Tadpole Behavior Automation

Project: Behavior Rig Overview
Authors: Aaron Ta

MONDAY, 11/18/2019

Originally documented by Christopher Marshall (08/03/2018). Last edited by Aaron Ta (11/18/19).

TadFunctionTest, BatchRunTAD9000

This is a Matlab script and function for use in automating the avoidance behavior analysis of Xenopus tadpoles. To run this script it is necessary to download the BatchRunTAD9000.m and the TadFunctionTest.m. Currently the Tad_Det_Assign_9001.m is the test file kept up to date with the TadFunctionTest.m that is used for testing and confirmation purposes.

Program Goals:

This script was designed to be used for expedited and automated analysis of visual response behavior of Xenopus tadpoles to a moving dot stimulus. The program uses blob filtering methods to find the locations of the tadpoles then assigns each tadpoles location based on distance to the next location. The location and radius of each oncoming dot is then extracted from the video. Using the location of the tadpoles and the location and radius of the dots the data is analyzed to find the frames in which a tadpole comes in contact with a dot. Based on set criteria from previous analysis protocols avoidances, in relation to the dot, of the tadpole and encounters with a dot are counted. This provides useful data used to analyze recovery behavior.

Pre-Run Dependencies:

- https://www.mathworks.com/matlabcentral/fileexchange/12275-extrema-m--extrema2-m
- https://www.mathworks.com/matlabcentral/fileexchange/34040-simple-tracker

To run the BatchRunTAD9000.m script or the TadFunctionTest.m function the two above dependencies must be installed off the Matlab File Exchange. The Extrema2 package is used for finding the locations of the local maximums for detection of tadpoles, and the Simple Tracker package is used for the Munkres assignment algorithm. These packages must be installed on a computer running Matlab version R2017_a or later with the Image Processing Toolbox installed and at least 16 Gb of memory.

Running Program:

Once installed the BatchRunTAD9000.m should be opened in the same folder as the TadFunctionTest.m. From the BatchRunTAD9000.m script hit run and select the folder containing the videos taken during the stimulus trials. This will begin the video analysis process creating folders for each video run saving location and avoidance data throughout the process. Videos with detections of less than half of the detections in the first frame analyzed will be skipped and shown as an error.

Program (BatchRunTAD9000) Overview:

1. On each .mp4 in this folder, call TadFunctionTest(). The following steps refer to code within TadFunctionTest() and are repeated for each video file.
2. Initialize values.
3. Average video frames to produce a mean background image with moving objects (dots and tadpoles) averaged out.
4. Subtract each individual frame from the background, invert, and keep positive values to extract tadpole guts (dots are lighter than the background while tadpoles are darker, so tadpoles would have negative values that are then inverted to positive) while setting non-tadpole pixels to 0 (black). This creates a copy of each frame where only areas darker than the average background are kept as non-zero values.
5. Apply a Laplacian of Gaussian (LoG) filter to these frames to filter noise.
6. Detect "blobs" of non-zero pixels and count them as tadpoles. Store this data in *raw_tad_detections.mat*.
7. Use a Kalman filter to estimate tadpole positions, and then the Hungarian-Munkres algorithm to assign these estimated detections to the proper tadpoles (connecting the "blobs" per frame over time as distinct tadpoles).

8. Remove estimations or detections that go beyond the boundaries of the lanes. Store this data in *position_estimates.mat*. **Tadpole detection for this video is finished.**
9. Reiterate through frames, this time subtracting from background and keeping the positive values without inversion to retain dots without tadpoles.
10. Apply morphological opening to filter out non-zero value areas (non-black areas) below a certain size threshold.
11. Use imfindcircles() to attempt to fit circles of a certain radius range within the remaining non-zero areas, marking successful fits as dots. Store this data in *dots_centers_radii.mat*. **Dot detection for this video is finished.**
12. Calculate the direction and angle at which each tadpole is moving between consecutive frames. Compare dot detections with tadpole detections after accounting for distance between the eyes and gut as well as a 180° semicircle centered around the direction of movement to determine whether an encounter has occured for this tadpole at this frame. Store this data in *encounterMatrix.mat*.
13. For each tadpole, iterate through each consecutive encounter. If the tadpole's velocity is below threshold, do not consider this encounter. If the tadpole's approach angle towards the dot is below threshold (not sufficiently perpendicular), do not consider this encounter. Otherwise, this encounter will be considered, and discount any additional encounters within the next 4 frames to prevent double-counting of the same dot.
14. Check the tadpole's angle before and after the encounter. Mark it as a turn if it exceeds threshold, or as no turn if it does not. Store this data in *Final_Avoidance_Data.mat*. **Dot avoidance analysis for this video is finished**.
15. Store this data into an array. Once all videos have been analyzed, write the array of results as a summary into *Batch_Avoidance_Data.xls*. **Batch analysis for all videos is finished.**

Output Files:

- raw_tad_detections.mat
  - Raw coordinates of dark blobs after subtracting each frame from the background. Neither the Kalman filter nor the Hungarian-Munkres algorithm have been applied.
- position_estimates.mat
  - Processed/final tadpole coordinates. Post-Kalman filter, Hungarian-Munkres algorithm, and lane boundary thresholding.
- dots_centers_radii.mat
  - Coordinates and radii of each dot detected for each frame.
- encounterMatrix.mat
  - Raw table indicating whether a given tadpole on a given frame encounters a dot (represented by a '1'); does not consider turning, velocity, whether or not the tadpole may be in the same dot as the last frame, etc. This essentially just registers whether there is overlap between a detected dot and a detected tadpole's estimated range of vision.
- Final_Avoidance_Data.mat
  - Final table that only registers "valid" encounters above a certain velocity threshold and angle of encounter between dot and tadpole. Stores whether a tadpole turned in this valid encounter (represented by a '1') or not. This data is what is summarized in the Excel spreadsheet.
- Batch_Avoidance_Data.xls
  - Summary of encounter statistics for all video files in this batch.

Situational Adjustments:

Adjustable from the prompt:

- Gaussian hsize and Gaussian standard deviation (sigma)
  - Variables for the Gaussian filter that removes image noise prior to tadpole detection
- Detection thresholding value
  - Threshold for post-filter tadpole detection. Increasing the threshold value will cause more noise from the original pictures to be discarded below the specified threshold value. This will leave the deeper points, tadpole gut, found by the LoG filter as maximums allowing the program to search for all local maxima and save the maxima as X and Y coordinates. If there are too many detections that aren’t tadpoles this can be increased incrementally until a reasonable value is reached to eliminate excess detections.
- Tadpole speed variability, Tadpole X direction noise, and Tadpole Y direction noise

  - The speed variability is taken as an estimate of the tadpoles speed from frame to frame given in pixels/frame. The X and Y noise are also taken as estimates of how much potential there is for tadpole movement variability in the detections. These three values should not have to be changed as the default values are accurate.
- Tadpole distance between eyes and gut
  - The program uses the gut of the tadpole to find the location because it is the most visible part of the tadpole. This being said estimated coordinates of the tadpoles field of view needed to be found in order to assure that the locations of the dots crossed over the tadpoles eyes. After testing it was determined that the average distance between tadpoles gut, at this stage of growth, and eyes was about ??? pixels (new video size means this must be rechecked). Using this as a radius points in a semicircle around the head and eyes are found and taken as new coordinates for the tadpoles. Adjusting this value could be done either by increasing or decreasing this radius for a tadpole at either a later or earlier stage of growth respectively.
- Lower avoidance angle tolerance and Higher avoidance angle tolerance
  - Counting an avoidance event is dependent on how much a tadpoles swimming direction angle changes within sixteen frames of crossing the location of a dot. To quantify this turn upper and lower bounds for a change in swimming direction angle were set at 110° and 70° respectively. This means that if a tadpole is swimming, in the video, to the right an avoidance is counted if the tadpoles swimming angle changes greater than or equal to 70° of the angle it was previously going. Now, for a tadpole moving to the left an avoidance is counted if the tadpoles swimming angle is less than or equal to 110° of the angle it was previously at. The values for upper and lower avoidance determination can be changed by increasing the upper avoidance angle and decreasing the lower avoidance angle for a smaller turn to be counted as an avoidance. To increase the size of the turn required to count an avoidance the upper and lower angles can be decreased and increased up to 95° and 85° respectively.

Adjustable from the code (TadFunctionTest):

- NumBgFrames
  - Number of frames, starting from frame 0, to average and use as the background image with both dots and tadpoles removed. Only using pre-dot frames improves the contrast between dots and the background at the cost of not detecting tadpoles that don't move prior to the dots appearing, while only using post-dot frames improves detection of tadpoles that only start moving post-dots at the cost of reduced contrast between dots and the background. The default of using all frames (NumBgFrames = NumFrames) is usually the best choice.
- noDotFrames
  - Number of frames prior to when dots begin to appear. The default, 450, matches that of the behavior rig code.
- Tadpole speed variability, Tadpole X direction noise, and Tadpole Y direction noise
  - The speed variability is taken as an estimate of the tadpoles speed from frame to frame given in pixels/frame. The X and Y noise are also taken as estimates of how much potential there is for tadpole movement variability in the detections. These three values should not have to be changed as the default values are accurate.
- imopenThreshold
  - Size of disk that imopen() uses to morphologically open the dots. Acts as a threshold to remove noise below a certain size threshold, but can produce odd behavior when tadpoles partially obscure dots due to how morphological opening functions. Lower values are less likely to remove dots that are partially obscured, but too-small values will lead to false detections of noise.
- imfindcirclesMin, imfindcirclesMax, and imfindcirclesSensitivity
  - Variables for imfindcircles(). After morphological opening, imfindcircles() detects circles between the radii of imfindcirclesMin and imfindcirclesMax with a sensitivity of imfindcirclesSensitivity. Adjusting the min and max can help if circles of a certain size are consistently not being detected, while adjusting sensitivity may help if circles are being detected inconsistently. Using Tad9000ColorVisualization can help troubleshoot dot detection issues. Multiple dots being split from a single dot is a sign of a too-low minimum, while small dots not being detected at all is a sign of a too-high minimum or threshold. Partially-obscured dots not being detected likely has to do with either a low sensitivity, a high imopenThreshold, or some combination of both.

Tad9000ColorVisualization

This is a diagnostic tool for TadFunctionTest, mapping the outputted detection data back onto the original video for troubleshooting, debugging, and analysis verification. By default, this script will produce an annotated video that plays at half-speed compared to the original to make visual inspection easier, as 1-frame events are difficult to see at 28 fps.

Running Program:

This script requires the original video .mp4 as well as four of the .mat files TadFunctionTest outputs by default: position_estimates.mat, dots_centers_radii.mat, encounterMatrix.mat, and Final_Avoidance_Data.mat. The easiest way to do this is to move or copy the original video into the folder that contains these output files, and then change the Matlab directory to this folder before running the program.

Program Overview:

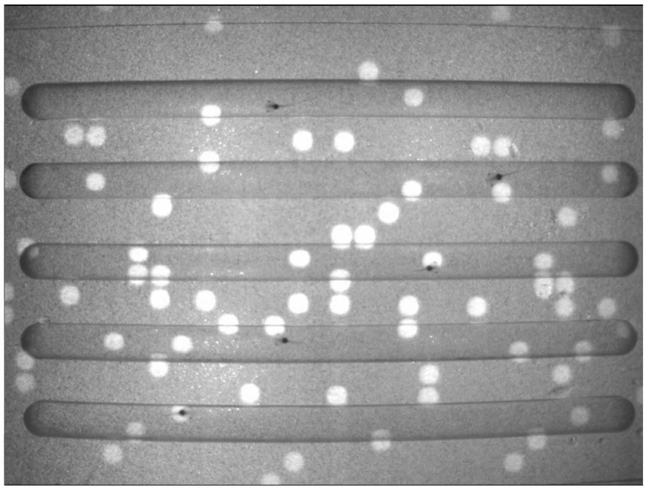
This program directly annotates the coordinates and data TadFunctionTest had written to file back onto the original video to allow for visualization of what the computer is "seeing." Gray dots represent what the computer "sees" as tadpoles, while black dots represent what the computer sees as dots. Tadpole dots will flash white when the computer detects that the tadpole is occupying the same space as a dot, but it is unable to consider it an encounter due to insufficient velocity or distance from the tadpole's supposed visual field. Tadpole dots will flash red when the computer recognizes a valid encounter where the tadpole turned, and will flash blue when it recognizes a valid encounter where the tadpole did not turn.

Output Files:

- {video_name}_AnnotatedCircleSlow.mp4
    - A copy of the original video, "video_name.mp4", that has been slowed to half-speed and annotated based on the output of TadFunctionTest.

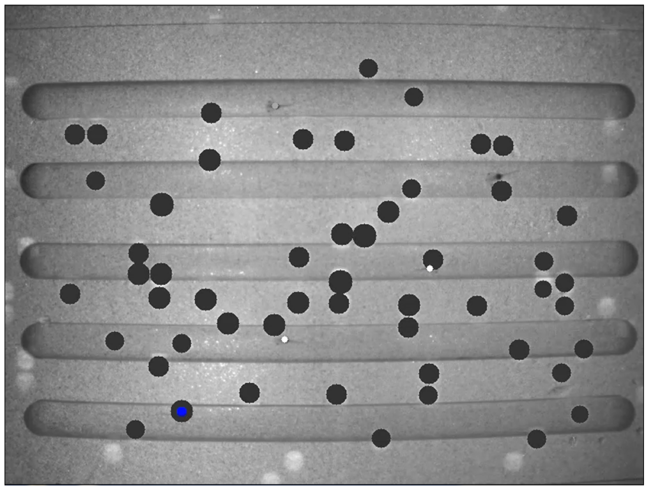
Screenshots:

UnannotatedSample.PNG



The original image from the behavior videos before any image processing is shown above.

AnnotatedSample.PNG



After running TadFunctionTest and then TadColorVisualization, the same frame can be seen here, annotated. Notice the gray dot in lane 1 representing a tadpole with no dot encounter. The white dots in lanes 3 and 4 indicate tadpoles who the computer has calculated as having dots within their field of view, but are either not moving quickly enough or at a perpendicular enough angle to consider these encounters. The blue dot in lane 5 indicates a tadpole who encountered a dot but did not turn this frame. The unannotated tadpole in lane 2 did not move in the video, and the lack of a dot indicates that the computer detected no tadpole here (a non-moving tadpole would be included in the background average and thus would be subtracted out in the analysis). The black dot annotation also reflects the constrained region of interest for analysis, as the computer excluded the ends of the lane as well as the non-lane regions near the top and bottom during processing.