Readme for FLLIT

Contents

1   Getting Started with FLLIT

1.1   Overview of the FLLIT program

The Feature Learning-based LImb segmentation and Tracking (FLLIT) program is compiled on MATLAB R2016a and runs on a Linux OS (eg. Ubuntu).

The program processes input data consisting of 512 pixels × 512 pixels video image sequences of the sample animal, e.g. *Drosophila* fruitfly or spiders. Currently, input supports TIFF and PNG files. The video should be in made in a single channel (grayscale), and taken at 250 frames per second or higher speeds. Currently, TIFF and PNG files are supported. The field of view should be held steady without movement throughout the video. The task is to:

1. Identify (at a pixel level) the legs of the sample animal via the 'Segmentation' module;

2. Track the leg claw positions via the 'Tracking' module;

3. Report tracking results consisting of:

    - Body centroid position
    - Angles of rotation of the body-axis (relative to the y-axis)
    - Leg claw positions in arena-centered frame of reference: Positional coordinates of each leg claw in each frame based on arena coordinates
    - Leg claw positions in body-centred frame of reference: Positional coordinates of each leg claw in each frame with reference to the body centroid

    These results can be further processed with the 'Data Processing' module and visualized via the 'Make Video' module.

The precompiled version of FLLIT can be downloaded from here.

As illustrated in Figure 1.1, the FLLIT program directory consists of the following folders and executables (as well as readme and license files):
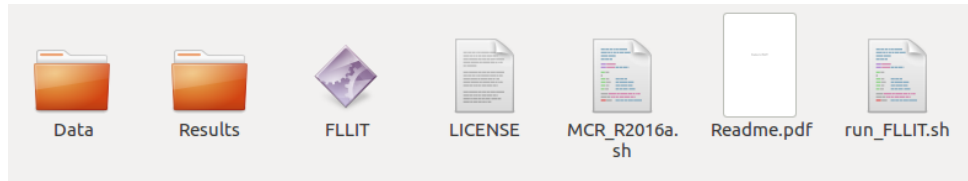

Figure 1.1: Folder Structure and Executables

- MCR_R2016a.sh: This bash script installs the MATLAB runtime libraries.

- Data folder: All image datasets to be analyzed are placed in the 'Data' folder.

- Results folder: After analysis with FLLIT, all results can be retrieved from the 'Results' folder.

- FLLIT and run_FLLIT.sh: These are the executables for the FLLIT program.

In the following sections, installation and execution of FLLIT on various Operating Systems will be outlined in detail:

- Ubuntu

- Windows

- MacOS

1.2 Running FLLIT on Ubuntu

1.2.1 Installation of required MATLAB runtime libraries

Open a terminal in the FLLIT directory and issue the following command:

```
bash MCR_R2016.sh
```

This will take a while to download (size ~ 1 GB) and install the MATLAB runtime libraries to the following location:

```
$HOME/MCR
```

Typical installation time on a desktop computer takes about 5 minutes.

1.2.2 Executing the program

To run the FLLIT program, first open a terminal in the FLLIT directory. To ensure that executable rights are accorded to the application, the following commands should be issued:

```
chmod +x run_FLLIT.sh FLLIT
```

The FLLIT program can then be executed with the following command:

```
bash run_FLLIT.sh $HOME/MCR
```

1.3   Running FLLIT on Windows

On Windows, it is necessary to deploy FLLIT with Docker.

Download and install Docker for Windows.

To enable execution of GUI applications in a Docker container on Windows, one has to install VcXsrv Windows X Server.

Make sure Docker and VcXsrv is up and running before starting FLLIT. On starting VcXsrv, configure the settings as follows.

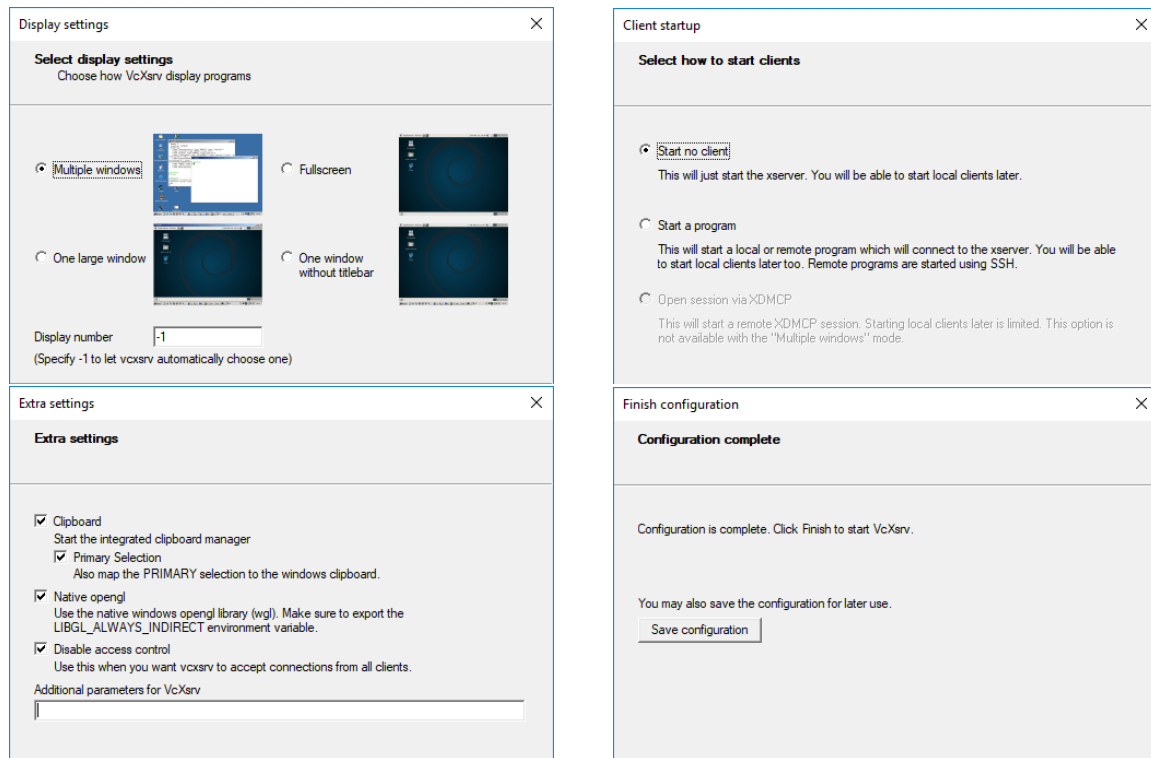Figure 1.2: Configurations for VcXsrv.

To run FLLIT, open a powershell and issue the following command

```
docker run -v <FLLIT_directory>:/FLLIT -e DISPLAY=<ip address>:0 feijianke/fllit
```

Here <FLLIT_directory> denotes the absolute path of the FLLIT directory and <ip address> denotes the computer's ip address. For example if FLLIT is located in D:/Downloads/FLLIT-master/Compiled and the ip address is 123.45.678.90, then the command will be

```
docker   run   -v   D:/Downloads/FLLIT-master/Compiled:/FLLIT   -e   DIS-
PLAY=123.45.678.90:0 feijianke/fllit
```

For first time execution, this will take some time to pull the fllit docker image from Docker Hub.

1.4 Running FLLIT on MacOS

On MacOS, it is necessary to deploy FLLIT with Docker.

Download and install Docker for MacOS.

Next, install and start socat by opening a terminal and issue the following commands

```
brew install socat
```

```
socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\" & disown
```

To enable execution of GUI applications in a Docker container on MacOS, one has to install XQuartz. Start XQuartz. Open up preferences and check the "Allow connections from network clients" in the **Security** tab.
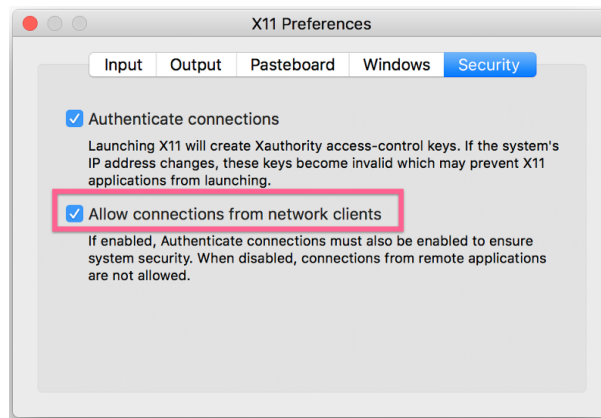

Figure 1.3: Check the "Allow connections from network clients" in the **Security** tab.

Make sure Docker is up and running before starting FLLIT. To run FLLIT, open a terminal and issue the following command

```
docker run -v <FLLIT_directory>:/FLLIT -e DISPLAY=<ip address>:0 feijianke/fllit
```

Here <FLLIT_directory> denotes the absolute path of the FLLIT directory and <ip address> denotes the computer's ip address. For example if FLLIT is located in $HOME/FLLIT-master/Compiled and the ip address is 123.45.678.90, then the command will be

```
docker run -v $HOME/FLLIT-master/Compiled:/FLLIT -e DISPLAY=123.45.678.90:0 feijianke/fllit
```

For first time execution, this will take some time to pull the fllit docker image from Docker Hub.

2   Program Workflow

The interface of the FLLIT program is shown in Figure 2.1. To begin, the user selects the image folder containing the image frames of the source video.
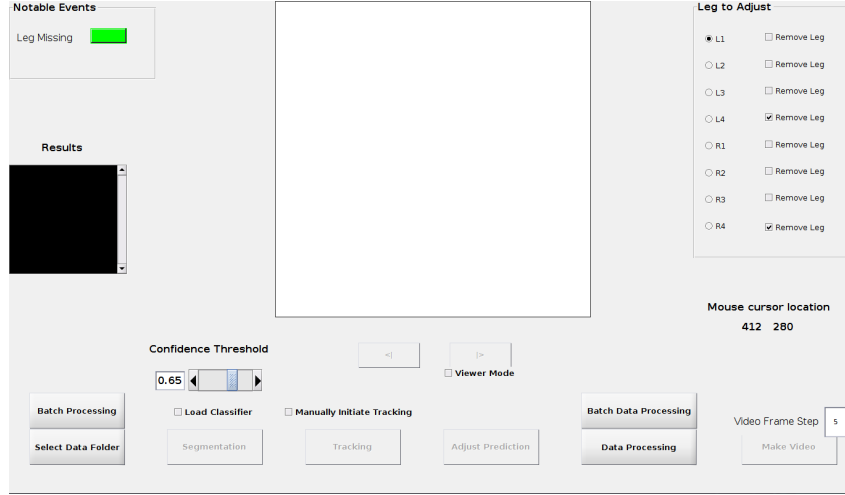

Figure 2.1: Graphic user interface of the FLLIT program

The workflow pipeline is depicted in Figure 2.2.

Segmentation is first run to identify legs. Tracking is then run to determine the body centroid trajectory, rotation angle and individual trajectories of each leg. Subsequently, the tracking results can be retrieved and visualized in the form of a short movie, and analysed using the Data Processing functions.

The entire workflow is automated. Expected run time for a demo dataset of 1000 video frames on a typical desktop computer is about 70 minutes. Mulclawle videos can be processed at a time by selecting the Batch processing option.
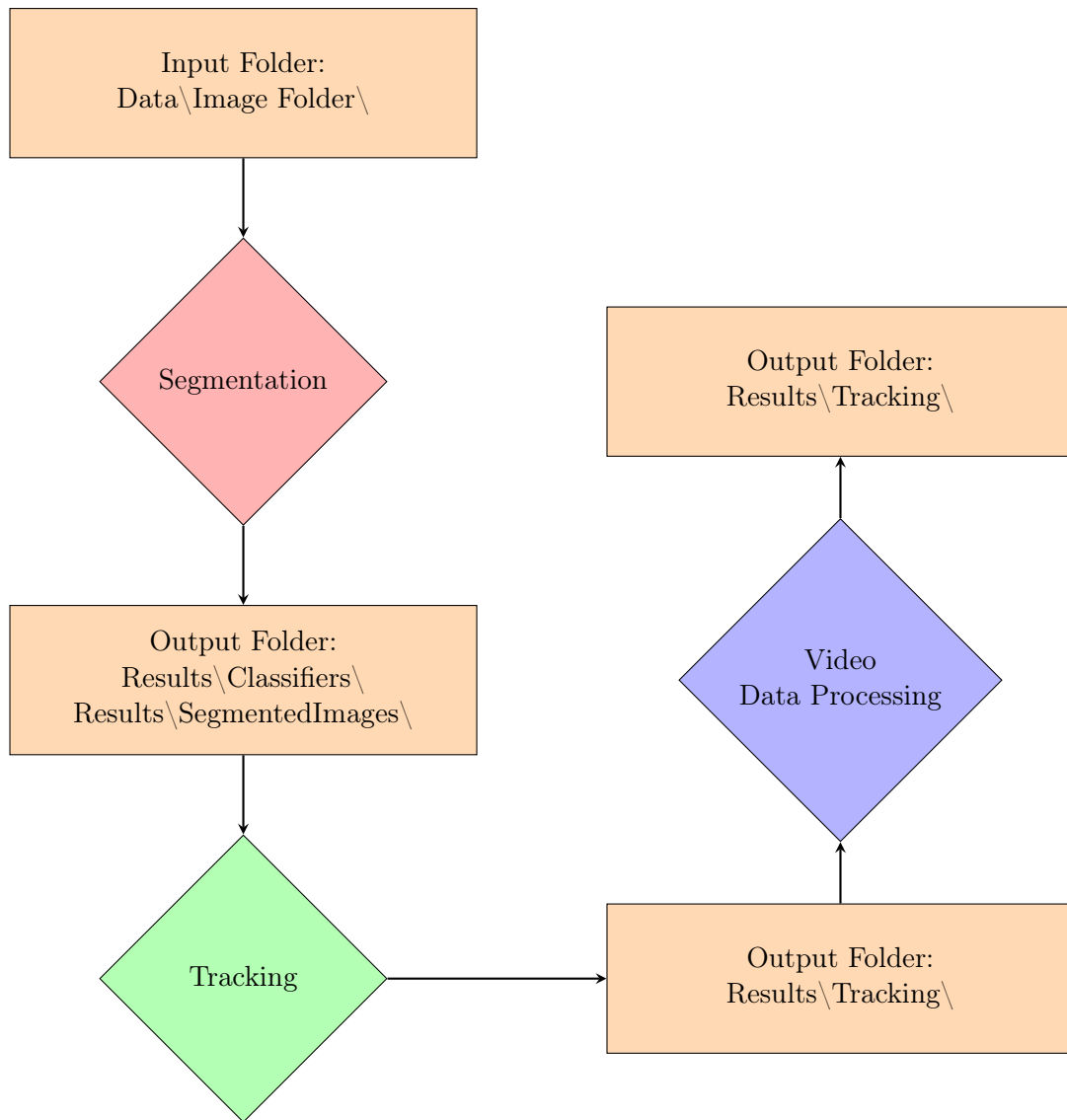
Figure 2.2: Flow chart of program

2.1 Segmentation

Segmentation is first carried out to identify the legs of the sample animal and is the most time-consuming portion of the analysis procedure. As illustrated in Figure 2.3, when the 'Segmentation' button is clicked, the program will proceed to extract the sample silhouette by automated background subtraction, and also generate training samples on a subset of the images. These training samples are used to train a classifier. After a classifier is learned, it is applied to novel images. The classifier assigns a confidence score to each pixel, ranging from 0 (least likely to belong to a leg) to 1 (most likely to belong to a leg). The default confidence score threshold is set at 0.65 and this can be adjusted by moving the slider bar or typing the desired threshold into the box.

The user also has the option to load an existing background image (which would be beneficial for videos where the sample moves slowly or not at all). This can be done by creating a background directory folder in the data folder and placing the background image (.png format Figure 2.4a) inside. There is also the option to load an existing classifier.
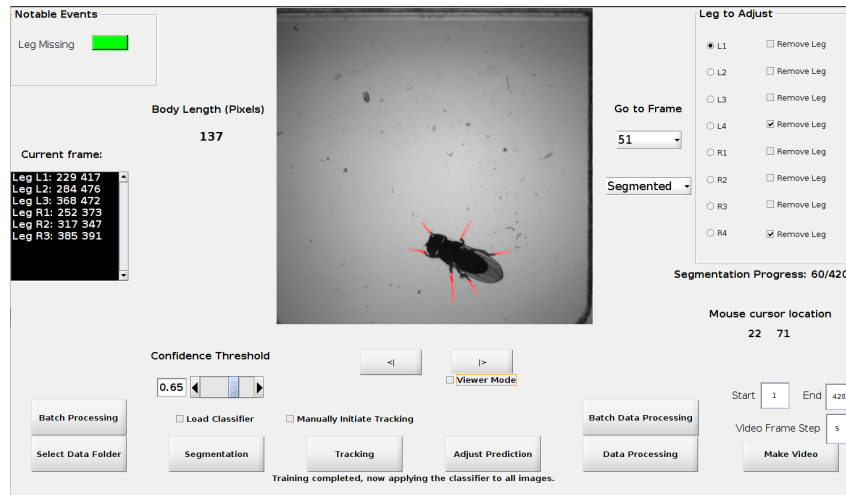

Figure 2.3: An example of a segmented image

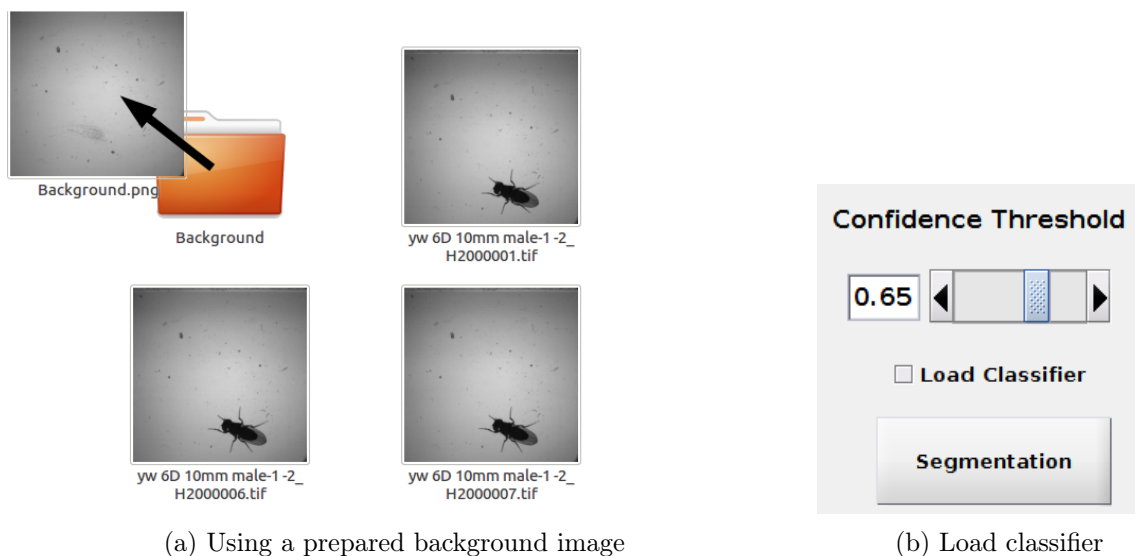(a) Using a prepared background image

(b) Load classifier

Figure 2.4: User options for segmentation

2.2 Tracking

Tracking is initiated by clicking on the tracking button after segmentation has been completed (Figure 2.5). Tracking will automatically initiate on the first frame where all legs can be confidently assigned and the legs will be assigned the labels L1−R3, corresponding to left fore leg, left middle leg, left hind leg and right fore leg, right middle leg and right hind leg respectively. The leg claws are circled out and their corresponding pixel coordinates are shown on the text display box to the left of the user interface.
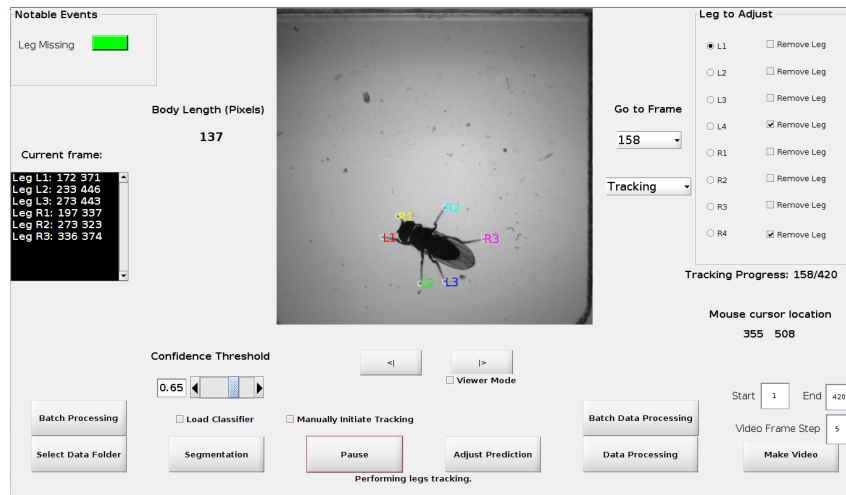

Figure 2.5: Tracking

Once tracking has started, the body length of the animal (anterior-most position on head to posterior-most position on wings) will also be automatically shown. During tracking, the green button next to the Legs Missing indicator will turn red if a leg is not identified. To correct an error, click 'Pause' followed by 'Adjust Prediction'. The user can then select a leg from the 'Select Leg for Labelling' panel and double click on the image to annotate or correct the claw position. The claw position coordinates of a leg can also be directly changed from the text display box on the left. Finally, the user can save corrections or discard changes to exit the 'Adjust Prediction' mode.

Navigating to the previous and subsequent frames can be done by clicking on the $<\|$ and $\| >$ buttons or via the 'left' and 'right' keys from the keyboard. The next frame $\| >$ button will perform the tracking algorithm on the subsequent frame. The 'Viewer Mode' box can be checked if the user wishes to browse through existing tracking data without running the tracking module. There is also a dropdown bar which allows selection of the frame of interest.

There is also the option to remove or include more legs during the tracking process. For example, decreasing the number of tracked legs allows analysis of datasets involving amputated flies, and inclusion of up to 8 legs allows for tracking of spider legs.

In some scenarios, the user may wish to manually initiate tracking from a particular frame, eg. as required for spider leg tracking. This can be done by checking the 'Manually Initiate Tracking' box. Clicking on the 'Tracking' button will turn the button text to 'Initial'. From here, the user needs to activate the 'Adjust Prediction' mode and label all the legs accordingly. Clicking on the tracking button again will then allow tracking to proceed normally from the user-initiated frame.

2.3   Video

Once tracking is complete, the tracking results are automatically saved to the Results folder as .csv and .mat files. Tracking results can also visualized with the 'Make Video' function (Figure 2.6).
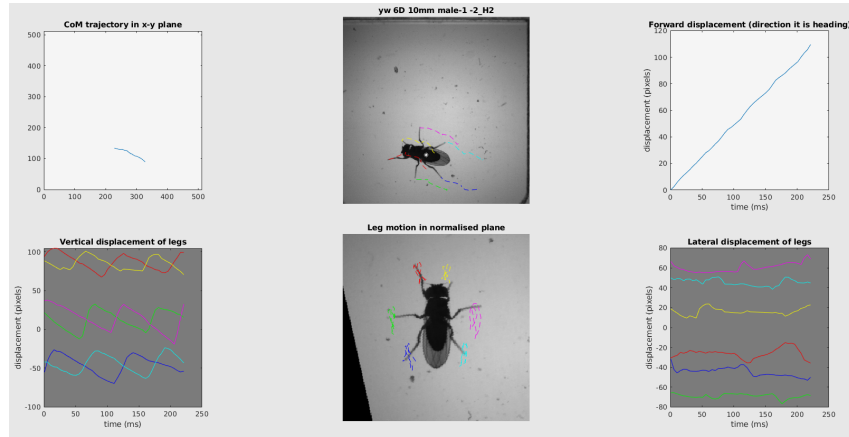

Figure 2.6: Sample video

2.4   Data Processing

The raw tracking results can be processed on clicking the 'Data Processing' button. The user can input the Frame Skip per Second (FPS) and image size (mm) of the dataset. The automatic data analysis will output the following parameters and plots:

Raw data

1. Body position: Positional coordinates of the body centroid in each frame.
   Output: Data saved to first two columns of CoM.csv.

2. Body trajectory: Angle of rotation of the body axis in degrees (relative to the y-axis).
   Output: Data saved as third column of CoM.csv.

3. Arena-centred leg claw positions: Positional coordinates of each leg claw in each frame based on arena coordinates.
   Output: Data save to trajectory.csv.

4. Body-centred leg claw positions: Positional coordinates of each leg claw in each frame with reference to the body centroid.
   Output: Data save to norm_trajectory.csv.

Body motion

1. Body length (in mm): Length of the sample animal estimated in each frame (anterior-most position on head to posterior-most position on the wings).
   Output: Data saved to bodylength.csv.

2. Instantaneous body velocity (in mm/s): Instantaneous velocity of the body (centroid) in the sample animal.
   Output: Data saved to BodyVelocity.csv, plot saved as BodyVelocity.pdf.

3. Turning points of the body trajectory: To locate the turning points, the trajectory is reduced to a piecewise-linear curve using the Dougl asâĂŞPeucker algorithm, following which a turning event is identified as involving an angle > 50 deg between two neighbouring linear segments constituting the simplified trajectory.
   Output: Plot saved as BodyTrajectory.pdf.

Individual stride parameters A stride event is identified as the movement of a leg with respect to the arena, lasting a minimum of 15ms.

1. Stride duration (in ms): The duration of a stride event.

2. Stride period (in ms): The duration from one stride event to the next.

3. Stride displacement (in mm): Displacement of the leg claw during a stride event.

4. Stride path covered (in mm): Total path covered by the leg claw during a stride event.

5. Anterior extreme position (in mm) [1]: Landing position (relative to the body) of a leg claw at the end of a stride event.

6. Posterior extreme position (in mm) [1]: Take-off position (relative to the body) of a leg claw at the start of a stride event.

7. Stride amplitude (in mm): Displacement along the direction of motion for a stride event.

8. Stance linearity (in mm) [1]: Defined as the deviation of the stride path from a curve smoothed over (at 20ms intervals) the corresponding anterior and posterior extreme positions of the stride.

9. Stride stretch (in mm): Distance of the leg claw position from the body centre in the middle of a stride event.

Output: All these parameters are saved to StrideParameters.csv.

Leg motion

1. Leg speed (in mm/s): The instantaneous speed of each leg.
   Output: Data saved to LegSpeed.csv, plot saved as Gait.pdf.

2. Gait index [1]: This measures the type of gait coordination exhibited by the (six-legged) sample animal during its motion. A gait index of 1 corresponds to a tripod gait, -1 corresponds to a tetrapod gait while 0 constitutes an non-canonical gait. In our implementation, the gait index is obtained by a moving average over a 120 ms window.
   Output: Data saved to GaitIndex.csv, plot saved as GaitIndex.pdf.

3. Movement percentage: Percentage of the time that a leg is in motion.
   Output: Data saved to LegParameters.csv.

4. Mean stride period (in ms).
   Output: data saved to LegParameters.csv.

5. Footprint regularity (in mm): Measured as the standard deviations of the posterior and anterior extreme positions of a leg.
   Output: Data saved to LegParameters.csv.

6. Leg trajectory domain area (in mm^2): The area of the minimal convex hull that contains the entire leg trajectory in the body-centred frame of reference.
   Output: Data saved to LegParameters.csv, plot saved as LegDomain.pdf.

7. Length and width of the leg trajectory domain (in mm): Obtained via the maximum projected distance of the claw positions onto the major (domain length) and minor (domain width) principal axes.
   Output: Data saved to LegParameters.csv.

8. Leg domain intersection/overlap (in mm^2): The intersection/overlap between each possible pair of leg domains. Output: Data saved to LegDomainOverlap.csv.

9. Stance width (in mm): Average of the distance between the AEP and PEP of the left and middle legs. Output: Data saved to StanceWidth.csv.
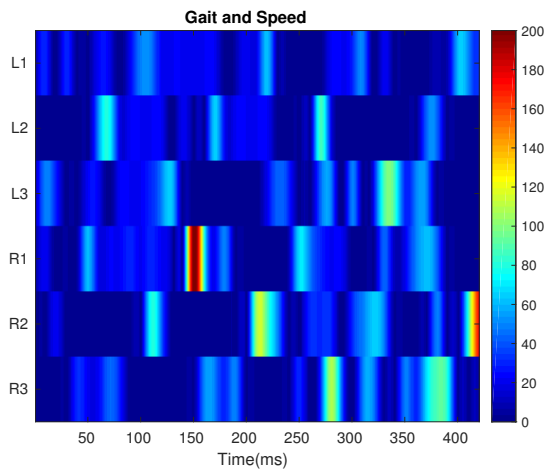
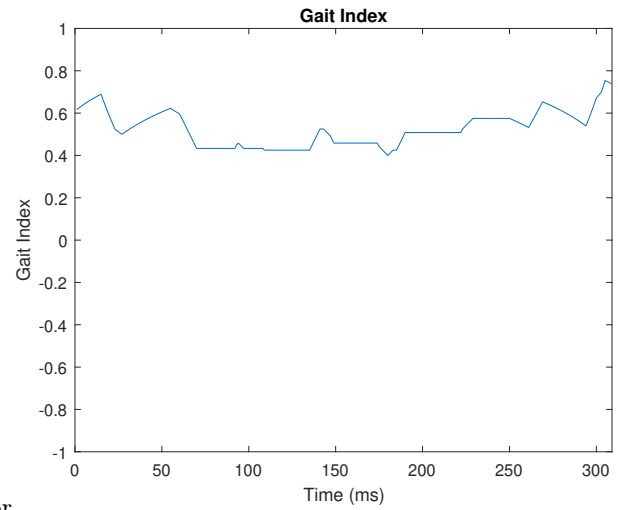Sample plots are shown in figure 2.7.
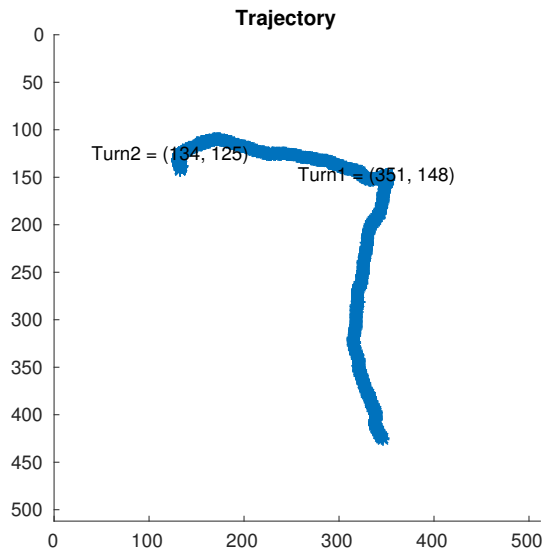
(a) Trajectory with turning points marked out. The units here are pixels.
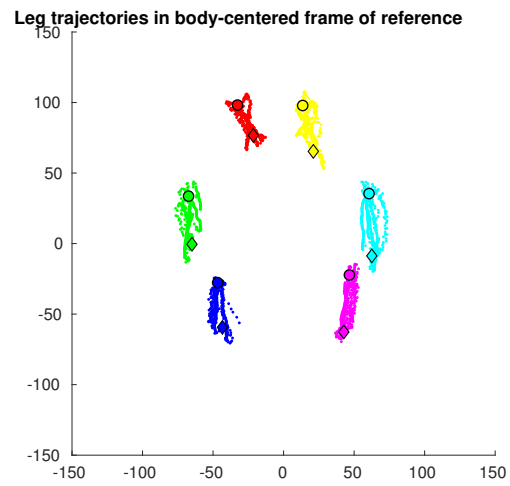
(b) Leg trajectory domains. The circle/diamond marker denotes the mean landing/take-off position.

(c) Gait plot for a sample drosophila dataset. Color intensities denote leg speeds in units of mm/s.

(d) Gait index plot

Figure 2.7: Sample output

Bibliography

[1] Mendes, César S and Bartos, Imre and Akay, Turgay and Márka, Szabolcs and Mann, Richard S. Quantification of gait parameters in freely walking wild type and sensory deprived Drosophila melanogaster. *elife*, 2:e00231, 2013.