The Zaslaver's Lab Multi-Animal Tracker V1.0 - The Manual

Eyal Itskovits

June 20, 2016

Contents

1 About this manual

This manual intends to instruct new users on how to use the Zaslaver's Lab Multi-Animal Tracker. It covers both the installation process and some basic usage example. It is recommended to follow the *examples* section successively as some examples are based on the previous ones.

2 General

The Multi-Animal Tracker is a *MATLAB*® (© The MathWorks, Inc., http://www.mathworks.com) code library that provides a suite of tools for acquisition, and analyses of different small model organisms, including, but not limited to, the *C.elegans* nematode.

2.1 The Multi-Animal Tracker suite

The Multi-Animal Tracker suites includes:

- WormsRecorder - a matlab interface for acquisition of behavioural assays.
- AnimalsTracker - Tracks worms' movement, and produces tracks objects that can be later analysed.
- Anlyeses Plugins - Plugins for out-of-the-box analyses of worm motility.

2.2 The AnimalsTracker module

The AnimalsTracker gets a video of behavioural experiment as an input and extract the tracks of the moving worms. To correctly track worms, and avoid tracking irrelevant objects in the video the tracker should be *trained* to identify worms (See the *examples* section for more information).

2.3 The WormRecorder module

The WormRecorder module provides a thin interface for the acquisition of behavioural assays videos. Additional documentation will be provided in the future.

2.4 The Analyses Plugins

We provide a small library of out-of-the-box analyses tools for extracted trajectories. One can readily use them to create plots based on a given assay. We encourage the users of this tracker to share their analysis tools with other labs (See *Examples* section for more information on usage of these tools).

3 Prerequisites

3.1 Software and Hardware prerequisites

- MATLAB (tested so far on 2014a, 2014b, 2012b). The following packages should also be installed:
  - MATLAB Image Processing package.
- At least 8GB of memory.
- Tested on Windows 7,8 and various Linux distributions.

For the WormRecorder module,

- Mathworks supported camera[1].
- MATLAB Image Acquisition package.

3.2 Video prerequisites

The tracker is pretty universal and can be used to extract trajectories from a wide variety of videos (there is no requirement to acquire movies with the WormRecorder). For accurate tracking, it is better to have:

- A frame rate of 1Hz or higher.
- Synchronized population of worms (or population with small variance in shape and size).

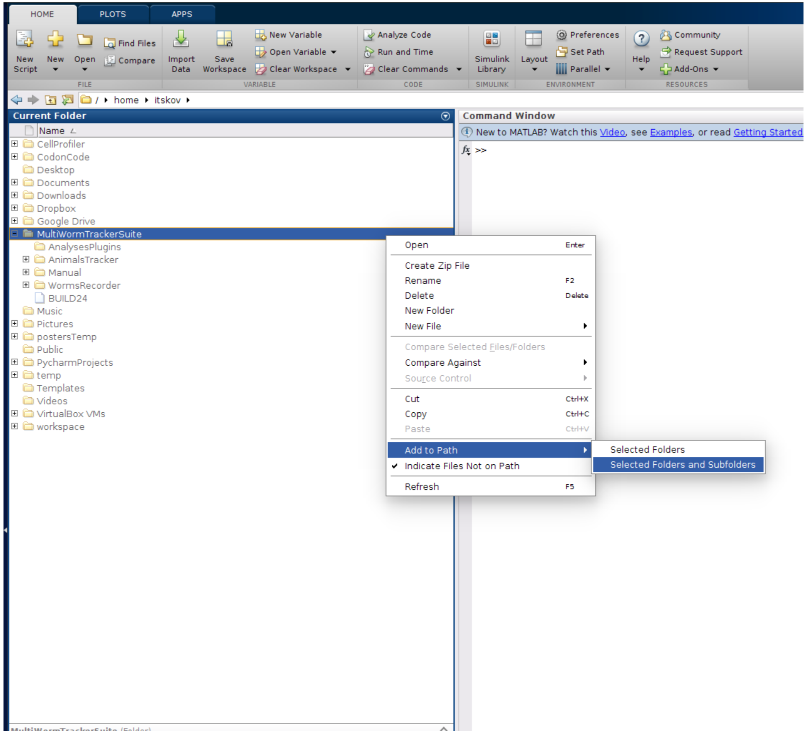
The tracker has been tested with worms occupying an average area of $\sim$ 30 pixels.

4 Installation

- Extract the content of the archive file *MultiwormTrackerSuite.zip* into a new directory of choice on your computer.
- Open MATLAB, and the directory to which you extracted the files on the directories tree.
- Right click on the directory. Go to the the *Add to Path* menu, and click on *Selected Folder and Subfolders.*

[1] Check Mathword supported vendors online.

Figure 1: Adding the MWT root directory to the path.



New to MATLAB? Watch this Video, see Examples, or read Getting Started

```
>>
```

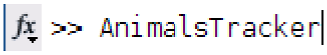
Installation is complete. In the example section we'll go over few usage scenarios from this point.

5 Examples

In the installation folder, under *SampleVideo* directory we provided a sample video which we'll use for the following usage examples.

5.1 Training the tracker and tracking worms

- Make sure you performed the installation steps mentioned in the *Installation* section.
- In MATLAB Command Window, type: *AnimalsTracker*

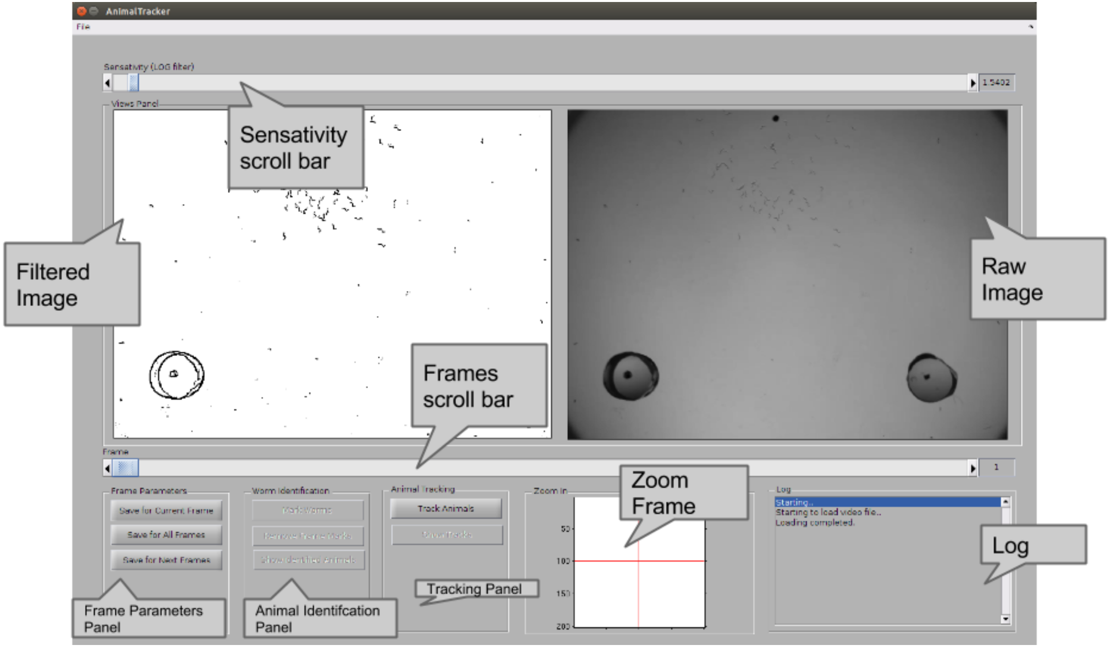


```
>> AnimalsTracker
```

- A graphical user interface will appear. Go to *file* menu, click *open* and choose the *SampleMovie.avi* that appears under *SampleVideo* directory in the MWT root installation folder.

Figure 2: The tracker's graphical user interface

.



```
Starting..
Starting to load video file..
Loading completed.
```

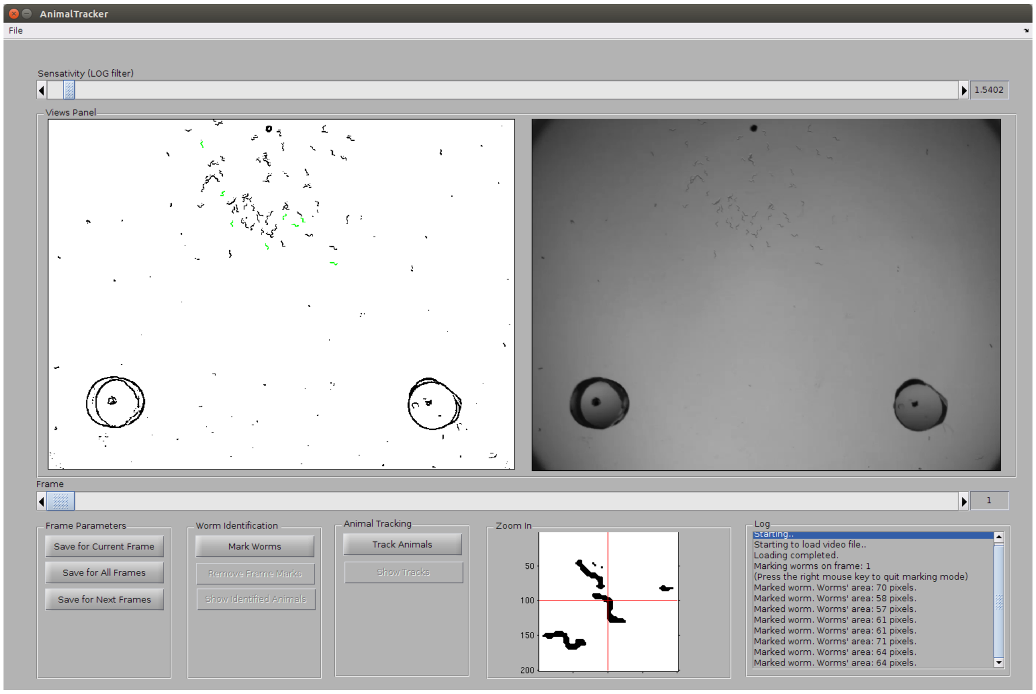
- The filtered frame may show the worms' shape as round blobs. To change this, draw the *sensitivity scroll bar* leftwards to increase filtering sensitivity or rightwards to decrease it until you can easily see the correct outlines of the worms on the one hand, and you don't get a lot of entities which are not worms (for the provided sample the optimal sensitivity is around 1.5).

- To save the right sensitivity threshold to the entire movie, press on the *Save For All Frames* in the *Frame Parameters* panel.

- Now that the sensitivity is set for all the frames we can start training the tracker to identify worms:

  - Press the *Mark Worms* button in the *Worm Identification panel*. The mouse cursor will change into a crosshair.
  - Click on worms[2] in the *Filtered Image* frame. You may use the *Zoom Frame* when the worms appear too small in the *Filtered Image* frame.
  - When you wish to stop marking worms, **right** click somewhere in the window, you should have around 24 worms marked in the whole video (the log window keeps tracks of marked worms).

[2] Try to avoid marking entities which are not single worms (i.e worms collisions or aggregates).

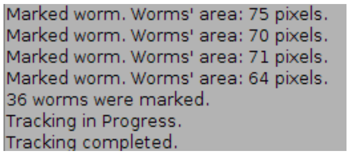


```
Starting..
Starting to load video file..
Loading completed.
Marking worms on frame: 1
(Press the right mouse key to quit marking mode)
Marked worm. Worms' area: 70 pixels.
Marked worm. Worms' area: 58 pixels.
Marked worm. Worms' area: 57 pixels.
Marked worm. Worms' area: 61 pixels.
Marked worm. Worms' area: 61 pixels.
Marked worm. Worms' area: 71 pixels.
Marked worm. Worms' area: 64 pixels.
Marked worm. Worms' area: 64 pixels.
```

- Now, use the *Frame scroll bar* to move to a different point in the video. And repeat the previous step (press *Mark Worms* and click on few worms). Go on and repeat this step for a couple of time points throughout the video.

- To verify how well the tracker was trained, choose a frame randomly. Then press on the *Show Identified Worms* button (Worm Identification Panel). The entities that were identified as worms should be marked green (or magenta, if they were also marked by you). If the tracker failed to mark the worms correctly, press the *Mark Worms* button again, and mark the worms that it had failed to identify.

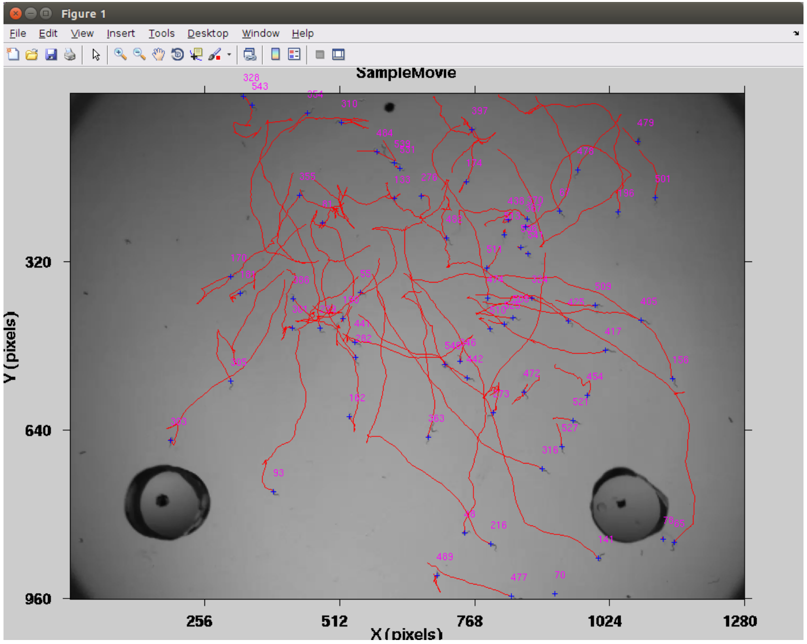
- When satisfied with the accuracy of the identification, we may proceed with the tracking by clicking the *Track Animals* button in the *Animal Tracking* panel. This process may take few minutes, during which messages will be printed in the main *MATLAB* window. Once finished, an indicative message will be printed in the *log* frame.



```
Marked worm. Worms' area: 75 pixels.
Marked worm. Worms' area: 70 pixels.
Marked worm. Worms' area: 71 pixels.
Marked worm. Worms' area: 64 pixels.
36 worms were marked.
Tracking in Progress.
Tracking completed.
```

- Once the tracking is complete, you may press the *Show Tracks* button in the *Animal Tracking* panel to see the tracking results.

Figure 3: Viewing the tracks after the tracking procedure is completed.

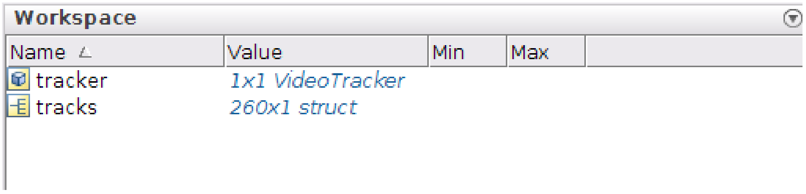


5.2 Exploration and tracks analyses

Apart from few out-of-the-box analyses modules we provide with the tracker, analysing the output of the MWT requires some basic skills in *MATLAB*. Here we will go through some common exploration and analyses of the tracking data.

5.2.1 Saving the tracks for further analyses

To save the tracks extracted by the tracker, go to the *File* menu and click on *Save Tracks*. You would be asked to choose a location in which a **.mat* file will be saved. This file contains all the information extracted from the video. Next, go back to the *MATLAB* main window, and find the file you've just saved on the directory tree. **Double clicking** on that file will introduce two new variables into your *MATLAB* workspace: *tracker* and *tracks*.



Workspace

Name	Value	Min	Max
tracker	1x1 VideoTracker		
tracks	260x1 struct		

The *tracks* variable is actually an array of all tracks acquired from the movie. The *tracker* is an object that holds information and functionality required for the exploration and analyses of the tracks. For instance,

```
>> tracker.numberOfFrames
```

will output the number of frames in the tracked video, and

```
>> tracks(54).numberOfSteps
```

will output the number of steps in track number 54 in the tracks list.

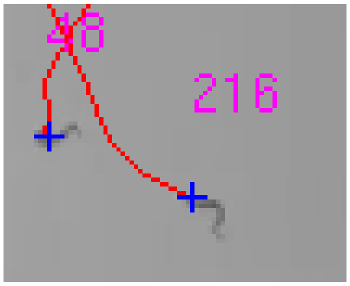
5.2.2 Viewing tracks

The *tracker* object allows viewing the trajectories of the tracks. For example, the following command,

```
>> tracker.viewTracks(tracks,[1, tracker.numberOfFrames]);
```

will view the tracks trajectories between frames 1 an the end of the video.

Each track is given an ID. The ID is printed right next to the worm when viewing the tracks using the *tracker.viewTracks* command[3].



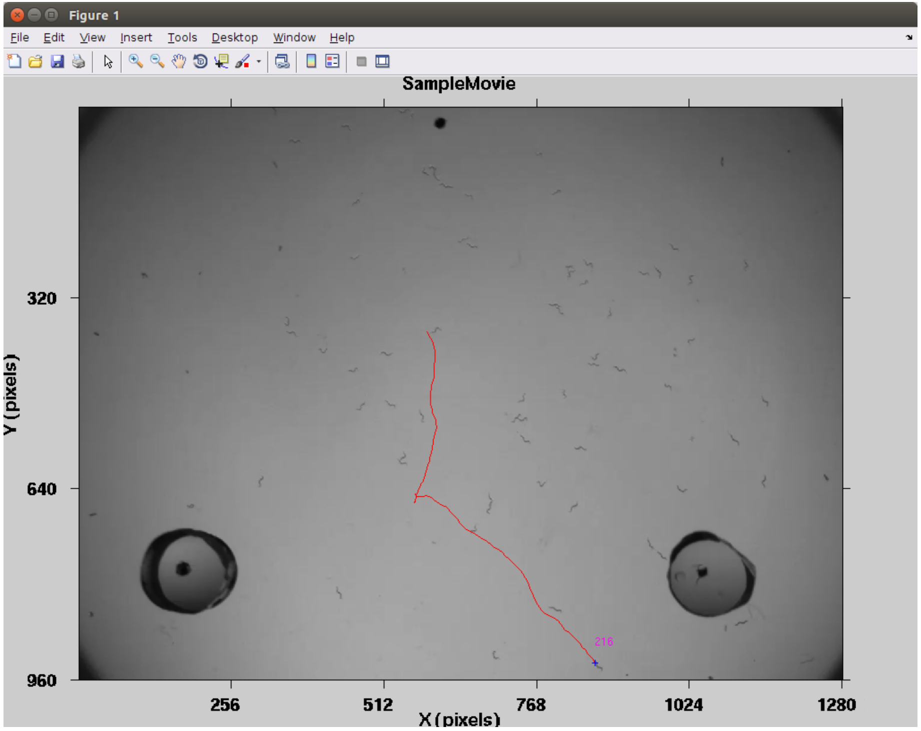
You can use the *viewTracks* command to view specific tracks based on their tracks ID. For example,

```
>> tracker.viewTracks(tracks([tracks.id] == 216), [1, tracker.numberOfFrames])
```

will view only the track which ID is 216.

[3] Keep in mind that its position in the list is not necessarily its ID.

Figure 4: Choosing tracks based on their track ID



5.2.3 Saving tracked videos

It is possible to save the videos with the tracked trajectories on top of it. This can easily be done by the same *viewTracks* method. The following command,

```
>> tracker.viewTracks(tracks,[1, tracker.numberOfFrames],0,'c:\MyVideos\')
```

will save the tracked video into *MyVideos* directory. The third parameter passed to the *viewTracks* method tells the tracker whether or not to show the worms' center of mass. Setting it to 1 will show the center of mass as a green cross.

5.3 Analysis of velocities

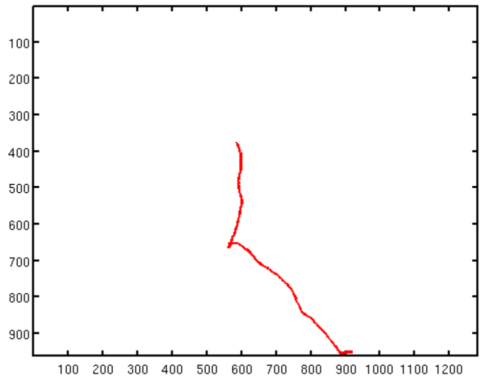
Each *track* in the *tracks* array contains multiple fields. The *path* field holds a list of coordinates that describe the track's trajectory over time. The track's *path* field contains 4 columns:

1. A serial frame number for this particular point.
2. The X coordinate.
3. The Y coordinate.
4. A boolean value that indicates whether this point was observed or predicted (the tracker tries to predicts tracks positions once it loses them).

The following code,

```
>> track216 = tracks([tracks.id] == 216)
>> plot(track216.path(:,2), track216.path(:,3),'r')
```

will plot[4],



We can calculate the size of step in each point time point, and average it over the entire video, and come up with the mean velocity of the worms in the current assay.

```
>> for t=tracks'
>> trackSteps = diff(t.path(:,2:3));
>> trackVelocities = sqrt(trackSteps(:,1).^2 + trackSteps(:,2).^2);
>> allVelocities = [allVelocities; trackVelocities];
>> end
```

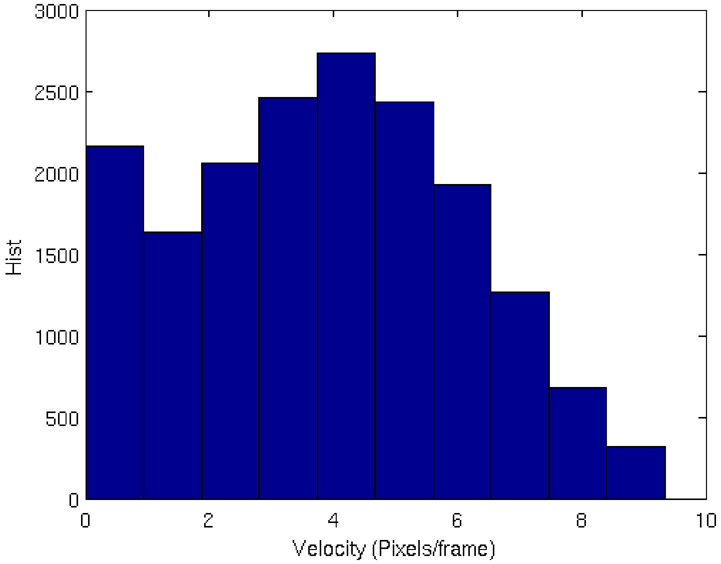
Before we continue, it would be a good idea to get rid of outliers that might be due collisions.

```
>> realVelocities = allVelocities(allVelocities < quantile(allVelocities,0.99))
```

And now, we can plot, for instance, the velocity histogram,

```
>> hist(realVelocities);
```

[4] Since the origin of axis in Matlab image is different from a Matlab plot, the following code will actually plot a mirror image. To solve this, exceute *set(gca,'YDir','reverse')* prior to the *plot* command.



In this assay, the velocity grossly distributes around 4 pixels/frame. However, a close-to-zero velocity is also quite common.

5.4 Using the Analyses Plugins library

We provide the following analyses tools as part of the *Analyses Plugins* library.

- PolygonDynamics - Allows marking a polygon on the field, and plot the number of worms inside the ploygon as a function of time.
- AttractionVectorField - Plots a vector field that describes the average worms movement.

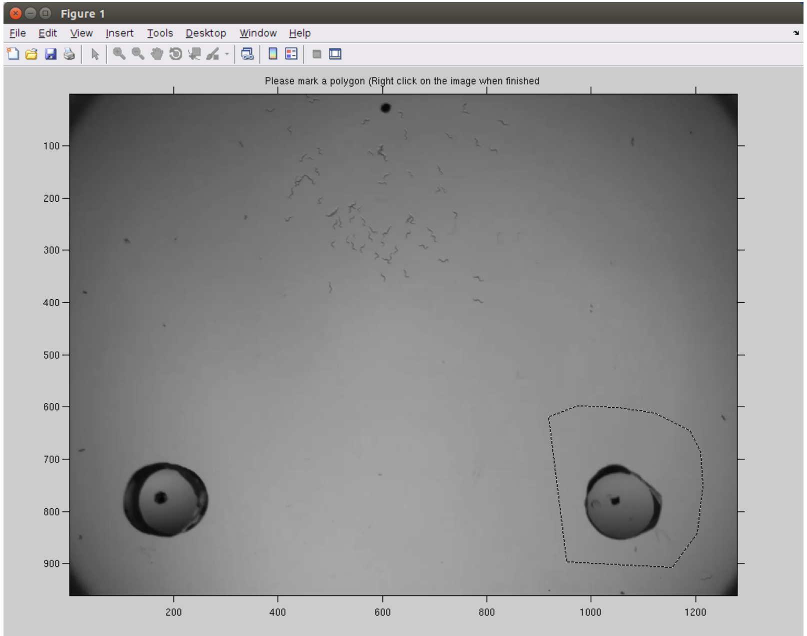
5.4.1 Polygon Dynamics

Execute the following command in MATLAB command window:

```
>> polygonDynamics(tracker,1);
```

The first frame of the the video should appear, and you'd be ask to mark a polygon of interest on the image. Use the left button key to mark points, and the right mouse key to stop the marking. Mark, for example, the chemoattractant point (right circle-shaped entity on the *SampleVideo*).

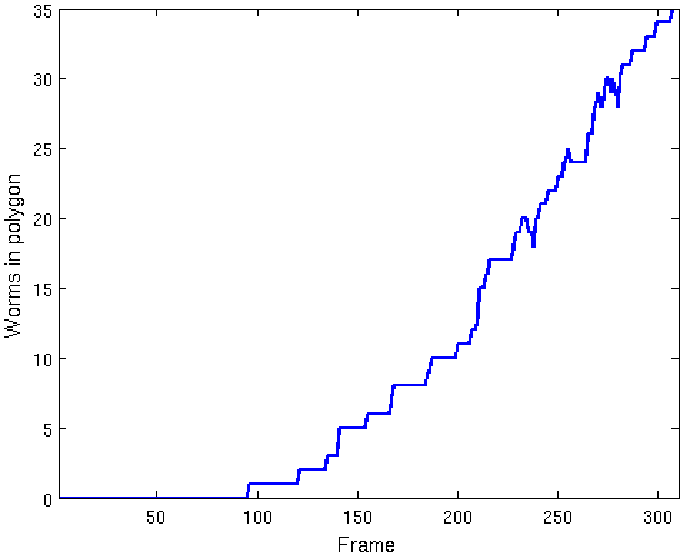
Figure 5: Marking a polygon of interest.



Please mark a polygon (Right click on the image when finished

Once you're done marking the polygon, a curve will appear showing the number of worms in that marked polygon as a function of time -

Figure 6: Number of worms in the polygon of interest.



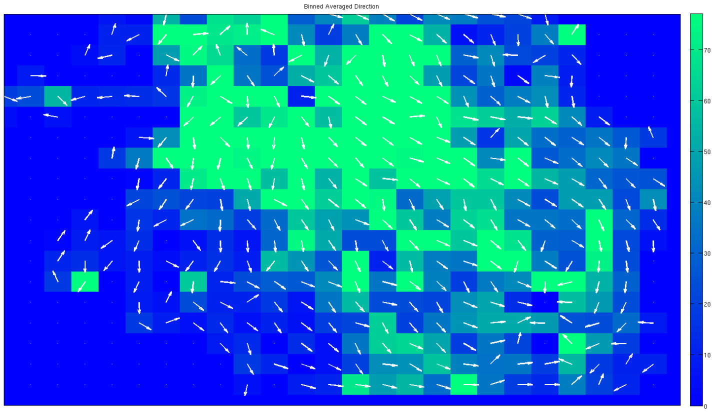
5.4.2 Attraction Vector Fields

The following command,

```
>> attractionVectorField(tracker)
```

will plot a vector field that shows the average direction of the worms in different areas of the field. For the provided *SampleVideo* the result would be,

Figure 7: Worms' directionality vector field.



6 License Information

This library is provided as is under the GNU General Public License v3.