FISH TRACKING

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirement for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Xinjia Huan

Zhao Wei

MEng Field Advisor: Dr. Bruce Land

MEng Outside Advisor: Ni Feng

Degree Date: May 2012

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title:

Fish Tracking

Author:

Zhao Wei & Xinjia Huan

Abstract:

In this project, we constructed a vision-based real time motion capture system to monitor daily locomotor rhythm of Plainfin Midshipman, a vocal fish. We used infrared cameras to collect input data and a GUI based Matlab program to display and analyze useful information. We want to monitor midshipman's locomotion over 24-hour period of time, but it is almost impossible for human to manually record for such a long time, a reasonable approach is to design a vision-based self-recording system which is reliable and user-friendly. Also, infrared camera does not impose any physical restrictions on fish, which provides a natural way of measuring fish's daily activity. In this system, every frame of the picture captured by camera is treated as input of Matlab Program. With this program, we can find the fish's location on every frame. Those locations will be saved in a file, so we can analyze the data to master fish's activity rhythm.

Individual Contribution

Xinjia Huan

- Researched on, purchase and constructed the hardware system
- Accumulated error problem
- Memory problem
- Designed data analysis system

Zhao Wei

- Researched and created the main functions in tracking algorithm
- Static error problem
- Designed user interface
- Designed data analysis system

Executive Summary

This project is the creation of software program to satisfy the specific needs of graduate students in Department of Neurobiology and Behavior to better study the behavior of a special kind of fish. The program created in this project is called Fishtrack3, which is run in Matlab to take advantage of its image processing functionality.

The program is used to study the locomotor rhythm of a sound‐producing teleost, the Plainfin Midshipman. Currently, most of the methods used to study the daily activity of the fish are very crude and inefficient, a more sophisticated and accurate way is to design a vision-based real time motion capture system to monitor daily locomotor rhythm of the fish, which is what we did in this project.

The project is a concatenation of hardware and software, it takes a infrared camera-based video as input, followed by a reliable program which can efficiently analyze and save data. The program is capable of locating the fish on every frame from the video, thus the fish's every move and activity is under monitoring, it can also analyze these data to provide plenty of information about the fish, like during what time period the fish is most active. We designed the program to support tracking 3 fish simultaneously for 24-hour period, at the same time be system and memory friendly. Accuracy is the most important requirement for this program at all stages of the project, other requirements include reliability, user-friendly interface. We are pleased to say they are all well met.

Analysis and study of animal behavior has always been a big topic in biology, this program should see good use for this area and it was made to be easily extendable for other functionalities.

Table of Contents

1 Introduction ……...……6

2 Requirements...……..…….7

3 Approach…….8

3.1 Hardware Setup ……...…….……....8

3.2 Tracking Algorithm ……...……...10

3.3 Error Elimination ……..….……..13

3.4 Data Storage and Analysis……...……16

4 User Interface Control…….….18

5 Test and Result…….……..…...19

6 Conclusion……..23

7 Acknowledgements…….……..…..24

8 Reference…….……...……....24

Appendix……..25

1 Introduction

The main focus of this project is to study the locomotor rhythm of a sound‐producing teleost, the Plainfin Midshipman. The male midshipman shows regular day-night behavior cycle during the summer breeding season, including vibrating muscles to produce long duration courtship calls at night to attract females and remain quiescent during the day. We want to study whether the fish also performs a day-night activity cycle during this breeding season. It is believed that the midshipman is more active during night than day. In order to reach a statistical result, we need to monitor multiple fish, each with multiple times for 24-hour period, which incurs huge amount of data and calculation, this leads us to the idea of designing a system concatenated with software and hardware to monitor the fish's activity for at least 24 hours.

The most straightforward configuration for such a system would be taking a camera-based video as input, followed by a reliable program which could efficiently analyze and save data. Since visible light source would change the activity rhythm of midshipman, an infrared-based video recording camera is recommended in this system. Other than that, a Graphical User Interface (GUI) based program which is easy to read and use is also recommended in this system.

The video input from camera is analyzed on a frame base, this program locates the fish's location on every frame and all the locations during the 24 hours will be saved into a file. We will then analyze these data to find the distances the fish moved during the 24 hours and find out during which time period the fish moved the most, which corresponds to the most active time period. Other behaviors of the fish could also be analyzed after collecting the data, like in which area of the tank the fish is most likely be, how many turnings the fish made and so on, in this project, we focus on the distances.

2 Requirements

In order for the program to function well, the following requirements should be met:

- accuracy

Accuracy is crucial since any deviation of the acquired location from the actual location would give us inaccurate data which leads to inaccurate distances. In this program, we should make sure the red dot (acquired fish location) always stays on the fish without sudden jump.

- reliable for 24 hours

This program should continuously run for at least 24 hours, so any minor flaws may accumulate during this long time period and easily leads to instability to the program and burden to the system. Especially, this program should be able to release useless memory and be memory-friendly.

- 3 fish in 3 tanks

This program should be able to monitor 3 fish in 3 different tanks. In order to reach a statistical result, this program should provide some efficiency.

- user-friendly

Our program is mainly designed for non-software background users. Therefore, a user-friendly interface which is easy to use and understood is essential.

- easy data access for later analysis

The data acquired by this program should be saved and accessible in a later time so as to support summation and reanalysis.

3 Approach

Our project requires both hardware and software support, we need hardware to obtain video input and software program to process the input video to find fish location.

3.1 Hardware Setup

Since the main challenge of this project is to develop a sophisticated tracking algorithm, we wish we could spend as less time as possible on hardware system. As a result, we focused on those commercially available, off-the-shelf products for hardware setup. After a little bit research, we decided to construct the hardware system as follows:

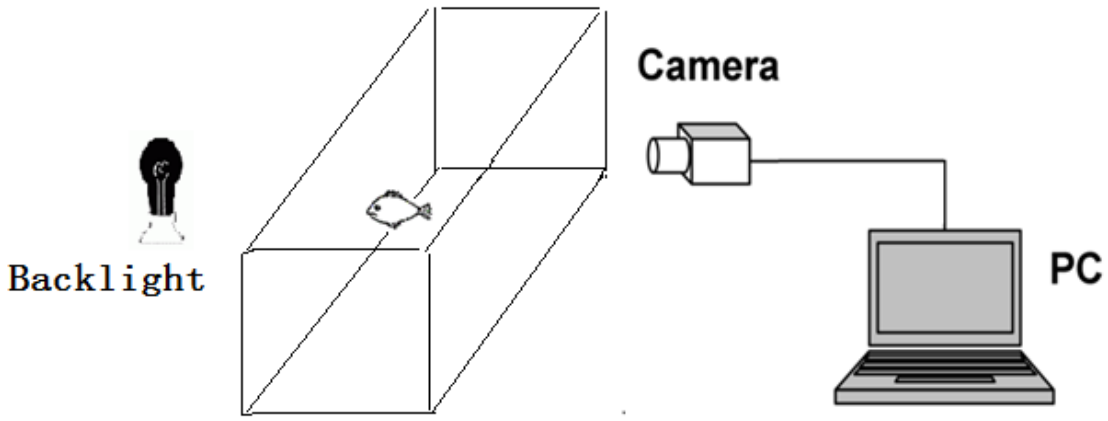


Figure 1: Hardware Setup

The camera is placed to face the side of tank with the largest area, which provides a more broad view of the fish. Ideally, placing another camera on top of the tank will provide more information about the fish in great detail in 3-D, but that is more complex and requires a lot more computation, we want to simplify the project since we only care about fish's activity level, so we use one camera.

In this setup, we need a PC, an infrared camera, an infrared LED backlight, a diffuser and a TV-USB adaptor.

- PC

To process the input video and save the acquired data from program

- Infrared camera

To monitor the 24-hour activity of the fish and provide video input for program. Infrared camera can capture the fish during both day and night.

- Infrared LED backlight

To provide infrared light to highlight the fish's location for camera, also infrared light does not affect the fish's daily activity rhythm.

The LED backlight should be strong enough so as to penetrate the tank, a LED array consists of hundreds of LEDs is recommended.

- Diffuser.

A diffuser is recommended to stick at the back of tank to smoothen the light.

Without a diffuser, the camera will see a big light spot and everywhere else black, which affects the accuracy. A diffuser can distribute the light evenly and light up the whole tank. A diffuser could be something like a translucent, white plate, in this project, we used food mat and it worked good.

- TV-USB adaptor

To transfer the data from camera to computer, also to transform analog signal to digital signal for program to process

Bill of Materials

Name	Serial Number	Unit Price
PC	-	-
Infrared camera	SVAT-SVVU5	$29.99
Infrared LED backlight	IR045	$48.95
TV-USB adaptor	Hauppauge USB-Live2	$44.95

Table 1: Bill of Materials

3.2 Tracking Algorithm

The basic idea of the tracking algorithm is to subtract well defined background $\boldsymbol{I_B[n]}$ from current frame $\boldsymbol{I[n]}$, resulted in a subtracted frame $\boldsymbol{I_\Delta[n]}$. Then we use Matlab to deal with $\boldsymbol{I_\Delta[n]}$ to eliminate noises in recording and find the pixel of maximum intensity in $\boldsymbol{I_\Delta[n]}$, then find out the centroid of the area with maximum intensity to locate the fish.

The subtracted frame $\boldsymbol{I_\Delta[n]}$ could be like the figure below, the circular area means the differences between background and current frame, which could be potential fish. Among them we then find the area with maximum intensity and find the centroid of this area. The centroid is the calculated location of fish. This algorithm is based on the fact that fish is the largest moving object in the tank. Largest not only means large in area but also means large magnitude in intensity (more details about intensity is in Section 3.2.2).

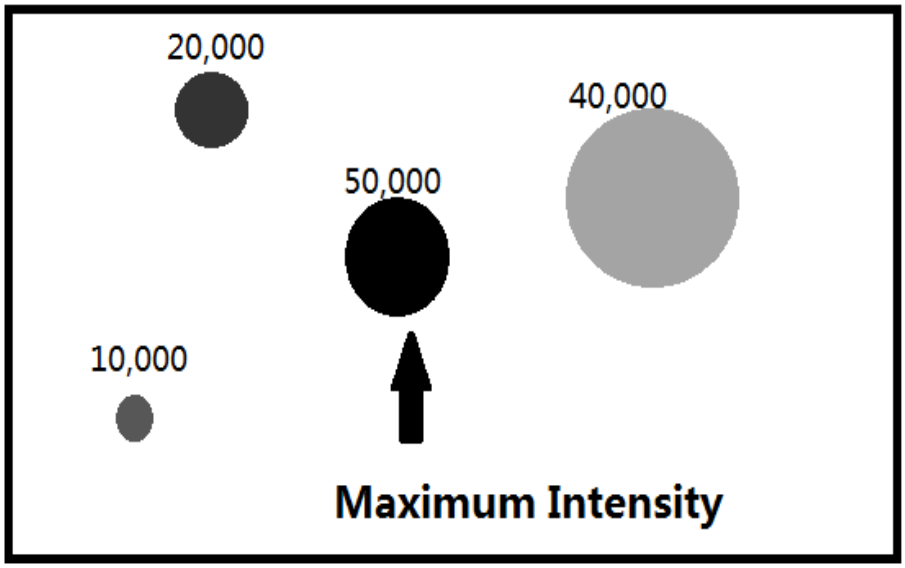


Figure 2: Subtracted frame $\boldsymbol{I_\Delta[n]}$ with area of interest

Since a frame is represented as a two-dimensional matrix in Matlab, each number in the matrix represents a pixel, and the value of the number represents the color in that pixel, so the subtraction between background and current frame is actually subtraction between two matrixes.

So in the above figure, darker area means larger difference between background and current frame.

As stated above, the basic idea is to subtract current frame from updating background. But we wish this program could be more sensitive to fish such that it would give more weights on black object. A direct way is to enlarge the difference between fish and background based on the fact that fish is black and background is white. So we squared the result and defined it as $\boldsymbol{I_{\triangle}[n]}$, which is calculated below.

$$\boldsymbol{I_{\triangle}[n] = (I[n] - I_B[n])^2} \qquad \textbf{(1)}$$

3.2.1 Background Frame

The very first problem is how to efficiently and accurately define a background. We used the algorithm from *Analysis of the Trajectory of Drosophila melanogaster in a Circular Open Field Arena* [1] written by Dan Valente, Ilan Golani and Partha P. Mitra. The background can be acquired first by averaging the first 20 frames and then update the background through running average, which means, we will combine the current background with current frame to get new background, the equation can be expressed as:

$$\boldsymbol{I_B[n+1] = \alpha I_B[n] + (1-\alpha)\, I[n]} \qquad \textbf{(2)}$$

Where $0.9 \leq \alpha \leq 1$

Besides, a subset image $\boldsymbol{I_{sub}}$ around the location of fish is extracted from the new background $\boldsymbol{I_B[n+1]}$ and replaced with the pixels of the same area in the original background $\boldsymbol{I_B[n]}$. This process prevents the fish been included into background.

3.2.2 Area of Maximum Intensity

In order to simplify the calculation, we converted frames to gray based images (white and black only) from RGB based images (red, green and blue). As a result, calculation could be based on

image intensity rather than relative ratio of RGB. As shown in **Figure 3**, fish is black and background is white in such setup. After defining background, the next step is to find the maximum intensity area.

The intensity is calculated by summing up the square of the pixel number (see **equation 1**) in the above area. So the maximum intensity is associated both with magnitude of area and darkness of the area. The area with the largest sum of squared pixel value is our target area.

Next, we find the centroid of this target area as the location of fish. A subset image $\boldsymbol{I_{\triangle sub}}$ around this pixel is extracted, where $\boldsymbol{I_{\triangle sub}}$ means our target area. The center of intensity of the subset image is calculated and used to calculate the object's *x* and *y* locations. The equations are given below.

$$\boldsymbol{x[n]} = \frac{\sum_{j=1}^{L_x} j I_{\triangle sub}[n]}{\sum_{j,i}^{L_x,L_y} I_{\triangle sub}[n]} + (I_{\triangle}^{max}[n])_x - L_x \qquad \textbf{(3)}$$

$$\boldsymbol{y[n]} = \frac{\sum_{i=1}^{L_y} i I_{\triangle sub}[n]}{\sum_{j,i}^{L_x,L_y} I_{\triangle sub}[n]} + (I_{\triangle}^{max}[n])_y - L_y \qquad \textbf{(4)}$$

Here *i* and *j* correspond to row and column indices respectively; L_x and L_y are the dimensions of the subset image; and $I_{\triangle}^{max}$ is the pixel of maximum intensity in the subset image.

Below is a sample image, the red dot on fish is the calculated centroid, which is the fish's location.

Figure 3: Sample Image

3.3 Error Elimination

Fish could be found in ideal environment based on the algorithm described in Section 3.2. Unfortunately, environment is not always as good as we expected. There are some facts which could affect the accuracy of the tracking algorithm.

3.3.1 Environmental Noise

Noise is inevitable in this project. Major noises could come from the process of transforming analog signal to digital signal of the input video, small variation of light, water ripple and bubble. Noises have significant influence in accuracy, so eliminating them seems crucial. As we can see from **Figure 2** on page 10, other than the area of maximum intensity, which is the fish, there is some other area, which is referred as noise.

The area of noise also has pixel intensity, so we can set a threshold value of pixel intensity, if the intensity of area is below this threshold value, then they are regarded as noise and ignored. In this project, we set the threshold value as $5*10^4$, actually, most noises could be eliminated and it worked fine.

Finally, the setup of hardware also has effect on noise reduction. Properly setup would increase the immunity of the program to noises. Refer to **Figure 1**, to set up the equipments well, following steps should be followed;

1. Camera should be in the same horizon level with corresponding tank and directly face the tank.
2. Backlight should be placed at the back of tank in the same horizon level with tank. Choose proper distance with tank so that the whole tank could be lighted up.
3. Diffuser should cover the whole back side of tank.
4. Light spot should be avoided, if not, no objects should be in the highlighted area, since a small movement of objects in highlighted area would cause large pixel intensity.

3.3.2 Static Error

Static error exists because most of the time, fish does not move, so it's difficult to find the fish's location just by subtracting current frame with background. When fish does not move, there exist some frames in which the calculated location is not for fish but for some other moving object which has largest intensity in that frame, it could be a bubble or water ripple. An obvious solution is to define a flag with value one when fish is moving and vice versa. When the flag is one do the calculation and find location, otherwise keep the location the same as last frame. However, it's not that easy to directly define such a flag to identify a moving fish from an image.

An alternative approach is to define such a flag in an indirect way. We define a threshold value of intensity **T** and pixel distance **D**. Fish is the largest moving object, we could easily define a threshold value **T** which is larger than normal noise but smaller than the fish. Noise occurs randomly on the whole image, if the calculated location is further than a distance **D** from previous location and at the same time the intensity is below **T**, then this location would be

ignored and treated as noise and the flag would be set to zero. In this scenario, the value of location remains the same as last frame. Otherwise, the program would do calculation and update the location. This mechanism is also intuitive because a sudden jump of location is not expected since fish moves continuously.

Then the major issue is how to define such threshold **T**. We measured average pixel difference between fish and background is 50(black is 0; white is 255), and smallest size of fish is at least 10 pixels by 12 pixels (camera is 360 pixels by 240 pixels). So the resulting **T** is $3*10^5$. We tested this **T** and everything was fine.

3.3.3 Accumulated Error

When reviewing the video recorded for the first test, we found that even the fish is not moving the red dot representing the stored location varied in a small range. Although the variation is quite small which is usually couple of pixels, the overall accumulated error could be very large for a long-time duration when come to the total distance fish moved. The accumulated error occurred because of the change of the centroid. This happened when fish moves its tail but stays in the same location.

We attacked this problem with following approach. Set first location point as a reference point, and put its successive points within 5 pixels in one subset. Then the first point which is out of 5 pixel range would be the next reference point and get another associated subset. Repeat these procedures until allocate all location points into different subsets. Then take average for each subset, the resulting set of points are the new modified location points.
Here is a plot which compares the original location points with modified location points. This plot is in 3D format z-axis represents time.

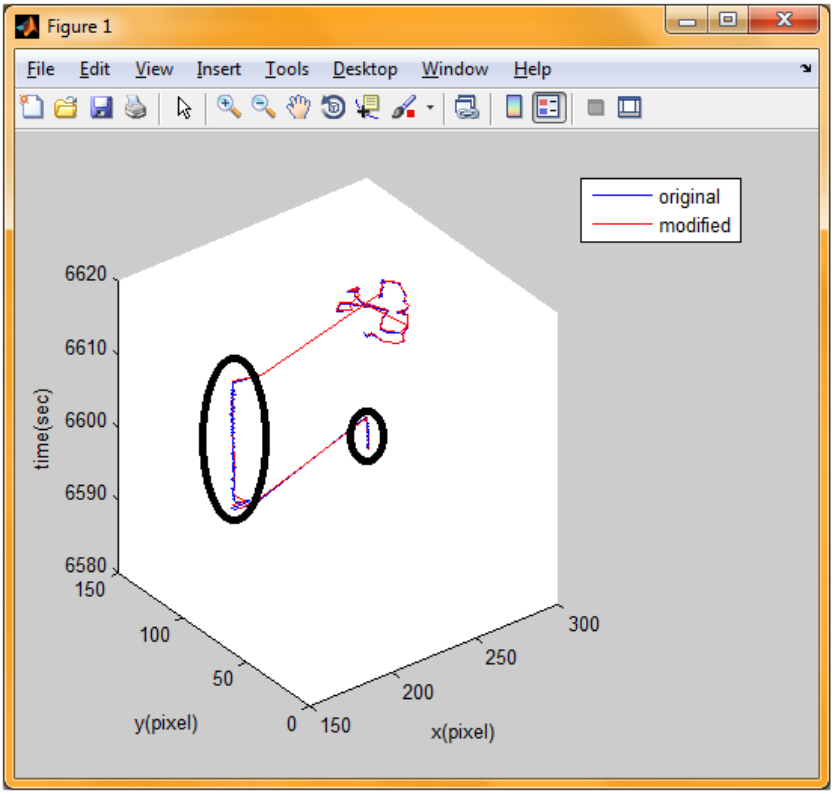


Figure 4: Comparison between original points (blue) and modified points (red)

3.4 Data Storage and Analysis

3.4.1 Data Storage

Our objective is to measure the fish's activity level in different time periods. A simple way to achieve that goal is to measure the moved distance in a fixed period. Since everything is based on pixels and a fair comparison should be based on inches, we define a parameter called **PixelLength** to convert pixels to inches. **PixelLength** is the length in inch two adjacent pixels stands for, which is used to convert the pixel distance to real distance in tank. For example, if the number of pixels fish traversed is **N** in a time period, then the fish moved about **N*PixelLenght** inches in the tank. **PixelLength** is calculated based on the size of tanks as below:

$$\text{PixelLength} = \left(\frac{tank\ width}{pixel\ width\ of\ ROI} + \frac{tank\ height}{pixel\ height\ of\ ROI}\right)/2 \qquad \textbf{(5)}$$

Where ROI means region of interest, which is selected by user.

Basically, locations in pixels are stored to hard drive every minute and release the memory they occupied, which could prevent loss of data when unexpected crash happens and also being memory friendly for long time recording. Besides, total moving distances of fish for each minute and associated time periods will be stored to hard drive also. When program stops, what we saved is the location of fish on every frame and the moving distance of the fish for every minute of the monitoring duration.

The fish's location (which is array of (x,y)) is saved in .mat format Matlab file, the array of distance for every minute is saved in both .mat file and .xls file, all of them can be reused.

3.4.2 Analysis

To meet the requirements of later analysis, we developed a program for data plotting. Bar plot is the default way to present data. And the time interval to save data could be changed based on the user's need. The bar plot is given below:

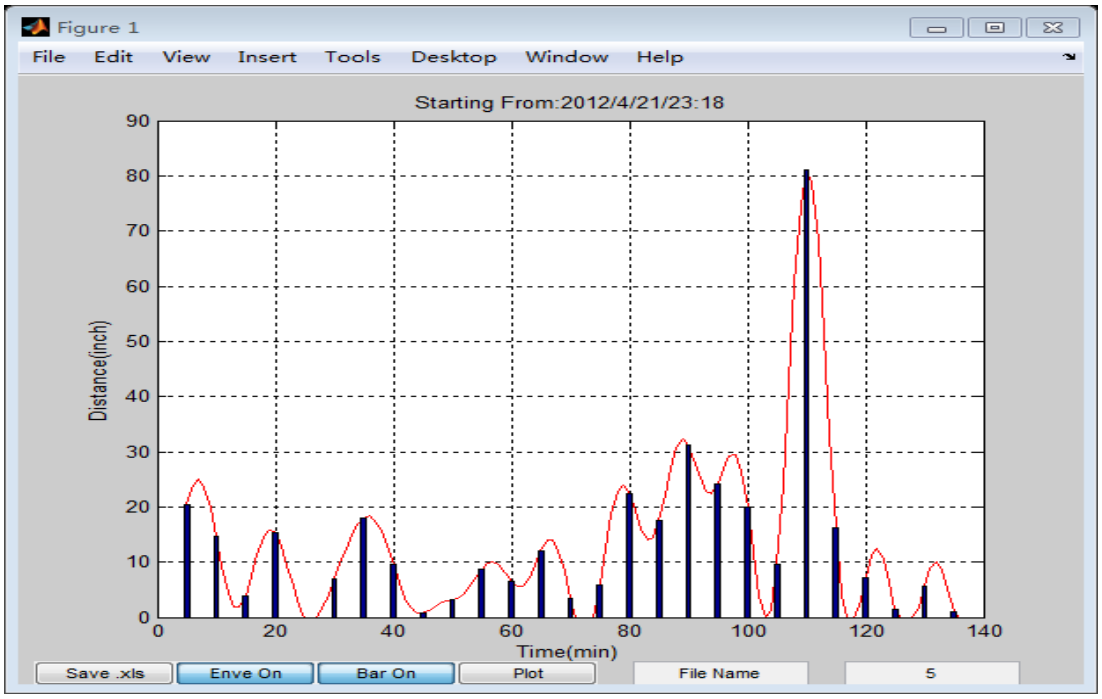


Figure 5: Data analysis for fish distance every 5 minutes

Each bar means the distance fish moved in the selected time period, in this example, it shows the distances fish moved every 5 minutes.

These data could also be saved into a file in .xls format if the user wants.

4 User Interface Control

In this project, a user interface (UI) is required to successfully run the program since some variable values need to be set manually by user, like tracking duration, directory and file name to save data. We designed the UI such that user can easily understand and use. The following figure shows the user interface and UI control.

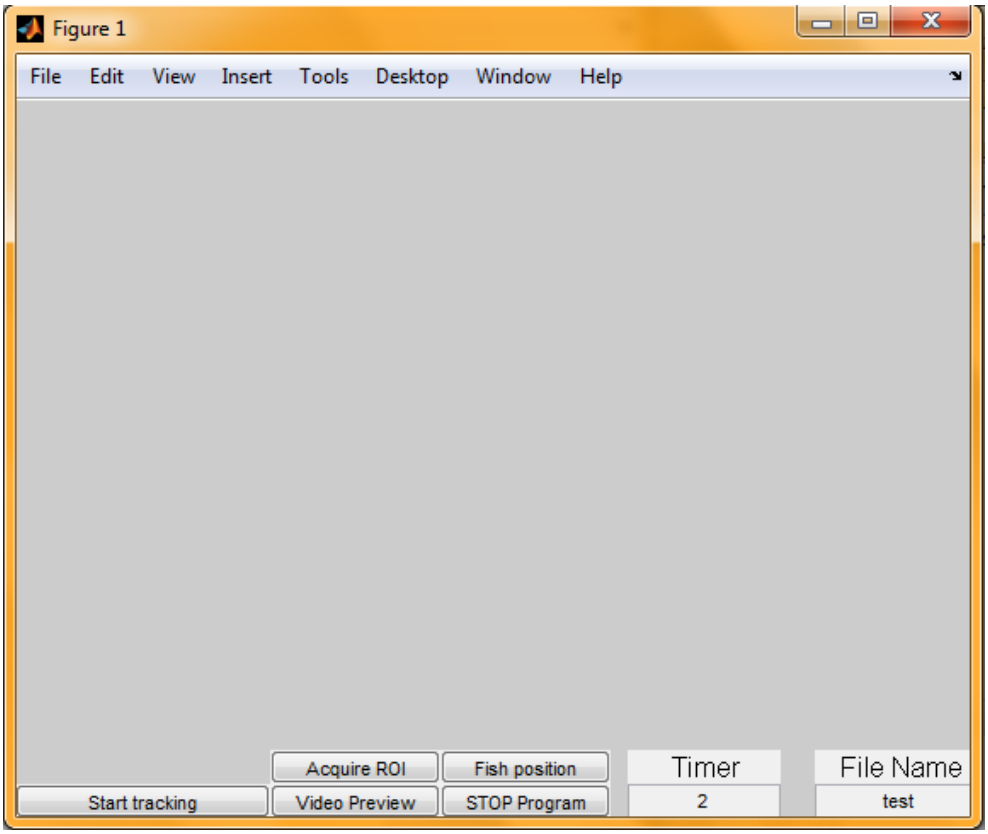


Figure 6: User Interface

There are several controls on the interface, all with different functionalities:

1. **Timer**: the number filled in by user indicates the duration of the program, when time is up, the program stops automatically (in hour).
2. **File Name**: name of file to save data, if the entered file name does not exist, the program creates a folder with the name.
3. **Acquire ROI**: acquire region of interest. When clicked, a snapshot of the camera will show up for the user to manually select the tank, so the program only focus on the selected the area (tank) and anything moves outside tank will be ignored. At the same time, the background will be initialized.
4. **Fish position**: when clicked, a snapshot will show up for the user to manually click on the fish's location. This is designed because in this project, fish rarely moves, when program started, fish's position could not be found by program since nothing moves, so we need to manually locate the fish.
5. **Video Preview**: preview the real time video captured by camera.
6. **Start tracking**: program starts to track when clicked.
7. **STOP Program**: to stop the program manually.

5 Test and Result

In order to meet the requirement of long time reliability, we tested this program 4 times in real setting. The test environment was:

1. Room temperature, 20 °C
2. PC CUP: Intel(R) Core(TM) i7-2620M CPU @2.70GHz
3. PC memory: 4.00 GB
4. Frame number: 20 frames/second

All videos were recorded and more details are shown in following table:

	Duration(Hour)	Date	Total Frames Analyzed	Comment
1	24	3/22/2012	$1.7*10^6$	1 camera
2	9	4/6/2012	$1.3*10^6$	2 cameras
3	72	4/13/2012	$1*10^7$	2 cameras
4	24	5/2/2012	$5.2*10^6$	3 cameras

Table 2: Duration and Data

The test result for the last test, with 3 cameras is shown below:

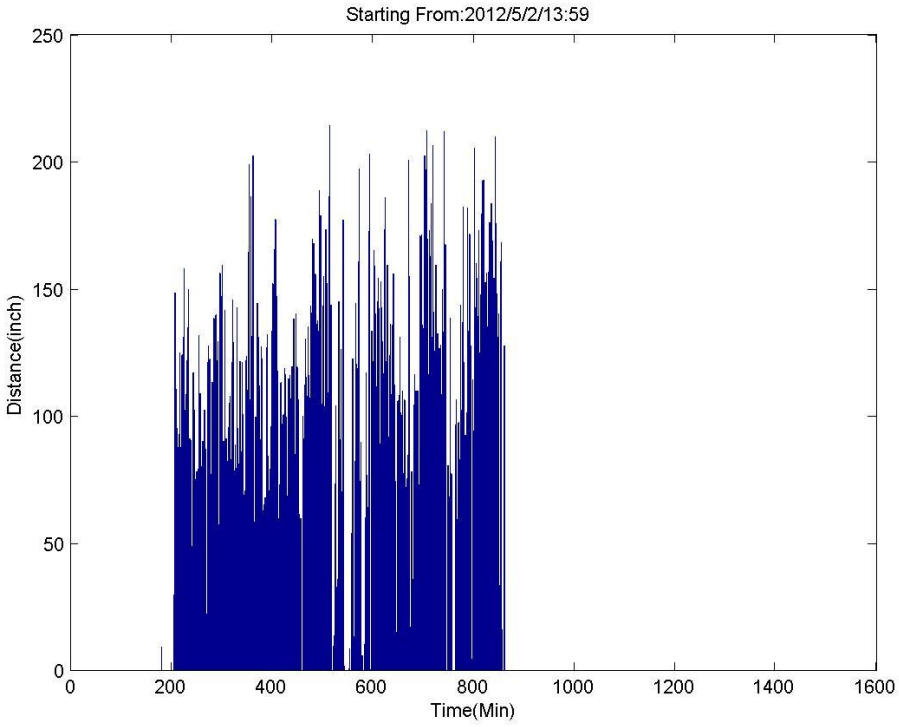


Figure 7: test result for fish 1

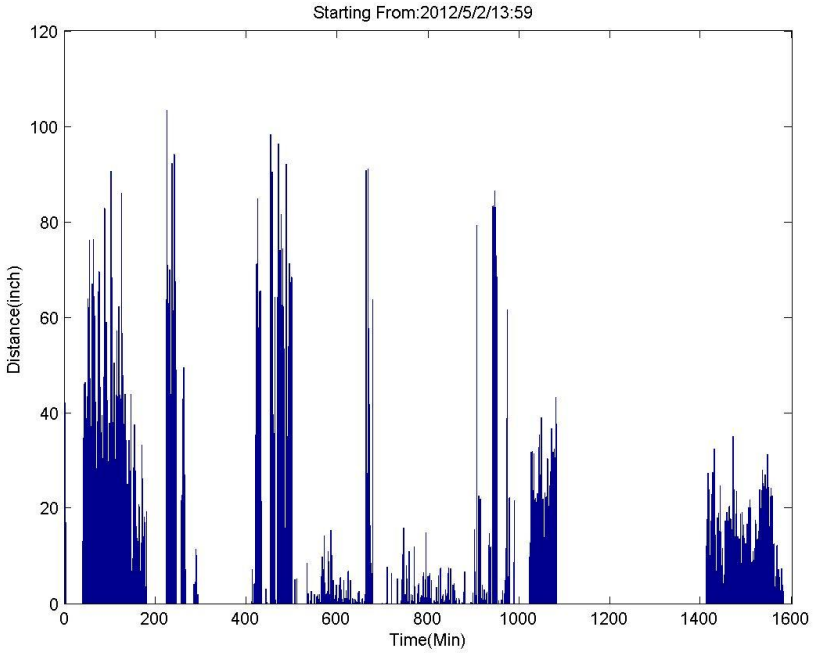


Figure 8: test result for fish 2

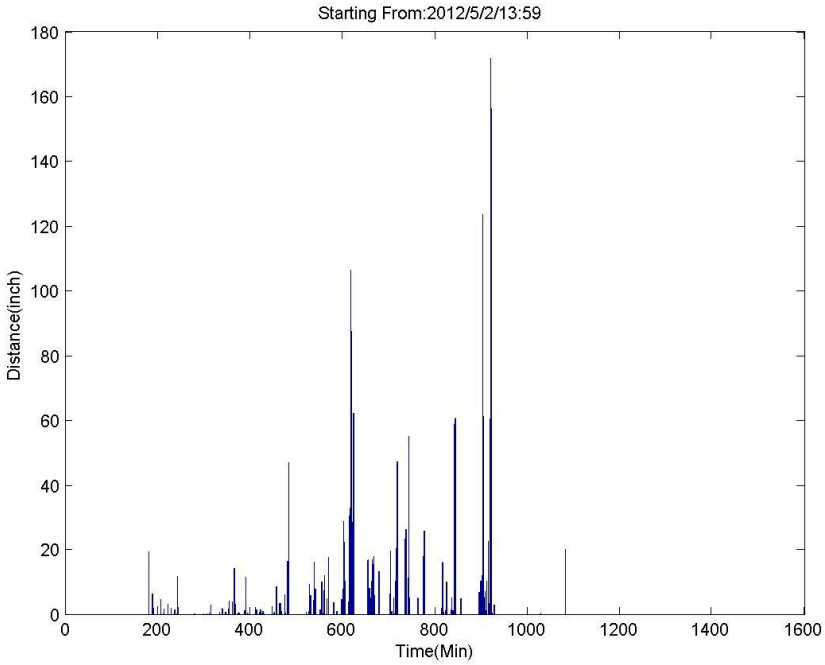


Figure 9: test result for fish 3

We met static error problem (Section 3.3.2), accumulated error problem (Section 3.3.3) and memory problem when doing tests in lab.

Memory occupation kept increasing even we deleted everything we used. We spent couple of nights debugging the program line by line. Finally memory problem was perfectly solved after we re-wrote a build-in Matlab function called *regionprops* which kept eating memories.

regionprops is a function that we used to find the area of maximum pixel intensity, however, it also has a lot of other functionalities. It's recursive and each time called, reginprops calculates lots of variables even though what we need is only one or two of them and fails to release those variables. Since this is not a typical functional error, traditional debug cannot find the problem, so we tested every module and put them into a loop to check the memory, finally found the origin to memory problem. We re-wrote a function similar to *regionprops,* but more simplified, which solves the memory problem.

Our test showed that on the PC, when running the program for 3 cameras, 25% CPU was used, 35% memory used, the memory occupied by Matlab was less than 1GB. The occupancy of CPU and Memory keeps constant during the monitoring and no accumulation happened. Depends on these, we believe our program can support 10-15 cameras simultaneously on this PC, but it also depends on the data processing capability of Matlab.

It showed that this program functions well for long time duration, the longest test lasted for 72 hours. Besides, accuracy was proved after we compared stored data with recorded video. A screenshot of the 3-camera monitoring is:

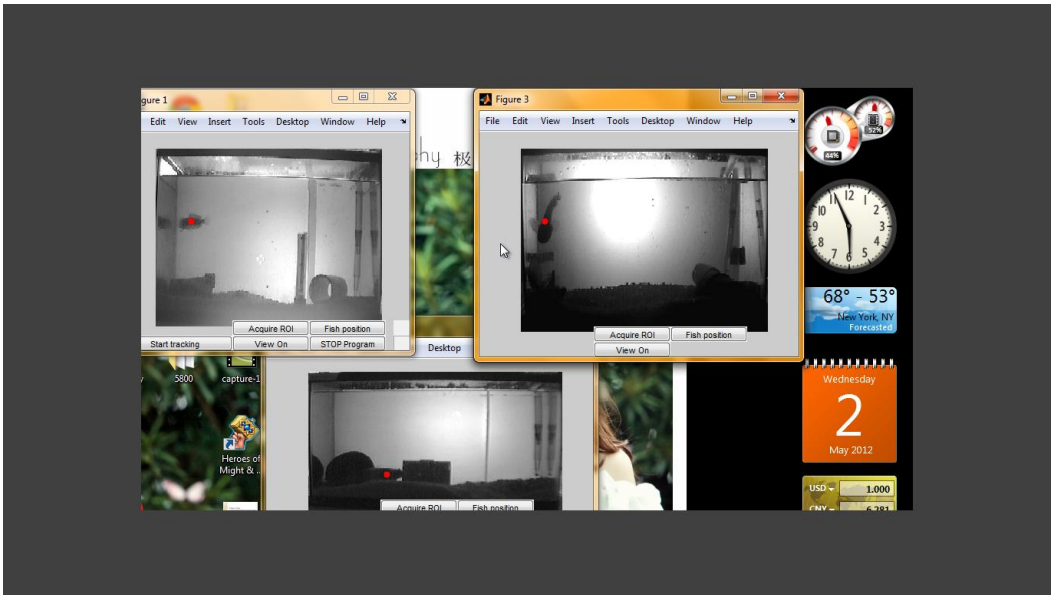


Figure 10: 3 fish in 3 tanks

6 Conclusion

This program works well and all requirements are met.

- Accuracy: comparison between recorded videos and stored data shows high accuracy
- Reliability: the 4 tests showed very good reliability of the program, it supports 72-hour recording and does not occupy much memory.
- 3 fish in 3 tanks: we could track 3 fish in 3 different tanks simultaneously, besides, more fish could be tracked just by making a slight change to the program, but that is not the scope of the project, we focus on 3 fish in 3 tanks here.
- User-friendly: the program interface is easy to understand and use for non-software background users.
- Easy data access for later use: all the fish locations will be saved into files and fish distances saved into a different file. These data is accessible and could be used any time.

7 Acknowledgements

1. We would like to acknowledge the advice and guidance of our ECE advisor, Bruce Land and outside ECE advisor, Ni Feng.
2. We also want to thank Dawnis Chow from Department of Neurobiology and Behavior for sharing information and materials with us for the project.

8 Reference

[1] Dan Valente, Ilan Golani, Partha P. Mitra, (2007). Analysis of the Trajectory of Drosophila melanogaster in a Circular Open Field Arena, PLoS ONE 2(10): e1083.

Appendix

User Manual

1 About

This program was created to meet the specific needs of Cornell's Bass lab in the Department of Neurobiology and Behavior.

All Matlab functions were created and tested in the Matlab 2010b version.

2 Hardware Setup

-connect cameras to PC via adapter

-properly place cameras (it should be in the same horizontal level with corresponding tank)

-place backlight on the back of tank (choose proper distance so that the whole tank could be lighted up)

-place diffuse material on the back of tank (try to smoothen light source and avoiding light spot)

Be cautious to:

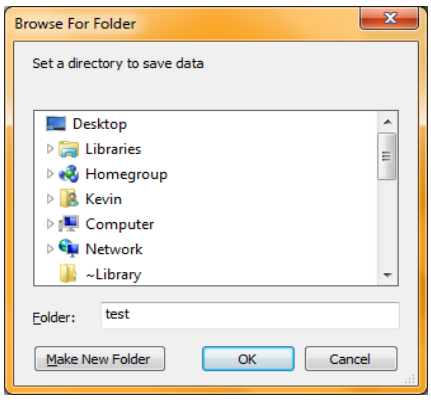
1. place camera, backlight and tank at the same horizontal level

2. try to avoid light spot.if can't, make sure no objects in highlighted area. (program is more sensitive to objects in highlighted area)

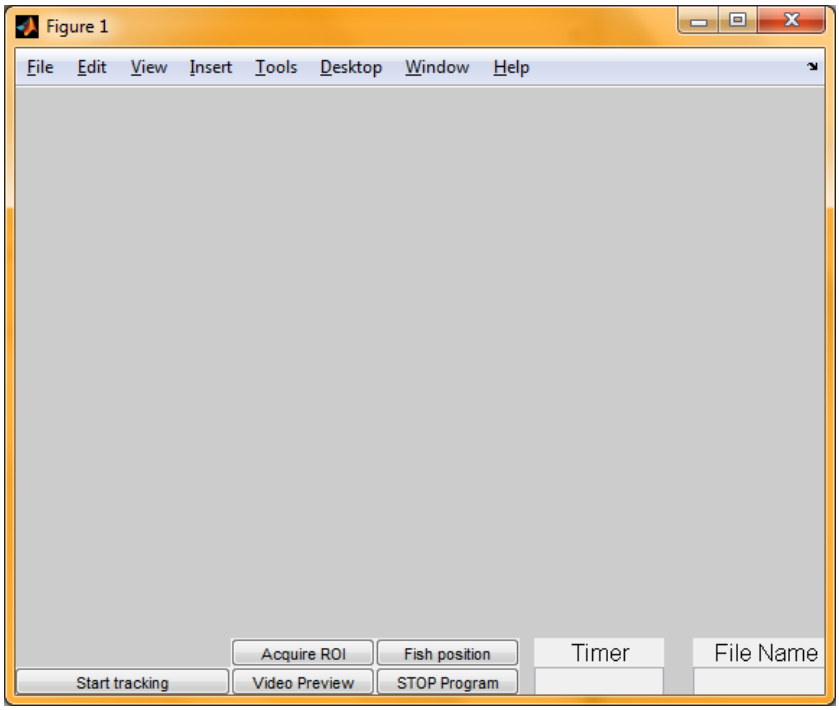
3 FishTrack

1. Run **Fishtrack3.m**

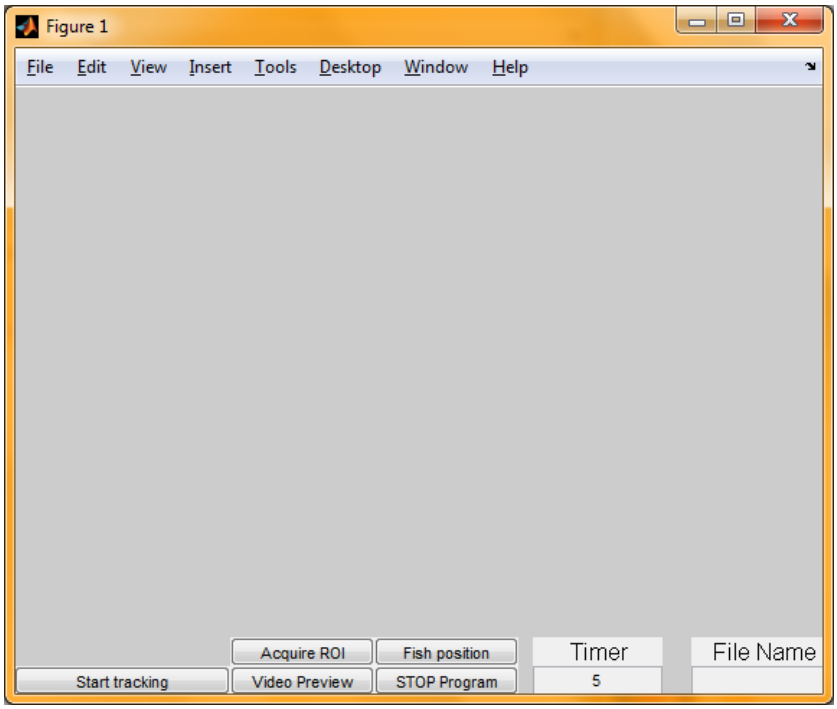
2. Choose directory to save data



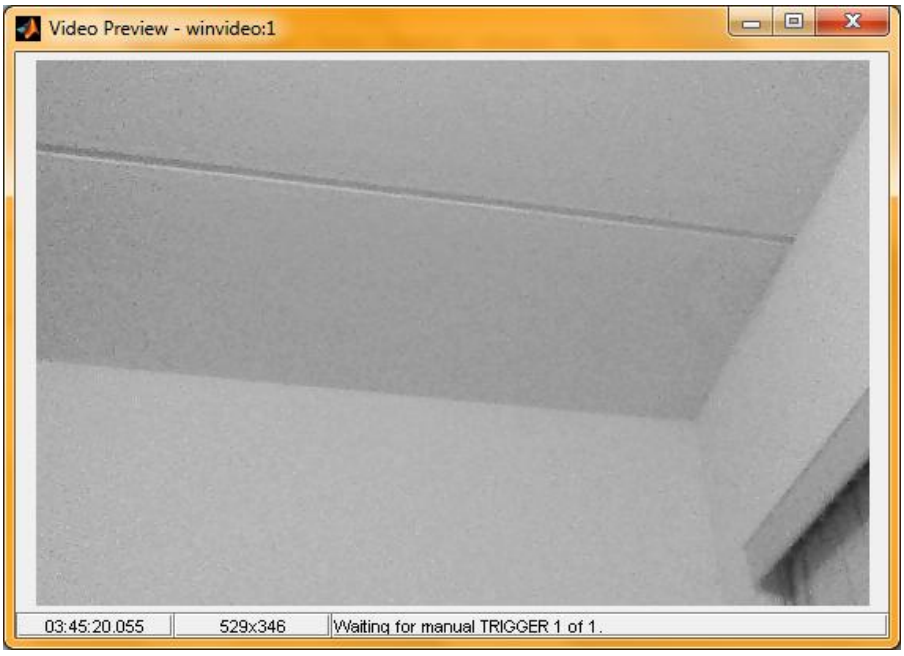
3. Set **Timer**: enter the time duration you want to monitor the fish. For example, for 5 hours, enter 5.



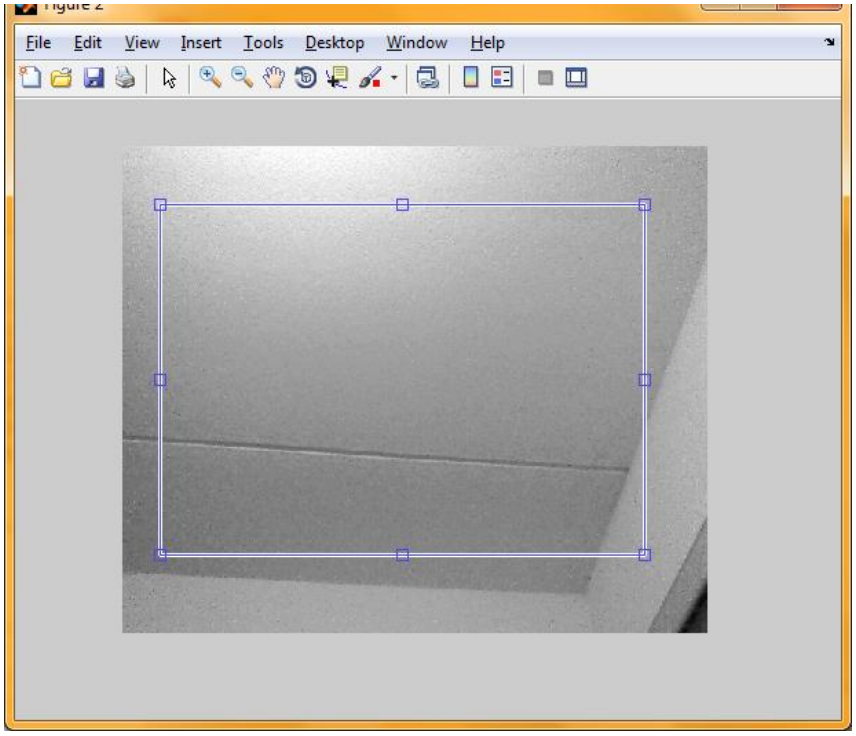
4. Set **File Name:** enter the file name where to save the data, like "test"



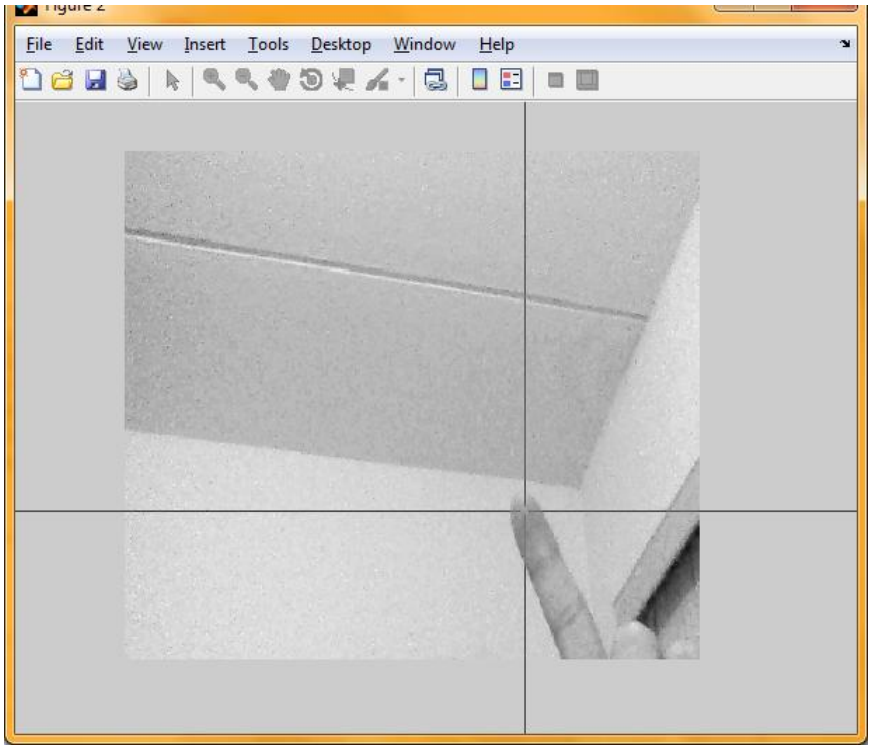
5. Click **Video Preview** to preview the real time video and adjust the position of camera



6. Click **Acquire ROI**, use mouse to drag the square to select the region of interest, and then double click. (ROI should include the whole tank, and be as large as possible)



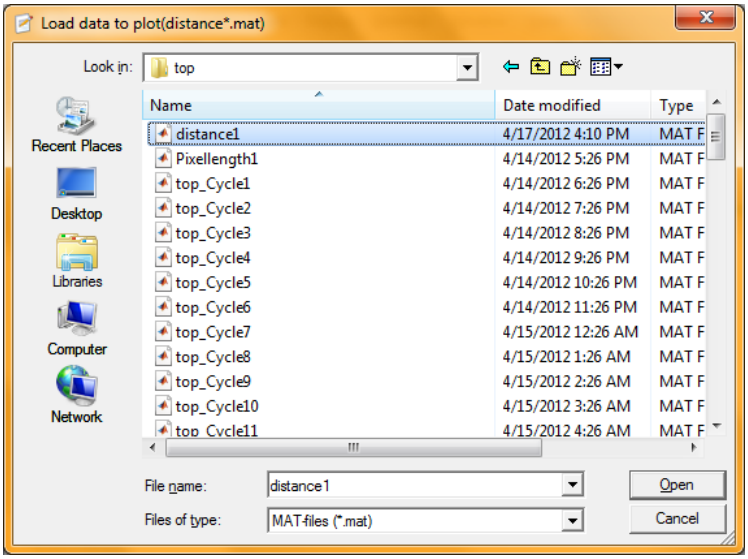
7. Click **Fish position**, to manually point out the fish position



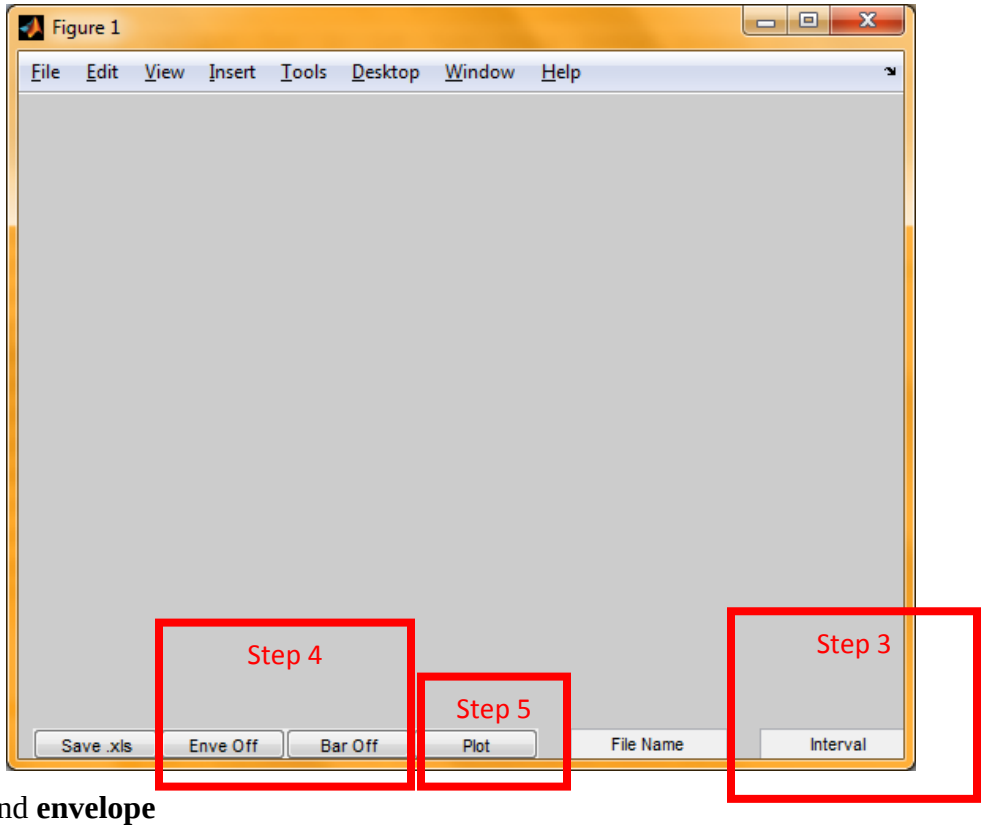
8. Click **Start Tracking**, the program will start to monitor the fish.
9. Click **STOP Program** to terminate.

4 FishTrackPlot

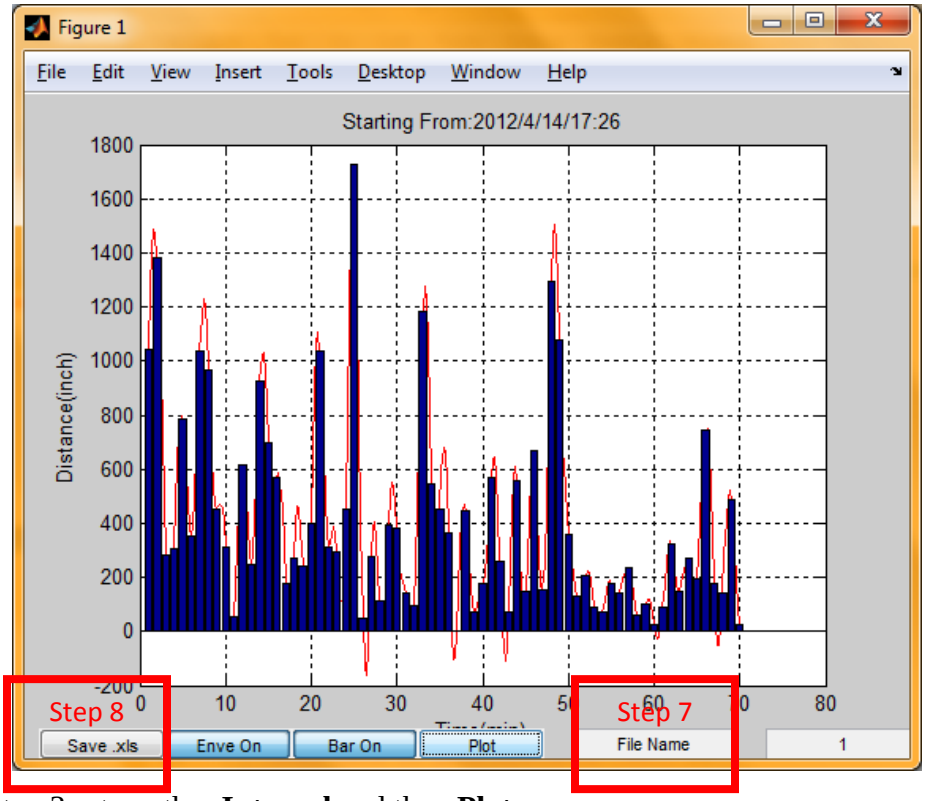
1. run **FishTrackPlot.m**
2. choose distancex.mat (must be distancex)



3. set **Interval** (must be integer)



4. set **bar** and **envelope**
5. click **Plot**



6. back to step 3 set another **Interval** and then **Plot**
7. set **File Name**
8. after click **Plot,** modified data could be saved as excel file by click **Save .xls** (**File Name** and **Plot** must be set before click **Save .xls**)
9. figure could be saved by **menu bar** => **File**=>**Save As** (could be .jpg or .fig by choosing from **Files of type**)