Geotaxis Functions

1: **RecordVideos**(numVideos,userFrameRate,duration,waitTime,filename)

-Function:

to record multiple videos in succession directly using MATLAB. This function does include the connection to the fly slamming mechanism.

-Input:

**numVideos** = the # of videos to record in succession

**userFrameRate** = the frame rate (Hz: frames per second) for each video (could be limited by the camera)

**duration** = duration of each recording (in seconds)

**waitTime** = the time between each video (in seconds)

**filename** = name to label videos (the number of the recording will be appended)

For example: ‘exp\_1’ for a numVideos=3 will output:

‘exp1\_1.mp4’ ‘exp1\_2.mat’ ‘exp1\_3.mp4’

-Output:

‘.mp4’ video files saved to the open folder in the MATLAB path.

-Notes:

**\*\*\***The function can overwrite videos with the same name but will check with the user prior to execution.

**\*\*\***This code works with XXXXXX camera specified in line 25. User should change this if using a different type of camera.

**\*\*\***This code connects directly to the motor for the fly slamming mechanism. If the user is not connecting to the same motor (or to any motor) comment out lines 34-36, 60-62, 87.

2: **VideoProcessor**(prev\_roi,n\_ROI,startFrame,saveROI,videoName)

-Function:

Uses an input video from a *D. melanogaster* geotaxis experiment.

Removes the initial frames containing the movement of the slamming (in short specifying a start frame)

Cuts the video into regions of interest (ROI)

Waits for the user to be satisfied with the ROI (using button/mouse click)

Saves the separated ROI into new vial videos for tracking.

-Inputs:

**prev\_roi** = previous ROI matrix from the MATLAB workspace

**n\_ROI** = number of ROI’s desired

**startFrame** = number of frames removed from the beginning

**saveROI** = specify if you would like to save the ROI information into a .mat file. 1 = save, 0 = do not save.

**videoName** = user already has the FULL PATH to the desired video file

-Outputs:

‘.mp4’ video files corresponding with the original video and then the ROI number.

For example, if input file is ‘exp1\_3.mp4’ and n\_ROI=2 the output would be: ‘exp1\_3\_1.mp4’, ‘exp1\_3\_2.mp4’

-Notes:

**\*\*\***If using the prev\_roi you must load it into the workspace.

**\*\*\***For the first video in each experiment, saveROI is recommended. Due to the nature of negative geotaxis experiments any following ROI should be identical barring any human error. For future videos in the same experiment, use the prev\_roi to make sure you are using the same cropping.

2.5: **VideoProcessIterative**(prev\_roi,n\_ROI,startFrame,saveROI)

-Function:

**NOT REQURIED: Created for convenience.**

Identical to VideoProcessor but iteratively cuts multiple videos in succession.

-Inputs:

**prev\_roi** = previous ROI matrix from the MATLAB workspace

**n\_ROI** = number of ROI’s desired

**startFrame** = number of frames removed from the beginning

**saveROI** = specify if you would like to save the ROI information into a .mat file. 1 = save, 0 = do not save.

\***vidoeName** = dialog box will pop up and ask the user to input one or more video files

-Outputs:

‘.mp4’ video files corresponding with the original video and then the ROI number.

For example, if input file is ‘exp1\_3.mp4’ and n\_ROI=2 the output would be: ‘exp1\_3\_1.mp4’, ‘exp1\_3\_2.mp4’

-Notes:

**\*\*\***If using the prev\_roi you must load it into the workspace.

**\*\*\***Recommended use of this function is after using VideoProcessor and utilizing prev\_roi for the remainder of the experiment videos.

**\*\*\***Watch to make sure the videos are framed correctly for each video regardless of the iteration. Some videos can occasionally be cropped incorrectly and can only be rectified manually.

**\*\*\***All inputs will be identical for each iteration. If parameters need to be changed for one of the videos, it is best to use the VideoProcessor.

3: **Calibrate**(sameFile,saveInfo)

-Function:

Creates a background for each input video file

Loads frame from every 0.5 second of each recording

Subtracts background from frame and detects blobs

Asks the user to categorize all blobs into 3 possible groups

One fly

Multiple flies

Noise

Saves area information in 3 separate excel files

Axis lengths

Single area

Multi area

-Inputs:

**sameFile**: specifies if user desires to save to existing excel files. Must be all three of them.

1 = yes

0 = no

default = 0

**saveInfo:** specifies if user desires to save the calibrated information.

1 = yes

0 = no

default = 1

**\*\*\***If saveInfo=1 and sameFile=0, user will need to define the name of the new calibration files in a pop-up window. For example, “Straight\_wing\_”.

**\*\*\***If saveInfo=1 and sameFile=1, user will need to identify the desired file they wish to append to.

**\*\*\***Recommendation is to first create the new file with a single ROI video. Then append subsequent ROI videos in groups of 3. The task is tedious, and it will help break up the workload.

-Outputs:

3 separate excel files=

**Axis lengths**:

first column = single fly maximum axis length

second column = single fly minimum axis length

third column = multiple flies maximum axis length

fourth column = multiple flies minimum axis length

named: user\_defined\_name”\_axisLengths.xls”

example: Straight\_Wing\_axisLengths.xls

**Single area**:

first column = single fly pixel area

named: user\_defined\_name”\_pixelarea.xls”

example: Straight\_Wing\_pixelarea.xls

**Multi area**:

first column = multiple flies pixel area

named: user\_defined\_name”\_multiarea.xls”

example: Straight\_Wing\_multiarea.xls

**\*\*\***This could be changed to one excel file output if necessary. I like to keep things separate. However, if you change this to a single excel output, keep in mind you have to edit the FlyTracking() function to be able to read it properly.

* Notes:

**\*\*\***Highly recommend calibrating the tracking to the metrics of your experimental set up. The camera quality, distance of camera to vials, size of flies, and sharpness of image all influence these parameters. Despite having identical equipment, small variations in fly size can strongly influence the accuracy of tracking.

4: vidMatrix = **FlyTracking** (numFliesArray,saveTrackVideo,showTracking,saveCoord,timeOfInterest,numVials)

-Function:

Designed to locate individual flies and track their progress through time.

This function:

Detects flies per frame by background subtraction and blob analysis.

Predicts the future location of the fly using a constant velocity Kalman filter.

Assigns tracking numbers to each detected object using Munkres' version of the Hungarian algorithm.

Uses a cost matrix combining the Euclidean distance cost matrix between the predicted and detection coordinates and the Euclidean distance between the previous and current centroids, as well as the cost of non-assignment.

Checks that number of tracks <= number of flies. And rectifies discrepancy.

Follows the progress of each fly frame by frame updating the Kalman filter as it goes.

The coordinates of the flies will be saved in a MxN matrix within a 1xP cell matrix.

P = number of vials per video

M = number of frames,

N = 2\*number of flies

Every two columns correlate to one fly (x-coordinate,y-coordinate)

The cell matrix name will correlate to the vial number received from the filename

for example: video exp3\_1 will have a corresponding cell array coord3\_1

-Inputs:

**numFliesArray**: an array containing the number of flies per vial per experiment selected. This could also be a partial array or a single number. The function will use the last number input to fill in any missing information.

For 3 videos, possible inputs could be:

ex1) [6,7,7]

ex2) [6,7] ---> [6,7,7]

ex3) 7 ---> [7,7,7]

default = 7

**saveTrackVideo**: if saving the tracking video with tracking numbers present is desired.

1 = save

0 = do not save

default = 0

**showTracking**: if user desires to watch the videos with tracking numbers present as it runs through the function.

1 = show the video as function tracks

0 = track without showing video

default = 0

**saveCoord:** if the user desires to save the final coordinate result from tracking to current file opened in MATLAB.

1 = save to ‘.mat’ file

0 = output coordinates to workspace without saving to file

default = 1

**timeOfInterest**: the amount of time (in seconds) user desires to track flies in video.

default = duration of video input

**numVials**: number of vials per video

default = 10

**calibration files:** these are excel files that are output by the Calibrate() function. There will be a pop-up window directing the user to input the files. If the user opts out of calibration default settings are used.

default =

maxLengthMult=50;

minLengthMult=5;

minLengthSingle=3;

flyMin=20;

flyMax=129;

multMax=500;

multMin=130;

-Outputs:

**vidMatrix:** an 1xP cell array with MxN matrices enclosed

P = number of vials in selected experiment

M = number of frames, N = 2\*number of flies

Every two columns correlate to one fly (x-coordinate,y-coordinate). This is why N=2\*Number of flies.

**\*\*\***if saveFile = 1, the coord will automatically save to the current folder.

**\*\*\***the cell array corresponds to a single trial within the experiment. If you have multiple trials per experiment they will be consecutively numbered according to the video name.

-Notes:

**\*\*\***Meant to be used in tandem with the Automated Geotaxis Monitoring (AGM) system.

5: **groupTracksPerVial()**

* Function:

This function is used to separate the vial matrices from the coord cell matrices output from the FlyTracking() function.

* Inputs:

**Coord#.mat:** 1xP cell matrix output from FlyTracking function

* Outputs:

**Vial#.mat:** All trials from the same vial will be appended together to produce a MxN matrix. M is the number of frames. N is the (number of flies \* number of trials \* 2 (x and y coordinates) for each vial).

**Exp#.mat:** 1xq matrix. q = coordinate matrices input by user.

-Notes:

**\*\*\***Vial#.mat is used for Geotaxis analysis

**\*\*\***Exp#.mat is used to compile all data for easy storage

6: **separateSheets()**

**NOT REQURIED: Created for convenience.**

* Function:

Used to separate the coord matrices output from the FlyTracking() function into trial matrices corresponding to the vial number and the trial number.

* Input:

**Coord#.mat** 1xP cell matrix output from FlyTracking function

* Output:

**Trial#\_#.mat** is an MxN matrix with M = # of frames. N = number of flies in vial \*2. The first # represents the vial number, the second # after the underscore is the number of the trial.

* Notes:

This function separates the cell coordinate matrix to the smallest degree. It is not necessary to do, but helpful when checking the differences between trials. (From our experiments there is no significant variation between trials)

7: **combineVials()**

**NOT REQURIED: Created for convenience.**

* Function:

Used to append different matrices together. If you have multiple vials of the same mutations, for example, you would be able to use all the standard functions and then append them together. It will append the columns as additional columns; it will not add rows.

* Input:

Any MxN matrices you wish to append together.

Type in desired name when prompted.

* Output:

New matrix under the user’s desired name saved to the folder you are working in.

* Notes:

**\*\*\***The matrices appended should be in the same folder together. If, for example, you are trying to append vial1.mat from two different experiments after using groupTracksPerVial(), to put them in a folder together you must rename them.

8: **plotTracks(**vialNumber, flyNumbers, onVideo, separate, colored**)**

**NOT REQURIED: Created for convenience.**

* Function:

Plot coordinates of the desired fly/flies. User decides if they want to plot onto the corresponding video or simply on a graph.

* Input:

**vialNumber:** the vial in which the desired fly/flies are in

**flyNumbers:** the track number that corresponds to the fly/flies of interest. This input if there are more than one fly should be listed within “[]”

**onVideo:** if the user wants to plot the flies onto the corresponding vial video.

1 = on video

0 = on graph

**separate:** if you want the flies plotted together on the graph/video or if you want them plotted separately.

1 = separate

0 = don’t separate

**colored:** choose if you want the flies to be plotted through time with changing color purple to red.

1 = color with time

0 = single color per fly

9: **plotCoordinatesOnVid(**flyNumber**)**

**NOT REQURIED: Created for convenience and ease of use.**

* Function:

Plot coordinates of the desired fly onto the corresponding video to produce another video with a visual representation of the tracking over time. The passage of time is represented by the color change from purple to red points.

* Input:

**flyNumber**: in the video the number corresponding to the fly you wish to visually track.

**Coord#.mat:** output of FlyTracking() function corresponding to the trial in question.

**Exp#\_#\_#.mp4:** the corresponding vial video for the fly and trial in question. The first # corresponds to the experiment number. The second # corresponds to the trial. The third # corresponds to the vial number.

* Output:

**Exp#\_#\_#\_plotted.mp4**: the same video input with the coordinate of the fly plotted through time in color from purple to red, changing with time.

* Notes:

**\*\*\*** cannot plot more than one fly due to the color change with time.

10: **datastr=Geotaxis\_analytics(**distTOpixel,framerate,maxYpixel,group**)**

* Function:

Used to extract geotactic information from coordinate matrices from fly tracking. The function will prompt user to pick one or more \*.mat files

containing fly x-y coordinates data from multiple trials. The \*.mat

files are expected to be in one folder.

Example use:

% y=Geotaxis\_analytics(4.243,30,4,{'THGal4female','csmale','yw'});

* Input:

distTOpixel = number of pixels representing 1 millimeter.

e.g.: 4.243

framerate = number of video frames captured per second.

e.g.: 30

maxYpixel = position of the highest y pixel in camera

frame. Note: Matlab assigns (0,0) to top left

of image, near the highest position reached by

flies. So, maxYpixel is paradoxically a small

number. e.g.: 4.

group = not a required input. The only non-numeric

input variable, it is used to label figure

window and Excel sheets which show results from

all group. For e.g.: {'DEMO 1','CS','YW'}

data = MxN coordinate matrices to be compared. M = number

of frames, N = number of flies \* 2 (x and y

coordinates)

* Output:

**datastr**: A structure that keeps all the information extracted for each input matrix.

datastr.Group.Names = user-provided names of groups

(INPUT: group) of genotypes,

trials, strain names, etc.

corresponding to the datafiles

selected upon prompt. e.g: for the

first group name, type

datastr.Group(1).Name

datastr.Group.ClimbCurves = matrices containing climbing

rate data of each group or the

percentage of flies in a group that

are at a given height at a given

time. Number of matrices is equal

to the number of groups or input

files. Each matrix is of size #rows

x 4. The 1st column is time, 2nd

column is climbing rate for max

height, 3rd column is climbing rate

for half max height and 4th column

is climbing rate for third of max

height. e.g: To output the climbing

data for group#3, type

datastr.Group(3).ClimbCurves.

datastr.Group.Fly\_no = a structure with 5 fields:

SpdAngle, NRMSE, AveSpeed, Slips,

Falls. 'SpdAngle' contains

frame-by-frame speed (col 1) and

angular (column 2) movement data of

individuals; 'NRMSE' contains

normalized RMSE of fitting fly

(x,y) coordinates to a straight

line, as a crude measure of the

type of individual trajectory;

'AveSpeed' is the average speed of

an individual fly. It can differ

from the average of all the

frame-by-frame speed measurements

when a fly was undetectable for

several frames because it was

immobile. So for any fly

Avespeed<=mean(SpdAngle(:,1));

'Slips' contains list of sudden

drops in height, more than ~ 4mm

but less than ~11.5mm. Slips(i,1)

is the time when the event

happened, Slips(i,2) is the height

from which the drop happened,

Slips(i,3) is the size of the drop;

'Falls' is similar to 'Slips' but

for drops > 11.5 mm. e.g: To output

the slips data for fly# 31 in

group# 2, type

datastr.Group(2).Fly\_no(31).Slips

**Excel file**:saves the comparison between the matrices in one place. Excel workbook containing results from each group in a

different sheet. Last sheet contains collated data from all groups being compared. The workbook is saved in the same folder as the .mat files that were fed to this function. The name of the Excel file bears the name of the first input file that was read.

**Figure:** plotted comparison of the matrices input.