^2 and memory parameters. Click "Link Tracks."
- Select "Show Tracks" to display the tracks superimposed on the image.
- Save the output using "Save Output." (Or Save TXT to save the position information in a simplified form, as a tab-delimited text file.)

SCREENSHOT

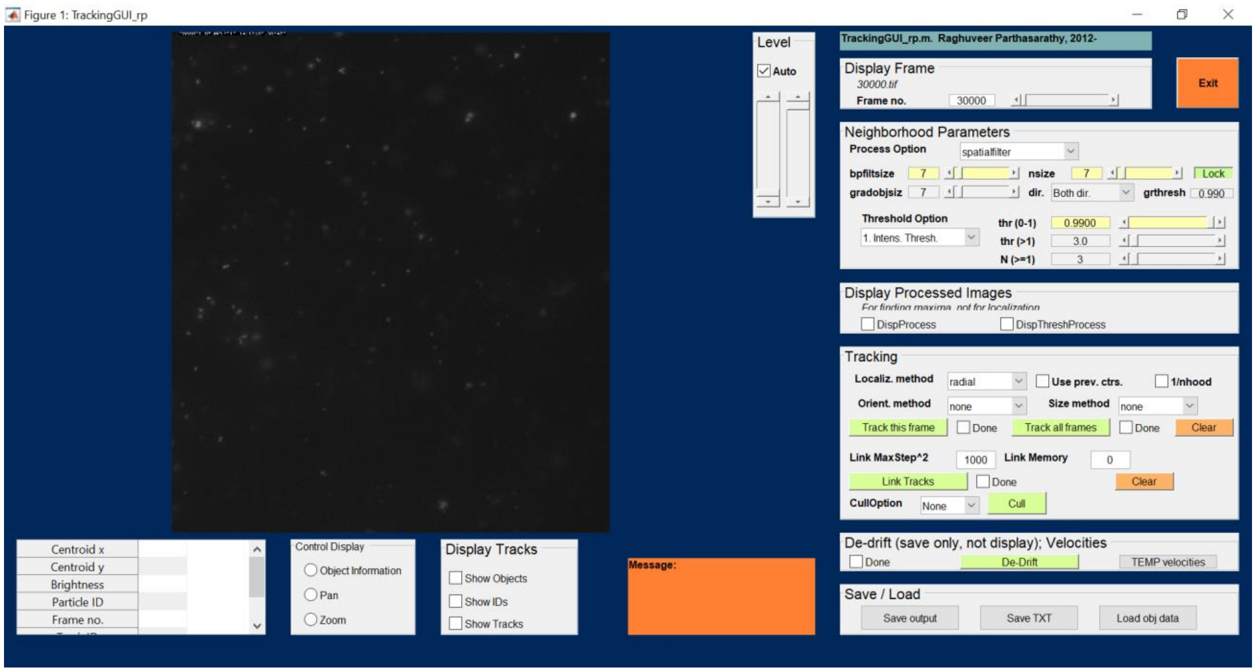
Figure 1: TrackingGUI_rp

TrackingGUI_rp.m. Raghuveer Parthasarathy, 2012-

Display Frame
30000.tif
Frame no. 30000

Neighborhood Parameters
Process Option: spatialfilter
bpfiltsize 7 nsize 7 Lock
gradobjsiz 7 dir. Both dir. grthresh 0.990
Threshold Option: 1. Intens. Thresh. thr (0-1) 0.9900 thr (>1) 3.0 N (>=1) 3

Display Processed Images
For finding maxima, not for localization
DispProcess DispThreshProcess

Tracking
Localiz. method: radial Use prev. ctrs. 1/nhood
Orient. method: none Size method: none
Link MaxStep^2 1000 Link Memory 0
CullOption: None

De-drift (save only, not display); Velocities

Message:

INSTRUCTIONS (EXPANDED)

- **The image data** should consist either of a series of consecutively labeled grayscale TIFF or PNS images (e.g. "*myImage001.tif*," "*myImage002.tif*," "*myImage003.tif*," ...) or a mulitpage TIFF stack. For consecutive 2D images, the numbers should come at the end, but they do not have to start at 0 or 1, and do not need to have the same number of digits (e.g. *myImage8.tif*," "*myImage9.tif*," "*myImage10.tif*" is fine).
- **Call `TrackingGUI_rp.m`** either with no input arguments, in which case you'll be prompted to enter image file name(s), or with a 3D array containing all image frames as input (e.g. "`TrackingGUI_rp(im)`", where `im` is a 3D matrix of images). If the array is not input, the GUI does

not in itself load all the images into a matrix – analysis is done on each 2D image sequentially, to avoid large memory requirements.

- Select the **processing option and parameters.** These, described further below, determine filter sizes and region sizes. Local maxima in the filtered image identify possible particles. Those that exceed the brightness threshold are kept. A neighborhood around *each* local maximum of is sent to the localization function which returns the particle center. Note that the filtered and thresholded pixel values are **not** used for particle localization – it's the actual measured pixel values that are fit.
- Since **spatial filtering** is generally applicable, I'll just describe this. See the list of all buttons below for a description of gradient magnitude voting. **Object size.** There are two "size" parameters for spatial filtering, both measured in pixels. Typically these are the same (and should be a bit greater than the expected object diameter) and so by default they are "locked" together; they can be "unlocked." (1) `bpfiltsize`: The function `fo5_rp.m` uses this to define the upper filter size for bandpass filtering; the lower size is 1 px. Enter "0" for no bandpass filtering or select "none" as the processing option. (2) `nsize`: The size of the image dilation structuring element and the neighborhood size (`nsize x nsize` pixels) for localizion. **Troubleshooting:** In general, using too small an object size will increase the likelihood of tracking noise, and of finding multiple maxima from one particle. Using too large an object size will slow the program and may lower the accuracy of particle center finding especially if the density of particles is high.
- **Thresholding:** There are three options for intensity thresholding. These apply to either processing option:
    - (1) Set a threshold (`thresh`) for the local intensity maxima, keeping all points with intensity above the *thresh*100* percentile. (E.g. *thresh*= 0.999 keeps pixels in the top 0.1% of intensity.)
    - (2) Keep pixels with intensity greater than *thresh* standard deviations above the median.
    - (3) Keep brightest *thresh* number of particles by finding the brightest regions and allowing only one maximum in each. (E.g. *thresh = 5* keeps only the five brightest particles.)
- **Viewing processing and thresholding parameters.** Select the '`Dispthreshfilt`' checkbox to see what the filtered and thresholded image looks like. (You can scroll through images, and check or uncheck the '`Dispthreshfilt`' box.) You should, ideally, find parameters for which there is a one-to-one correspondence between particles and the post-processing features found. Again, note that the filtering and thresholding are used only to find regions; localization is performed on the unprocessed pixel values.
- **Select the particle localization algorithm.** In brief: 2D Gaussian fitting by maximum likelihood estimation, and 2D Gaussian fitting by least-squares minimization all provide very high accuracy, close to theoretical limits. Radial-symmetry-based fitting[2] is much faster (typically > 100× faster), because of its analytically exact, non-iterative form, and has accuracy similar to the above Gaussian fitting (and to theoretical bounds). Linearized Gaussian fitting and centroid fitting are supported but are both *dangerously inaccurate* – use caution!
- **Other parameters.** The "1/nhood" option forces there to be at most one "object" found per neighborhood. Without this, it's possible that we might detect two bright maxima from the same particle/bacterium, which then confuses the linkage into tracks. This option will slow the analysis; if the slowdown is acceptable, I recommend using this option. See also "Descriptions of each button," below.

[2] R. Parthasarathy, "Rapid, accurate particle tracking by calculation of radial symmetry centers," *Nature Methods* **9:** 724-726 (2012). DOI: 10.1038/nmeth.2071

- Click **“Track this frame”** to apply the present parameters and fitting method to the presently displayed frame. Select **“Show objects”** in the “Display Tracks” menu to see if the found objects correspond to particles. If not, “Clear” the localization information, and try different parameters.
- Click **“Track all frames”** to apply the present parameters and fitting method to the all the 2D images.
- **Link objects** across frames into “2D + time” tracks. This is done by matching nearest neighbors across frames, with an extra constraint that the mappings “forward” and “backward” in time must be the same. There are two parameters, neither of which is very sensitive. The “`max step ^2`” is the square of the maximum step size that a particle is allowed to have between frames; if the distance-squared is greater than this, the found objects are necessarily considered to belong to different particles. This can be set to a large value (e.g. 100 if we’re sure a particle will never move more than 10 pixels between image frames.) The “`memory`” term attempts to link particles despite their absence over some number of frames, preserving their ID numbers. To be safe, set `memory=0` (no absence allowed). Click “Link Objs -> Tracks” – this is very fast.
- Select **“Show Tracks”** in the “Display Tracks” menu to display the tracks superimposed on the image. Note that you can zoom and pan the image. Clicking “object information” and then clicking the image shows the object properties of the object that is closest to the clicked point.
- **Culling objects and tracks.** See “Descriptions of each button / procedure, below.
- **Save the output** (“Save Output”). This saves the `objs_link` matrix and other things into a MAT file. The form of the `objs_link` matrix, which includes position and brightness information as well as “track” information, is given in the description of the “*Save Output*” button, below.
- Alternatively, save the output as a simple tab-delimited text file using “Save TXT”. This saves just the *x* and *y* positions of each linked object, without additional information. Row 1 = *x* positions (px) of object #1 in each frame in which it exists; Row 2 = *y* positions. Row 3 = *x* positions of object #2; Row 4 = *y* positions of object #2, etc. Note that the columns correspond to frames, but the set of frames in which each object exists need not be the same. (I.e. object 1 may exist in frame 1, 2, 3, 4, and object 2 may exist in frames 3, 4, 5, 6 – each would appear as a 4-column set of points in the output file.) The tab-delimited text file can be opened e.g. by WordPad or Excel. Note that the .MAT file output (with the object matrix construction) conveys much more information.

Descriptions of each button / procedure in TrackingGUI_rp:

DISPLAY FRAME

Level

Auto-adjust image levels, or manually adjust minimum and maximum.

Frame number
Selects the frame to display. Enter the frame number in the text box or use the slider.

NEIGHBORHOOD PARAMETERS

Process option
Choose the processing option that will be used to determine neighborhoods in which to apply precise particle localization algorithms (i.e. to find neighborhoods encompassing single particles). Choices:

- spatial filtering: Apply a bandpass filter with edges 1 px and *bpfiltsize* px. Determine particle neighborhoods by finding local maxima in the filtered image. (Note that the filtered image is not used for the sub-pixel particle localization.) The bandpass filtering is done by the bandpass filter function **bpass.m** by John Crocker and David Grier. (This is straightforward, and could easily be replaced.)
- gradient voting: At each pixel, calculate the local intensity gradient. Assign to each pixel within distance *gradobjsize* a vote proportional to the gradient magnitude. The parameter *dir* allows use of only positive or negative intensity gradient directions, or both. The parameter *grthresh* uses only the pixels *i* with gradient magnitude above this relative threshold (0-1). With this processing option, we use maxima of the total gradient vote as indicators of particle neighborhoods.
- None. No processing. Determine particle neighborhoods by finding local maxima in the raw image, with neighborhood size *nsize*.

bpfiltsize
See "Process option," above. Enter a value using the text box or the slider.

nsize
The neighborhood size in which to finely determine particle location (*nsize* x *nsize* px). Enter a value using the text box or the slider.

Lock
If selected (default), bpfiltsize and nsize are constrained to be equal.

gradobjsize, dir, grthresh
See "Process option," above.

Threshold option
There are three options for intensity thresholding, each of which can apply to whatever processing (filtered intensity, gradient magnitude, unfiltered intensity) is chosen:

(1) Set a threshold (`thresh`) for the local intensity maxima, keeping all points with intensity above the *thresh*100* percentile. (E.g. *thresh*= 0.999 keeps pixels in the top 0.1% of intensity.)
(2) Keep pixels with intensity greater than *thresh* standard deviations above the median.
(3) Keep brightest *thresh* number of particles by finding the brightest regions and allowing only one maximum in each. (E.g. *thresh = 5* keeps only the five brightest particles.)

The top, middle, and bottom text entry boxes and sliders apply to options 1, 2, and 3, respectively.

Display Processed Images

DispProcess
If checked, display the processed image (bandpass filtered or gradient vote image).

DispThreshProcess
If checked, display the above-threshold regions of the processed image.

Localization

Localization method
Selects the particle localization algorithm. In brief: Radial-symmetry-based localization, 2D Gaussian fitting by maximum likelihood estimation, and 2D Gaussian fitting by least-squares minimization all provide very high accuracy, close to theoretical limits. Radial-symmetry-based fitting is much faster (typically > 100× faster), because of its analytically exact, non-iterative form. Linearized Gaussian fitting and centroid fitting are both supported, but are both *dangerously inaccurate* – use caution!

Use prev. ctrs.
If checked, the localization will use the particle positions in frame *i*-1 as the neighborhood centers in frame *i*. Obviously, this should be avoided if the number of particles is not constant, or if positions from frame-to-frame are significantly different. This option is useful for speeding up localization for a series of images of the same particles, moving over time, since it bypasses processing and neighborhood calculations.

1/nhood
If checked, we consider only one local maximum per neighborhood in the processed image. If there are multiple maxima within regions, keep the brightest. (This is always done for threshold option 3, regardless of this input variable.) Multiple local maxima, and therefore multiple detected objects, can confuse linkage into tracks. Selecting this option slows the program. Whether it is useful depends on the images. It can be the case that for extended objects like bacteria, it's easy to find multiple maxima per bacterium, in which case forcing 1/nhood helps.

Orientation method
Selects the algorithm for determining the rotational orientation of the particle. (Default: none.) At present, only one method is supported: *momentcalc*, for a simple calculation of angle using principal moments.

Track this frame
Apply the selected parameters and localization method to the presently displayed frame. Select "Show objs" in the "Display Tracks" menu to display the found objects and positions. The *Done* checkbox will be checked if the present frame has already been analyzed. *Clear* will clear the found object information.

Track all frames
Apply the selected parameters and localization method to the presently displayed frame. Select "Show objs" in the "Display Tracks" menu to display the found objects and positions. The *Done* checkbox will be checked if all the frames have already been analyzed. *Clear* will clear the found object information.

Link MaxStep^2
This parameter gives maximum step size, squared, that a particle is allowed to have between frames; if the distance-squared is greater than this, the found objects are necessarily considered to belong to different particles. This can be set to a large value (e.g. 100 if we're sure a particle will never move more than 10 pixels between image frames.)

Link Memory
If *memory*>1, the program attempts to link particles despite their absence over some number of frames, preserving their ID numbers. To be safe, set `memory=1` (no absence allowed).

Link Tracks
Link objects across frames (e.g. into "2D + time" tracks). Linkage is done by matching nearest neighbors across frames, with an extra constraint that the mappings "forward" and "backward" in time must be the same. The results are quite insensitive to the two parameters noted above. Linkage is very fast. *Clear* will clear the linkage information, but not the found object information.

CULL OPTION (CULLING OBJECTS AND TRACKS)

CullOption Menu: Can cull objects (based on localization algorithm output), tracks (based on a variety of properties); can Undo the preceding culling; can revert to the original linked-object matrix (before any culling). Chose an option, then click "Cull."

- Cull **Objects** that are outliers in width (*sigma*) and /or gradient-line distance to center (*dmin*) for radial-symmetry-based localization). The thresholds of *sigma* and *dmin* above which objects are discarded are entered in line 1 of the dialog box, expressed as the number of standard deviations above the median. Or leave the box blank to use the default values of 0.3 for each. Or enter -1 to be prompted for input based on the histogram of *sigma* and *dmin* values. (If prompted, note that the program asks for absolute *sigma* and *dmin* values, not standard deviation values.) If a localization method other than radial-symmetry is used, *dmin* is not considered. For the culling method, one can consider a rectangle in sigma-dmin space, or a diagonal line – see `cullobjects.m` for details. If objects have been linked, culling is applied to both the unlinked *objs* matrix and the linked *objs_link* matrix.
- Cull **Tracks** based on various properties. Evaluates the chosen property, displays a histogram of the values, prompts the user for max & min values to allow, and culls tracks outside this range. Can be applied to de-drifted trajectories (see below for de-drifting) or the normal linked-object trajectories. If the checkbox indicating drift correction is un-checked, the user is first asked if de-drifting should be done, before evaluating and culling. If the drift-correction checkbox is checked (i.e. drift correction has already been done), the user is asked whether the "regular" (objs_link) or de-drifted trajectories should be used, and then whether de-drifting should be performed on these. (Note that it would be strange to use de-drifted trajectories and then *again* de-drift!) De-drifting is particularly useful / important for the "StdDev" criterion: if there is lots of drift, objects will have a large standard deviation of position; de-drifting and then culling based on standard deviation will remove "passive" objects. In general, I suggest culling on the dedrifted data only once. Calls **`cullTracks_function.m`**.
    - **Standard deviation of positions / sqrt(Number of frames).** Useful for removing stuck objects. Note that for Brownian motion, the standard deviation scales as the square root of

the Number of Frames, so this property should be independent of track length. Culling parameters: min and max values to allow.

- **Track length,** number of Frames. Culling parameters: min and max values to allow.
- **Track straightness.** Evaluate the Straightness Index, defined as the ratio of the end-to-end distance to the contour length of the track, and therefore a number between 0 and 1. Useful for removing drifting objects that are not rendered motionless by de-drifting. Culling parameters: min and max values to allow.
- **Track angle.** Calculated from the slope of the best-fit line to the trajectory. Degrees: 0 (rightward) to 360. Can also assist with uniform drift or bias. Culling parameters: min and max values to allow. If "max" < "min," assume the desired range crosses 360 degrees – for example, [340 20] would allow all angles from 340 to 360 degrees and 0 to 20 degrees.
- **Step Speed.** A crude measure of velocity – simply a trajectory's average frame-to-frame displacement. Culling parameters: min and max values to allow. Units: pixels/frame . Note that multiplying these values by the the um/px spatial scale **x** frames/second temporal scale gives speed in um/second.

- **Undo:** Undo the preceding culling.
- **Revert:** revert to the original linked-object matrix (before any culling).

De-Drift

De-Drift. Corrects for overall drift of objects and the medium; de-drifts positions in tracks by calculating the mean or median displacement of all objects and subtracting this from each object's position, possibly with averaging or shear correction. Note that applying the median shift (default) works better than the mean, because of outliers. Calls **`dedrift_rp.m`**. Dialog box for parameters. Note that the de-drifted trajectories are not displayed – they couldn't be superimposed on un-translated images – but they may be used for culling (see above).

Velocities. Call **`trackveldist.m`** to determine the track velocities. Note that calculating velocities can be slow if there are lots of tracks! **NOTE:** so far, nothing is done with the velocity information! This is rather crudely written, not accounting for runs/flicks/tumbles. Also, the outputs are not presented well. Still, it provides a simple assessment of the velocities of the tracked objects.

In brief, the function calculates the velocity of each track as the best-fit slope of position vs. time over some (user-input) sliding window of frames. This can be calculated for the original or de-drifted positions; de-drifting is of course preferred. The velocities of each segment and overall average velocity of each trajectory are calculated, and the averages of these are output to the display window, not the GUI. Histograms are shown, and the output can be saved / exported.

This is rather crudely written, and the outputs should be better displayed and integrated with the rest of the GUI.

Save / Load

Save Output

Saves the localization and linkage results, and other variables, in a MAT file. The most important array in the MAT file is the `objs_link` matrix. Each column corresponds to one particle in one frame. Rows indicate:

```
        x position, px (1.0 == the middle of the top left pixel, increasing right)
        y position, px (1.0 == the middle of the top left pixel, increasing down)
        brightness (integrated intensity)
        particleid
        frame
        trackid , determined by the linking function;
        sigma, px Gaussian width, or the square root of the second moment
        meand2, px^2, mean distance-squared between gradient lines and center (nonzero
only for 'radial' localization; small == good fit)
```

(I.e. The columns for which row 6 is "23" contain all the x,y positions of track number 23. The column for which row 6 is 23 and row 5 is 18 gives in rows 1 and 2 the *x*, *y*, position of track 23 in frame 18. Row 1 for all the columns for which row 6 is "23" , in MATLAB `objs_link(1,objs_link(:,6)==23)` denotes the x-positions of track number 23.)

If orientation is also determined, there are additional rows:

```
        theta, ellipse orientation angle, radians
        ra, px , ellipse semimajor axis
        rb, px , ellipse semimajor axis
```

Save TXT

Save the output as a simple tab-delimited text file. This saves just the *x* and *y* positions of each linked object, without additional information. Row 1 = *x* positions (px, with 1==the center of the upper left pixel) of object #1 in each frame in which it exists; Row 2 = *y* positions (px, with 1==the center of the upper left pixel) . Row 3 = *x* positions of object #2; Row 4 = *y* positions of object #2, etc. Note that the columns correspond to frames, but the set of frames in which each object exists need not be the same. (I.e. object 1 may exist in frame 1, 2, 3, 4, and object 2 may exist in frames 3, 4, 5, 6 – each would appear as a 4-column set of points in the output file.) The tab-delimited text file can be opened e.g. by WordPad or Excel. Note that the .MAT file output (with the object matrix construction) conveys much more information.

Load obj data

Loads output (`objs_link` array, processing and localization parameters, etc.) from a previously saved MAT file. Can import a variable with a name other than objs_link in the MAT file to be the new obj_link array – for example, output saved from culling tracks using **`cullTracksGUI.m`** .

MESSAGE

Displays messages produced by the program.

DISPLAY TRACKS

Show objects

If selected, superimpose found object positions for this frame on the display image.

Show IDs

If selected, superimpose found object IDs for this frame on the display image. If the objects have been linked across frames into tracks, the track IDs are used. If not, the particle IDs for this frame are used.

Show Tracks

If selected, and if objects have been linked across frames into tracks, superimpose the tracks from all frames on the display image.

Control Display

Object Information
If selected, clicking will show in the object information table (to the left) the properties of the object closest to the clicked point.

Pan
If selected, the user can pan the displayed image. (Irrelevant if not zoomed in.)

Zoom
If selected, the user can zoom into the displayed image. Double-click to return to the original scale.

[Object Information Table]

See *Object Information*, above.