Nematodes Project

Kyle L. Moy

December 22, 2015

Contents

1 Introduction

Welcome to the Nematodes project! The aim of this document is to serve as a comprehensive user manual for all of the tools that I created for this project. Additionally, you will find a sort of postmortem discussion about our approaches, what worked, what didn't work, and other details.

Here's a list of people (as far as I know) who have ever been involved in this project.

Valerie Simonis wrote the first version of the tracker in Python[1], and worked with Christopher Brandon in the design and testing of the tracking hardware.

Chandler Gegg wrote Python code for calculating movement features from the centroid information given from Ron's work.

Ron Neihaus wrote a great deal of MATLAB code for extracting morphological features from video data, as well as movement features and centroid locations. However, limitations in MATLAB and minor bugs prevented us from using his code on long videos. However, since no one remaining in the lab (ie. me) was familiar with his code or MATLAB in general, we pursued other alternatives for extracting movement features.

Max Weiyu Li explored new ways to analyze movement feature data in R, researched Lvy flight, the use of step-lengths, Locality (Proximity), and how these concepts can be applied to our data. Some of his work is discussed in [2].

Huu Phoc Tran continued Max's work in R, refining the step length discretization process and then went on to do distribution analysis of our step length data. This work is discussed in [2].

Evan Story developed a web application in Javascript for visualizing/graphing movement feature data. (*http://medixsrv.cstcis.cti.depaul.edu/nematodes/visualizer/*)

Daniela Raicu, Ph.D involved in basically everything.

Jacob Furst, Ph.D involved in basically everything.

Hongkyun Kim, Ph.D is a professor from Rosalind Franklin University, in the Department of Cell Biology and Anatomy. He initially conceived the project, and uses the tracker to produce video recordings. He is the biological expertise in this project.

Christopher Brandon, Ph.D designed and built the tracker hardware.

2 Information

2.1 Data Directory, Projects

In `//MEDIXSRV/Nematodes/data`, a folder exists for each video recording, named by strain type (Ex. N2, tph1, egl19) and the food environment (f for on-food, nf for off-food), followed by a unique index. These folders contain all files related to each video, and are henceforth are referred to as "projects." Each folder have the following subdirectories:

data The data folder is for all calculated information. Generally, everything in this folder should be "finished" data in CSV format.

input This folder contains the video recording as a series of images. From the Java tracker, this should come as a series of JPG files named in sequence with a 7 digit number, padded with zeroes (Ex. 0000001.jpg).

log This contents of this folder aren't particularly well defined, but I use it for "raw" data files, including log files from the tracker, and data from the segmentation process.

output This folder is more or less deprecated. The segmentation process has the option of rendering the segmented image for every frame, which would be placed in this folder. However, this feature only exist because I modeled my software after Ron's MATLAB code- I later realized this doesn't have much use.

visual This folder contains graphs, paths, and any other data visualizations.

2.2 Data Files

Many of the files in the *data* subdirectory are the results of intermediate calculations, and only exist to make re-calculation (parameter changing) quicker. The file of most importance is `movementFeatures.csv`, which should include the information from all other files. The exception being `occupancy.csv`.

Below are the column headings for each comma separated values file, followed by the unit. All values are either signed integers or signed doubles.

`centroid.csv`

Frame	Time Elapsed	Time Δ	X	Y
Frame	ms	ms	px	px

`movementFeatures.csv`

Frame	Time Elapsed	Time Δ	X	Y	Speed	Acceleration	Angle	Angular Velocity
Frame	Seconds	Seconds	px	px	mm/s	mm/s^2	deg	deg $/s$

`movementFeaturesBinned.csv`

Minute	Speed	Acceleration	Angle	Angular Velocity
Minute	mm/s	mm/s^2	deg	deg $/s$

`occupancy.csv`

Time	Unique Cells Visited
Seconds (1 min intervals)	Cells

`tracker.csv`

Frame	Time Elapsed	Time Δ	X Step	Y Step	Is Moving
Frame	ms	ms	0‡	0‡	boolean

‡ X and Y Step are supposed to keep track of the camera offset, and were values recorded by Valerie's tracker, and are kept here for compatibility in other calculations but are not actually used. These values are also recorded by the Java tracker, however it is more accurate to calculate the offset by other means. Point is, these values will always be 0.

`feature.log`

Output from WormSeg.

Frame	X	Y	Area
Frame	px	px	px

`performance.log`

Output from WormSeg.

Computer	FPS	Time Per Frame
Name	Frames	ms

`log.dat`

Output from the tracker. This is binary data, can't be read as text.

Frame	Time Elapsed	X	Y
Frame	ms	px	px
int	long	int	int

3 Instruction Guide

3.1 Tracker Software

The first version of the tracker is discussed in detail in [1].

3.1.1 The GUI

3.2 Filing new data

The workflow was designed to process data in batches, so I highly recommend completing each section for every new data set before moving on to the next. Although these sections are divided by task, they're meant to be followed sequentially. First, you need to set up the folder structure in the "data" directory on the MEDIXSRV. You could do this manually, or you can use the WormSeg software to automatically generate them:

1. Navigate to `//MEDIXSRV/Nematodes/src/WormSeg/util/`. This folder contains scripts for executing various parts of the Java software. I never got around to writing a GUI, sorry.

2. Launch `server.bat`. This is the core of the WormSeg program, "WormSeg Host.bat" specifically puts in server mode, but we won't be needing that just yet.
3. To create a new project folder, use the command "project create [name]". Create a project folder for each new video- the naming is up to you, because this predates our current naming scheme. Hint: you can press the Up Arrow key to input your last command, this is helpful for creating many projects with similar names (e.g. N2_f1, N2_f2, N2_f3, etc...).
4. A new folder under `//MEDIXSRV/Nematodes/data` will be created for each project, including the necessary subdirectories, as well as a copy of the template WormSeg configurations.
5. You may close the WormSeg terminal.
6. If you are adding data collected using the Python tracker, refer to the **Legacy Data** section before continuing.
7. For each video, move the video frames into the respective project's **input** folder, and the "log.dat" file into the "log" folder. It is likely that these files will be tarballed (.tar.gz), which will require extraction twice using archival software such as 7-Zip, which is available online. The log file will be among the images.

Legacy Data Video data produced by Valerie's python tracker needs some preparation before we can use the WormSeg program. If you are working with data from the Java tracker, skip this section.

Data from the original tracker comes as two files per recording; one **AVI** video, and one **LOG** type file. Parsing the "legacy" log file is now automated by the movement feature calculation software later, but the video still needs to be rendered as an image sequence (JPG) before we can continue. To do this, we will use a free open source software called "Virtual Dub," which should be available on MEDIXSRV.

1. Remote Desktop into MEDIXSRV (`medixsrv.cstcis.cti.depaul.edu`). You will be reading the video from MEDIXSRV and then writing images back to it; although this can be done over the network via shared folders, I don't recommend it as it would take a very long time. Remoting into MEDIXSRV allows you to read and write locally.
2. Locate and execute `//medixsrv/Nematodes/src/utils/vdub/VirtualDub.exe`.
3. Open a video recording by clicking **File** $\rightarrow$ **Open video file...** or by simply clicking and dragging the video into the Virtual Dub window.
4. Now, we need to truncate the video to remove frames where the worm is not present (the tracker has lost the worm). Select a time in the video just before the tracker has lost the worm by clicking on the seek bar. If the worm is present during the whole video, skip to step 7.

5. Mark this time as the end of the crop by clicking **Edit** → **Set selection end**, or by pressing the **END** key.

6. Crop the video by clicking **Edit** → **Crop to selection**.

7. Add this job to the queue by clicking **File** → **Queue batch operation** → **Save image sequence...**. Alternatively, if you are only processing one file, you can use **File** → **Export** → **Image sequence...**

8. Fill in the settings exactly as shown below.

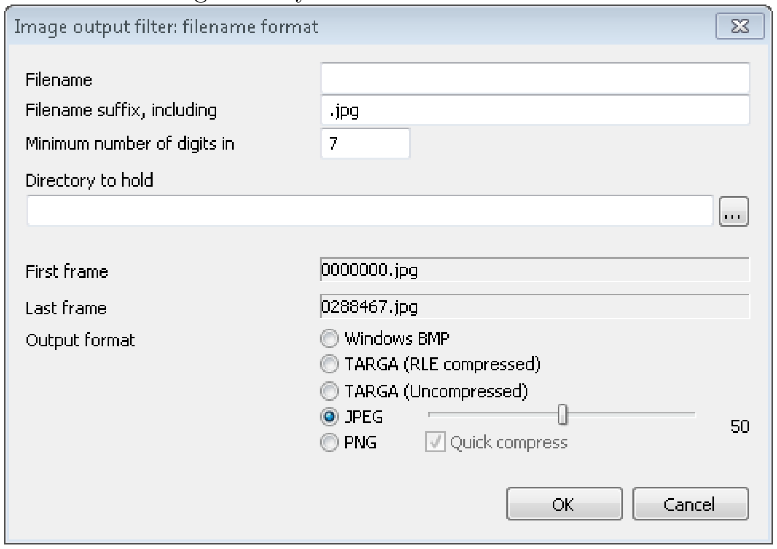

Image output filter: filename format

Filename	
Filename suffix, including	.jpg
Minimum number of digits in	7
Directory to hold	
First frame	0000000.jpg
Last frame	0288467.jpg
Output format	( ) Windows BMP ( ) TARGA (RLE compressed) ( ) TARGA (Uncompressed) [● JPEG] 50 ( ) PNG ☑ Quick compress

9. For the "Directory to hold" path, use the "input" directory of the corresponding project folder. (Ex. `D:/Nematodes/data/N2_f1/input`) **Be sure to update this field between projects!** Click OK to continue.

10. For a single video, you are now done after the video is converted. Otherwise, repeat steps 3 - 9 for multiple projects.

11. Once you've added all videos to the queue, open the Job Control window by clicking **File** → **Job control...**, or by pressing **F4**.

12. Ensure that all the listed jobs are correct, and then click Start. This will now begin converting all jobs, this may take a long while.

Configuring Frame Count After video frames have been placed in the "input" directory, the "frame-start" and "frame-end" parameters in the configuration file `config.yml` must be updated to reflect the start and end point for the WormSeg segmentation process. If all frames are to be used, the configuration file can be updated automatically using the ConfigureFrameCount utility.

1. Remote Desktop into MEDIXSRV (`medixsrv.cstcis.cti.depaul.edu`). Note that the ConfigureFrameCount **must** be run locally on MEDIXSRV.

2. Launch the ConfigureFrameCount utility (`//MEDIXSRV/Nematodes/src/WormCalc/util/ConfigureFrameCount.bat`).
3. Enter one or more data set names to update. You can enter multiple names separated by space (e.g. `n2_f1 n2_f2 n2_f3`), or even enter a regular expression to match (e.g. `n2_nf[1-6]` or `n2_.*`), or multiple regular expressions. You can also use `*` to select all.

3.3 Configuration

Each data set directory contains a configuration file `config.yml` that declare a number of parameters for the WormSeg Host and Node. The default parameters are more or less optimal at the time of writing, and doesn't need attention. Any parameters not described below are no longer used.

filename-errorlog The filename to log frame numbers on which segmentations fail fatally. This is technically no longer possible.

filename-featurelog The filename (relative to `log/`) to log segmentation results to.

filename-extension The file extension on the images in the `input/` directory.

filename-padding The number of digits in the image file names. WormSeg will automatically pad zeros on the left to match this parameter.

frame-toggle-output Enable or disable rendering the segmentation output. 1 to enable, 0 to disable. This is a computationally expensive feature.

frame-width The width of the image frames.

frame-height The height of the image frames.

frame-font The font to use in rendering output frames.

frame-color-text The text color in hex for use in rendering.

frame-color-background The background color in hex for use in rendering.

frame-color-foreground The foreground (worm) color in hex for use in rendering.

search-window-radius The WIDTH (no longer a circle radius) of a square centered on the last known worm position in which to search for the worm. Sorry about the misleading name.

search-area-value The expected area of the worm, in pixels. This is used in evaluating the confidence that a particular segment is a worm.

search-area-tolerance The tolerable range for a segment's area (from search-area-value) to be considered for worm candidacy, in decimal form. Segments with an area outside this range will be discarded. For example, a search-area-value of 300 and a search-area-tolerance of 0.1 (10%) will give a range of 270-330 pixels. There is not yet any evidence that a tighter tolerance improves results or performance...

search-gaussian-radius The radius of the Gaussian filter that is passed over the frame before segmentation.

threshold-window-size The width of a square used in the calculation of local averages, for use in dynamic segmentation.

threshold-value The ratio threshold below which a pixel will be considered as being on-worm, for use in dynamic segmentation. The ratio being compared is the pixel's intensity divided by the local average intensity.

frame-start The frame at which to start the segmentation process.

frame-end The frame at which to end the segmentation process. This can be automatically configured using the ConfigureFrameCount utility. (See Configuring Frame Count)

thread-count The number of threads on the Node client machines to use.

thread-priority The priority given to threads started by the WormSeg process.

3.4 Segmentation with WormSeg

Once you have the video data in place, you won't ever have to touch or move it manually again. You can now begin to process it using WormSeg to extract the centroid data. The WormSeg software is designed to distribute compute tasks over a number of computers, using MEDIXSRV (specifically) as a hub or controller.

1. Remote Desktop into MEDIXSRV (`medixsrv.cstcis.cti.depaul.edu`). Note that the WormSeg Host **must** be run on MEDIXSRV specifically. Not only is the data stored there, but I also hard coded the address[1].

2. Launch the WormSeg Host (`//MEDIXSRV/Nematodes/src/WormSeg/util/WormSeg Host.bat`) Run this script in-place (without copying it elsewhere). It is important that the Host is running before Nodes are launched (next step).

[1]The offending code is com.kylelmoy.WormSeg.Networking.Client Line 49. This could easily be modified to accept the server location as an argument.

3. On another computer (not MEDIXSRV), and copy WormSeg Node (`//MEDIXSRV/Nematodes/src/WormSeg/util/WormSeg Node.bat`) to the desktop, and the run the script. This will download the WormSeg code and launch the client.

4. Repeat the last step on any number of other computers. Note that Java 1.8 JRE must be installed to run the Java code.

5. In the WormSeg Host terminal (on MEDIXSRV), you can begin a fully automatic[2] batch processing using the command `job` followed by one or more project names. For example, `job N2_f1 N2_nf2 tph1_nf3` will process those three projects.

6. This will take some time, depending on the length of the video and the number of WormSeg Nodes you have running. Once a project has been processed, the "log" folder will contain "feature.log" which contains the centroid and area data, and "performance.log" which contains information about the Node performance.

3.5 Movement Feature Calculations

To calculate movement features from the centroid data, use the WormCalc code. Technically WormCalc can be run from anywhere, but it would be fastest to run on MEDIXSRV.

1. Remote Desktop into MEDIXSRV (`medixsrv.cstcis.cti.depaul.edu`).

2. Launch the WormCalc script "ProcessDataSet" (`//MEDIXSRV/Nematodes/src/WormCalc/util/ProcessDataSet.bat`). Although there are many functions of the WormCalc program, this script will do pretty much everything.

3. ProcessDataSet will ask for one or more data set names. You can enter multiple names separated by space (e.g. `n2_f1 n2_f2 n2_f3`), or even enter a regular expression to match (e.g. `n2_nf[1-6]` or `n2_.*`), or multiple regular expressions. You can also use `*` to select all.

4. ProcessDataSet will output a number of files in the "data" folder.

[2]The whole process is a series of steps which can be run independently, however I don't feel like documenting all the available commands as you really won't be needing them. The `job` command simply strings all these commands together.

4 Discussion

4.1 Tracker

4.2 Segmentation Process

4.3 Movement Feature Calculations

References

[1] Valerie Simonis *Tracking C. Elegans* 2014.

[2] Kyle Moy *Computational Methods for Tracking, Quantitative Assessment, and Visualization of C. elegans Locomotory Behavior* 2015.