Multiple-object Tracking and Its Application to Fly Behavior Analysis

TOHOKU UNIVERSITY
Graduate School of Information Sciences

Multiple-object Tracking and Its Application to Fly Behavior Analysis

Submitted to the Department of System Information Sciences
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy (Information Sciences)

by

Sirigrivatanawong Pudith

March 24, 2017

Multiple-object Tracking and Its Application to Fly Behavior Analysis

Sirigrivatanawong Pudith

Abstract

The use of computer has been applied to many fields in our daily life including personal use to a larger scale like industry applications. Furthermore, some specific tasks are beyond human ability can be achieved. In this way, using computer to process visual data that may be in form of digital image or video, computer vision comes to take action and makes the thing possible. Multiple-object tracking is one of the most important tasks in computer vision and dealing with multiple objects that present in a scene is the main focus in this study. Together with knowledge of control, multiple-object tracking can be applied to many applications. Normally, dealing with multiple objects is nontrivial and requires additional processes comparing to a situation of single object. Identity assignment is very simple in the scene of a single object. However, in a scene of multiple objects, the identity assignment may not be a problem in case of distinguishable objects that may be form manually tagged or different in appearance of the objects. However, it will be a very challenging task in the case of indistinguishable objects. Dealing with indistinguishable objects can fulfill some issues as difficulty of labeling or tagging the objects and requirement of addition tool to observe the tagged objects.

In the first part of this work, a group of distinguishable objects was studied by means of the distribution of mobile lake monitoring sensors. Under an assumption that each sensor has its own identity throughout the experiment, identity assignment becomes unnecessary. A controller that control the group distribution by using the centroid and the variance of the group is proposed. The proposed controller considers the existing water flow field and uses it in the calculation process to achieve lower energy consumption by the help of the water flow field. With a distribution method that requires less energy, the sensors can reach their objective with sufficient amount of energy that would allow a greater overall operating time of the monitoring system, which would in turn result in higher monitoring system efficiency. Later in this work, multiple-object tracking of a model organism was focused as a group of indistinguishable objects. Study of behavior changes in model organism is an important part of biologists research. This requires successful tracking of each individual in a group of the moving animals. But, manually performing tracking analysis is a laborious work. Though, machine vision is a useful tool for locomotion analysis since it can provide more precise and time efficient measurements. For analysis of a group of Drosophila (common fruit fly) as the model organism using machine vision, many image processing techniques are required in order to obtain good detection result, which is the first step of the tracking system. The most challenging tasks in tracking indistinguishable objects is identity assignment between frames since all objects have the same appearance.

For the tracking step, combining information such as position and heading direction with assignment algorithms gives a successful result robust to identities swapping. Unlike other methods, introducing the use of heading direction increases the system efficiency of dealing with identity loss and flies swapping situations. The system can also operate with variety of videos with different light intensity condition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation of the Study

Now a day, the use of computer has been applied to many fields in our daily life. From a small scale like personal use to a larger scale like industry, computer is used to perform a specific task to make life easier and even used for the task that is beyond human ability. Using computer to process visual data that may be in form of digital image or video, computer vision comes to take action and makes the thing possible. Multiple-object tracking is one of the most important tasks in computer vision and dealing with multiple objects that present in a scene is the main focus in this study.

Computer vision techniques used in multi-object tracking are simply sequential uses of single object tracking. In single object tracking, variety of algorithms are used. Some examples of these algorithms are as follows

1. Pixel-pattern-matching-based algorithms

   (a) Lucas-Kanade [7]

   (b) ESM (Efficient Second-order Minimization) [8]

2. Feature-based algorithms
    (a) SIFT (Scale Invariant Feature Transform) [9]
    (b) HOG (Histograms of Oriented Gradients) [10]
3. Machine learning
    (a) Boosting [11]
    (b) MIL (Multiple Instance Learning) [12]
    (c) TLD (Tracking-Learning-Detection) [13]
    (d) Median Flow [14]
    (e) KCF (Kernelized Correlation Filters) [15]

In multi-object tracking, human tracking is a very popular topic among researchers all over the world. Fig. 1.1 shows a human tracking in action with colored-rectangles. In the figure, three people are tracked separately. Computer vision can distinguish these people by using features that consists of color information etc. Apart from human tracking, which many features can be used for the tracking, *Drosophila* or common fruit fly has some more difficulty. Fig. 1.2 illustrates an appearance of three *Drosophila* in different pose. According to the figure, it is quite difficult to track the flies in the following aspects

1. they have the same appearance
2. individual fly occupies few pixels that does not have enough features to detect head direction
3. they may fly (their position and orientation can change suddenly)

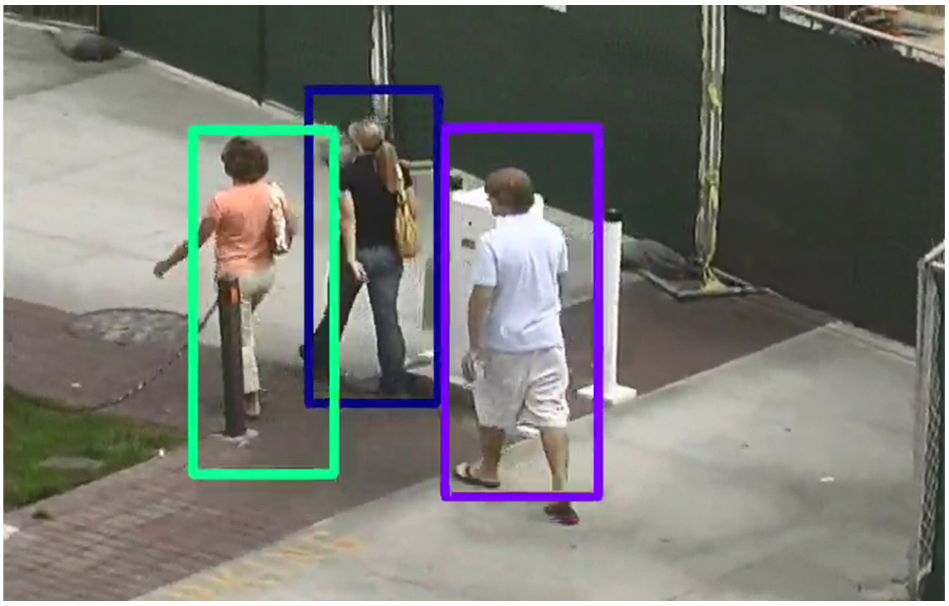


Figure 1.1: Human tracking scenario. Three people are tracked separately with colored-rectangles.

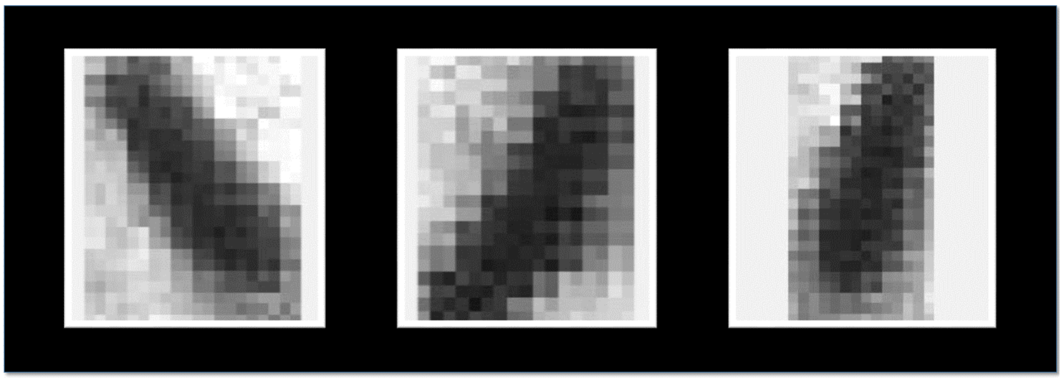


Figure 1.2: Fly appearance in three different poses. All fly look the same since they have similar appearance.

In similar appearance object tracking, dealing with multiple objects is nontrivial and requires additional processes comparing to a situation of single object. Identity assignment, which is the main part of tracking process, is very simple in the scene of a single object since there is only one identity. An example of single object tracking

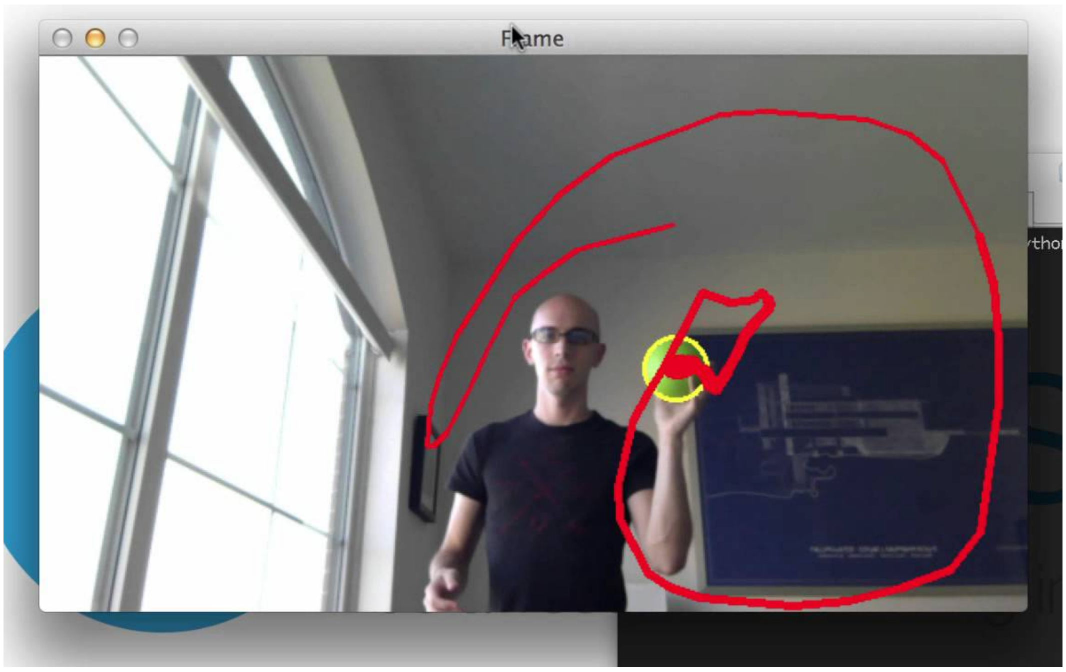


Figure 1.3: Single object tracking using shape and color data [1]. The identity of the object can be easily preserved since there is only one object presence in the scene.

using shape and color data is shown in Fig. 1.3. According to the figure, a green ball is being tracked and encircled by the yellow circle. The red line represents the trajectory of the tracked ball.

In a scene of multiple objects, the identity assignment may not be a problem in case of distinguishable objects that may be form manually tagged or different in appearance of the objects. Fig. 1.4 gives a scene of multiple-object tracking with distinguishable features. There are two controllers with a illuminating ball at the end. Each ball has different color that helps identifying whether it is left or right controller. However, it will be a very challenging task in the case of indistinguishable objects. Dealing with indistinguishable objects can fulfill some issues as difficulty of labeling or tagging the objects and requirement of addition tool to observe the tagged objects.

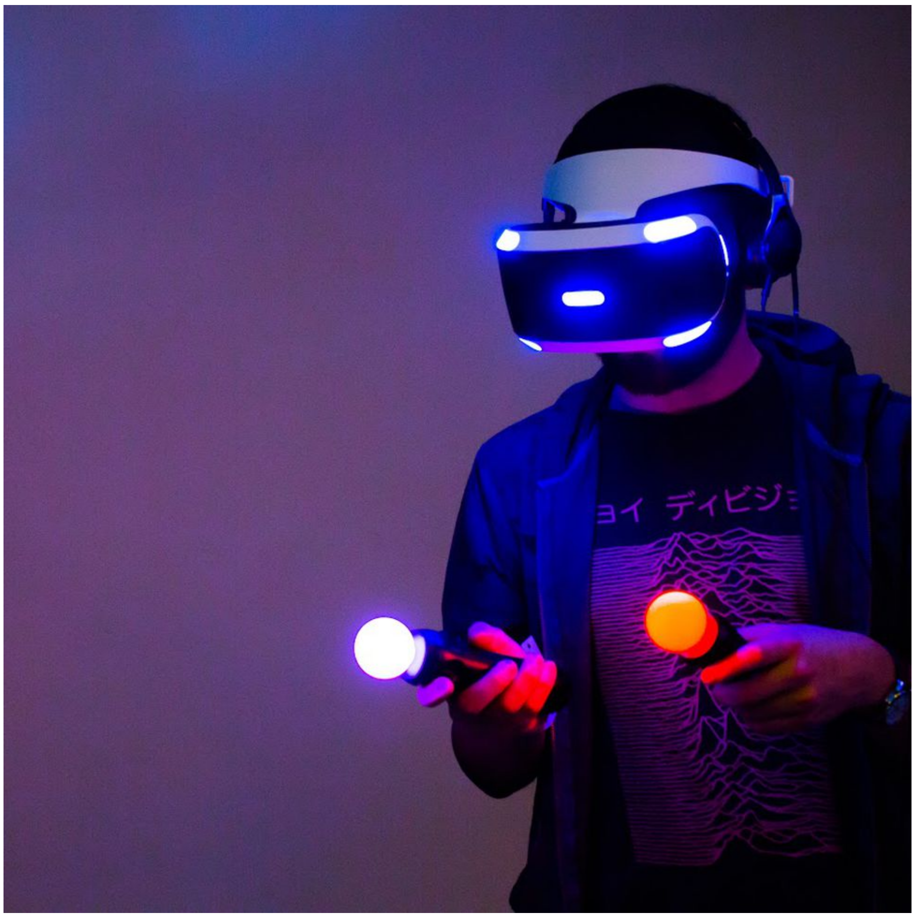


Figure 1.4: Multiple-object tracking with distinguishable features [2]. Illuminating balls at the controllers are tracked with a camera to detect the movement of the player. Identity of each track traces left and right data separately.

For big objects like robot and human, the objects can be distinguished by attaching some marks varying in color or pattern. But in tiny object observation, which the requirement of computer vision is obviously more demanded, it is very difficult to mark each object. Especially in biological field, even if it is possible to mark, after an experiment has been done all the samples are normally replaced with new samples. It would be very time consuming if we have to do the marking every time. Furthermore, attaching the tiny object with observable marker may affect the movement of the object and this will lead to non accuracy observation For biological behavior analysis, tracking a lot of objects individually is required. *Drosophila* is a good target to analyze the behavior differences that come form genetic modification since its manipulation technique has been well-developed.

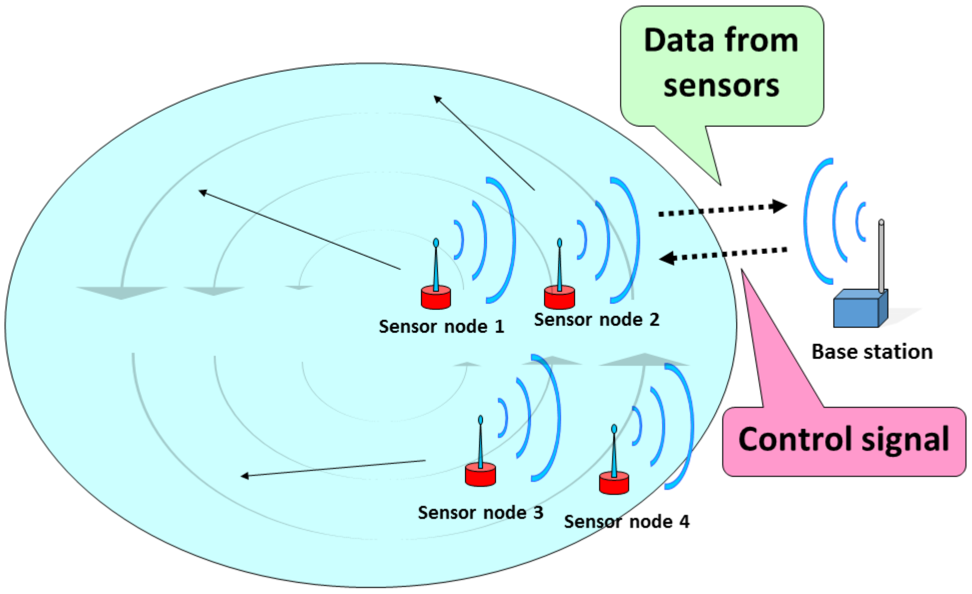


Figure 1.5: Distribution concept of floating lake monitoring system consisting of multiple-object. All objects are distinguishable with ability to gather data around their position. In this scenario, there are four floating sensor nodes initially deployed at the same area. The purpose of the operation is to distribute all the sensor nodes throughout the lake with using control signal from the base station located at the shore.

1.2 Objective

In this work, we focus on tracking of multiple-object and its application to fly behavior analysis. From the perspective of this study, each object in the scene is affected by the state of other objects in the same scene. In order to achieve the successful tracking result, various image processing and algorithm techniques are used in the process. This work especially aims at achieving tracking flies without restriction of their motion while keeping IDs in cases of crossing, piling up, suddenly backing, and suddenly flying. The multiple-object tracking are divided into four main chapters. In chapter 2, a work concerning multiple-object with distinguishable capability is discussed. The control method used in this chapter is a basic technique of statistical process of a group consisting of multiple objects. Fig. 1.5 shows the distribution concept of floating lake monitoring system consisting of multiple-object.

In chapter 3, multiple *Drosophila* detection and tracking system in frame pair approach is discussed. This part focuses on a situation of multiple indistinguishable objects that are in form of *Drosophila Melanogaster* or a common fruit fly in a closed arena. Fig. 1.6 shows a scene of multiple *Drosophila* in a circular closed arena. Flies are illustrated as black oval shapes. Image processing techniques used in the work are explained. For the tracking section, position of each fly is used as the attention control to achieve the corresponding pair of each fly in two consecutive frames. The result provides useful information of average velocity in specific frame pairs that can be use in further behavior analysis.

In chapter 4, multiple *Drosophila* detection and tracking system in multiple consecutive frames approach is introduced. This chapter provides a detail of improved multiple *Drosophila* tracking program. All the flies are indistinguishable and combination of various techniques that are applied to achieve a successful tracking result. Not only providing a successful result of tracking in two consecutive frames pair, the program also gives well tracking result of each fly throughout all frames in an input video.

In chapter 5, behavior analysis of a fruit fly is discussed. The output data from the tracking program is used with random forest classification to classify the behavior of a fly in each consecutive frame. In addition, the result of the classification is compared with manual process done by human to show the ability of the tracking program.

1.3 Contributions of Individual Chapter

1.3.1 Contributions of Chapter 2

In this chapter, the main contribution is the usage of attention control that is in form of force from water flow field to assist the operation of lake monitoring sensors group. Using the tracking data obtained from each object together with water flow information, the group of the monitoring sensor nodes can utilize the water flow to help preserving the battery power, resulting in longer operation time. The design of the lake monitoring sensors system will also be introduced. The dynamics model of each sensor node that is considered in this work will also be discussed.

1.3.2 Contributions of Chapter 3

In this chapter, the system was used for observing locomotion activities such as linear velocity and absolute angular velocity of *Drosophila* in [16]. Their work is a study of functions in sugar receptor neurons between and within taste organs of

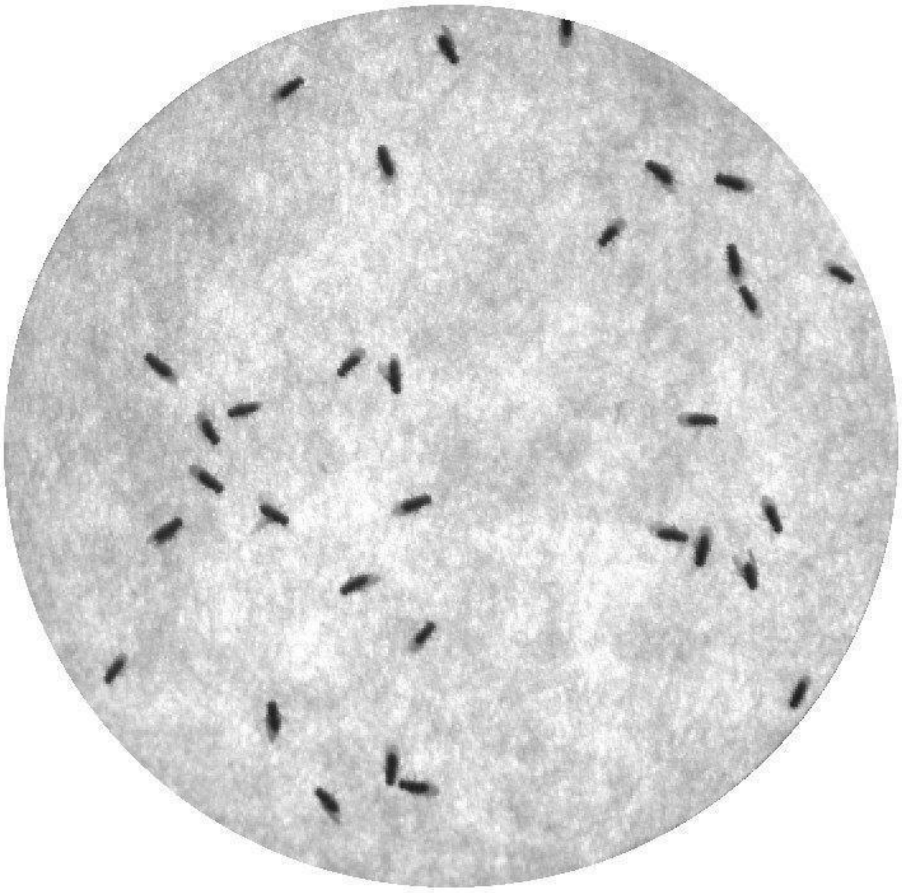


Figure 1.6: A scene of multiple *Drosophila* in a circular closed arena. This is an example of indistinguishable multiple-object tracking. Since the size of the fly is approximately 2 mm, it is very difficult to tag or mark each fly.

Drosophila. The result from the locomotion activities observation shows the response of *Drosophila* to the presence of sugar. By blocking sweet taste receptor neurons, the velocity of *Drosophila* will not decrease even if these flies encounter sugar, because their sugar detection is impaired.

1.3.3 Contributions of Chapter 4

In this chapter, *Drosophila* tracking software that can successfully track each individual fly throughout the video is introduced. There are many *Drosophila* tracking software available and C-Trax [17] and idTracker [18] are some of the most well-known tracking programs. These tracking programs require wing clipping before the experiments to restrict the only walking. In addition, both systems have difficulties when tracking multiple flies especially for identities swapping. This work provides a solution to handle identity swapping in crossover and touching scenarios that is an identity assignment failure in the tracking.

1.4 Thesis Outline

The rest of this thesis is organized as follows. In chapter 2, a work concerning multiple-object with distinguishable capability is discussed. In this chapter, the usage of attention control that is in form of force from water flow field to assist the operation of lake monitoring sensors group. Simulation result of sensors distribution on the simulated lake is shown at the end of this chapter. In chapter 3, multiple *Drosophila* detection and tracking system in frame pair approach is presented. The results of posture estimation and tracking in frame pairs are provided followed by multiple *Drosophila* detection and tracking system in multiple consecutive frames approach in chapter 4. In chapter 5, behavior analysis of fruit fly by the usage of tracked data and by human are discussed. Lastly, conclusion and future work will be discussed in chapter 6.

Chapter 2

Control of distributed sensors

2.1 Introduction

Water is a natural resource that is the most important to humans, animals, and plants [19]. Water resources are divided into many categories such as river, lake, and ocean, according to the physical characteristics. These natural resources are being polluted because of various human activities. From this reason, it makes the demand of water monitoring system increase for a safer environment [20]. In recent years, there are many studies on physical quantities of the water resources and many researches have been done on the measurement of the flow field of water in river, lake, and ocean, such as [21], [22], [23], [24], [25], [26]. For the water monitoring system, many aspects can be monitored such as chemical component, pH, local weather condition, behavior of fish, and invertebrates [27]. Water flow field is one of the most important information of the water resources for the monitoring system. Water flow field data is important for studying mechanism and ecosystem of the source. Furthermore, with flow field data, scientist can predict the flow of contaminant in case of accidental leakage of pollution, which is a part of water quality warning system.

The water flow field data can be mainly collected by using

1. Satellite information;
2. High frequency radar;
3. Sensor nodes.

Sequence image from satellite is used in [24] to trace inverted ocean surface flow field. Surface ocean flow field sensing by using satellite is discussed in [28] and [29]. Using satellite for tracking other information like ice sheet velocity is done by [30]. Larger coverage area is an advantage of using satellite to measure the surface flow field. The work of water flow field measurement using high frequency radar is shown in [21], [22], and [25]. Ultra high frequency is used by [23] to observe river flow velocity. Using sensor nodes can measure the water flow field and water temperature directly from the source. Moreover, the method can perform various kinds of measurement e.g. oxygen dissolved, fish tracking, and toxic level measurement. The disadvantages of this method are limitation of battery life and the additional requirement of control strategy to control the group of mobile sensor nodes. Floating sensor networks is used by [31] to perform local measurement and to track flow in water systems. Wireless floating sensor system is used by [32].

In this research, we focus on using mobile multiple sensor nodes to collect data from a lake. Besides river and ocean, dynamic of lake is smaller comparing with those two resources. This simplifies the simulation. Since the sensor nodes need to use energy from the batteries attached with them, limitation of energy for mobile monitoring unit has been a problem for the monitoring system. Therefore, if the energy consumption for the sensor nodes distribution can be reduced, there will be more energy left in the limited battery that will allow the monitoring system to last longer, which result in higher monitoring system efficiency. An example of energy consumption problem is stated in [33]. Each raft has only 5 hours of battery life, which is not suitable for long-time monitoring system.

Since there exists water flow in the lake, the idea of utilizing the flow to assist the movement of each sensor node can be the answer to the power consumption problem. In this research, a water monitoring distribution system is developed under the following two points: low power consumption of sensors distribution.

Some works involve with controlling a system with a group of agents. [34] addresses the problem of controlling a group of agents towards a consensus point. A leader-follower formation control of multiple autonomous mobile robots has been studied by [35]. [36] focuses on the control problem of multiagent systems under unknown and persistent disturbances. Although these works have provided the control input results, but none of them lowers the amount of control input by using help of water flow field. For the flow field estimation, [37] shows a new developed approach to reconstruct a 3-dimensional incompressible flow from noisy data in an open domain using a two-scalar (toroidal and poloidal) spectral representation. Recovering Fluid-type Motions Using Navier-Stokes Potential Flow is addressed by [38].

The main objectives of this research are to distribute lake monitoring sensors by using low energy consumption. The distribution is expected to be able to achieve the desired goal with using help of water flow field for low power consumption.

2.2 Overview of the lake monitoring sensors system

In this section, the overview of the lake monitoring sensors system will be briefly discussed. Moreover, a concept of the proposed monitoring sensor node will also be introduced. In water resource monitoring system, there are some examples of currently used floating sensor node to perform water monitoring task in a river or a lake. Floating sensor nodes without actuators are shown in Fig. 2.1. These floating sensor nodes can directly measure water quantities such as water flow velocity, temperature, and depth. Unfortunately, they cannot perform movement to gather data around a wider area. The ability to control the floating sensor node is not available for this kind of floating sensor node. Each sensor node is carried around by the flow of water. Therefore, the area of interest may not be reached.

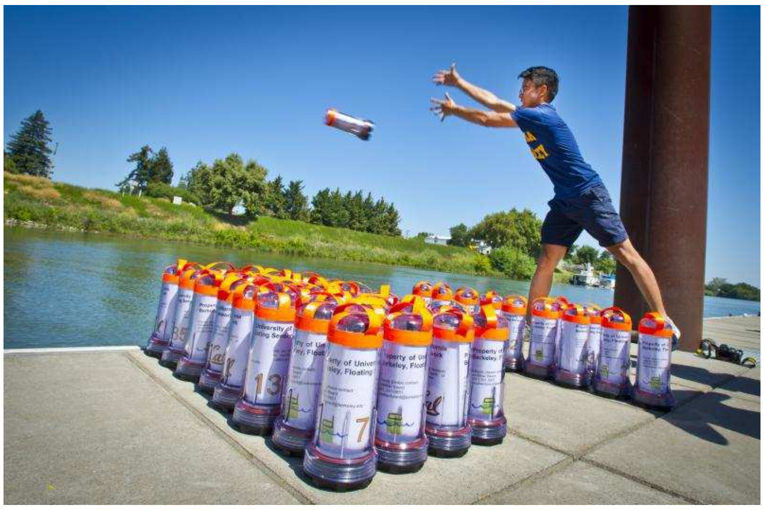


Figure 2.1: Non-actuated floating sensor node is used to do the water monitoring task [3]. The sensor node cannot perform movement by itself, therefore the target area may not be reached.

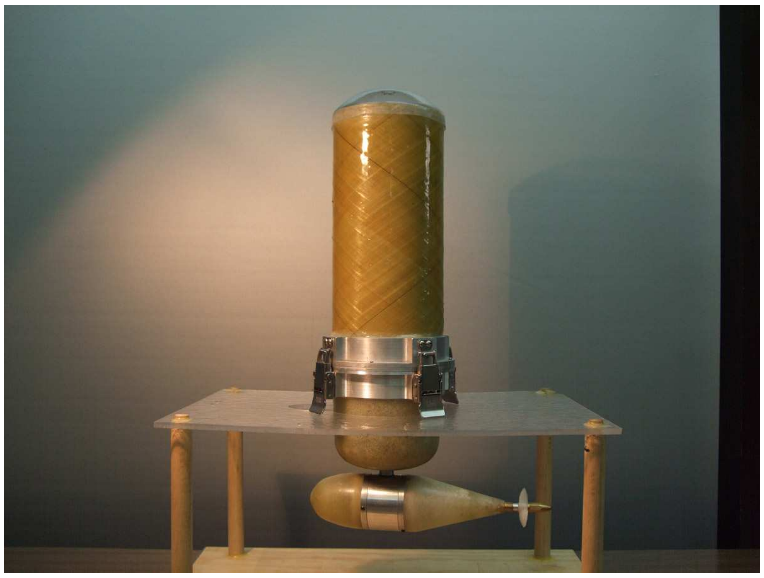


Figure 2.2: Early prototype of an actuated water monitoring sensor node was developed based on the drifter project of University of California, Berkeley in 2007. At the bottom of the sensor node, there is a single rotating motor added to it. [4]

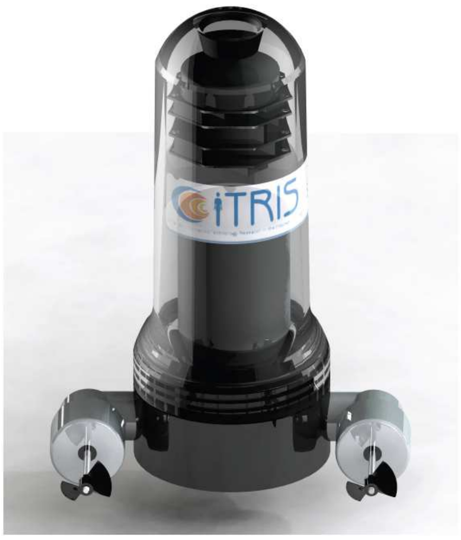


Figure 2.3: A movable floating water monitoring sensor node with cylinders 5″ in diameter and 17″ tall float (motorized, with their major axis vertical) and communicate water quality information via cell phone and short-range radio. [5]

The sensor node that was developed from a non-actuated sensor node toward an actuated sensor node is shown in Fig. 2.2. A single rotating motor pod is attached to the bottom hull to make the sensor node possible to move around on the water surface. Another example of currently used floating sensor node for water monitoring system is shown in Fig. 2.3.

This floating network project is a project at University of California, Berkeley joining with the Lawrence Berkeley National Laboratories and the California Department of Water Resources, in USA. The design of this floating sensor node and the sensors contained inside the body are shown in Fig. 2.4.

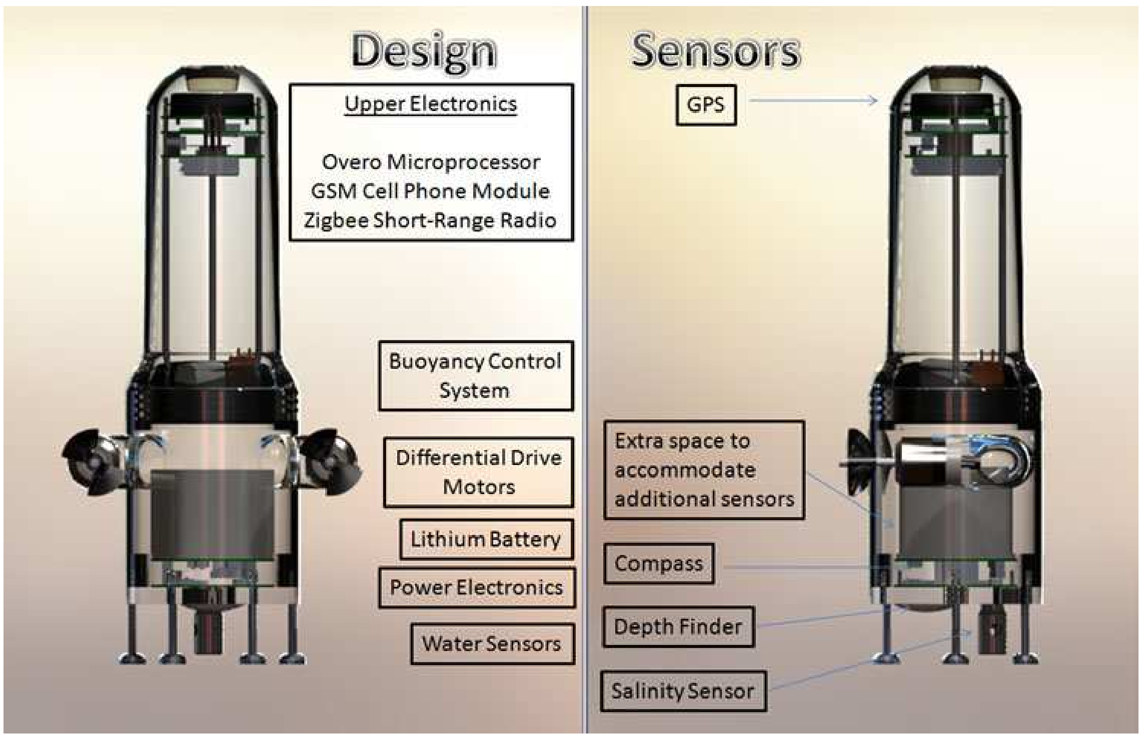


Figure 2.4: Design of the movable water monitoring sensor node and sensors contained inside the sensor node [6]. There is extra space for additional sensors depending on each specific task.

The capabilities of this floating sensor node are as follows

1. Use GPS to track surface water flow
2. Can move to a desired GPS point at approximately 0.5 m/s speed
3. Can send flow and quality data in real time using GSM mobile network
4. Can communicate with each other over Zigbee wireless radio
5. 72 hour operational lifetime battery

For our lake monitoring sensors system, the system mainly consists of two parts, a groups of monitoring sensor nodes and a central control unit. The group of sensor nodes is composed of movable monitoring sensor nodes that can gather monitoring data, such as water flow field and temperature, on the water surface. The central control unit is a computational unit located on the shore of the lake. This unit performs two main tasks, which are data monitoring and computing control input for each sensor node. It receives data such as sensor node position and flow field from the group of sensor nodes and calculate the control input signal to control the group.

2.3 Sensor node

A sensor node in this research is considered to be a floating sensor node having shape of cylinder. This sensor node is composed of a battery, actuators, GPS device, an inertial measurement unit (IMU), a communication unit, and a micro computer. All electric devices are supplied the electric power by the battery. The actuators make the sensor node to be able to move around on the water surface. The GPS device and the IMU are used to measure the position and the velocity of the sensor node, respectively. The communication unit performs data transferring between the sensor node and the central server via wireless communication. The data transferred in this process are monitoring data, such as water flow field, position and velocity of the sensor node, and control data, input signal to drive the sensor node.

Each sensor node is assume to be able to perform movement in 2-dimension, x and y Cartesian coordinate system, on the surface of water source without need of turning. This assumption makes the dynamics motion model of each sensor node simple. Furthermore, the dynamics motion model in x and y direction can be considered separately. The continuous-time dynamics model of each sensor node is represented as

$$\ddot{x} = \frac{1}{m}(u_x + w_x), \tag{2.1}$$

$$\ddot{y} = \frac{1}{m}(u_y + w_y), \tag{2.2}$$

where x and y are the displacement in x and y direction, respectively. The variable m represents mass of a sensor node. The variable u represents the input force to drive the sensor node where u_x and u_y is an input force in x direction and y direction, respectively. These input forces enable the sensor node to perform movement. The disturbance force from environment is defined by w where w_x and w_y represents the disturbance force that affects the dynamics the sensor node in x direction and y direction, respectively.

The state space representation of the dynamics motion model of a sensor node is defined as

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} u_x + w_x \\ u_y + w_y \end{bmatrix}. \tag{2.3}$$

In this research, mass of every monitoring sensor nodes in the group is assumed to be the same that is 50 kg in order to avoid complicated calculation. The disturbance force affecting the dynamics of each sensor node is considered as drag force. In fluid dynamics, drag force is a force that acts on a solid object located in fluid with difference in velocity. The direction of the drag force is in the direction of the relative velocity between the object and the fluid. Quantities that have effect on the magnitude of drag force are square of the relative velocity, shape of the object, and density of the fluid. For simple calculation, the shape of each sensor node is assumed to be upright cylinder. This makes the cross sectional area of the object to be the same in every direction in 2-dimensional planes of motion. The equation to calculate the drag force is as follows

$$Drag = C_d A \rho \left| \boldsymbol{v}_{fluid} - \boldsymbol{v} \right|^2 \frac{(\boldsymbol{v}_{fluid} - \boldsymbol{v})}{\left| \boldsymbol{v}_{fluid} - \boldsymbol{v} \right|}, \tag{2.4}$$

where C_d is as drag coefficient. Reference area A is typically defined as the area of the orthographic projection of the object on a plane perpendicular to the direction of motion. For non-hollow objects with simple shape, such as a sphere, this reference area is the same as a cross sectional area of the object. Fluid density is represented by ρ, which is set to be a constant of $1,000$ kg/m^3. The vectors $\boldsymbol{v}_{fluid} = [v_{x,fluid}, v_{y,fluid}]^\top \in \mathbb{R}^2$ and $\boldsymbol{v} = [v_x, v_y]^\top \in \mathbb{R}^2$ are the velocities of the fluid and the velocities of the sensor node, respectively. The velocities of the fluid in x and y directions and the velocities of the object in x and y directions are $v_{x,fluid}$, $v_{y,fluid}$, v_x, and v_y, respectively.

The drag coefficient is a quantity that is used to quantify the drag of resistance of the sensor node in a lake. This quantity depends on a particular surface area. By assuming the shape of each sensor node to be upright cylinder, which has the same cross sectional area perpendicular to the motion in every direction on 2-dimensional planes, the drag coefficient C_d and the reference area A are constant throughout the simulation.

To define the drag force in x and y directions, Eq. 2.4 can be separated into two components in x and y directions as

$$w_x = C_d A \rho \left| \boldsymbol{v}_{fluid} - \boldsymbol{v} \right| \left(v_{x,fluid} - v_x \right), \tag{2.5}$$

$$w_y = C_d A \rho \left| \boldsymbol{v}_{fluid} - \boldsymbol{v} \right| \left(v_{y,fluid} - v_y \right), \tag{2.6}$$

where w_x is a drag force acting on the sensor in x direction and w_y is a drag force acting on the sensor in y direction.

2.4 Mathematical Models for Simulation

In this section, mathematical models used in the simulation will be introduced. These important mathematical models include dynamics model of the whole monitoring sensors group and water flow field simulation for simulated lake.

2.4.1 Dynamics model

A group of N lake monitoring sensors moving in 2-dimension in simulated lake is considered in this part. The continuous-time state space dynamics model of the group is represented by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{u} + \boldsymbol{w}), \tag{2.7}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{D} & 0 & \cdots & 0 \\ 0 & \boldsymbol{D} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \boldsymbol{D} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \boldsymbol{E} & 0 & \cdots & 0 \\ 0 & \boldsymbol{E} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \boldsymbol{E} \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \boldsymbol{E} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}. \tag{2.8}$$

The matrix $\boldsymbol{A}$ is $2N \times 2N$ matrix and the matrix $\boldsymbol{B}$ is $2N \times N$ matrix, where N is the number of sensor nodes in the group.

Denote that the state vector $\boldsymbol{x} = [x_1, \dot{x}_1, y_1, \dot{y}_1, ..., x_N, \dot{x}_N, y_N, \dot{y}_N]^\top \in \mathbb{R}^{4N}$. The input vector $\boldsymbol{u} = [u_{x1}, u_{y1}, ..., u_{xN}, u_{yN}]^\top \in \mathbb{R}^{2N}$ and the disturbance vector $\boldsymbol{w} = [w_{x1}, w_{y1}, ..., w_{xN}, w_{yN}]^\top \in \mathbb{R}^{2N}$. The state variables are represented by displacement x_i and velocity $\dot{x}_i$ of i-th sensor in x direction. State variables y_i and $\dot{y}_i$ are the displacement and velocity of i-th sensor in y direction. The variable u represents the input force obtained from the central server where u_{xi} and u_{yi} is an input force of i-th sensor in x direction and y direction, respectively. These input forces drive the sensor node, making them to perform movement according to the forces. The variable w represents disturbance force acting on each sensor node from the water flow field in the lake. The disturbance force of i-th sensor node in x direction and in y direction is represented by w_{xi} and w_{yi}, respectively.

2.4.2 Water flow field simulation

In order to simulate a flow of water in a lake, understanding of hydrodynamics is important for the flow field simulation since the distribution of lake monitoring sensors is involved with the hydrodynamics of surface water flow. In this research, we consider distributing a group of lake monitoring sensors in a lake that is assumed to be rectangle in shape. For water flow field simulation, the flow of water needs to satisfy the incompressible Navier-Stokes equations, which are a set of equations that describe the motion of fluid substances. Since the distribution is performed on the surface of water, which is two dimensions space, the incompressible Navier-Stokes equations are as follows

$$\frac{\partial U}{\partial t} + \frac{\partial P}{\partial x} = -\frac{\partial\left(U^2\right)}{\partial x} - \frac{\partial\left(UV\right)}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right), \tag{2.9}$$

$$\frac{\partial V}{\partial t} + \frac{\partial P}{\partial y} = -\frac{\partial\left(UV\right)}{\partial x} - \frac{\partial\left(V^2\right)}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right), \tag{2.10}$$

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0, \tag{2.11}$$

where U and V are the flow velocity in x and y Cartesian coordinate directions, respectively. The pressure is denoted by p. The Reynolds number is represented by Re. This number tells how much the flow is affected by inertia. In this research, we assume the Reynolds number to be a constant in order to provide simple calculation. Eq. 2.9 and Eq. 2.10 are momentum equations that describe the time evolution of the water flow field (U, V) under effects of inertial and viscous (friction) forces. The divergence of the fluid velocity that is equal to zero as seen in Eq. 2.11 implies the incompressibility of the fluid. The nonlinear terms in the momentum equations are

$$-\frac{\partial\left(U^2\right)}{\partial x} - \frac{\partial\left(UV\right)}{\partial y} = -\frac{U\partial U}{\partial x} - \frac{V\partial U}{\partial y}, \tag{2.12}$$

$$-\frac{\partial\left(UV\right)}{\partial x} - \frac{\partial\left(V^2\right)}{\partial y} = -\frac{U\partial V}{\partial x} - \frac{V\partial V}{\partial y}, \tag{2.13}$$

which can be obtained by applying the chain rule together with Eq. 2.11.

The derivation of the Navier-Stokes equations can be found in [39]. A good example of solving the incompressible Navier-Stokes equations on rectangular domain can be found in [40].

A centered stencil is used to approximate the first order derivative in a grid point as follows

$$\frac{\partial U^{i,j}[k]}{\partial x} \approx \frac{U^{i+1,j}[k] - U^{i-1,j}[k]}{2h_x}, \tag{2.14}$$

where $U^{i,j}[k]$ represents the flow velocity at (i,j) discretized space $i \in \{1,2,...,m\}$, $j \in \{1,2,...,n\}$. k denotes the discretized time. The parameter h_x is grid point distance in x direction.

Finite differences is used to approximate the second order derivatives in a grid point. At an interior point $U^{i,j}[k]$, we approximate the Laplace operator by

$$\frac{\partial^2 U^{i,j}[k]}{\partial x^2} + \frac{\partial^2 U^{i,j}[k]}{\partial y^2} \approx \frac{U^{i-1,j}[k] - 2U^{i,j}[k] + U^{i+1,j}[k]}{h_x^2} + \frac{U^{i,j-1}[k] - 2U^{i,j}[k] + U^{i,j+1}[k]}{h_y^2}, \tag{2.15}$$

where the parameter h_y is grid point distance in y direction. For y direction component V and for the Laplacian of the pressure P, the same formula holds.

For the time derivative, the approximation is defined by

$$\frac{\partial U^{i,j}[k]}{\partial t} \approx \frac{U^{i,j}[k] - U^{i,j}[k-1]}{\Delta t}, \tag{2.16}$$

where Δt is time difference between k^{th} time step and $(k+1)^{th}$ time step.

Initial conditions for the water flow field simulation are water flow at the boundary of the lake and initial water flow field at grid points inside the lake. In our research, the flow at boundary of the lake is set to be constant throughout the simulation. The initial water flow field are set to be zero at every point apart from the flow at boundary of the lake.

2.5 Proposed Control Law

To perform multi-agent control, a control law is needed to be designed. In this section, the proposed controller will be introduced. Our proposed controller aims at distributing the group of lake monitoring sensors by taking control of centroid and position variance of the group. Beside using position control that controls each sensor node individually, using centroid and position variance as the control parameters controls the whole group at once [41]. The centroid means the geometric center of a two-dimensional region. It is simply the arithmetic mean or average of position. The centroid can be calculated by using the following equation,

$$\mu_x(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t), \tag{2.17}$$

where $\mu_x(t)$ is the centroid of the sensors group at time t in x direction.

The position variance of the sensors group tells how far the group expands from the centroid. It can be calculated by using the following equation,

$$Var_x(t) = \frac{1}{N}\sum_{i=1}^{N} \left(x_i(t) - \mu_x(t)\right)^2, \tag{2.18}$$

where $Var_x(t)$ is the position variance of the sensors group at time t in x direction.

Notice that the centroid and the position variance of the sensors group can be considered in x direction and in y direction separately.

In order to derive a controller for the proposed control method, damping effect from the second-order differential equation of damped harmonic oscillator is used. The damping effect is an effect that can reduce the amplitude of oscillation in an oscillatory system. In this way, the effect can be used to minimize error, the difference between current value and desired value, of the centroid and the position variance. This means that the current values will gradually approach to the desired values of the centroid and the position variance as time approaches to infinity. The second-order differential equation of damped harmonic oscillator is represented by the following equation

$$\ddot{\alpha} + c\dot{\alpha} + k\alpha = 0, \tag{2.19}$$

where α is a valuable to be reduced by the damping effect. The constants k and c are positive constants that affect the system behaviour. These constants can be adjusted in order to obtain various kinds of damped system such as overdamped, critically damp, underdamped, and undamped.

In this part the derivation of the proposed controller in x direction will be discussed. For the proposed controller in y direction, it can be achieved by the same mean. For the derivation of centroid controller in x direction, the error of the centroid is represented by

$$\bar{\mu}_x(t) = \mu_x(t) - \mu_{x,r}, \tag{2.20}$$

where $\mu_{x,r}$ is the desired centroid in x direction of the system and $\mu_x(t)$ is the x direction position of the centroid at time t. The term $\mu_x(t)$ can be obtained from Eq. 2.17.

The first- and second-order time derivative of the centroid error can be shown as

$$\dot{\bar{\mu}}_x(t) = \frac{1}{N}\sum_{i=1}^{N} \dot{x}_i(t), \tag{2.21}$$

$$\ddot{\bar{\mu}}_x(t) = \frac{1}{N}\sum_{i=1}^{N} \ddot{x}_i(t). \tag{2.22}$$

Next, the centroid error is substituted into the second-order differential equation of damped harmonic oscillator, Eq. 2.19, to make the centroid error obey the damping effect. The result of substituting is as follows

$$\frac{1}{N}\sum_{i=1}^{N}\ddot{x}_i(t)+c_1\dot{\bar{\mu}}_x(t)+k_1\left(\mu_x(t)-\mu_r\right)=0. \tag{2.23}$$

Together with the relationship between the acceleration, the input force, and the disturbance force from water flow field as seen from equation Eq. 2.1 and Eq. 2.2, Eq. 2.23 becomes

$$\frac{1}{N}\sum_{i=1}^{N}(u_{xi}(t)+w_{xi}(t))+c_1\dot{\bar{\mu}}_x(t)+k_1\left(\mu_x(t)-\mu_r\right)=0. \tag{2.24}$$

By rearranging Eq. 2.24, the centroid controller in x direction can be shown as

$$\sum_{i=1}^{N}u_{xi}(t)=-Nmk_1\left(\mu_x(t)-\mu_r\right)-Nmc_1\dot{\bar{\mu}}_x(t)-\sum_{i=1}^{N}w_{xi}(t), \tag{2.25}$$

which is the first controller in the proposed controller.

For the derivation of position variance controller in x direction, the error of the position variance is defined by

$$\bar{Var}_x(t)=Var_x(t)-Var_{x,r}, \tag{2.26}$$

where $Var_{x,r}$ is the desired position variance in x direction of the system and $Var_x(t)$ is the x direction position variance of the sensors group at time t. The term $Var_x(t)$ can be obtained from Eq. 2.18. The first- and second-order time derivative of the position variance error are as follows

$$\dot{\bar{Var}}_x(t)=\frac{2}{N}\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right), \tag{2.27}$$

$$\ddot{\bar{Var}}_x(t)=\frac{2}{N}\left(\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\left(\ddot{x}_i(t)-\ddot{\mu}_x(t)\right)+\sum_{i=1}^{N}\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right)^2\right). \tag{2.28}$$

For the second-order time derivative term, since the term

$$\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\ddot{\mu}_x(t)=0, \tag{2.29}$$

then Eq. 2.28 can be rewritten into

$$\ddot{Var}_x(t) = \frac{2}{N}\left(\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\ddot{x}_i(t) + \sum_{i=1}^{N}\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right)^2\right). \tag{2.30}$$

By substituting the position variance error into the second-order differential equation of damped harmonic oscillator, Eq. 2.19, the result is as follows

$$\begin{aligned}&\frac{2}{N}\left(\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\ddot{x}_i(t) + \sum_{i=1}^{N}\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right)^2\right)\\ &+ c_2\dot{Var}_x(t) + k_2\left(Var_x(t) - Var_{x,r}\right) = 0.\end{aligned} \tag{2.31}$$

Once again, together with the relationship between the acceleration, the input force, and the disturbance force from water flow field as seen from Eq. 2.1 and Eq. 2.2, Eq. 2.31 becomes

$$\begin{aligned}&\frac{2}{N}\left(\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\left(u_{xi}(t)+w_{xi}(t)\right) + \sum_{i=1}^{N}\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right)^2\right)\\ &+ c_2\dot{V}_x(t) + k_2\left(V_x(t) - V_{x,r}\right) = 0.\end{aligned} \tag{2.32}$$

By rearranging equation Eq. 2.32, the position variance controller in x direction can be shown as

$$\begin{aligned}&\sum_{i=1}^{N}\left(x_i(t)-\mu_x(t)\right)\left(u_{xi}(t)+w_{xi}\right)\\ &= -\frac{N}{2}mk_2\left(V_x(t)-V_{x,r}\right) - \frac{N}{2}mc_2\dot{V}_x(t) - m\sum_{i=1}^{N}\left(\dot{x}_i(t)-\dot{\mu}_x(t)\right)^2,\end{aligned} \tag{2.33}$$

which is the second controller in the proposed controller.

Next, the proposed controller will be modified. The modified version of the proposed controller is expected to provide a distribution with low energy consumption. To achieve the modified proposed controller, an additional quantity delta is introduced. This quantity is considered to be an additional force in the forced damped harmonic oscillator shown by

$$\ddot{\alpha} + c\dot{\alpha} + k\alpha = \delta, \tag{2.34}$$

under a performance index or cost function of

$$J = \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u} + \boldsymbol{\delta}^\top \boldsymbol{Q} \boldsymbol{\delta}, \tag{2.35}$$

where R and Q are positive-define matrices.

Denote that the vector $\boldsymbol{u} = [u_{x1}, u_{y1}, ..., u_{xN}, u_{yN}]^\top \in \mathbb{R}^{2N}$ and the vector $\boldsymbol{\delta} = [\delta_{x\mu}, \delta_{xVar}, \delta_{y\mu}, \delta_{yVar}]^\top \in \mathbb{R}^4$.

By adding this additional quantity $\boldsymbol{\delta}$, the proposed controller can be simply written as

$$\begin{bmatrix} 1 & 0 & 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 1 \\ x_1 - \mu_x & 0 & x_2 - \mu_x & 0 & \cdots & x_N - \mu_x & 0 \\ 0 & y_1 - \mu_y & 0 & y_2 - \mu_y & \cdots & 0 & y_N - \mu_y \end{bmatrix} \begin{bmatrix} u_{x1} + w_{x1} \\ u_{y1} + w_{y1} \\ u_{x2} + w_{x2} \\ u_{y2} + w_{y2} \\ \vdots \\ u_{xN} + w_{xN} \\ u_{yN} + w_{yN} \end{bmatrix}$$

$$= \begin{bmatrix} -Nmk_{1,x}\left(\mu_x - \mu_{x,r}\right) - Nmc_{1,x}\dot{\bar{\mu}}_x \\ -\frac{N}{2}mk_{2,x}\left(Var_x - Var_{x,r}\right) - \frac{N}{2}mc_{2,x}\dot{\bar{Var}}_x - m\sum_{i=1}^{N}\left(\dot{x}_i - \dot{\mu}_x\right)^2 \\ -Nmk_{1,y}\left(\mu_y - \mu_{y,r}\right) - Nmc_{1,y}\dot{\bar{\mu}}_y \\ -\frac{N}{2}mk_{2,y}\left(Var_y - Var_{y,r}\right) - \frac{N}{2}mc_{2,y}\dot{\bar{Var}}_y - m\sum_{i=1}^{N}\left(\dot{y}_i - \dot{\mu}_y\right)^2 \end{bmatrix} + \begin{bmatrix} \delta_{x\mu} \\ \delta_{xVar} \\ \delta_{y\mu} \\ \delta_{yVar} \end{bmatrix}. \tag{2.36}$$

For more simple form, the Eq. 2.36 can be written as

$$\boldsymbol{F}(\boldsymbol{u} + \boldsymbol{w}) = \boldsymbol{g} + \boldsymbol{\delta}. \tag{2.37}$$

The additional quantity $\boldsymbol{\delta}$ can be determined by combining the Eq. 2.37 that represents the modified proposed controller with the cost function in Eq. 2.35. By substituting

$$\boldsymbol{u} = \boldsymbol{F}^{\dagger}\boldsymbol{g} - \boldsymbol{F}^{\dagger}\boldsymbol{F}\boldsymbol{w} + \boldsymbol{F}^{\dagger}\boldsymbol{\delta}, \tag{2.38}$$

where $\boldsymbol{F}^{\dagger}$ represents the pseudo-inverse of $\boldsymbol{F}$, into the cost function in Eq. 2.35, the cost function becomes

$$J = (\boldsymbol{g}^{\top}\boldsymbol{F}^{\dagger\top} - \boldsymbol{w}^{\top}\boldsymbol{F}^{\top}\boldsymbol{F}^{\dagger\top} + \boldsymbol{\delta}\top\boldsymbol{F}^{\dagger\top})\boldsymbol{R}(\boldsymbol{F}^{\dagger}\boldsymbol{g} - \boldsymbol{F}^{\dagger}\boldsymbol{F}\boldsymbol{w} + \boldsymbol{F}^{\dagger}\boldsymbol{\delta}) + \boldsymbol{\delta}^{\top}\boldsymbol{Q}\boldsymbol{\delta}. \tag{2.39}$$

Rearranging Eq. 2.39, the result is

$$\begin{aligned} J =& \boldsymbol{\delta}^{\top}(\boldsymbol{F}^{\dagger\top}\boldsymbol{R}\boldsymbol{F}^{\dagger} + \boldsymbol{Q})\boldsymbol{\delta} + \boldsymbol{\delta}\top\boldsymbol{F}^{\dagger\top}\boldsymbol{R}(\boldsymbol{F}^{\dagger}\boldsymbol{g} - \boldsymbol{F}^{\dagger}\boldsymbol{F}\boldsymbol{w}) \\ &+ (\boldsymbol{g}^{\top}\boldsymbol{F}^{\dagger\top} - \boldsymbol{w}^{\top}\boldsymbol{F}^{\top}\boldsymbol{F}^{\dagger\top})\boldsymbol{R}\boldsymbol{F}^{\dagger}\boldsymbol{\delta} \\ &+ (\boldsymbol{g}^{\top}\boldsymbol{F}^{\dagger\top} - \boldsymbol{w}^{\top}\boldsymbol{F}^{\top}\boldsymbol{F}^{\dagger\top})\boldsymbol{R}\boldsymbol{F}^{\dagger}(\boldsymbol{g} - \boldsymbol{F}\boldsymbol{w}), \end{aligned} \tag{2.40}$$

or it can be simply written as

$$J = \boldsymbol{\delta}^{\top}\boldsymbol{L}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\boldsymbol{M}^{\top} + \boldsymbol{M}\boldsymbol{\delta} + \boldsymbol{M}(\boldsymbol{g} - \boldsymbol{F}\boldsymbol{w}), \tag{2.41}$$

where

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{F}^{\dagger\top}\boldsymbol{R}\boldsymbol{F}^{\dagger} + \boldsymbol{Q}, \\ \boldsymbol{M} &= (\boldsymbol{g}^{\top}\boldsymbol{F}^{\dagger\top} - \boldsymbol{w}^{\top}\boldsymbol{F}^{\top}\boldsymbol{F}^{\dagger\top})\boldsymbol{R}\boldsymbol{F}^{\dagger}. \end{aligned} \tag{2.42}$$

By adding a term

$$\boldsymbol{M}\boldsymbol{L}^{-1}\boldsymbol{M}^{\top} - \boldsymbol{M}\boldsymbol{L}^{-1}\boldsymbol{M}^{\top} = 0, \tag{2.43}$$

to the Eq. 2.41, the cost function becomes

$$J = \boldsymbol{\delta}^{\top}\boldsymbol{L}\boldsymbol{\delta} + \boldsymbol{\delta}^{\top}\boldsymbol{M}^{\top} + \boldsymbol{M}\boldsymbol{\delta} + \boldsymbol{M}\boldsymbol{L}^{-1}\boldsymbol{M}^{\top} - \boldsymbol{M}\boldsymbol{L}^{-1}\boldsymbol{M}^{\top} + \boldsymbol{M}(\boldsymbol{g} - \boldsymbol{F}\boldsymbol{w}). \tag{2.44}$$

Combine the terms involving with the additional quantity $\boldsymbol{\delta}$ together, the cost function becomes as follows

$$J = (\boldsymbol{\delta}^{\top} + \boldsymbol{M}\boldsymbol{L}^{-1})\boldsymbol{L}(\boldsymbol{\delta} + \boldsymbol{L}^{-1}\boldsymbol{M}^{\top}) - \boldsymbol{M}\boldsymbol{L}^{-1}\boldsymbol{M}^{\top} + \boldsymbol{M}(\boldsymbol{g} - \boldsymbol{F}\boldsymbol{w}). \tag{2.45}$$

Notice that the solution of the additional quantity $\boldsymbol{\delta}$ that minimizes the term $(\boldsymbol{\delta}^\top + \boldsymbol{M}\boldsymbol{L}^{-1})\boldsymbol{L}(\boldsymbol{\delta} + \boldsymbol{L}^{-1}\boldsymbol{M}^\top)$ can be obtained from

$$\boldsymbol{\delta} + \boldsymbol{L}^{-1}\boldsymbol{M}^\top = 0. \tag{2.46}$$

Rearranging the equation Eq. 2.46, the equation to determine the value of the additional quantity $\boldsymbol{\delta}$ is as follows

$$\boldsymbol{\delta} = -(\boldsymbol{F}^{\dagger\top}\boldsymbol{R}\boldsymbol{F}^\dagger + \boldsymbol{Q})^{-1}((\boldsymbol{g}^\top\boldsymbol{F}^{\dagger\top} - \boldsymbol{w}^\top\boldsymbol{F}^\top\boldsymbol{F}^{\dagger\top})\boldsymbol{R}\boldsymbol{F}^\dagger)^\top. \tag{2.47}$$

2.6 Water Flow Field Simulation Example

According to the incompressible Navier-Stokes equations, the water flow field can be simulated by the following time update equations

$$\boldsymbol{\xi}^{\mathbb{I}_{\text{inside}}}[k+1] = f_\xi(\boldsymbol{\xi}^{\mathbb{I}_{\text{inside}}}[k], \boldsymbol{\xi}^{\mathbb{I}_{\text{boundary}}}[k]), \tag{2.48}$$

$$\boldsymbol{\xi}^{\mathbb{I}_{\text{boundary}}}[k+1] = \boldsymbol{\xi}^{\mathbb{I}_{\text{boundary}}}[k], \tag{2.49}$$

where for the elements of the index set $\mathbb{I}$ are all possible combination of position (i,j) in the discretized space, the boundary set $\mathbb{I}_{\text{boundary}}$ represents all points (i,j) that are located on the boundary of the lake. The inside set $\mathbb{I}_{\text{inside}}$ represents all points (i,j) that are located in the lake beside the boundary. The relationships between these sets are defined by $\mathbb{I}_{\text{inside}} \bigcup \mathbb{I}_{\text{boundary}} = \mathbb{I}$ and $\mathbb{I}_{\text{inside}} \bigcap \mathbb{I}_{\text{boundary}} = \emptyset$. The symbol $\boldsymbol{\xi}^{\mathbb{I}}[k]$ is the vector whose elements are represented by flow velocity in x direction $\xi^{i,j}[k]$ and flow velocity in y direction $\eta^{i,j}[k]$, where $(i,j) \in \mathbb{I}$. Note that for given flow field information at time k, $\boldsymbol{\xi}^{\mathbb{I}_{\text{inside}}}[k]$ and $\boldsymbol{\xi}^{\mathbb{I}_{\text{boundary}}}[k]$, we can obtain the flow field information at the next time step, $\boldsymbol{\xi}^{\mathbb{I}_{\text{inside}}}[k+1]$ and $\boldsymbol{\xi}^{\mathbb{I}_{\text{boundary}}}[k+1]$, by using the time update function f_ξ.

At time step k of solving Navier-Stokes equations, known parameters are the flow velocity in x direction $\xi^{i,j}[k]$ and flow velocity in y direction $\eta^{i,j}[k]$, where $(i,j) \in \mathbb{I}$. Unknown parameters are $p_x^{(i,j)_{\text{inside}}}[k+1]$, $p_y^{(i,j)_{\text{inside}}}[k+1]$, $\xi_x^{(i,j)_{\text{inside}}}[k]$, $\xi_y^{(i,j)_{\text{inside}}}[k]$, $\xi_{xx}^{(i,j)_{\text{inside}}}[k]$, $\xi_{yy}^{(i,j)_{\text{inside}}}[k]$, $\eta_x^{(i,j)_{\text{inside}}}[k]$, $\eta_y^{(i,j)_{\text{inside}}}[k]$, $\eta_{xx}^{(i,j)_{\text{inside}}}[k]$, and $\eta_{yy}^{(i,j)_{\text{inside}}}[k]$, where $(i,j)_{\text{inside}} \in \mathbb{I}_{\text{inside}}$. The time update function f_ξ can be defined by a combination of the following equations

$$\begin{aligned}\xi^{(i,j)_{\text{inside}}}[k+1] = \xi^{(i,j)_{\text{inside}}}[k] + \Delta t(-p_x^{(i,j)_{\text{inside}}}[k+1] - \xi^{(i,j)_{\text{inside}}}[k]\xi_x^{(i,j)_{\text{inside}}}[k] \\ - \eta^{(i,j)_{\text{inside}}}[k]\xi_y^{(i,j)_{\text{inside}}}[k] + \frac{1}{Re}(\xi_{xx}^{(i,j)_{\text{inside}}}[k] + \xi_{yy}^{(i,j)_{\text{inside}}}[k])),\end{aligned} \tag{2.50}$$

$$\begin{aligned}\eta^{(i,j)_{\text{inside}}}[k+1] = \eta^{(i,j)_{\text{inside}}}[k] + \Delta t(-p_y^{(i,j)_{\text{inside}}}[k+1] - \xi^{(i,j)_{\text{inside}}}[k]\eta_x^{(i,j)_{\text{inside}}}[k] \\ - \eta^{(i,j)_{\text{inside}}}[k]\eta_y^{(i,j)_{\text{inside}}}[k] + \frac{1}{Re}(\eta_{xx}^{(i,j)_{\text{inside}}}[k] + \eta_{yy}^{(i,j)_{\text{inside}}}[k])).\end{aligned} \tag{2.51}$$

Notice that the unknowns $\xi_x^{(i,j)_{\text{inside}}}[k]$, $\xi_y^{(i,j)_{\text{inside}}}[k]$, $\xi_{xx}^{(i,j)_{\text{inside}}}[k]$, $\xi_{yy}^{(i,j)_{\text{inside}}}[k]$, $\eta_x^{(i,j)_{\text{inside}}}[k]$, $\eta_y^{(i,j)_{\text{inside}}}[k]$, $\eta_{xx}^{(i,j)_{\text{inside}}}[k]$, and $\eta_{yy}^{(i,j)_{\text{inside}}}[k]$ can be obtained by using the first-order space derivatives and the second-order space derivatives. For the unknown terms $p_x^{(i,j)_{\text{inside}}}[k+1]$ and $p_y^{(i,j)_{\text{inside}}}[k+1]$, which are the first-order space derivatives of pressure in x and y directions, we need to calculate the pressure field p first. The derivation of the pressure field will be explained later.

The first-order space derivatives of $\xi^{i,j}[k]$, $\eta^{i,j}[k]$, and $p^{i,j}[k]$ are as follows

$$\xi_x^{i,j}[k] = \frac{\xi^{i+1,j}[k] - \xi^{i-1,j}[k]}{2h_x}, \tag{2.52}$$

$$\xi_y^{i,j}[k] = \frac{\xi^{i,j+1}[k] - \xi^{i,j-1}[k]}{2h_y}, \tag{2.53}$$

$$\eta_x^{i,j}[k] = \frac{\eta^{i+1,j}[k] - \eta^{i-1,j}[k]}{2h_x}, \tag{2.54}$$

$$\eta_y^{i,j}[k] = \frac{\eta^{i,j+1}[k] - \eta^{i,j-1}[k]}{2h_y}, \tag{2.55}$$

$$p_x^{i,j}[k] = \frac{p^{i+1,j}[k] - p^{i-1,j}[k]}{2h_x}, \tag{2.56}$$

$$p_y^{i,j}[k] = \frac{p^{i,j+1}[k] - p^{i,j-1}[k]}{2h_y}, \tag{2.57}$$

where the subscript x of $\xi_x^{i,j}[k]$, $\eta_x^{i,j}[k]$, and $p_x^{i,j}[k]$ represents the first-order space derivative with respect to x direction and the subscript y of $\xi_y^{i,j}[k]$, $\eta_y^{i,j}[k]$, and $p_y^{i,j}[k]$ represents the first-order space derivative with respect to y direction. The parameter h_x and h_y are grid point distance in x and y direction, respectively.

The second-order space derivatives of $\xi^{i,j}[k]$, $\eta^{i,j}[k]$, and $p^{i,j}[k]$ can be obtained by

$$\xi_{xx}^{i,j}[k] = \frac{\xi^{i-1,j}[k] - 2\xi^{i,j}[k] + \xi^{i+1,j}[k]}{h_x^2}, \tag{2.58}$$

$$\xi_{yy}^{i,j}[k] = \frac{\xi^{i,j-1}[k] - 2\xi^{i,j}[k] + \xi^{i,j+1}[k]}{h_y^2}, \tag{2.59}$$

$$\eta_{xx}^{i,j}[k] = \frac{\eta^{i-1,j}[k] - 2\eta^{i,j}[k] + \eta^{i+1,j}[k]}{h_x^2}, \tag{2.60}$$

$$\eta_{yy}^{i,j}[k] = \frac{\eta^{i,j-1}[k] - 2\eta^{i,j}[k] + \eta^{i,j+1}[k]}{h_y^2}, \tag{2.61}$$

$$p_{xx}^{i,j}[k] = \frac{p^{i-1,j}[k] - 2p^{i,j}[k] + p^{i+1,j}[k]}{h_x^2}, \tag{2.62}$$

$$p_{yy}^{i,j}[k] = \frac{p^{i,j-1}[k] - 2p^{i,j}[k] + p^{i,j+1}[k]}{h_y^2}, \tag{2.63}$$

where the subscript xx of $\xi_{xx}^{i,j}[k]$, $\eta_{xx}^{i,j}[k]$, and $p_{xx}^{i,j}[k]$ represents the second-order space derivative with respect to x direction and the subscript yy of $\xi_{yy}^{i,j}[k]$, $\eta_{yy}^{i,j}[k]$, and $p_{yy}^{i,j}[k]$ represents the second-order space derivative with respect to y direction.

For the derivation of the pressure $p^{(i,j)_{\text{inside}}}[k+1]$, it can be obtained by solving the following Poisson's equation

$$p_{xx}^{(i,j)_{\text{inside}}}[k+1] + p_{yy}^{(i,j)_{\text{inside}}}[k+1] = \frac{1}{\Delta t}(\xi_x^{(i,j)_{\text{inside}}}[k] + \eta_y^{(i,j)_{\text{inside}}}[k]), \tag{2.64}$$

where the terms $\xi_x^{(i,j)_{\text{inside}}}[k]$ and $\eta_y^{(i,j)_{\text{inside}}}[k]$ are considered to be given in this step and can be obtained by using Eq. 2.52 and Eq. 2.55.

For the pressure at boundary $p^{(i,j)_{\text{boundary}}}$, we prescribe the Dirichlet boundary conditions,

$$p^{(i,j)_{\text{north}}}[k] = p^{i,j-1}[k], \tag{2.65}$$

$$p^{(i,j)_{\text{south}}}[k] = p^{i,j+1}[k], \tag{2.66}$$

$$p^{(i,j)_{\text{east}}}[k] = p^{i-1,j}[k], \tag{2.67}$$

$$p^{(i,j)_{\text{west}}}[k] = p^{i+1,j}[k], \tag{2.68}$$

where $(i,j)_{\text{north}}$, $(i,j)_{\text{south}}$, $(i,j)_{\text{east}}$, and $(i,j)_{\text{west}}$ represent a point at north boundary, south boundary, east boundary, and west boundary, respectively. This equation shows that the unknown pressure at the boundary point has the same value as the pressure located at the grid point inside the boundary next to it.

For more detail, we will use a simple example to show how to solve for the pressure $p^{(i,j)_{\text{inside}}}[k+1]$. Assuming that we have a discretized lake with resolution of 4×4 as shown in Fig. 2.5. The pressure at position (i,j) is located in the middle of a cell that is labeled (i,j) at its bottom right. For example, $p^{1,1}$ is located in the middle of the cell $(1,1)$. The red cells represent the boundary region while the blue cells represent the inside region.

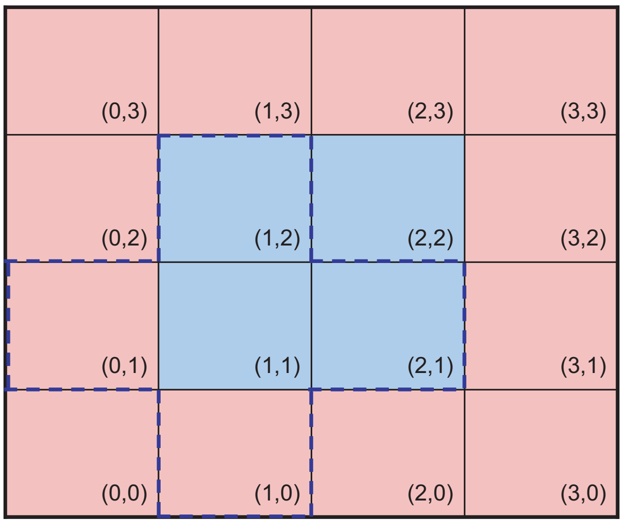


Figure 2.5: Example of a rectangle lake having resolution of 4×4. The red cells represent the boundary region of the lake and the blue cells represent the inside region of the lake. The region covered by the blue dotted line represents neighbour cells of the cell $(1,1)$ that are necessary for the use of Eq. 2.64 for $(i,j) = (1,1)$.

If we consider the pressure at $(1,1)$, the region covered by the blue dotted line represents the values that are necessary for the use of Eq. 2.64. For $(i,j) = (1,1)$, Eq. 2.64 can be written as

$$\frac{p^{0,1}[k+1] - 2p^{1,1}[k+1] + p^{2,1}[k+1]}{h_x^2} + \frac{p^{1,0}[k+1] - 2p^{1,1}[k+1] + p^{1,2}[k+1]}{h_y^2}$$
$$= \frac{1}{\Delta t}(\xi_x^{1,1}[k] + \eta_y^{1,1}[k]). \tag{2.69}$$

According to the Dirichlet boundary conditions, we have $p^{0,1}[k+1] = p^{1,1}[k+1]$ and $p^{1,0}[k+1] = p^{1,1}[k+1]$. Normally for a square lake, the grid point distances in x and y direction are the same that is $h_x = h_y$. Then we can rewrite Eq. 2.69 as

$$\frac{-2p^{1,1}[k+1] + p^{2,1}[k+1] + p^{1,2}[k+1]}{h_x^2} = \frac{1}{\Delta t}(\xi_x^{1,1}[k] + \eta_y^{1,1}[k]). \tag{2.70}$$

In the same way for the consideration of the pressure at the other three points, $p^{2,1}$, $p^{1,2}$, and $p^{2,2}$, we have

$$\begin{aligned}
\frac{-2p^{2,1}[k+1]+p^{1,1}[k+1]+p^{2,2}[k+1]}{h_x^2} &= \frac{1}{\Delta t}(\xi_x^{2,1}[k]+\eta_y^{2,1}[k]), && (2.71)\\
\frac{-2p^{1,2}[k+1]+p^{1,1}[k+1]+p^{2,2}[k+1]}{h_x^2} &= \frac{1}{\Delta t}(\xi_x^{1,2}[k]+\eta_y^{1,2}[k]), && (2.72)\\
\frac{-2p^{2,2}[k+1]+p^{2,1}[k+1]+p^{1,2}[k+1]}{h_x^2} &= \frac{1}{\Delta t}(\xi_x^{2,2}[k]+\eta_y^{2,2}[k]). && (2.73)
\end{aligned}$$

According to the lake in Fig. 2.5 and assuming that $h_x = h_y = 1$, to complete the pressure field at the inner region, we will get four equations written in matrix form with four unknowns, $p^{1,1}$, $p^{2,1}$, $p^{1,2}$, and $p^{2,2}$, as follows

$$\begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -2 & 0 & 1 \\ 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} p^{1,1} \\ p^{2,1} \\ p^{1,2} \\ p^{2,2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}(\xi_x^{1,1}[k]+\eta_y^{1,1}[k]) \\ \frac{1}{\Delta t}(\xi_x^{2,1}[k]+\eta_y^{2,1}[k]) \\ \frac{1}{\Delta t}(\xi_x^{1,2}[k]+\eta_y^{1,2}[k]) \\ \frac{1}{\Delta t}(\xi_x^{2,2}[k]+\eta_y^{2,2}[k]) \end{bmatrix}, \qquad (2.74)$$

which is in the form of $\boldsymbol{Ax} = \boldsymbol{b}$. Note that this leaves us with a small detail that the matrix $\boldsymbol{A}$ is not full rank. Then we modify the matrix $\boldsymbol{A}$ so that it becomes invertible. This is achieved by multiplying the first element of $\boldsymbol{A}$ with 1.5. Then we can solve these equations for the pressure field at the inner region of the lake.

2.7 Simulation Experiment

Firstly, the simulation of sensors distribution under uniform constant flow field will be carried out. The controllers used in this part are the proposed controller and the modified proposed controller. The purpose of this simulation is to shows the energy consumption reduction ability of the modified proposed controller over the original proposed controller. In this simulation, lake is assumed to be a rectangular lake with the width of 30 m and length of 30 m. The reference origin in the Cartesian coordinate system is set at the center of the lake. Water flow field is set to be constant with magnitude of 0.05 m/s flowing from the east to the west of the lake. Notice that in y direction, there is no flow from water.

A group of 4 monitoring sensor nodes are initially placed at the following positions: $(12.0, 0.25)$, $(12.5, -0.25)$, $(12.5, 0.25)$, and $(12.0, 0.25)$ with initial velocity of zero in both x and y directions. The mass of a sensor node is 50 kg for all sensor nodes in the group. The reference area and the drag coefficient for drag force calculation are assumed to be constants of 1 m^2 and 0.5 for all simulations. The desired centroid of this simulation is $(0, 0)$ and the desired position variance is 9 in both x and y directions. For both controllers, the constants k and c are all set to be 1 and 20, respectively. In modified proposed controller, the positive-define matrices are set as $\boldsymbol{R} = \text{diag}(5, 5, 5, 5, 5, 5, 5, 5)$ and $\boldsymbol{Q} = \text{diag}(0.5, 0.5, 0.5, 0.5)$. We define distribution time as the time measured from the beginning of the distribution until the centroid of the sensors group reaches the desired centroid value within error of 0.05. Fig. 2.6 shows the trajectory of each sensor of the proposed control method with the distribution time of 110 sec. The trajectory of each sensor of the modified proposed control method is shown in Fig. 2.7 with the distribution time of 200 sec. The initial positions are plotted as circle marks and the final positions of the simulation are plotted as cross marks. The blue vector field represented the water flow field.

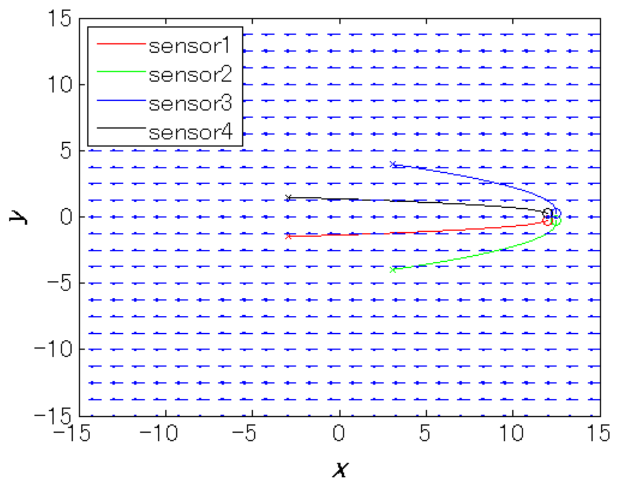


Figure 2.6: Trajectory of each sensor in the distribution using the original proposed controller from $t = 0$ to $t = 110$. The water flow field is set to be uniform and constant with magnitude of 0.05 m/s in direction of negative x and represented by blue vector field. The group of sensor can reach the desired centroid and desired position variance within 110 [sec] period of time.

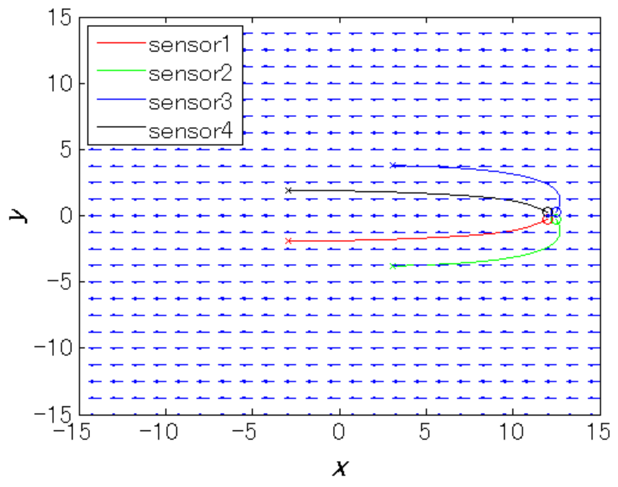


Figure 2.7: Trajectory of each sensor in the distribution using the modified proposed controller from $t = 0$ to $t = 200$. The group of sensor uses 200 [sec] period of time to reach the desired centroid and desired position variance.

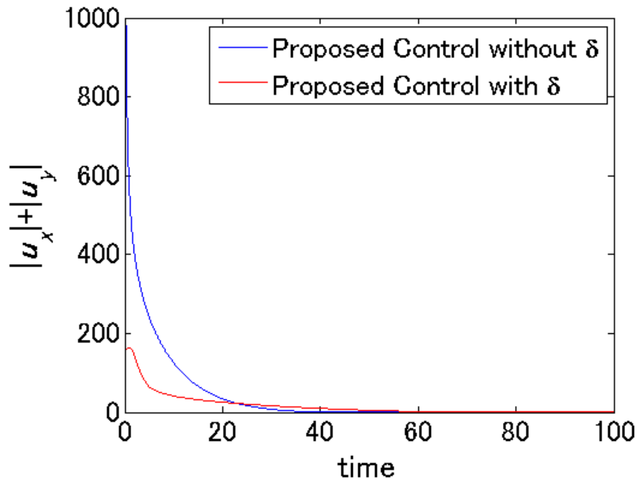


Figure 2.8: Time series graph of summation of absolute values of input forces from x and y directions.

The energy usage for each methods is shown in Fig. 2.8, which is time-series graph of absolute summation of input forces. As seen from the graph, the modified proposed control method can reduce the consumption of input forces better than the original proposed control method. Numerical comparison can be done by calculating the area under the graph from $t = 0$ to $t = 100$. For the original proposed control method, this area is 4.137×10^3, where the area for the modified proposed control method is 1.671^3.

Next, various flow field magnitudes are applied for the simulation. All settings are the same as the previous simulation except the magnitude of the water flow field will be varied. In this part, the modified proposed controller is used. Tab. 2.1 shows the area under the time-series graph of absolute summation of input forces and distribution time in various flow field magnitudes.

Table 2.1: Input forces consumption and time used in various flow field magnitudes.

Flow field magnitude in x direction [m/s]	Input forces consumption	Distribution time [sec]
0.00	2.3394×10^3	167
0.01	2.1830×10^3	160
0.02	2.0495×10^3	165
0.03	1.9455×10^3	174
0.04	1.8805×10^3	186
0.05	1.8652×10^3	200
0.06	1.9132×10^3	216
0.07	2.0356×10^3	233
0.08	2.2466×10^3	251

According to the table, the input forces consumption continues to decrease as the flow magnitude increases from 0.00 to 0.05. This shows that the modified proposed controller can use the flow field to help the distribution. However, when the flow magnitude is larger than 0.05, more inputs forces are needed. The reason is that the larger of flow magnitude produces larger forces acting on the sensors group. Therefore, more input forces are needed to control the group distribution.

2.7.1 Sensors distribution under Navier-Stokes simulated flow field

In this section, 3 cases of different controllers, which are the position-velocity controller, the proposed controller, and the modified proposed controller will be used. Position-velocity controller is a simple control of each sensor node. This controller controls each sensor node individually by driving each sensor node toward desired position and making the sensor node maintain its position at the desired position. The position-velocity controller is defined by the input for each sensor node for x direction as

$$u_{xi}(t) = -\alpha_{xi}(x_i(t) - x_{ri}) - \beta_{xi}\dot{x}_i(t) - w_{xi}(t), \tag{2.75}$$

where x_{ri} is the desired position in x direction of the i-th sensor. The parameters α_{xi} and β_{xi} are x direction constants of the i-th sensor that can be calculated by using the linear quadratic(LQ) optimal control. Note that in this experiment, all weighting matrices in the quadratic continuous-time cost function in LQ optimal control are set to be identity matrix. The control input of this controller in y direction can be determined by the same way as in x direction. Unlike the proposed controller, this controller requires a desired position for each sensor. This means if there are N sensors in the monitoring system, we have to provide N desired positions for the sensors group. The distribution will be done under the water flow field in a simulated lake that follows the incompressible Navier-Stokes equations.

Simulated flow field in rectangular lake

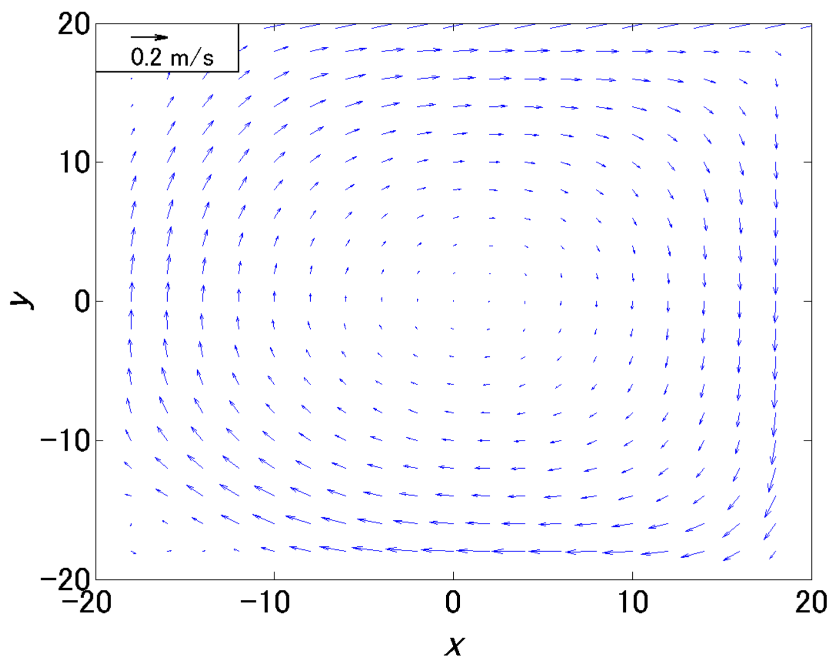


Figure 2.9: The steady flow field of water in a rectangular lake. After several times of solving the incompressible Navier-Stokes equations, the flow field velocity after the time of 2×10^4 [sec] from the beginning of flow field simulation becomes the steady flow field as shown in this figure. The flow field is represented by blue vector field, where the length of the arrow represents the magnitude of flow velocity and the direction of the arrow is in the same direction as the flow velocity.

First of all, the Reynold number is assumed to be constants throughout the simulation with value of 1×10^3, describing that the flow is transient or unsteady. The lake is assumed to be rectangular with the width of 40 m and the length of 40 m. The origin point in the Cartesian coordinate system is set to be at the center of the lake. The water flow field in the lake has resolution of 80×80 in grid pattern. The boundary condition for the flow field simulation is that there is wind blowing at the northern boundary of the lake with magnitude of 1 m/s in the direction of positive x. In this way, the flow velocity of water at the northern boundary is 1 m/s in positive x direction as well. The other boundaries conditions are assumed to be zero. The flow field simulation is done by continuously solving the incompressible Navier-Stokes equation until stable flow field is achieved. The steady simulated flow field is shown in Fig. 2.9. The resolution of grid points shown in this figure is reduced by half for easy observation. This water flow field will be used for the lake monitoring sensors distribution and the flow field estimation later on.

Sensors distribution simulation

In this simulation of the lake monitoring sensors distribution, there are 8 sensor nodes in the group. All sensor nodes are placed at the same position that is $(14, 14)$ with initial velocity of zero. The reasons why all the sensor nodes are place at the same position are we would like to reduce the cost of initial sensor arrangement done by human. The releasing time for each sensor node is different. To put it more simple, the releasing time of sensor $1, 2, ..., 8$ is set to be $t = 0, 40, ..., 280$ [sec]. There will be no controller applied to the system from the initial time to the time the last sensor node is released. The controller will be applied to the system at the same time when the last sensor node is released, which is when $t = 280$ [sec]. The desired centroid of the distribution is set to be at the origin point. The desired position variance is set to be 30 in both x and y directions. To achieve the same desired centroid and desired position variance, the desired positions of sensor node 1 to 8 of the position-velocity controller are set to be $(2, 4)$, $(4, -8)$, $(6, -2)$, $(-4, 2)$, $(-2, 8)$, $(8, 4)$, $(-6, -6)$, and $(-8, -6)$, respectively. These desired positions give the centroid of (0,0) and the position variance of (30,30) that are the same as the desired values. Fig. 2.10 shows the trajectory of each sensor node for the position-velocity controller with simulation time of $2,000$ [sec]. The initial position of each sensor node is plotted as a dot mark. The final position of each sensor node when the simulation time reaches $2,000$ [sec] is plotted as a cross mark. Note that, the blue vector field in trajectory graphs will show just only the direction of the flow for easy observation.

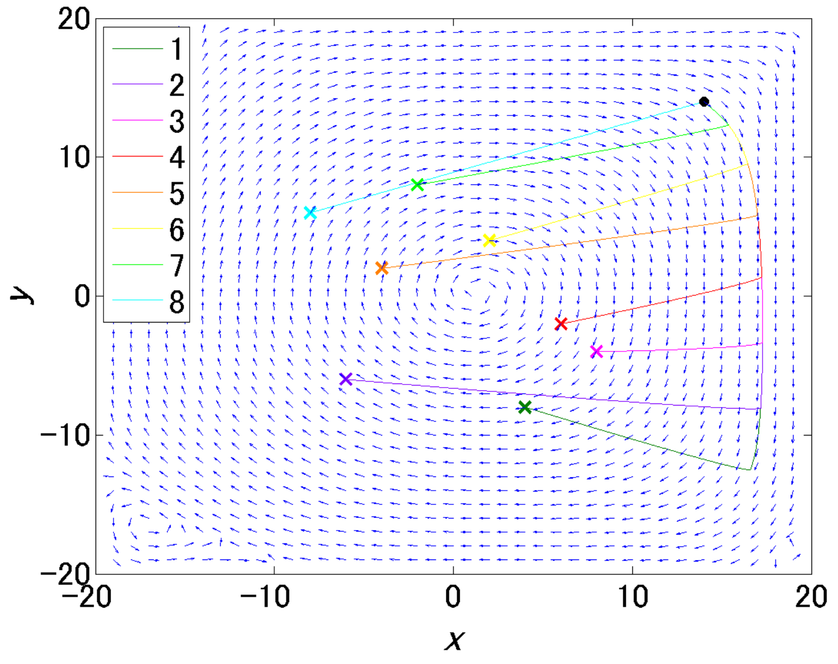


Figure 2.10: Trajectory of each sensor in the distribution using the position-velocity controller from $t = 0$ to $t = 2,000$. There are 8 sensor nodes moving toward the desired positions directly. The initial positions of all sensor nodes are set to be the same position which is plotted by black dot mark in this figure.

Time series graphs of the centroid and position variance from $t = 0$ to $t = 800$ of the position-velocity controller can be seen from Fig. 2.11 and Fig. 2.12.

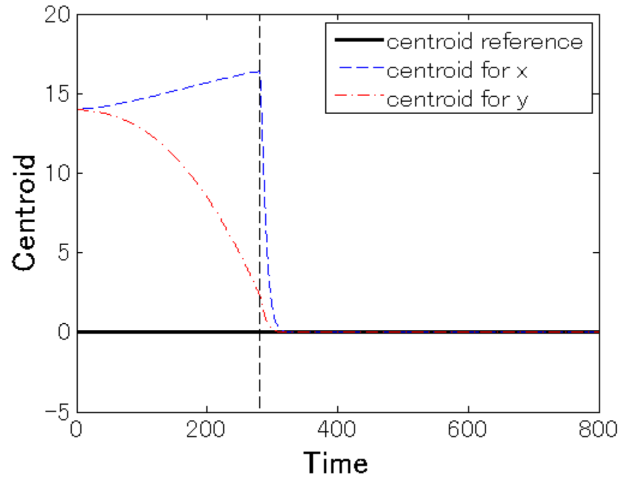


Figure 2.11: Time series graph of the centroid of the sensors group controlled by position-velocity controller from $t = 0$ to $t = 800$. The vertical dashed line represents the time when all sensor nodes are released and the time when the controller is applied to control the group of sensors. The desired centroid is set to be $(0, 0)$.

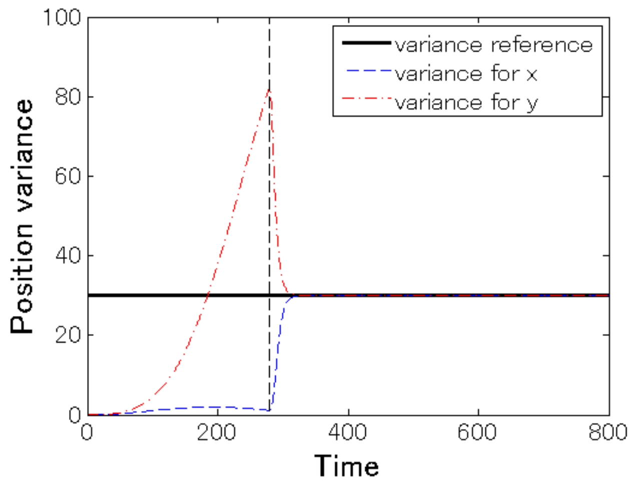


Figure 2.12: Time series graph of the position variance of the sensors group controlled by position-velocity controller from $t = 0$ to $t = 800$. The vertical dashed line represents the time when all sensor nodes are released and the time when the controller is applied. The desired position variance is set to be 30 in both x and y directions.

For the proposed controller case and the modified proposed controller case, the constants are set as $k_{1,x} = 1$, $c_{1,x} = 20$, $k_{2,x} = 0.2$, $c_{2,x} = 20$, $k_{1,y} = 1$, $c_{1,y} = 20$, $k_{2,y} = 1$, and $c_{2,y} = 10$. Fig. 2.13 shows the trajectory of each sensor node for the proposed controller with simulation time of $2,000$ [sec]. We start applying the proposed controller to the sensors group at the same time with the last sensor node is released, which is when $t = 280$ [sec].

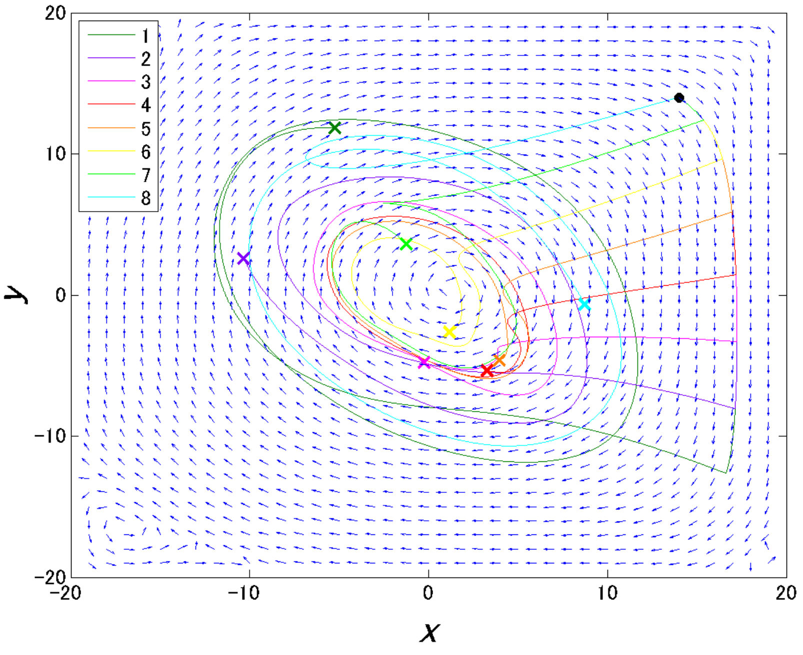


Figure 2.13: Trajectory of each sensor in the distribution using the proposed controller from $t = 0$ to $t = 2,000$. 8 sensor nodes moves with influence of the water flow field. The initial positions of all sensor nodes are set to be the same position which is plotted by black dot mark in this figure. The controller is applied to the sensors group after $t = 280$

Time series graph of centroid and position variance of the sensors group from $t = 0$ to $t = 800$ of the proposed controller can be seen from Fig. 2.14 and Fig. 2.15.

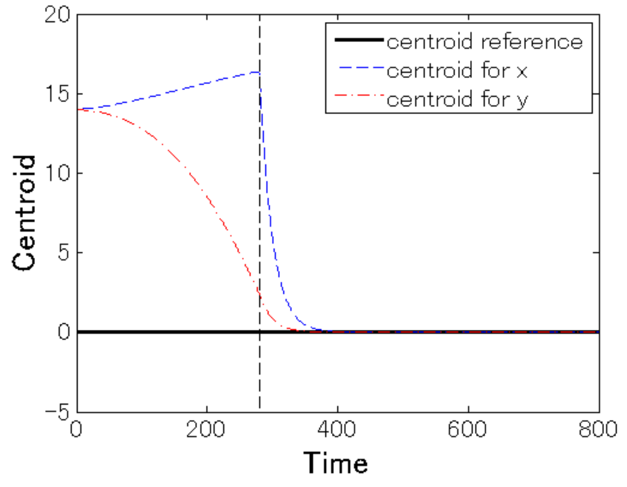


Figure 2.14: Time series graph of the centroid position controlled by the proposed controller for x and y directions. The vertical dashed line represents the time when all sensor nodes are released and the time when the controller is applied to control the group of sensors. The desired centroid is set to be $(0, 0)$.

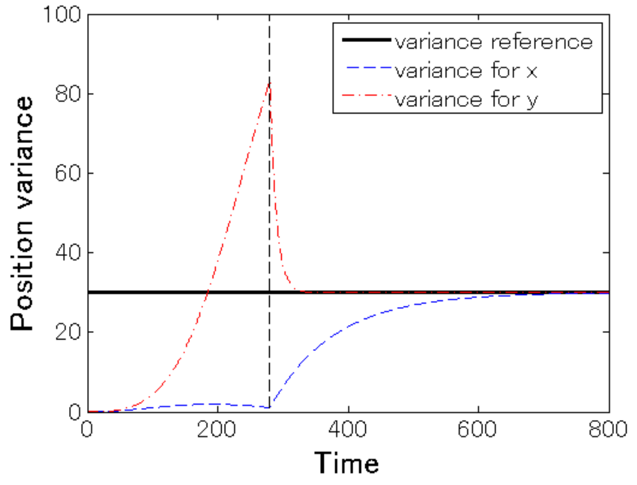


Figure 2.15: Time series graph of the position variance controlled by the proposed controller for x and y directions. The vertical dashed line is the time when all sensor nodes are released and the time when the controller is applied. The desired position variance is set to be 30 for both position variance in x and y directions.

For the modified proposed controller, the positive-define matrices are set as $\boldsymbol{R} = \mathrm{diag}(5, 5, ..., 5)$ and $\boldsymbol{Q} = \mathrm{diag}(0.5, 0.5, 0.5, 0.5)$. Fig. 2.16 shows the trajectory of each sensor node for the modified proposed controller with simulation time of $2,000$ [sec]. As the same with the proposed controller, we start applying the proposed controller to the sensors group at the same time with the last sensor node is released, which is when $t = 280$ [sec].

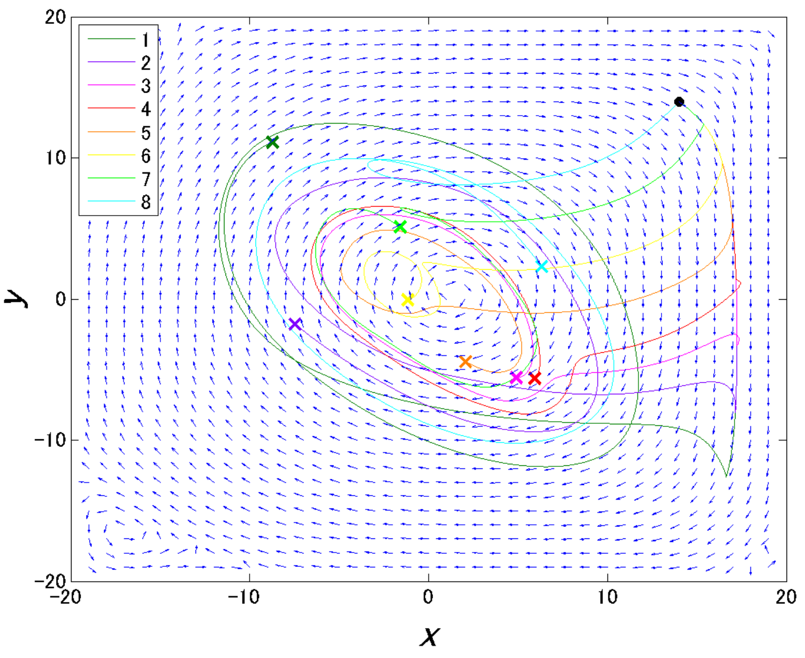


Figure 2.16: Trajectory of each sensor in the distribution using the modified proposed controller from $t = 0$ to $t = 2,000$. The initial positions of all sensor nodes are set to be the same position which is plotted by black dot mark in this figure. The controller is applied to the sensors group after $t = 280$

Time series graphs of centroid and position variance from $t = 0$ to $t = 800$ of the modified proposed controller can be seen from Fig. 2.17 and Fig. 2.18.

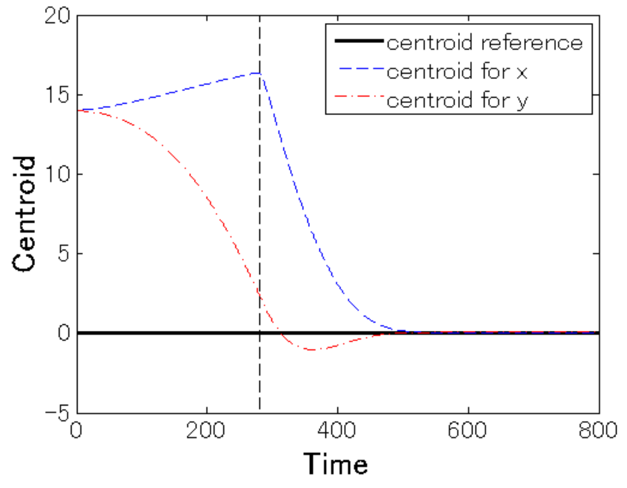


Figure 2.17: Time series graph of the centroid position of all sensor controlled by the modified proposed controller for x and y directions. The vertical dashed line represents the time when all sensor nodes are released and the time when the controller is applied. The desired centroid is set to be $(0, 0)$

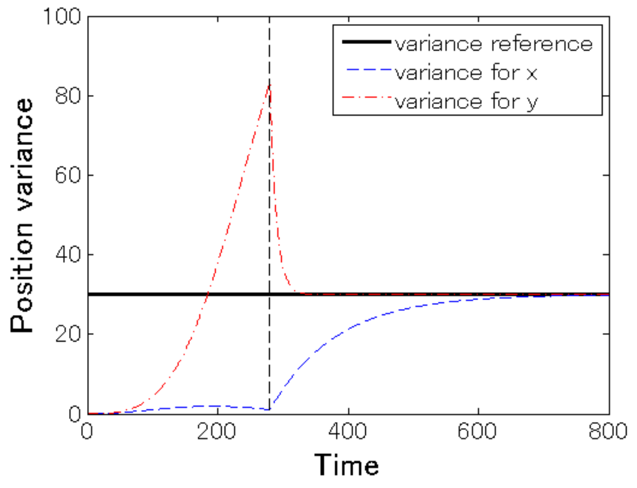


Figure 2.18: Time series graph of the position variance of all sensor controlled by the modified proposed controller for x and y directions. The vertical dashed line represents the time when all sensor nodes are released and the time when the controller is applied. The desired position variance is set to be 30 for both x and y directions.

The energy consumption is defined as

$$E(T) = \int_0^T \boldsymbol{u}^\top(t)\boldsymbol{u}(t)\mathrm{d}t, \tag{2.76}$$

where T is the time period. Fig. 2.19 shows the comparison of energy consumption among the modified proposed controller, the proposed controller, and the position-velocity controller. According to Fig. 2.19, the modified proposed controller can provide the lowest energy consumption than the position-velocity controller and the original proposed controller. This make the modified proposed controller be able to perform lake monitoring over a longer period of time.

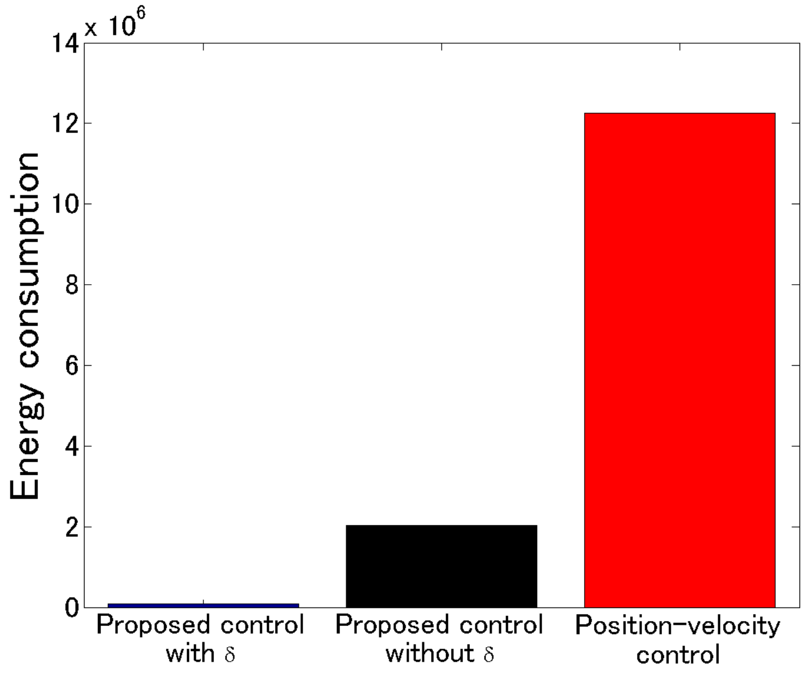


Figure 2.19: Energy consumption comparison among the modified proposed controller, the proposed controller, and the position-velocity controller.

2.8 Conclusion and Discussion

In this chapter, the simulation of lake monitoring sensors distribution with low energy consumption has been carried out. Energy consumption, which are in term of input forces, of considered controllers are compared. The distributions are performed in both uniform constant water flow field and complex steady flow field. The proposed controller provides less energy consumption comparing with a method using simple position-velocity control. Furthermore, the modified version of the proposed controller gives more effective group control with lower in energy consumption than the original proposed controller. This makes the proposed lake monitoring sensors system be able to achieve long time monitoring of water resource. The lower in energy consumption for the monitoring sensors distribution saves the amount of limited energy inside the battery attached to each sensor node. Therefore, with low energy consumption, the monitoring system operating time will last longer and the system efficiency will be increased.

Chapter 3

Pair-wise assignment of identity

3.1 Introduction

Drosophila melanogaster, generally known as common fruit fly, is an important model organism in neurobiology [42]. With *Drosophila*, many sophisticated genetic methods can be used to manipulate the activity of specific neurons in a targeted manner. This allows powerful experiments that examine the structure and function of the nervous system at the resolution of single neurons.

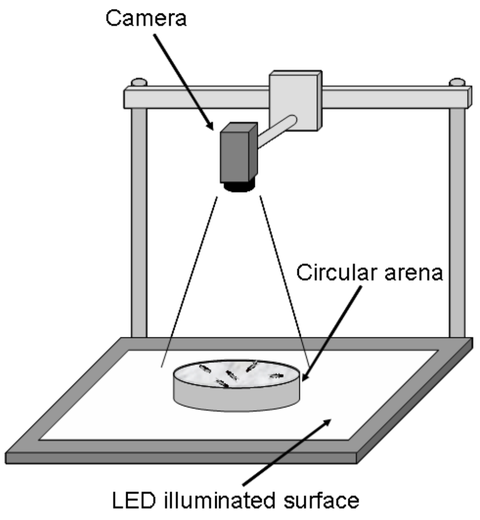
Machine vision systems have been worked for locomotion analysis of insects, such as bees [43], houseflies [44], and ants [45]. This work uses machine vision system on *Drosophila*. In order to understand behavior of *Drosophila*, one of the important information that need to be measured is their movement. To make an observation of their movement in a controlled environment, a high performance measuring system is required.

For a single fly, it is quite simple to observe its movement but it is not possible to provide some movement patterns that are related to interactive behavior among a group of multiple flies. Some examples of researches using movement information of a single fly are shown in [46–49]. Measurement of position of free-flying single *Drosophila* in 3D environment for a behavioural system analysis was done in [46]. Takeoff dynamic of a single *Drosophila* was studies by [50] and [51]. A study of a walking fly that was shortened its wings in order to limit its movement is shown in [49].

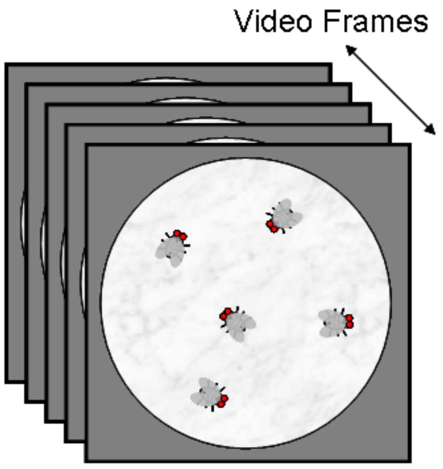
Instead of a single fly, generally a group of fruit flies are studied together in an experiment. The reason for studying a group of flies is that it provides some behaviors of interaction between flies that cannot be seen from observing only a single fly, such as touching and chasing. Observing multiple flies concurrently makes it difficult and challenging to track each fly using human eyes and even by using computer vision system since their appearance are similar. A study of *Drosophila* behaviour using two flies was done by [52]. However, studying with too few flies takes time because we need to wait until the flies do the behavior.

This work introduces a system for detecting and tracking in order to solve the problem of multiple flies tracking. Our experimental setup is shown in Fig. 3.1 (a). There is an LED illuminated surface located at the bottom. A circular arena containing a group of *Drosophila* is placed on the LED illuminated surface. A video camera is attached to the structure to have a view of the arena. The camera captures a video containing many frames of flies in the arena as shown in Fig. 3.1 (b). The captured video will be used for fly tracking and posture estimation later.

Given the video file containing a group of flies in a controlled arena, our system detects all flies in the arena, reliably identifies individuals between consecutive frame pairs, and measures the posture and the velocity of each fly in each frame in the



(a)



(b)

Figure 3.1: Experimental setup (a) and video frames for fly tracking and posture estimation (b). An LED illuminated surface is located at the bottom of the setup. A circular arena containing a group of *Drosophila* is placed on the LED illuminated surface. A video camera is attached to the structure in order to have a view of the arena.

video. As a result, using our system can help scientists to get information of the flies movement in an experiment.

Unlike traditional binarization, which does background subtraction first and then applies threshold, we also provide another binalization method done by firstly applying threshold to a background image and a current image and then performing background subtraction. The binarization method in this work can give better binarized result of flies image, which will be shown later.

Having multiple flies together in a controlled arena makes it possible for some flies to touch each other. This results in merging problem of flies in image processing, which two merging flies will be detected as a single fly. In our tracking system, this problem is solved by adjusting the threshold at the flies merging region. As a result, by using the new threshold two flies can be separated from a merging area.

For the approximation of posture of each fly, calculation based on the ellipse approximation is carried out. The posture of each fly includes the 2D-position and the orientation of its body. Notice that this ellipse approximation can be used even with non-ellipse shape.

At the last part of our tracking system, fly identification for each fly between every pair of two consecutive frames is done. The identification is considered as flies pairing based on distance. To do the job, we use the closest neighbor approach algorithm, which will be explained in detail later. Our fly tracking algorithm provides a robust flies detecting and tracking even when clusters of flies appear in the view. The accuracy of the fly identification is evaluated by manually inspecting more than $1,000$ fly pairs from random frame pairs and random videos. The result of fly identity has low error rate of 0.48%. For the posture of each fly, the average errors of the position

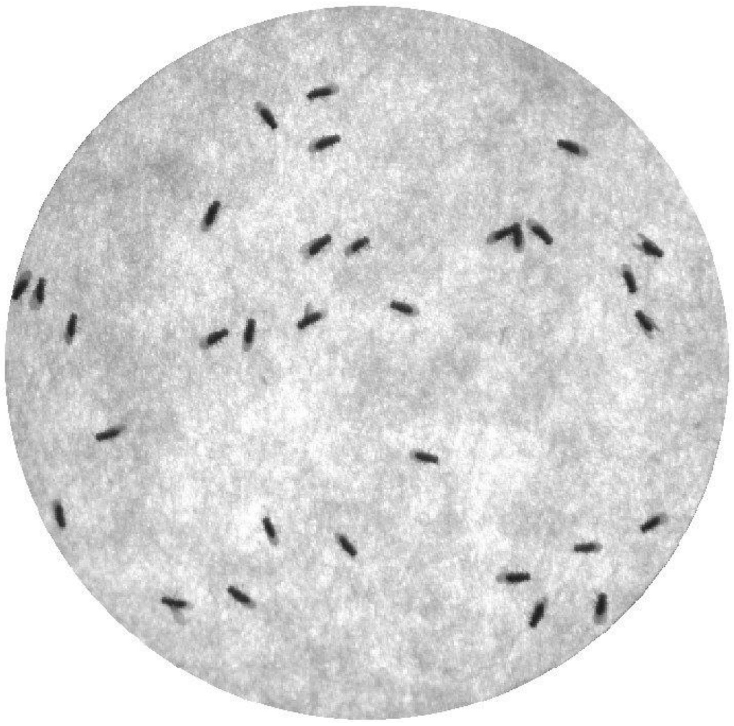


Figure 3.2: Grayscale image of *Drosophila* group in a closed circular arena is presented in this figure. It is closed in order to prevent fly from leaving the arena. Each fly is shown in a black oval shape. The area outside the circular arena is ignored.

of the flies and the orientation are approximately 5% of fly body length and $2.2 \pm 0.2°$, respectively.

3.2 Fly detection algorithm

Our fly tracking system is divided into three parts; fly detection, fly posture approximation, and fly identification. Given an input video containing a group of flies in a closed circular arena, the result from the first part is that each fly will be detected as a single fly. In this section, the detection of each fly will be explained in detail.

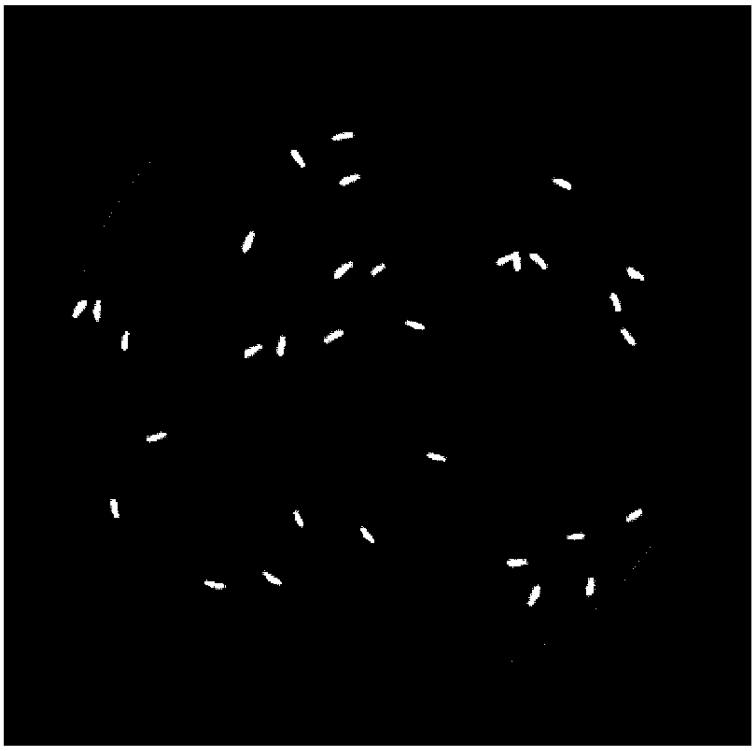


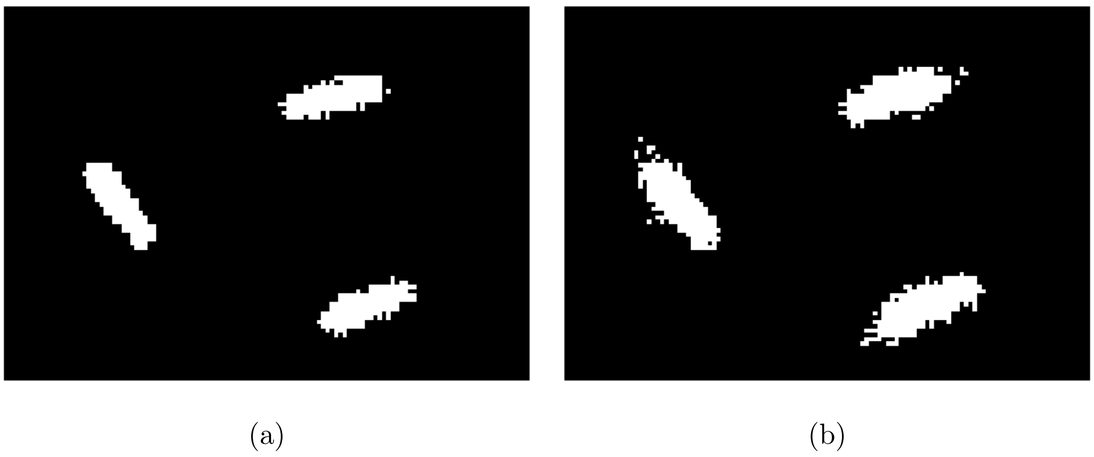
Figure 3.3: Binarized image showing *Drosophila* group in a circular arena. Each fly is shown as a white oval shape in the scene.

Firstly for each frame in the input video, RGB image is converted to Grayscale image. Fig. 3.2 shows an example of grayscale image of *Drosophila melanogaster* in a closed circular arena.

Next, binarization and background subtraction are performed. In this step, our work applies binarization to the image and the background image separately before doing background subtraction. This makes this work different from the others that normally perform binarization after doing background subtraction. Fig. 3.3 is a binarized image showing that only flies can be seen from the scene as the white oval areas and these areas are considered as detected regions. According to the figure, there are some noises appear as tiny white dots, which are not flies. Fig. 3.4 shows binarized image comparison between our method, which applies binarization first and then followed by background subtraction, and normal approach, which uses background subtraction first and then followed by binarization. Our approach results in clearer appearance of fly body excluding its wing part. This clearer appearance will provide more accuracy for the fly posture approximation.

To remove the noises and to make the flies look smoother, erosion followed by dilation operations using the same structure element are then applied to the image, respectively. The structure used in these operations is a circle with radius of 3 pixels. Then the measurements of a set of properties of the detected region are performed. These properties include area, centroid, eccentricity, orientation, bounding box (the smallest rectangle containing the region), and the linear indices of the pixels in the region.

In order to determine whether a detected region is actually a fly or not, likelihood of existence of fly is involved. The likelihood of existence is calculated based on



(a) (b)

Figure 3.4: Binarized image comparison between our method applying binarization first and then followed by background subtraction (a) and normal approach using background subtraction first and then followed by binarization (b).

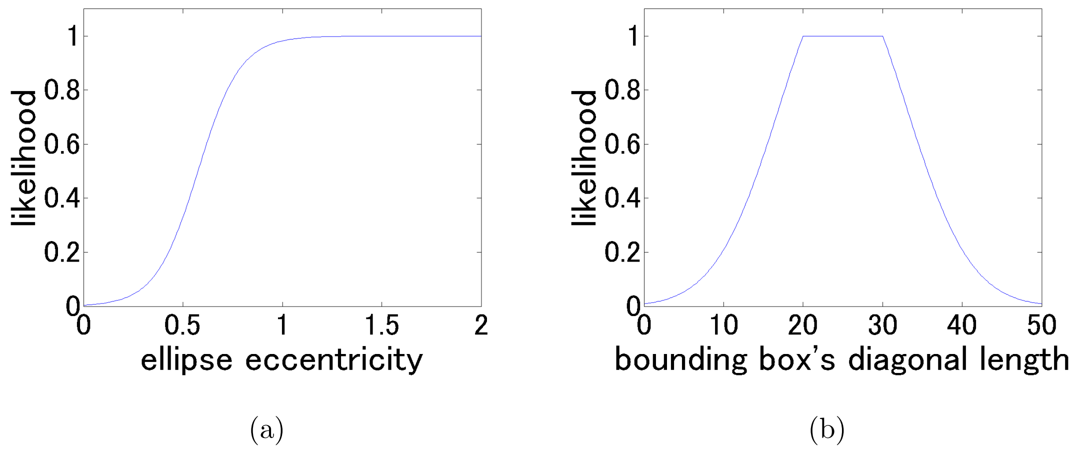


Figure 3.5: Graph of likelihood of existence of fly with respect to ellipse eccentricity (a). More ellipse eccentricity gives higher likelihood. Graph of likelihood of existence of fly with respect to bounding box's diagonal length. The likelihood is at the maximum value when the bounding box's diagonal length is in a curtain range.

ellipse eccentricity and bounding box's diagonal length of the region. In this step, two likelihood values are first calculated. They are a likelihood calculated based on eccentricity and a likelihood calculated based on bounding box's diagonal length. Then the likelihood of existence of fly is obtained by multiplying those two likelihoods together. Fig. 3.5 shows graphs of likelihood with respect to ellipse eccentricity and likelihood with respect to bounding box's diagonal length. For the likelihood based on the eccentricity, higher ellipse eccentricity gives higher likelihood. For the likelihood based on the diagonal length, the likelihood is at the maximum value when bounding box's diagonal length is between 20 to 30. The likelihood with respect to ellipse eccentricity can be computed using the following equation

$$p_1(\varepsilon) = \frac{1}{1 + e^{4(\frac{2(1-\varepsilon)}{0.085} - 1)}}, \tag{3.1}$$

where ε is input ellipse eccentricity and p_1 represents the likelihood corresponding to the input. The likelihood with respect to bounding box's diagonal length can be obtained as follows

$$y(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d-\mu}{\sigma}\right)^2}, \tag{3.2}$$

$$p_2(d) = \begin{cases} 1, & \text{if } y(d) \geq y(d_{\max}) \\ e^{\frac{(d_{\max}-\mu)^2 - (d-\mu)^2}{2\sigma^2}}, & \text{otherwise} \end{cases} \tag{3.3}$$

where d is input bounding box's diagonal. The output p_1 represent the likelihood corresponding to the input. μ and σ are mean and standard deviation, respectively. The value $d_{\max}$ represents the maximum value of bounding box's diagonal that gives the likelihood of 1. For Fig. 3.5 (b), μ, σ, and $d_{\max}$ are set to be 25, 8, and 30, respectively.

The area that has likelihood greater than likelihood threshold is extracted and average area of all extracted area is calculated. For the extracted area, we assign the estimate number of flies in that area to be 1.

For the region i whose likelihood is less than 0.5, the estimate number of flies ($N\text{likely}^i$) in that region is calculated by the following equation

$$N^i_{\text{likely}} = \text{Round}(\frac{A^i}{A}), \tag{3.4}$$

where the nearest integer function Round(x) gives the nearest integer to x, where $x \in \mathbb{R}$. A^i represents the area of the region i and A is the average area of all extracted area.

The regions having likelihood less than the likelihood threshold and N^i_{likely} equals to zero will be ignored from being recognized as flies. The regions having likely number of flies more than 1 will be considered. To separate those merging flies, we will adjust a new threshold and apply it only to the region. The new threshold for extracting merging flies is then calculated by summation of pixel intensity average and pixel intensity standard deviation within the region. Next, the binarization at the new threshold is performed and the number of detected regions is calculated.

An example of extracting merging flies in binary image is shown in Fig. 3.6. According to the figure, there are two flies merging in the region. The left side of Fig. 3.6 shows how the merging area looks like under binarization using the old threshold. On the other hand, the right side of Fig. 3.6 shows how the area looks like under binarization using the new threshold. By using the new threshold, the two flies in the merging region can be separated.

Let the new calculated number of regions i to be N^i_{new}. If N^i_{new} is equal to N^i_{likely}, then the process is completed. If N^i_{new} is not equal to N^i_{likely}, then the threshold

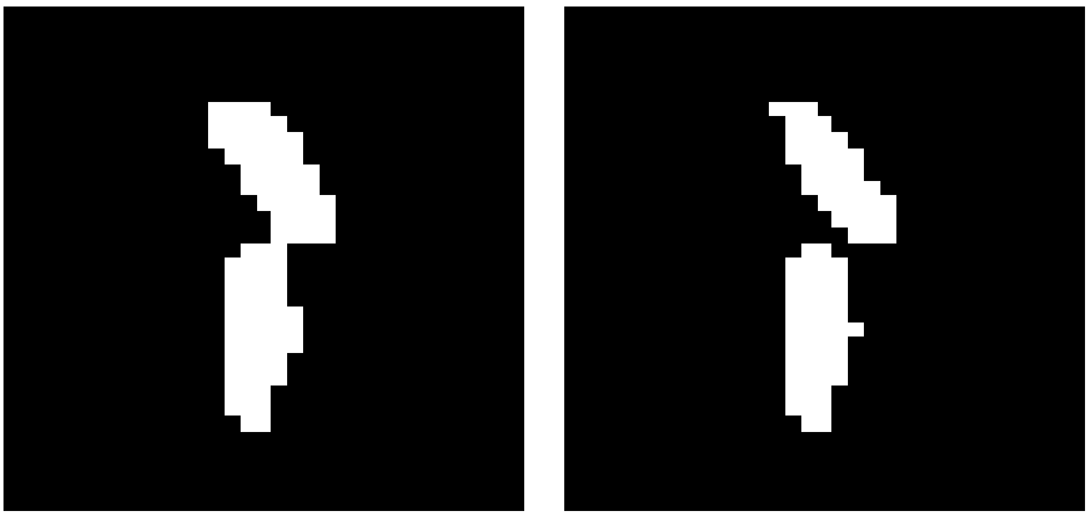

(a) (b)

Figure 3.6: Binary image of a flies merging area obtained by using the old threshold (a) and by using the new threshold (b). Two flies can be separated after using the new threshold for binarization.

value needs to be updated and we have to and repeat the binarization at the updated threshold. Let the threshold at the region i at the current iteration to be th^i_{current}, the threshold is updated by the following equation

$$th^i_{\text{update}} = \begin{cases} th^i_{\text{current}} + 1, & \text{if } N^i_{\text{new}} > N^i_{\text{likely}} \\ th^i_{\text{current}} - 1, & \text{otherwise} \end{cases} \tag{3.5}$$

where th^i_{update} is the updated threshold.

Notice that the process will also finish if the number of iteration reaches the maximum number of iteration or the new threshold reaches the threshold of fly body intensity.

After the process is completed, measurements of properties of a region after improvement process are performed. These properties are the same as previously stated which include area, centroid, eccentricity, orientation, bounding box, and the linear indices of the pixels in the region. The improved regions are extracted by searching the region that has likelihood greater than the likelihood threshold.

The posture of each fly is determined based on ellipse approximation. Furthermore, fly identification is performed based on the distance between flies in two consecutive frames. Posture determination and fly identification will be explained in detail in next two sections.

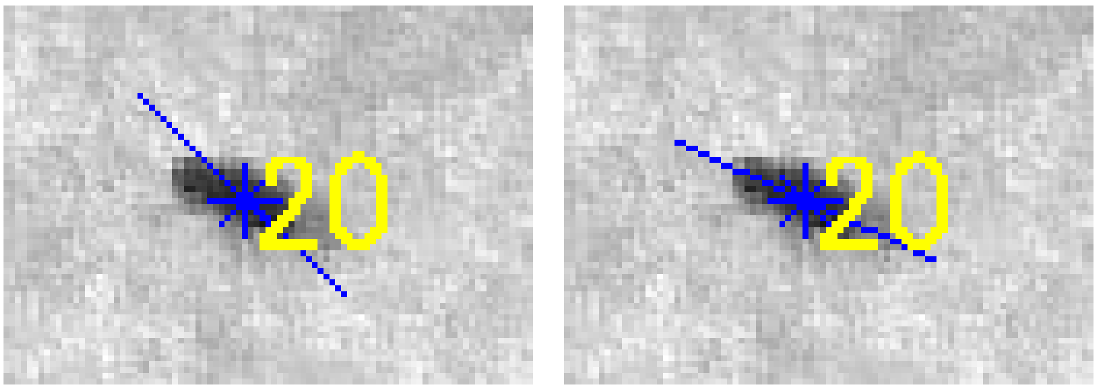
3.3 Fly posture approximation

In this section, the posture approximation will be described. In this part, posture calculation based on the ellipse approximation is carried out. The posture of a fly consists of 2D-position and orientation of its body. The approximated orientation of a fly body will be compared with the same fly orientation obtained from using RANSAC iterative method. For the ellipse approximation, region-based method using the moments of a shape is used for estimating the best-fit ellipse [53]. This ellipse approximation can be used as an expression of shape even though the shape is not an ellipse. First, the moment of order $p + q$ of a 2D region (G) is calculated by the following integration over the area of G

$$m_{pq} = \iint_G f(x, y)x^p y^q \; xy. \tag{3.6}$$

Here f is a function used to describe situations when some properties of each part in the region vary. Since we treat all the parts in the region equally, thus the function f will always be $f(i, j) = 1$. The zeroth moment (m_{00}) is basically the area of G.

For calculating the moments of a pixelated binary image region, it is assumed that



(a) (b)

Figure 3.7: Posture approximation by using RANSAC iterative method (a) and ellipse approximation (b). The fly is labelled with yellow numbers. The blue line indicates the postures of the fly. The center of the body is shown as the blue asterisk mark. The ellipse approximation provides better fly body orientation approximation than the RANSAC iterative method.

the region of interest is totally enclosed in a rectangle of size n by m pixels. The Eq. (3.6) for moments calculation reduces to

$$m_{pq} = \sum_{i=1}^{n} \sum_{j=1}^{m} x_i^p y_j^q b(i,j), \quad b(i,j) \in \{0,1\}, \tag{3.7}$$

where the function $b(i,j)$ has value of one if the i,jth pixel is in the foreground. If the pixel is in the background then the function has value of zero. The coordinate (x_i, y_j) is the coordinate of the i,jth pixel.

The central moments of area G are determined by

$$u_{pq} = \iint_G f(x-\bar{x})^p (y-\bar{y})^q xy, \tag{3.8}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \tag{3.9}$$

$$\bar{y} = \frac{m_{01}}{m_{00}} \tag{3.10}$$

are the centroids of the area G. They can be written in terms of the moments as

$$u_{00} = m_{00} \tag{3.11}$$

$$u_{10} = u_{01} = 0 \tag{3.12}$$

$$u_{20} = \frac{1}{m_{00}}\Big(m_{20} - \frac{m_{10}^2}{m_{00}}\Big) \tag{3.13}$$

$$u_{02} = \frac{1}{m_{00}}\Big(m_{02} - \frac{m_{01}^2}{m_{00}}\Big) \tag{3.14}$$

$$u_{11} = \frac{1}{m_{00}}\Big(m_{11} - \frac{m_{10}m_{01}}{m_{00}}\Big). \tag{3.15}$$

We use the central moments to determine the best-fit ellipse for the arbitrary shape with orientation ϕ of the long axis of the ellipse. The orientation ϕ is computed as follows

$$\phi = \frac{1}{2}\tan^{-1}\Big(\frac{2u_{11}}{u_{20} - u_{02}}\Big). \tag{3.16}$$

Fig. 3.7 show a labeled fly with identification number of 20 with posture approximation by using RANSAC iterative method (a) and the ellipse approximation (b). As seen from the figure, the position and the orientation of the fly are shown in a blue asterisk mark and a blue line, respectively. The ellipse approximation provides better fly body orientation approximation than RANSAC iterative method.

3.4 Fly identification

The fly identification method used in our fly tracking system will be explained in detail. The identification of each fly between two consecutive frames is considered as flies paring process based on distance. In this step we apply a closest neighbor approach to pair each fly between two frames. The closest neighbor approach algorithm is shown in Algorithm 1.

```
Data: two consecutive frames, $F_1$ and $F_2$
Result: pairing between detected objects in $F_1$ and $F_2$
create a set $\mathbb{P}$ containing all possible pairs between detected objects in $F_1$ and
  $F_2$;
create an empty set $\mathbb{Q}$;
for all pair in $\mathbb{P}$ do
    compute Euclidean distance between objects in each pair;
end
while $\mathbb{P} \neq \emptyset$ do
    find the pair that has minimum distance within $\mathbb{P}$;
    delete the pair from $\mathbb{P}$ and other pairs containing the element in the pair;
    put the minimum distance pair into $\mathbb{Q}$;
end
```

Algorithm 1: The closest neighbor assignment algorithm

Fig. 3.8 shows an example of pairing flies between two consecutive frames. From the figure, two flies are labeled with number 15 and 17. The asterisk marks represent the center of fly body at current frame. The positions in the next consecutive frame are indicated by the blue dots. They are paired with their corresponding position in the next frame by blue lines.

Notice that the maximum changes in position and angle of paired particles are set to be approximately 7mm and 90°, respectively. The change that is greater than these maximum will not be considered.

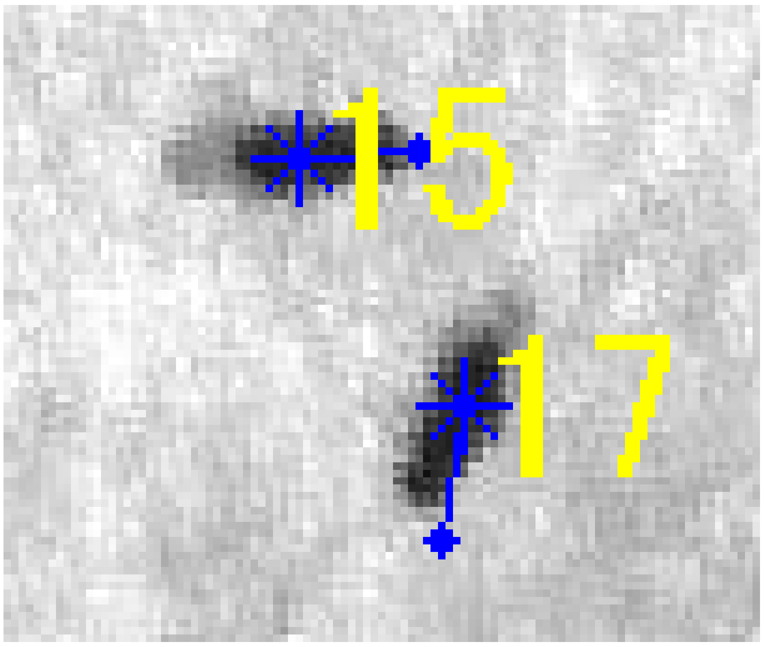


Figure 3.8: Two flies labeled with number 15 and 17 are paired with positions from the next consecutive frame. The asterisk marks represent the center of fly body at current frame. The positions in the next consecutive frame are indicated by the blue dots. The closest neighbor approach is used to pair these two flies.

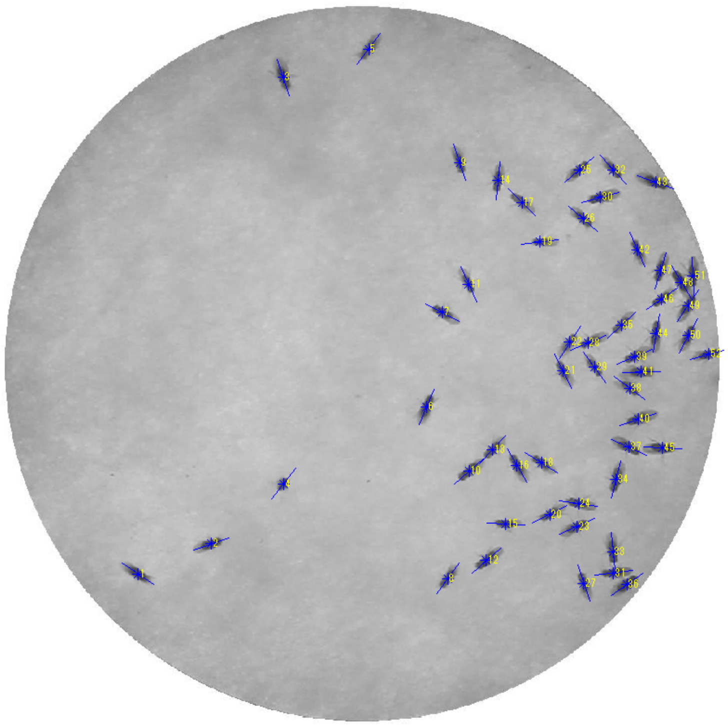


Figure 3.9: All flies in the arena are detected. The positions of the center of fly body are indicated with blue asterisk marks. Moreover, each fly is labeled with a unique number in yellow. The posture of each fly is shown by the blue line located over its body.

Table 3.1: Error of fly identity, position, and orientation

	Error	**Number of sampled data**
Fly identity	5 flies pairs	$1,049$ flies pairs
Position analysis	0.108 ± 0.006 mm	100 flies
Orientation analysis	$2.2 \pm 0.2°$	95 flies

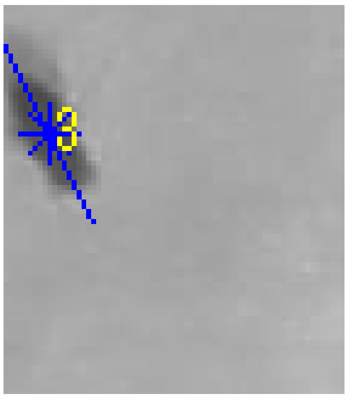
3.5 Experiment

In this section, a result of fly tracking algorithm is presented. A 20 fps video having the total of 52 flies in a closed circular arena is used as the input for the system. Considering about fly detection, Fig. 3.9 shows that all flies in the arena can be detected with our system. According to the figure, the positions of the center of fly body are indicated with blue asterisk marks. Further, each fly is labeled with a unique number and the posture of each fly is indicated with the blue line.

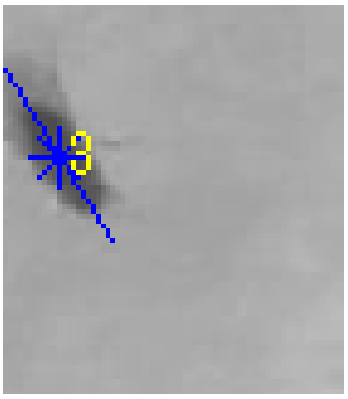
To see a tracked fly in many consecutive frame sequences, Fig. 3.10 shows a tracked walking *Drosophila* from $t = 0$ (a) to $t = 0.25$ (f) with its posture indicated by the blue asterisk mark. The time interval between two consecutive sub-figures is 50 ms.

All of the examples' input files were obtained from experiment of a group of flies with the presence of sugar. The computations are done on a parallel computer LX406Re-2 that has 68 nodes. Each node is equipped with two groups of 12-core Intel Xeon processors E5-2695v2 and a main storage of 128 GB. The algorithm is run in MATLAB on each node.

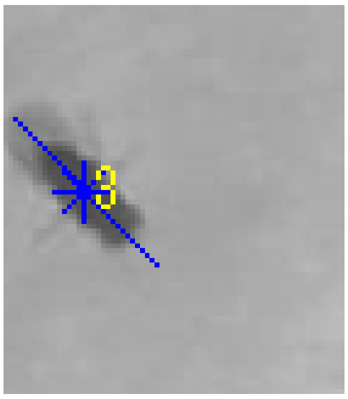
Table 3.1 shows error of fly identity, position, and orientation. The accuracy of flies identification in two frames is evaluated by manually inspecting $1,049$ fly pairs from random frame pairs and random videos. Fly identity has error of 5 flies pairs



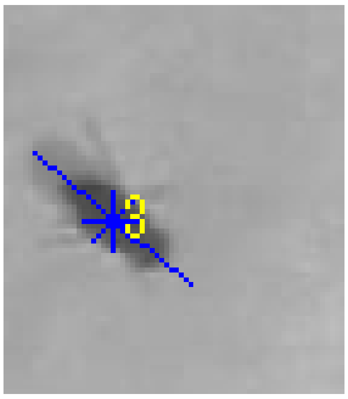
(a)



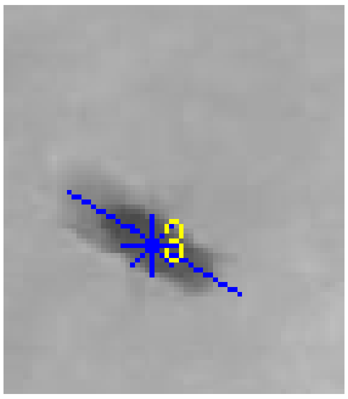
(b)



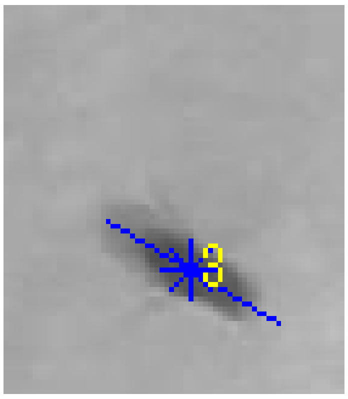
(c)



(d)



(e)



(f)

Figure 3.10: Tracked walking *Drosophila* from $t = 0$ to $t = 0.25$ with label number 3 and posture indicated by the blue asterisk mark. The time interval between two consecutive sub-figures is 50 ms.

that is about 0.48%. The failed fly identity occurred at flies merging area and at the case of two flies moving into each other with high speed. For the position and the orientation of the flies, 100 flies and 95 flies were inspected, respectively. They were from random flies, random frames, and random videos. The average errors of the position and the orientation are 0.108 ± 0.006 mm (approximately 5% of fly body length) and $2.2 \pm 0.2°$, respectively.

3.6 Conclusion and Discussion

In this work, an algorithm to detect a group of *Drosophila melanogaster* or common fruit flies is introduced. The fly tracking algorithm can provide accurate result with low error of flies identification and flies posture estimation including position and orientation. The next goal is to apply the system to more challenging fly experiment condition like applying electric shock to the fly. With the presence of electric shock, the fly will jump or fly with high speed. Moreover by using high-speed camera, the difficulty of fly tracking in case of jumping or flying flies will be reduced. In this way, better accuracy for the tracking and robustness of the algorithm will also be improved as well.

Chapter 4

Assignment of identity using head direction and moving prediction

4.1 Introduction

The study of model organism provides understanding of how biological processes work, which benefits humans in terms of medical knowledge and understanding of other biological behaviors [16]. One of the popular model organisms that has been heavily used in research is *Drosophila melanogaster*, or generally known as the common fruit fly. With this model organism, a variety of genetic methods can be used to manipulate the activity of specific neurons in freely moving and behaving animals [54].

A common way to analyze behavior is to categorize and quantify animal movements. Locomotion observation of insects such as bees [43], flies [44], and ants [55], using machine vision systems have been used for the analysis. In this work, we focus on using a machine vision system to track freely walking *Drosophila melanogaster* in a circular arena. Considering the fact that it has a small size and it can sometimes jump; this causes great difficulties for motion observation.

Dealing with only a single fly can provide its information in detail and the motion observation can be easily implemented, but the motion related to interactive behavior among a group of multiple flies will be absent. There are some researches working on behavior of a single fly. [50] and [51] study takeoff dynamic of a single *Drosophila*. A study of freely flying *Drosophila* is done by [46]. [52] focuses on behavior of *Drosophila* using two flies. However, it takes time until they show interactive behavior.

Machine vision system with a presence of multiple *Drosophila* fulfills the motion observation in the way that it can provide observation on the behaviors of interacting flies that cannot be observed from the single fly scenario. More flies provide more data and in addition, sexual behavior can also be observed. However, dealing with multiple flies is challenging to the system since the system has to be able to track each individual fly. This means the system has to give each fly an identifier and maintain its identifier throughout the observation period.

This work introduces a machine vision system that can detect and track multiple *Drosophila* in a video input. Overall process of this system is divided into three main parts: foreground detection, posture modeling, and tracking.

4.2 Related work

To achieve a successful result from a video input, firstly, the system needs to be able to distinguish *Drosophila* from the background. Based on the fact that *Drosophila* moves while other objects stays in the same frame, this gives us an opportunity to use mode value of intensity to extract background information from the image scene. Background extraction with a moving foreground is done by [56]. Once the background has been extracted, background subtraction [57] is then performed to split up *Drosophila* and other objects. This does not only improve the accuracy of *Drosophila* detection but also prevents the system from wasting processing time on other objects beside the fly. The background subtraction example can be seen from [58] and [59] with an application of video surveillance and monitoring. Other applications using the background subtraction are such as optical motion capture [60], human computer interaction [61], and manufacturing industry [62].

After the *Drosophila* have been isolated from the background, the resulting image needs to be polished in order to remove the remaining noises. Polishing process can be accomplished by normal binary thresholding or by applying opening mathematical morphology [63]. In this system, a Laplacian of Gaussian filter [64] is applied to the resulting image to provide smooth and noiseless blobs result. Next, ellipse fitting is performed at each blob and major axis, an axis representing the widest part of the ellipse, is calculated. Region-based method for ellipse fitting is discussed in [53].

There are many off the shelf tracking software packages available. One of the most well-known *Drosophila* tracking programs is C-Trax [17]. It is an open-source software that is freely available for estimating the positions and orientations of walking flies. The program comes with ability to automatically classify the flies' behaviors. In addition to C-Trax, another system that is well-known for measuring animal behavior is JAABA [65]. JAABA uses machine learning-based system to compute behavior of animals together with input trajectory from other tracking programs. idTracker [18] provides a tracking system that can be used with a wide range of animals including zebrafish, medaka fish, mice, flies, and ants. EthoVision XT is suitable for a wide range of set-ups but it has weak point when dealing with a large number of animals with interactions.

In this work, we present a multiple *Drosophila* tracking system that works without needing any modification of the test subjects. This system is able to detect and track each individual fly in the scene while providing an accurate result obtained from using a Kalman filter. In addition, it uses a combination of prediction from the Kalman filter and the closest neighbor principle in order to achieve better result in tracking selection. Heading direction is also used for confirmation of fly pairing between two consecutive frames. The proposed system is a platform for future development.

4.3 Method and Material

For an input video in a range of interested frames, foreground detection and posture modeling are processed first for all frames. Next, position and heading direction obtained from these two steps will be used in the tracking step.

4.3.1 Foreground detection

At each time step, an input image is taken from image sequences from an input video and converted into a gray scale image. The background image is extracted by considering only the static part of the image sequences. In this way, the pixel value that frequently appears in the image sequences is treated as background while the one with less frequency will be considered as foreground. In order to determine the value that has the highest frequency at each pixel, a vector containing pixel values from the image sequences at the same pixel position is created. Mode calculation on the vector is performed and the most frequent value is then obtained. This process is repeated for every pixel position of the image sequences, thus the background is created. 100 images are randomly selected from the image sequences and are used in the background extraction step instead of the whole image sequences. Fig. 4.1 shows an example of six consecutive images of *Drosophila* and Fig. 4.2 represents the extracted background images generated by using a number of frames from the video in Fig. 4.1 and calculating the mode value at each pixel.

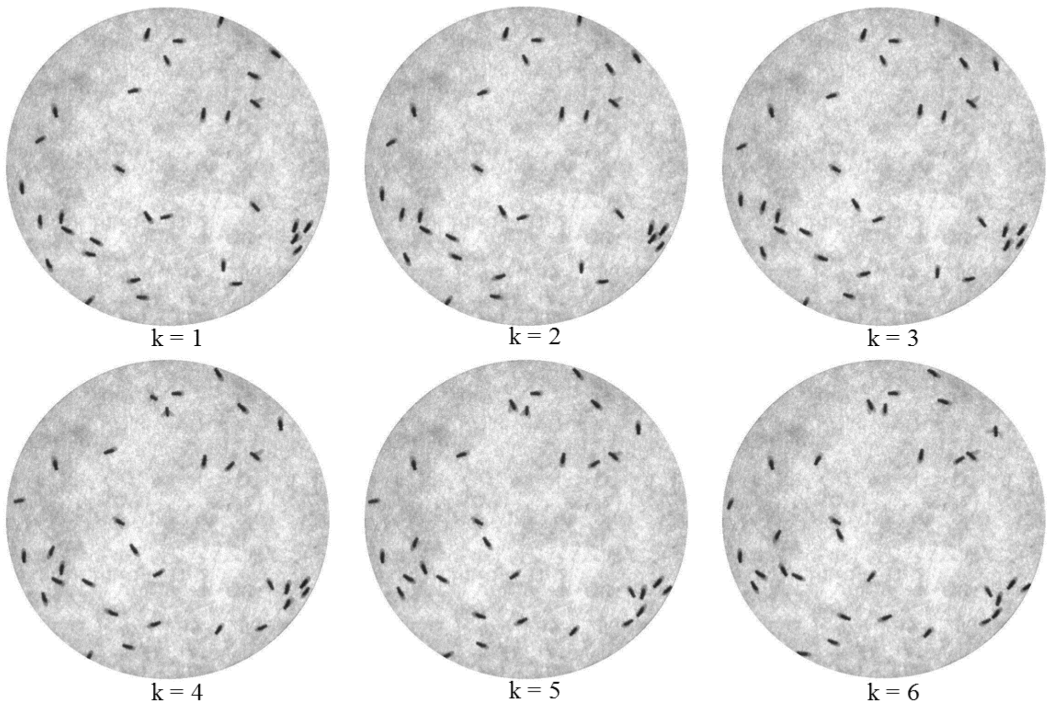


Figure 4.1: Input images of six consecutive frames. A circular arena contains 32 *Drosophila* as the black ovals inside. The frame rate is 19 fps.

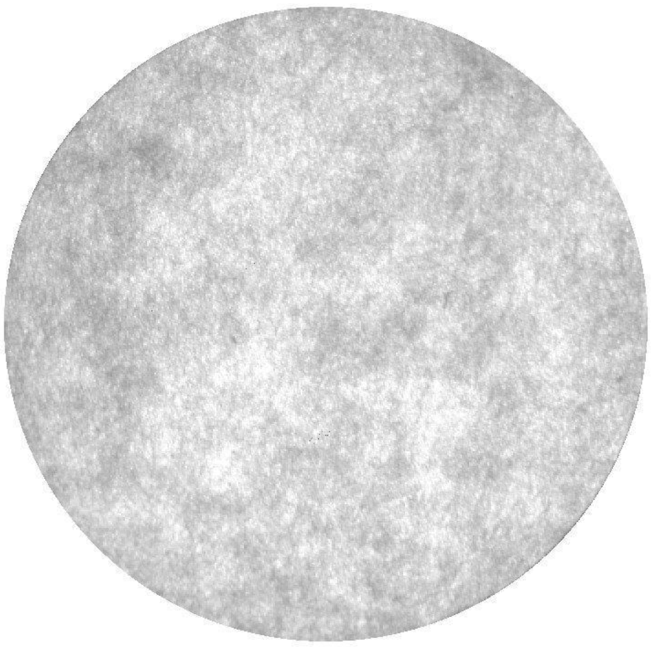


Figure 4.2: Extracted background image of the input video from Fig. 4.1. The background image was generated by using a number of frames from the same video file and calculating the mode value at each pixel.

In some cases, input video file suffers from varying of light in the scene due to non-perfect experimental setup. The varying light affects the ability of detection and may result in missed detections. Therefore intensity adjustment is performed, using the background image as a reference for intensity shifting. The intensity shifting is done by two steps: first, calculating intensity mean values of both the background image and the current input image, second, updating intensity of every pixel in the input image by adding the difference between the intensity mean values of the background image and the current input image to all pixels.

The foreground is detected by performing background subtraction with the current input image. In this step, some noises may appear in the result image. Fig. 4.3 illustrates the resulting image after performing background subtraction followed by intensity reversal. The static arena part is deleted and only *Drosophila* are left in the scene.

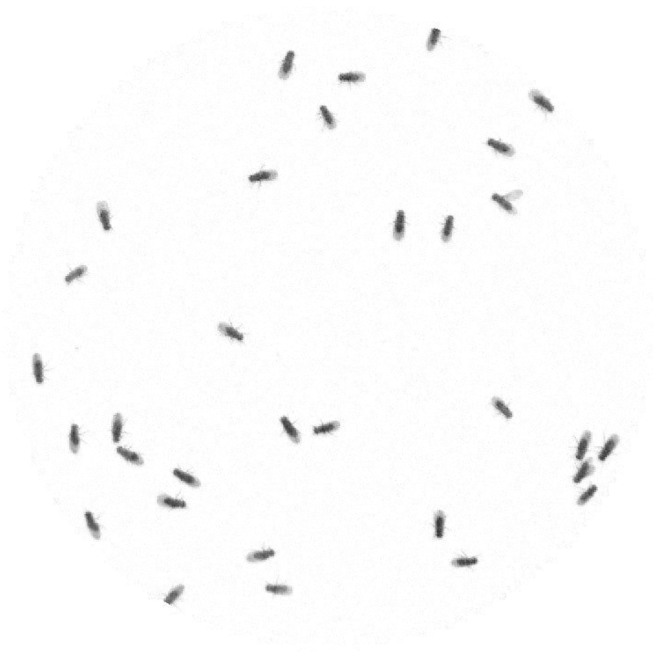


Figure 4.3: Result image from background subtraction followed by intensity reversal. The static arena part that belongs to the background was deleted. Only moving flies' body and their wings remain in the result image.

Next, blob filter is applied to the image. The blob filter uses Laplacian of Gaussian as a kernel matrix for convolution with the image. Laplacian of Gaussian appearance is shown in Fig. 4.4. According to the fact that the fly's body in Fig. 4.3 is dark, the Laplacian of Gaussian makes the body white. The convolution makes bright pixel dark and dark pixel white. In this way, fly's body will appear as a white blob. Next, a proper user-defined threshold is applied to the blob image in order to get rid of some noises. In this thresholding step, the unwanted part will be deleted from the scene and wanted part will be preserved in the scene. Here, this is not binary thresholding, therefore every pixel in every blob still possesses detailed intensity values. Fig. 4.5 shows a resulting image after applying blob filter and thresholding.

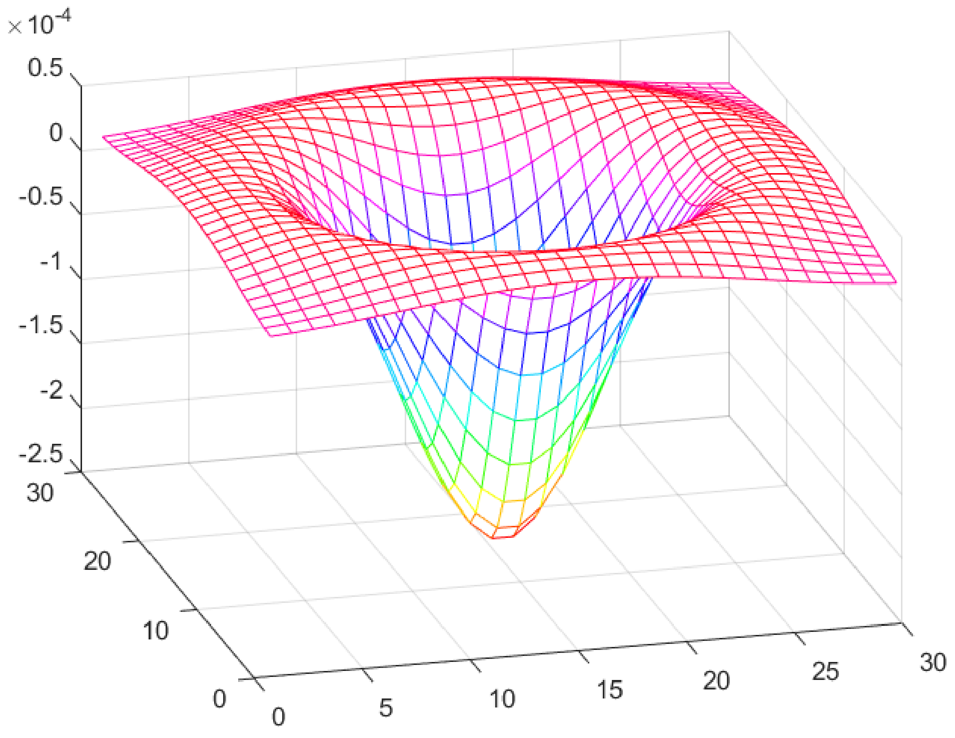


Figure 4.4: Laplacian of Gaussian as a kernel in 3-D view. The kernel is used for convolution with the resulting image from background subtraction (Fig. 4.3).

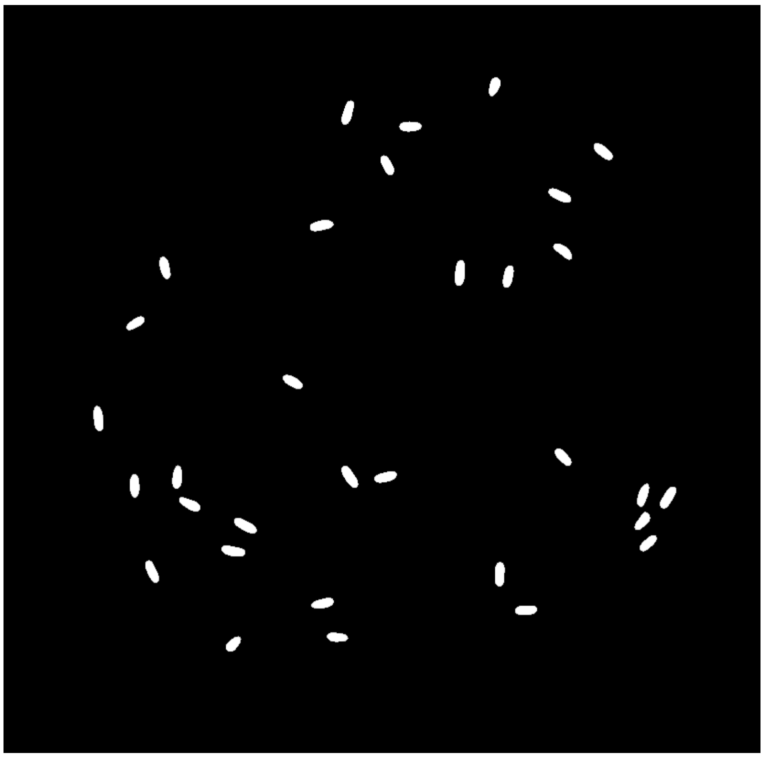


Figure 4.5: Filtered blob image. Thresholding is applied to the product of the convolution using the Laplacian of Gaussian as a kernel from Fig. 4.4, resulting in the filtered blob image.

A two dimensional Laplacian of Gaussian function is written as

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{4.1}$$

where x and y are Cartesian coordinates. Standard deviation is user-defined and is shown as σ. The value of standard deviation used in Fig. 4.4 is $\sigma = 6$.

In case of multiple foreground objects, sometimes a group of ones is detected as only one blob, meaning that one object is detected instead of multiple objects. This results in an error of detected positions and will decrease the tracking efficiency in later steps. This problem takes place when some foreground objects locate very close to one another. To overcome such a problem, blob analysis is introduced.

The blob analysis process starts by first calculating parameters of every blob in the blob image such as area, bounding box, and center of the blob. Second, average area of all the blobs is determined and considered as an expected area of one object. Third, at each blob, the expected number of objects is calculated by dividing the blob area with the expected area. If the expected number of objects is one, then that blob is considered to be correct, and no further modification is necessary. On the other hand, expected number of objects larger than one means that multiple objects are merging together and detected as a single blob.

In this case, the process of extracting objects that are merged is done by varying a threshold value and redoing the thresholding step in that specific blob region. The threshold value is increased by one step value per iteration and there are a maximum number of iteration steps. The process of threshold varying stops when the number of blobs obtained from applying the increased threshold value equals to the expected number of object or the iteration reaches its maximum. Measurements of parameters such as area, bounding box, and center of the blob are also performed at the new extracted blobs. Fig. 4.6 shows how merging flies look like in blob image and Fig. 4.7 shows the same scene in 3-D shaded surface. The threshold level is not high enough to separate the flies. The result after the system split the merging flies is shown in Fig. 4.8 and Fig. 4.9. Note that the threshold level was increased to be just enough for separation of the blob. The splitting of merging flies only works when the flies are not overlapping too much. If they are overlapping too much, the varying of threshold will reach its maximum without reaching the expected number. This makes the program detect them as a single fly. However, in this system, multiple identities can be assigned to the same detection.

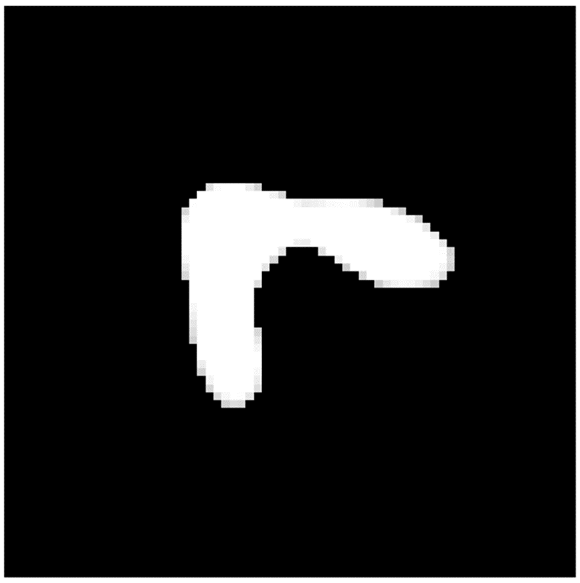


Figure 4.6: Merging flies in 2-D scene. The merging flies appears as a single white area.

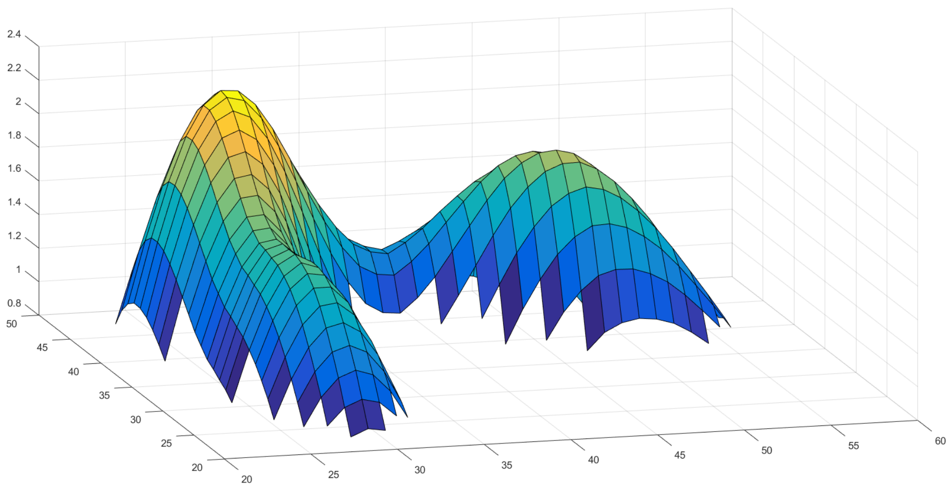


Figure 4.7: Merging flies in 3-D shaded surface. By observing the merging flies in 3-D shaded surface, this single area seems to contain two flies since there are obviously two mountains in the image.

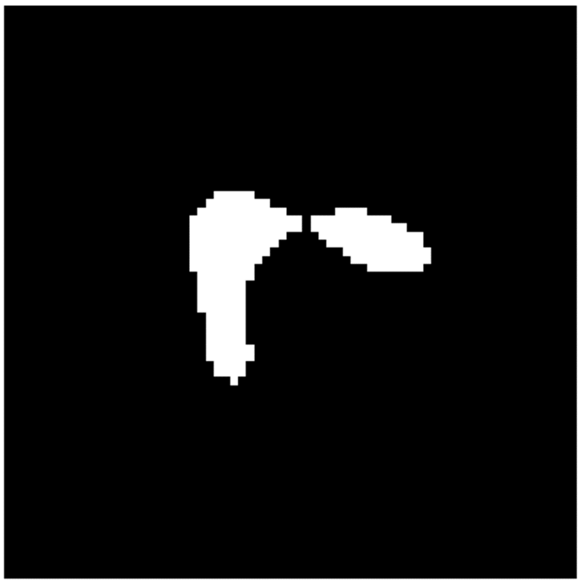


Figure 4.8: Split flies in 2-D scene. The merging flies blob from Fig. 4.6 was split by varying threshold value until the blob is split into an expected number of fly.

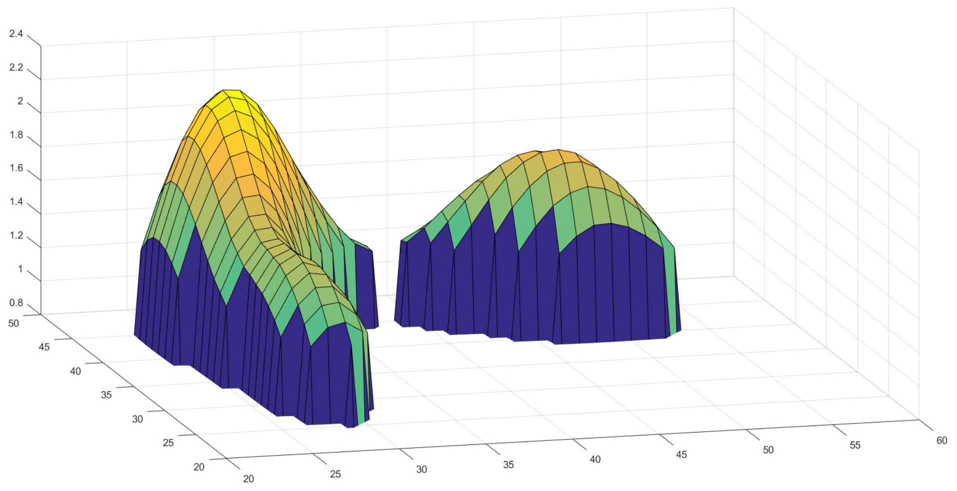


Figure 4.9: Split flies in 3-D shaded surface. The merging flies blob from Fig. 4.7 was split by varying threshold value until the blob is split into an expected number of fly.

4.3.2 Posture modeling

The posture of a *Drosophila* includes its position and heading direction. The position of each *Drosophila* is the center of its blob. Note that this has already been done in the blob analysis step. The center of a blob can be calculated by

$$\bar{x} = \frac{1}{N} \sum_{i \in \mathbb{I}} x_i, \tag{4.2}$$

$$\bar{y} = \frac{1}{N} \sum_{i \in \mathbb{I}} y_i, \tag{4.3}$$

where $\bar{x}$ and $\bar{y}$ are the center in x and y direction, respectively. The number of connected pixels that form the blob is N. Set $\mathbb{I}$ contains all indexes of those pixels.

To obtain the heading direction, orientation of each object is first approximated since the heading direction lies on the orientation vector. This is done by performing ellipse approximation to each blob and then calculating the major axis of the best-fit ellipse. Once the orientation of every blob is determined, measurement of heading direction is then performed. The ellipse approximation can be done by following Eq. 4.4 to Eq. 4.10. Eq. 4.4 to Eq. 4.6 calculate normalized second central moment for the region. The term $\frac{1}{12}$ is the normalized second central moment of a pixel with unit length.

$$u_{xx} = \frac{1}{N} \sum_{i \in I} (x_i - \bar{x})^2 + \frac{1}{12}, \tag{4.4}$$

$$u_{yy} = \frac{1}{N} \sum_{i \in I} (y_i - \bar{y})^2 + \frac{1}{12}, \tag{4.5}$$

$$u_{xy} = \frac{1}{N} \sum_{i \in I} (x_i - \bar{x})(y_i - \bar{y}) + \frac{1}{12}, \tag{4.6}$$

$$\Delta = \sqrt{(u_{xx} - u_{yy})^2 + 4u_{xy}^2}, \tag{4.7}$$

$$a = \sqrt{2(u_{xx} + u_{yy} + \Delta)}, \tag{4.8}$$

$$b = \sqrt{2(u_{xx} + u_{yy} - \Delta)}, \tag{4.9}$$

$$\theta = \begin{cases} \arctan(\frac{u_{yy} - u_{xx} + \Delta}{2u_{xy}}) & \text{if } u_{yy} > u_{xx} \\ \arctan(\frac{2u_{xy}}{u_{xx} - u_{yy} + \Delta}) & \text{if } u_{xx} > u_{yy} \\ 0 & \text{if } u_{xy} = 0 \text{ and } u_{xx} - u_{yy} + \Delta = 0 \end{cases} \tag{4.10}$$

where a and b are major and minor axes, respectively. The orientation is represented by θ. Fig. 4.10 shows an example of ellipse fitting to a blob. Fig. 4.11 shows the ellipse with the real scene. The major axis is represented by a red straight line while the minor axis is represented by a blue straight line.

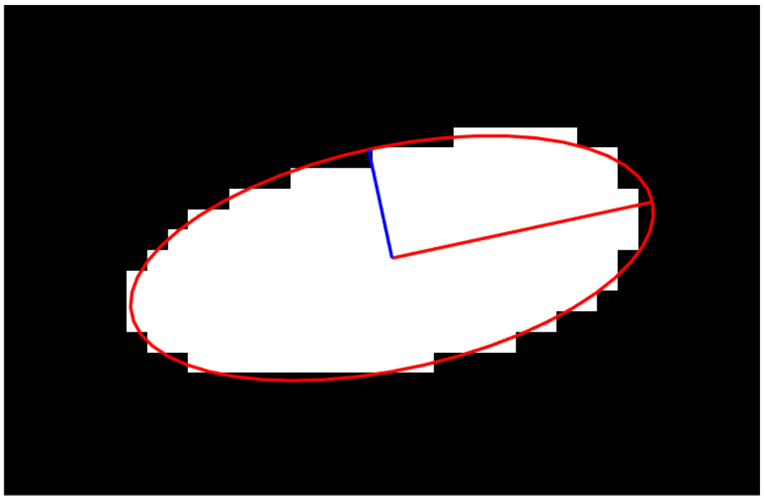


Figure 4.10: Ellipse fitting result in binary scene. The ellipse fits the filtered data in binary scene. The major and the minor axes are illustrated as the red and the blue lines, respectively.

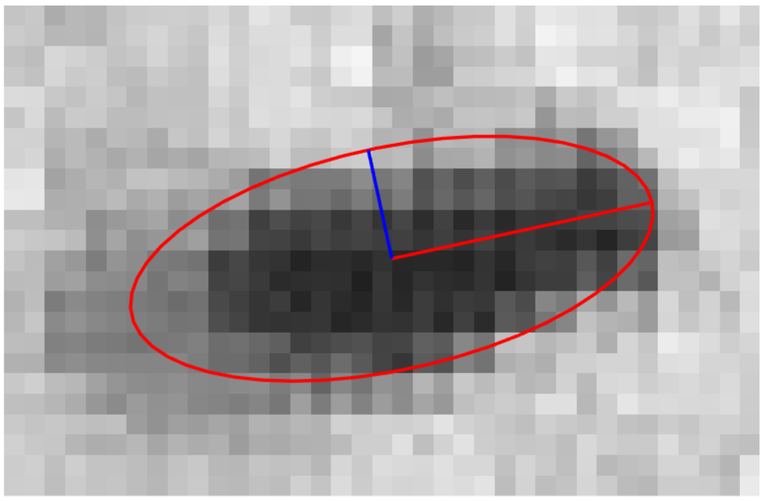


Figure 4.11: Ellipse fitting result in real scene. The major and the minor axes are illustrated as the red and the blue lines, respectively. According to the real scene, head of the fly and wings position both lie on the main axis. Being able to determine the position of the wings implies the position of the head.

According to the orientation of a blob, there are two directions of choices, one pointing in the direction of the head and another pointing in a reverse direction. The problem is how to determine which direction from the two choices is the correct one for heading direction. One method of detecting the heading direction is to detect positions of the wings, because they are always located at the opposite direction to the head. This system uses an idea that wings has an intermediate intensity between fly's body and background to distinguish them. Wing detection of a blob is done by deploying two search points to that blob along with applying a specific range of threshold couples to all the pixels in the bounding box of the blob. The two search points are decided from the points located at the end of the blob's major axis, which are normally the locations of the wings. A threshold couple is introduced to extract the wing parts. For each threshold couple, two threshold values are presented, floor value and ceiling value. When the threshold couple is applied to an image, the pixel value that is below the floor value and is above the ceiling value will be assigned with zero. On the other hand, the one with the value between the floor and ceiling values will be assigned with one. The system gives a score to each of the two search points and the score is obtained by collecting assigned values of all pixels located around each search point within a specific radius. The search point that has higher score that is above a specific value will be considered as the representation of wing position. The system also counts a score for the pixels outside the search area, noted as environment pixels. For a search point having high score with high score of environment, the search point will be rejected from being the representative and the score of that point will not be counted. In this case, what the search point found is not wing pixel but the environment pixel. The process runs by alternating various the threshold couple and collecting score for those search points for a certain iteration step. Finally, the heading direction for each blob is obtained by calculating a direction from the blob centroid to the search point that has a higher score. These position and heading direction will be used in the tracking step later on. The algorithm used for determining the heading direction is as follows.

```
**Data:** split blob image with orientation, gray-scale original image, searching
      radius $r$, environment threshold $TH_{en}$
**Result:** heading direction for each blob
create a set $\mathbb{TH}_{step}$ containing threshold steps;
create two integer variables, $A_{point}$ and $B_{point}$, and initially set them to be zero;
**for** *each blob* **do**
    create two search points, A and B, at the ends of the main axis;
    **for** *each $i^{th}$ threshold element in* $\mathbb{TH}_{step}$ **do**
        apply the $i^{th}$ threshold to the blob;
        count positive pixels within radius $r$ from point a and store in $p_A$;
        count positive pixels within radius $r$ from point b and store in $p_B$;
        count positive pixels outside radius $r$ from both point a and b and store
          in $p_{envi}$;
        **if** $p_{envi} > TH_{en} \vee p_A = p_B$ **then**
            do nothing;
        **else**
            **if** $p_A > p_B$ **then**
                increase $A_{point}$ by $p_A - p_B$
            **else**
                increase $B_{point}$ by $p_B - p_A$
            **end**
        **end**
    **end**
    compare $A_{point}$ and $B_{point}$, the point that has higher value should be in the
      direction of wings therefore, the heading direction will be from the
      centroid to the point that has lower value
**end**
```

Algorithm 2: Heading direction detection algorithm

Fig. 4.12 displays six filtered images with different threshold range. The image is the same as Fig. 4.11. Two search points are created at the end of major axis. Boundary of each search area is drown as a red circle and a green circle. The higher number of positive pixel that a search point found, the higher probability of that point to contain wings. Thus, heading direction is to the opposite. According to Fig. 4.12, wings are located at the green circle and this implies that the true head of the fly is at the red circle.

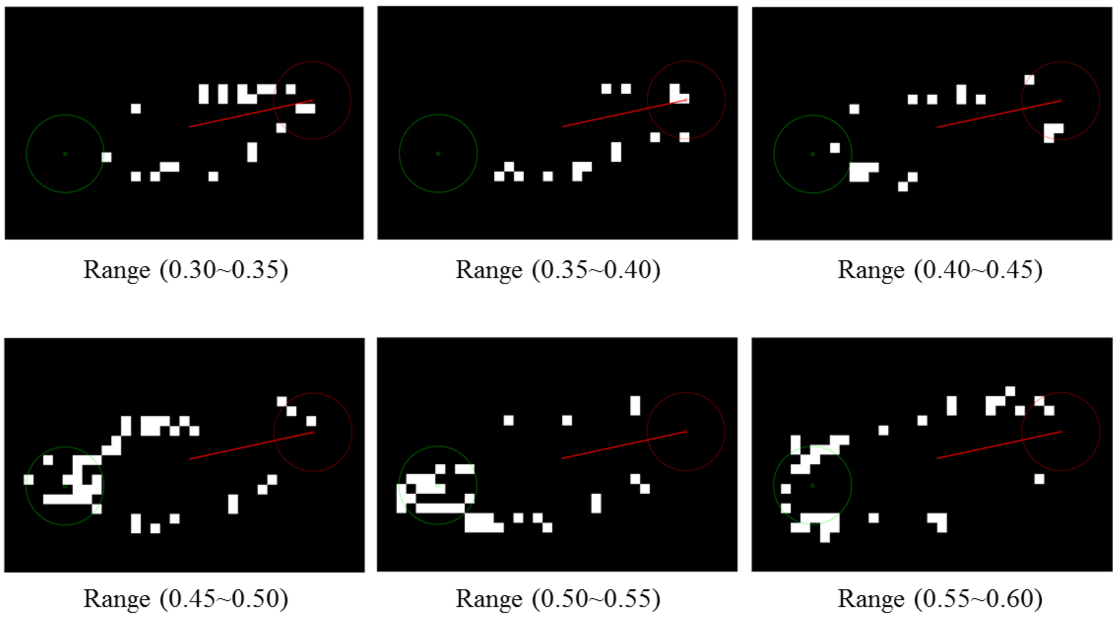


Figure 4.12: Filtered image at different threshold range level. Two search points illustrated as red and green are located at the end of the major axis. A number of threshold ranges are applied to a blob resulting in white pixels. Each search point collects a score by counting a number of white pixels located within its range. The higher number of white pixel that a search point found, the higher probability of that point to contain wings. In this figure, wings of the fly is likely to be located at the green circle.

4.3.3 Tracking

Once the moving pixels or the foregrounds are detected as blobs in each frame, the next task is to assign identities to those blobs and find their correspondence in the next frame. After a fly has been assigned an identity, the identity should be kept with the same fly throughout the frame sequences, unless it may result in identity loss that will obviously affect the accuracy of further measurements. Tracking is the process of retaining the identity of each blob. There are some situations that identity loss happens such as jumping behavior, overlapping, and crossing. These problems introduce difficulty to the tracking task and a tracking strategy that is robust to those situations is required. In this work, a Kalman Filter is used as a position predictor and velocity together with the use of Hungarian algorithm [66] to determine the best matching of identities. To solve Hungarian algorithm problem in MATLAB, we used an open-source function called "assignmentoptimal" by Markus Buehren [67]. The Kalman filter is also used for estimating the next state of the system. Unfortunately, the only prediction from Kalman Filter is not enough to deal with all the situations since the flies may suddenly move backward and this will result in tracking loss or identity swapping. Thus, we introduce a method using of heading direction to help confirming the identity matching. Exactly or nearly the same in heading direction of two blobs in consecutive frames increase the certainty that the two blobs are the same object. The pairs that are rejected from the confirmation of the identity matching will be treated with the closest neighbor assignment algorithm.

Kalman Filter

Theoretically the Kalman Filter is an estimator for linear-quadratic problem, which is the problem of estimating the instantaneous state of a linear dynamic system perturbed by white noise and the resulting estimator is statistically optimal with respect to any quadratic function of estimation error [68]. For the next consecutive state, the filter predicts state variables based on current state variables, compares and modifies with measured information from the next state, and then gives estimation of the next state variables with minimal error. In this work matching between predicted states and the measurement needs a confirmation from the Hungarian assignment algorithm and heading direction before states estimation. The estimated states will be kept as states for the next time step.

State vector and state transition matrix are as follows,

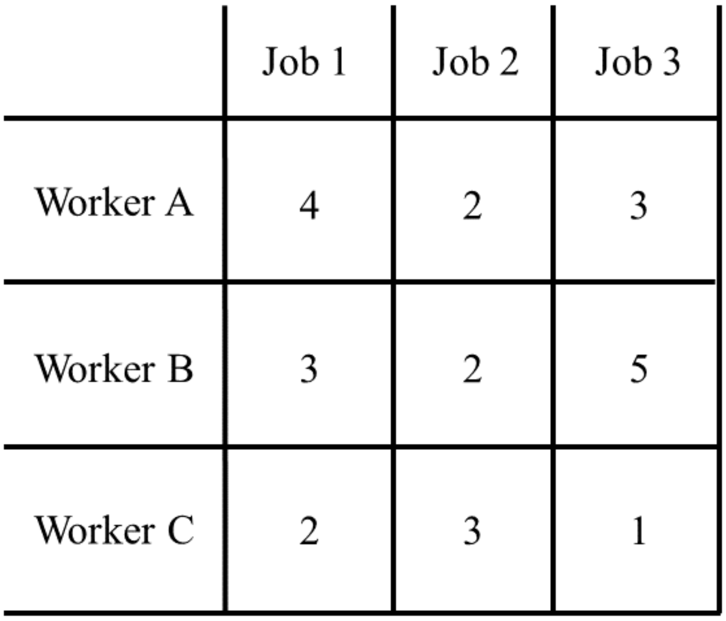
$$x_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}, A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

where v_x and v_y are velocity in x and y direction, respectively. Δt is time difference between two consecutive frames with state prediction equation as

$$x_{k+1} = Ax_k. \tag{4.12}$$

Hungarian assignment algorithm

Hungarian assignment algorithm is an optimization algorithm that solves assignment problem. Assignment problem can be considered as assigning workers with jobs. The optimum of the assignment is achieved by giving each working with suitable job that minimize overall cost. An example of cost matrix for assigning a job to a worker is shown in Fig. 4.13. In identity assignment between two consecutive frames, predicted result from detection stage in the first frame is considered as the worker while detection in the second frame is considered as the job from the cost matrix in Fig. 4.13. The cost of assigning a job to a worker is the Euclidean distance between the prediction and the detection.



	Job 1	Job 2	Job 3
Worker A	4	2	3
Worker B	3	2	5
Worker C	2	3	1

Figure 4.13: Example of cost matrix for assigning a job to a worker. This system assigns identities between two consecutive frames using predicted positions from the first frame and detected positions from the second frame as the workers and the jobs. The cost will be distance between all possible pairs of positions.

The closest neighbor assignment algorithm

The closest neighbor assignment algorithm solves assignment problem by considering the smallest cost as the highest priority. This algorithm is different from the Hungarian assignment algorithm in the way that it focuses on the individual cost instead of the overall cost. Therefore, the result of using this algorithm does not guarantee the minimum of overall cost. However, since sometimes the tracking movement is not properly estimated such as moving forward and then backward or jumping in the different direction from previous movement, which will result in failure of estimation, the closest neighbor assignment algorithm is then required to solve such a problem. Note that in this work, this algorithm will be used as a second assignment algorithm dealing with failed cases from heading direction checking after Hungarian assignment algorithm, which is the main algorithm.

The closest neighbor assignment algorithm shown in Algorithm 1 works by finding the minimum cost in the cost matrix. It takes the pair that has the smallest cost and put it into chosen pair set. The cost matrix is then modified by deleting a row and a column involving the chosen pair, thus other pairs that include elements in the pair is removed from consideration. The algorithm goes back to finding the minimum cost using the new cost matrix and this process continues until all the elements in the row or the column are removed.

Tracking process

In the tracking process, there will be a tracker following movement of each foreground object in the scene. The tracker keeps information of a foreground object like position, velocity, and heading direction. Velocity of each detection is initially set to be zero. Next, prediction step in Kalman Filter is applied to all the trackers. The predicted results will be treated as the worker in the Hungarian assignment algorithm. All detection in the next consecutive frame will be input to the assignment algorithm as the job. In this step, assignment cost matrix is created and the cost will be Euclidean distance between the worker and the job. Note that the number of workers and jobs may not be the same since some objects may disappear due to overlapping or miss-detection. This makes the assignment cost matrix non rectangular matrix that may cause the Hungarian assignment algorithm in some programs to fail. Adding dummy workers or dummy jobs makes the cost matrix to be a rectangular matrix.

After proper matching is obtained, heading direction of the tracker and its pair are checked in order to ensure that they are the same object. An angle between two directions tells the difference between them. If the angle lies within a specific range defined by user, the pair is preserved. If it does not, then the pair is rejected. The pair that includes dummy will be rejected as well.

For all pairs that are rejected, a new cost matrix is created based on those pairs. This new cost matrix is different from the previous cost matrix in the way that it uses original states as the worker instead of predicted states. After the pairing step, all the information such as position, velocity, and heading direction for the next time step will be kept in each tracker. All the processes repeat at each time step from the desired starting frame until the desired ending frame.

Fig. 4.14 and Fig. 4.15 shows the overview of the tracking process. Suppose that the current frame is k (Fig. 4.14), trackers are labeled with number from 1 to 4 at each current position. All current positions are shown as red circles. The blue cross marks are predicted positions based on current positions and their velocities. For the next frame $k + 1$ (Fig. 4.15), new positions are measured and shown as green triangles. The system then tries to pair each predicted position or blue cross mark to each measurement or green triangle by using Hungarian assignment algorithm. There will be obvious pairs of $(1', a)$, $(2', b)$, and $(3', c)$. These pairs match the state before prediction to the measurement as $(1, a)$, $(2, b)$, and $(3, c)$. However, the remaining 4 points, $4'$, $5'$, d, and e, are quite problem. If we continue without checking the heading direction, $(4', e)$ and $(5', d)$ will be the results from pairing. But actually, the tracker number 4 should be paired with measurement d and the tracker number 5 should be paired with measurement e since the heading directions make more sense.

To obtain proper pairing, heading direction needs to be checked. For those who were rejected, we apply the closest neighbor between its state before prediction (red circle) and measurement (green triangle). In this way, the result will be $(4, d)$ and $(5, e)$.

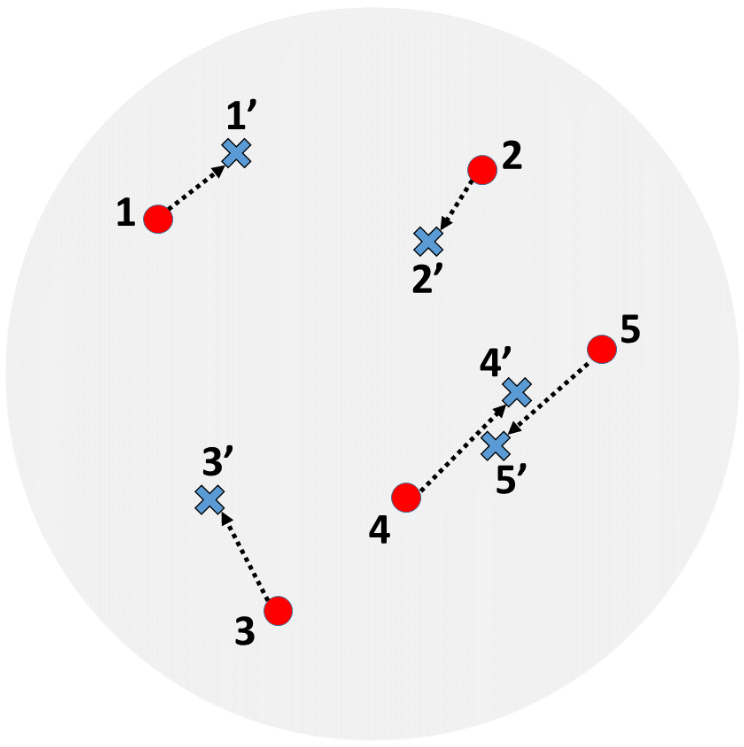


Figure 4.14: Current position with prediction. First position of each fly in the next time step is predicted by using its current position shown in red dot and current velocity shown in dotted vector.

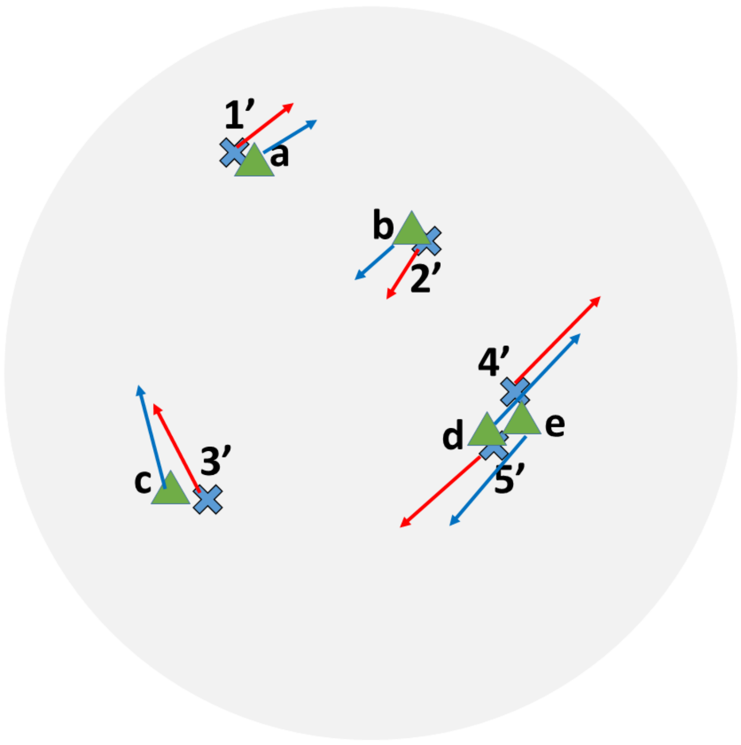


Figure 4.15: Prediction and next time step position. Detected positions in the next time step are paired with the predicted positions by using the Hungarian assignment algorithm. The result of pairing is confirmed by checking the heading directions. The system rejects the pair that has high direction difference. In this figure, the pairs {4’,e} and {5’,d} will be rejected and refined with the closest neighbor assignment algorithm.

4.4 Experiment

In this section, a video showing result of processing an example *Drosophila* is presented. The name of video is "Video_1". The example video was taken at 19 fps 827×818 resolution in the total of 1,184 frames. There are 32 flies in a closed circular arena. All flies are not marked or even have their wings cut.

According to the input video, each fly appears as a dark object in the scene. The floor of the arena appears as a brighter object. Based on this statement, the proposed system detects flies using pixel intensity, dark object will be detected as a fly. However, in some scene it is possible to have other objects, which are not the flies, appear in the scene such as small tools, foods, etc. These objects appear in dark and will affect the detection process of the system. To get rid of those unwanted objects, background subtraction is performed. Fig. 4.16 shows grayscale input image and Fig. 4.17 shows a scene after background subtraction. According to the figure, unwanted objects have been eliminated leaving only flies.

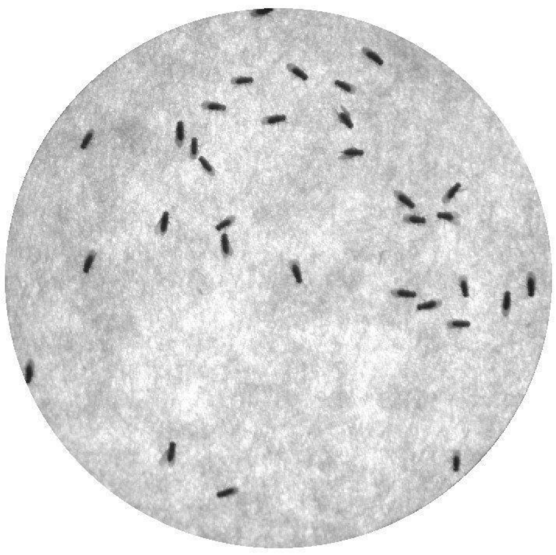


Figure 4.16: Original input image from Video_1. There are total of 32 files in a circular arena.

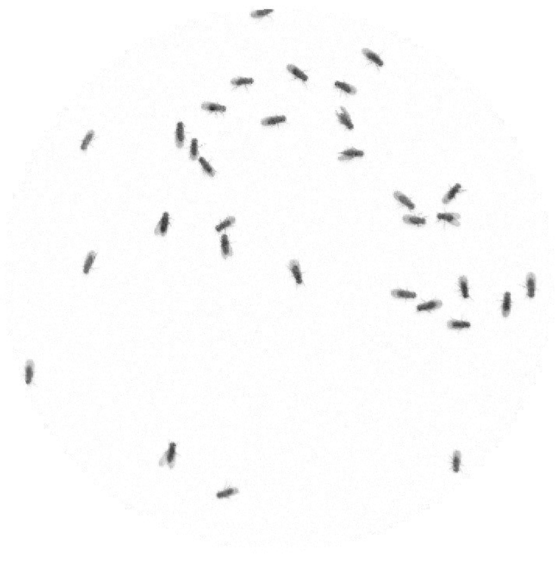


Figure 4.17: Extracted flies image from Video_1. The extracted flies image resulted from background subtraction followed by intensity reversal.

4.4.1 Blob filtering and splitting of merging flies

Fig. 4.18 shows the processed image using normal binarization and Fig. 4.19 shows the processed image using the blob filter. By comparing these two figures, less noises are presented in the case of using the blob filter. In this way, clearer appearance will provide more accurate result in posture modeling section.

Since the main situation that the system is dealing with is multiple *Drosophila* in a scene, there will be many situations where the flies merge. In case of two or more flies are located close to each other and the result is merging flies detected as a single blob. Merging flies combine positions of multiple flies together, forming a new position of the group that causes the correct position of each fly to be absent. To overcome this merging problem, the system applies various intensity thresholds to the merging area in order to separate the flies. Those separated blobs will be treated in blob analysis separately resulting in more proper position of each fly.

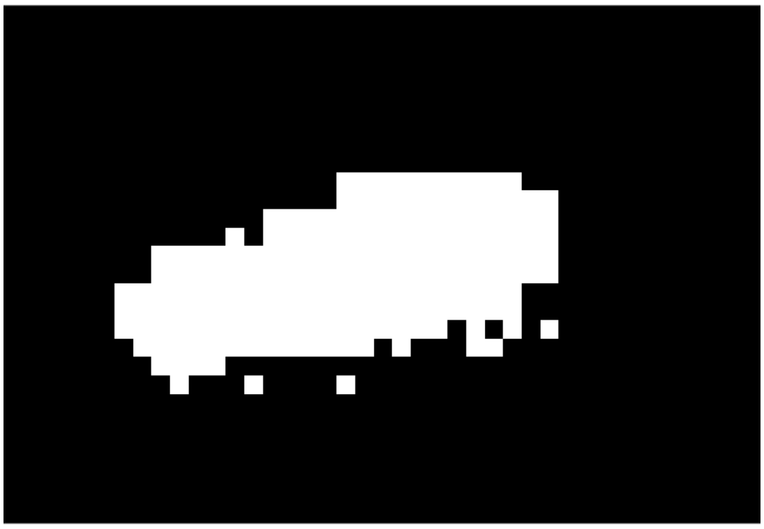


Figure 4.18: Normal binary thresholding image. The result of the normal binarization contains some noises.

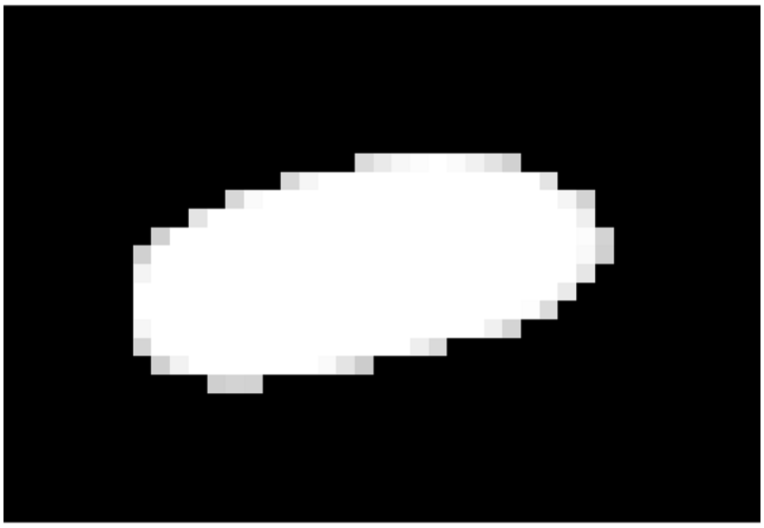


Figure 4.19: Blob filtered image. The result of the blob filter provides clear oval shape of a single fly.

4.4.2 Heading direction

Failure case

Heading direction is one of the important factors that help the tracking process. There are some situations that cause error in heading direction calculation. Fig. 4.20 and reffigfig:fail2 show two cases, fly performing side wings extension and abnormal shape, where heading direction fails. Side wings extension prevents the system from finding the position of wings regarding the main axis. Abnormal shape occurs when *Drosophila* fly and its detected heading direction becomes unknown even using human eyes to decide.

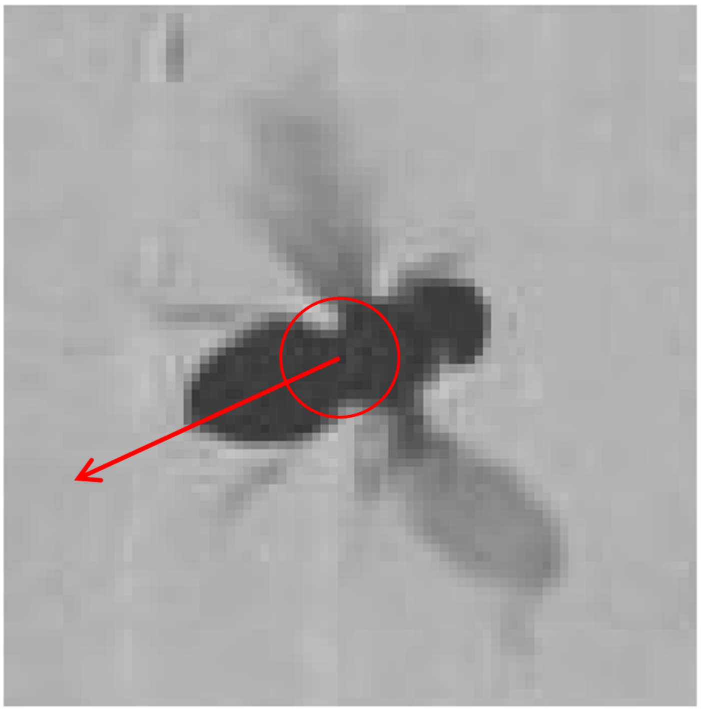


Figure 4.20: Fly performing side wings extension makes heading direction to fail. Having its wings in position different from the normal wings position causes the system to fail to determine the heading direction.

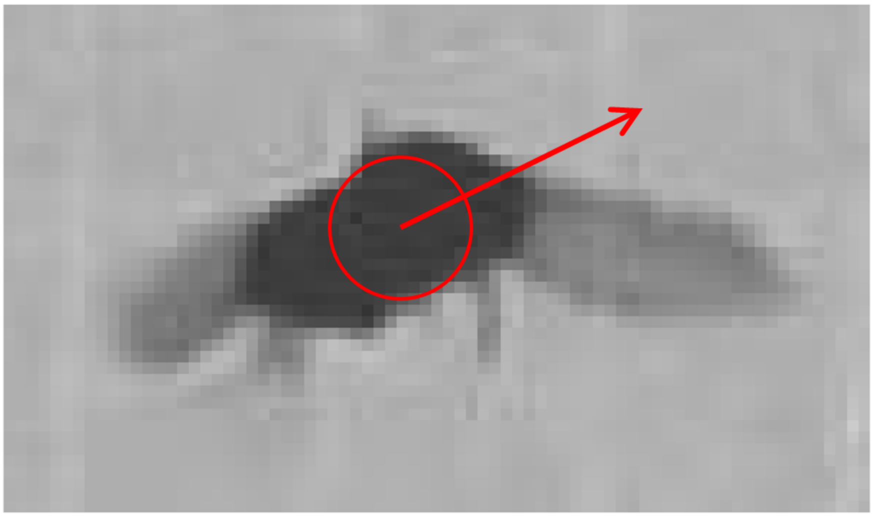


Figure 4.21: Abnormal shape detected by the camera makes heading direction to fail.

Heading direction result in real scene

Fig. 4.22 illustrates the result of the process of determining the heading direction of each fly from "Video_1". Heading direction is shown in red arrow for each fly. All the information such as position and heading direction, will be kept in each frame waiting to be used in the tracking step.

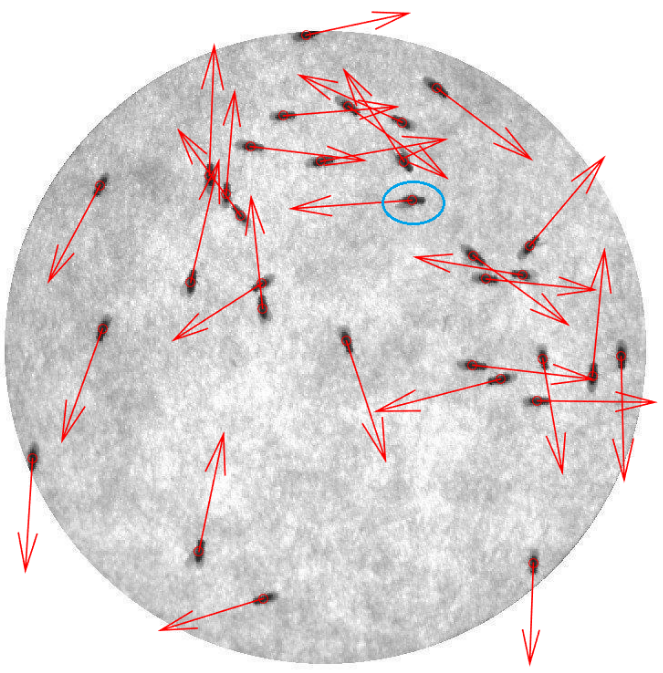


Figure 4.22: The result of determining heading direction of each fly using "Video_1". Heading direction of each fly is shown as a red arrow. Almost all heading directions were correctly determined except the fly with blue circle.

Heading direction result in various intensity videos

The system was also tested with various intensity shifting. Fig. 4.23 to Fig. 4.30 show heading direction result of eight intensity of Fig. 4.22. Having 207 mean intensity value of Fig. 4.22 as a reference, Fig. 4.23, Fig. 4.24, Fig. 4.25, and Fig. 4.26 has mean intensity shifted by +80, +60, +40, and +20, respectively. For negative mean intensity shift, Fig. 4.27, Fig. 4.28, Fig. 4.29, and Fig. 4.30 has mean intensity shifted by −20, −40, −60, and −80, respectively.

The system uses heading direction to help in identity assignment step, but the heading direction does not have to be very accurate. We are currently developing the software to be able to accurately determine the heading direction.

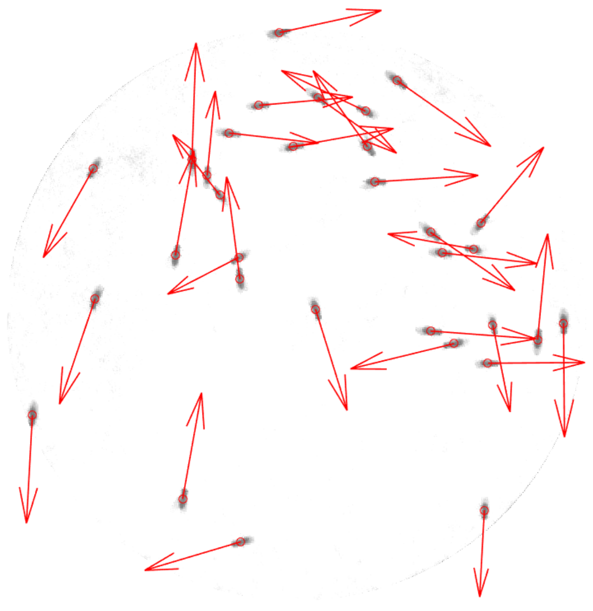


Figure 4.23: Heading direction results of +80 intensity shift.

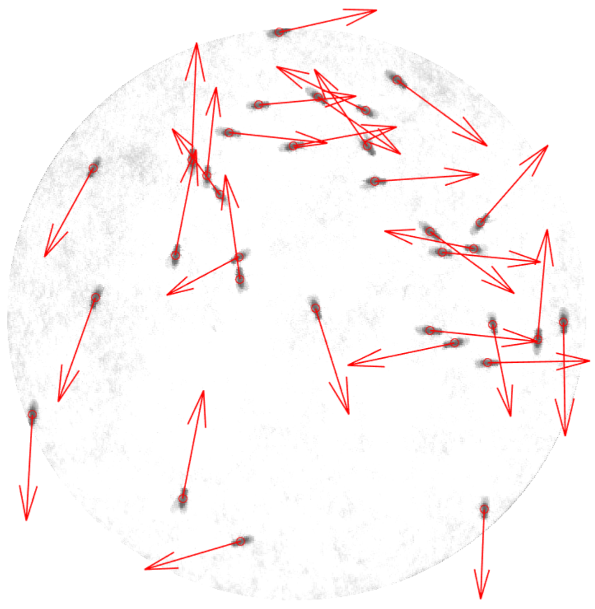


Figure 4.24: Heading direction results of +60 intensity shift.

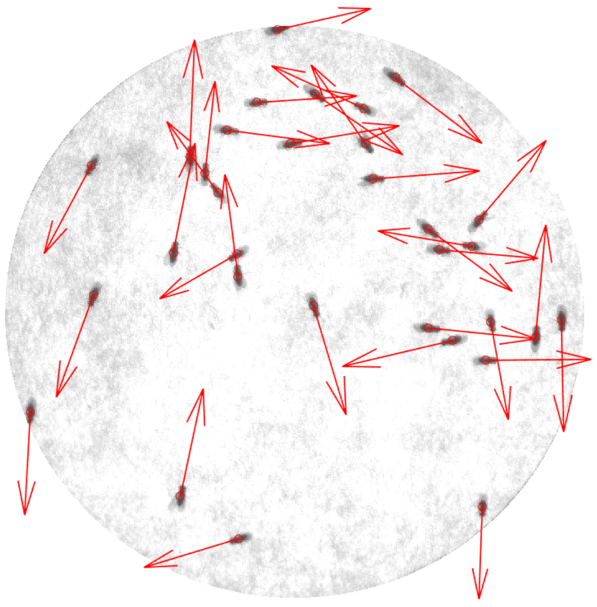


Figure 4.25: Heading direction results of +40 intensity shift.

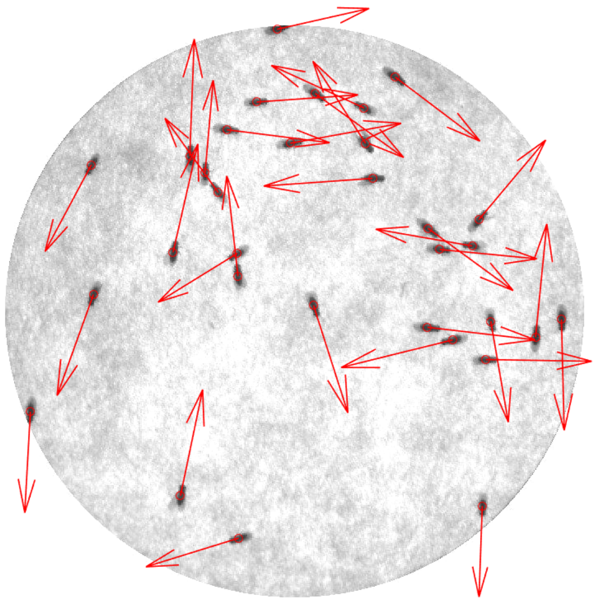


Figure 4.26: Heading direction results of +20 intensity shift.

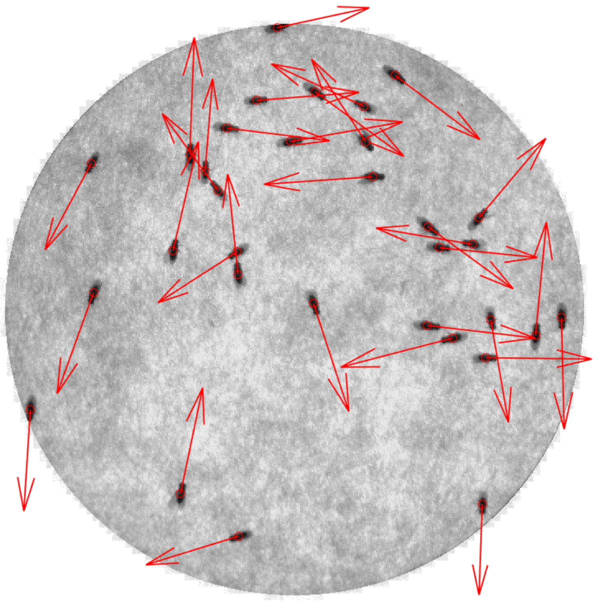


Figure 4.27: Heading direction results of -20 intensity shift.

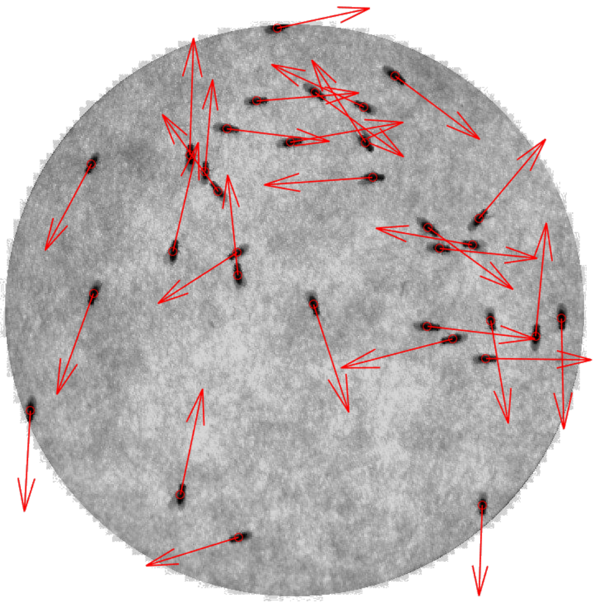


Figure 4.28: Heading direction results of -40 intensity shift.

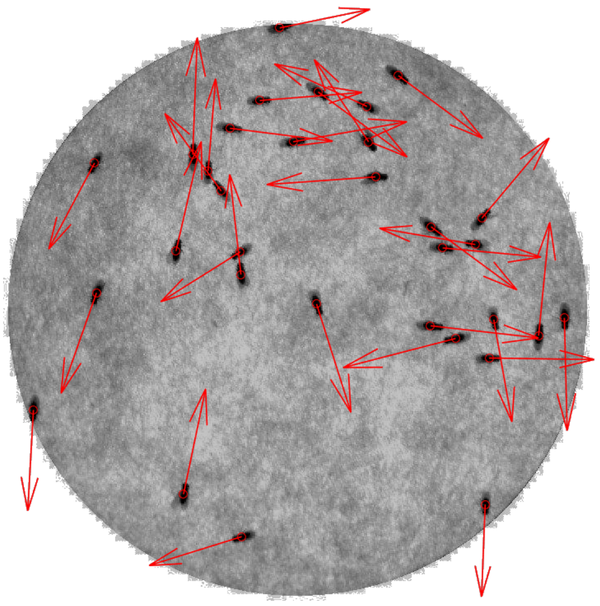


Figure 4.29: Heading direction results of -60 intensity shift.

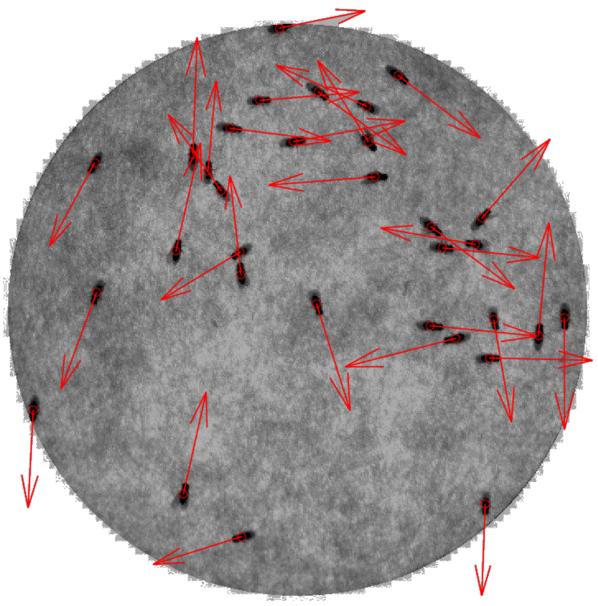


Figure 4.30: Heading direction results of -80 intensity shift.

4.4.3 Identity assignment

Fig. 4.31 to Fig. 4.34 show an example of tracking result. Each fly in the scene is labeled with number corresponding to its tracker number. Let the current frame to be frame t, trajectory of the movement of each fly from frame $t - 6$ to frame t is illustrated by colored-line. From time step k to k+3, there are 18 stationary flies (number 1, 2, 6, 7, 9, 12, 15, 18, 21, 23, 25, 26, 27, 28, 29, 30, and 31). 13 flies are detected to have small simple displacement (number 3, 4, 5, 8, 10, 11, 13, 14, 17, 20, 22, 24, and 32). There is a flying fly (fly number 16 enclosed by black dotted rectangle). At time step k, it flew from the top of the arena to the middle and to the bottom right at time step k+1. The next time step, it went a bit upward and finally flew back to the middle of the arena at time step k+3. Each time step is different from its consecutive frame by 0.0526 seconds. According to the figure, all identities are assigned properly to all *Drosophila*. In case of flying fly (fly number 16 that is enclosed by black dotted rectangle), it can also be tracked by this system.

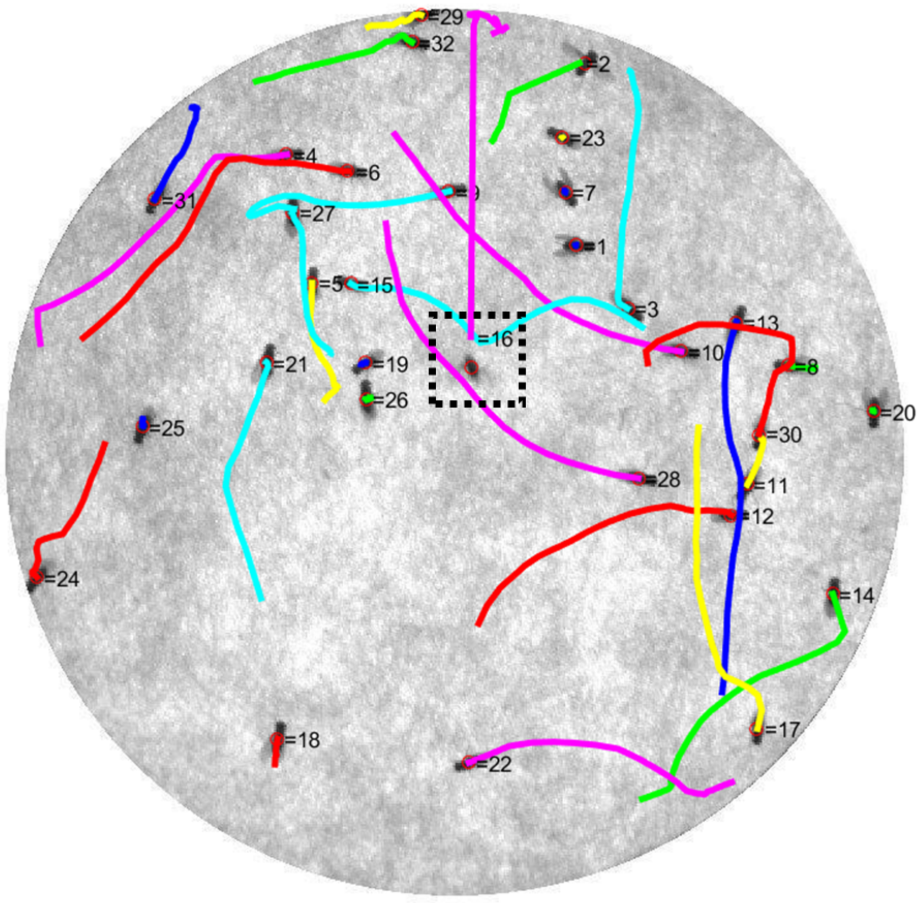


Figure 4.31: Trajectories images showing the result from identity assignment at time step k. The colored line represents each trajectory from last 6 time steps. A flying fly is detected (fly number 16 enclosed by black dotted rectangle). At this time step, it flew from the top of the arena toward the middle of the arena.

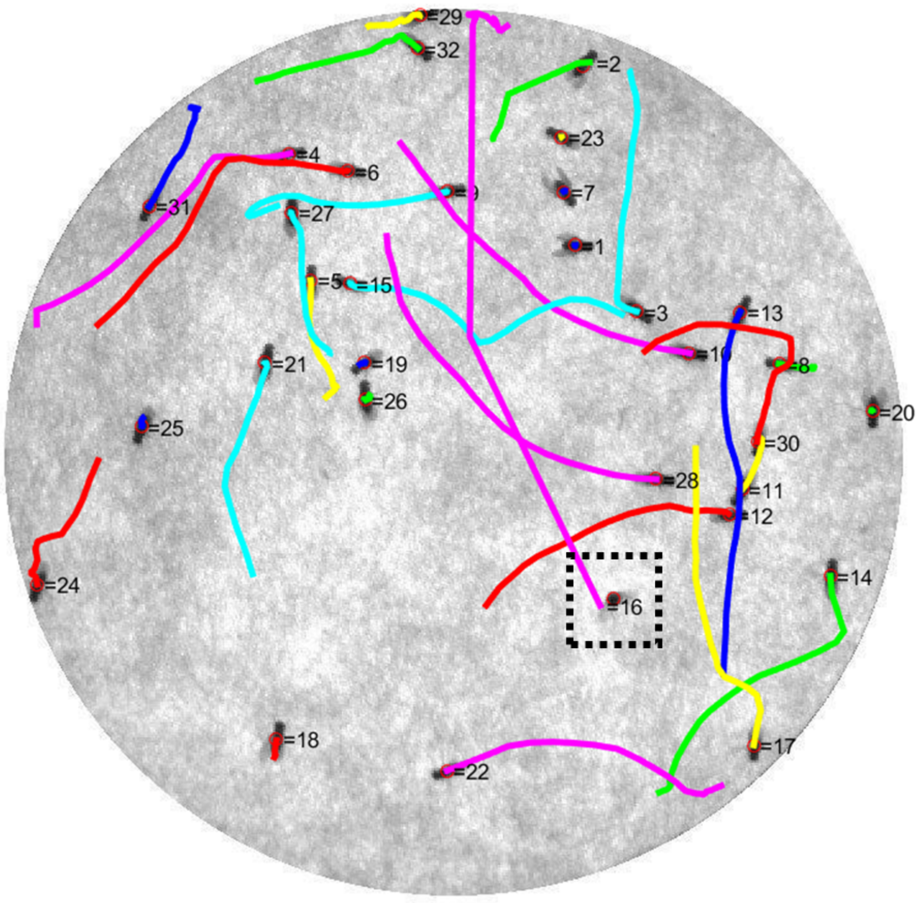


Figure 4.32: Trajectories images showing the result from identity assignment at time step k+1. The colored line represents each trajectory from last 6 time steps. At this time step, the flying fly (fly number 16 enclosed by black dotted rectangle) flew to the bottom right of the arena.

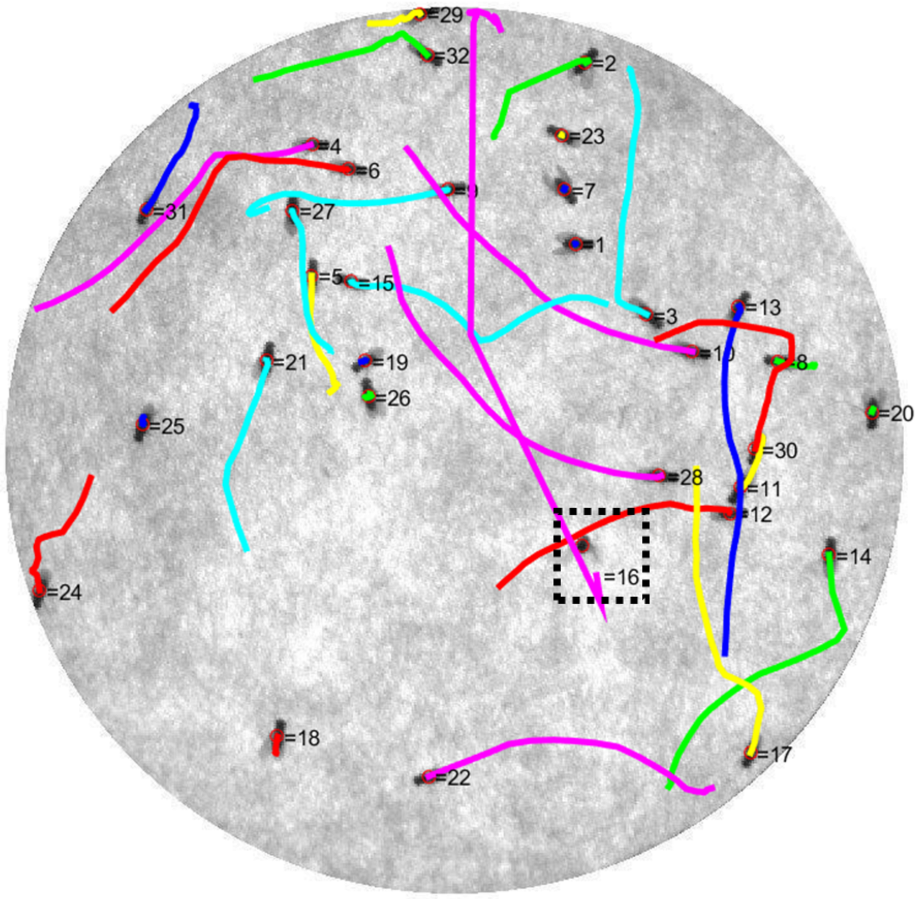


Figure 4.33: Trajectories images showing the result from identity assignment at time step k+2. The colored line represents each trajectory from last 6 time steps. At this time step, the flying fly (fly number 16 enclosed by black dotted rectangle) went a bit upward.

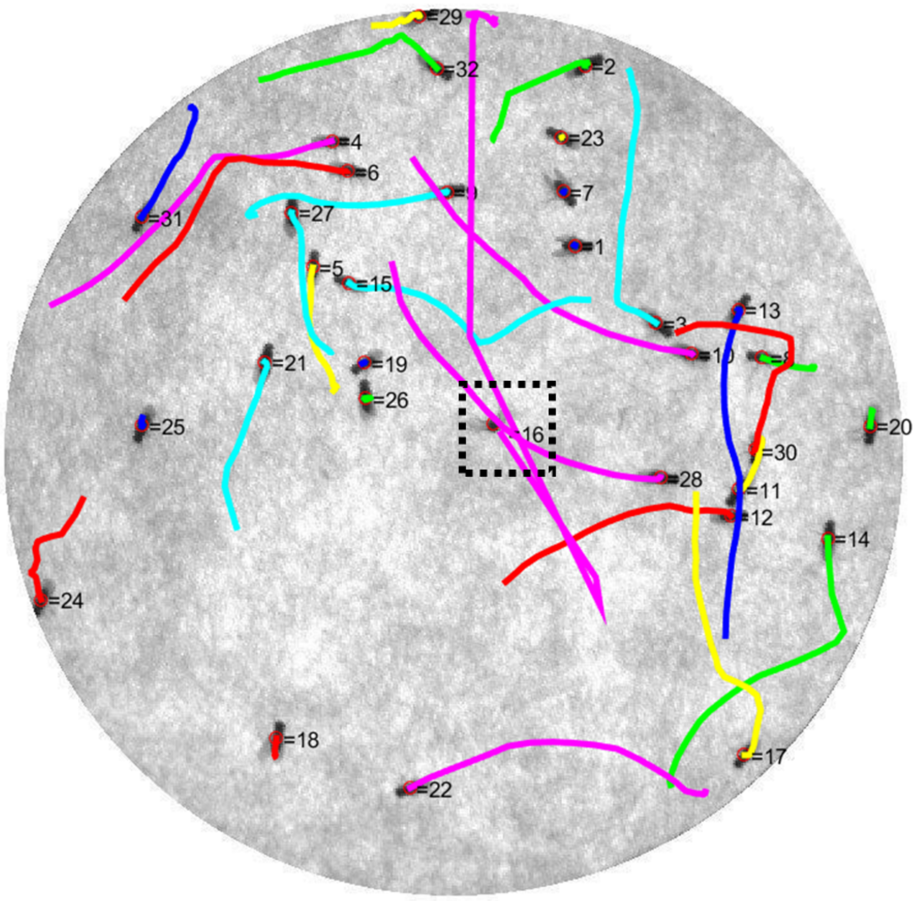
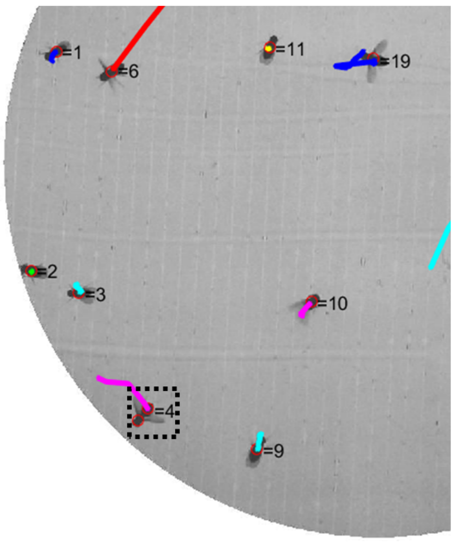
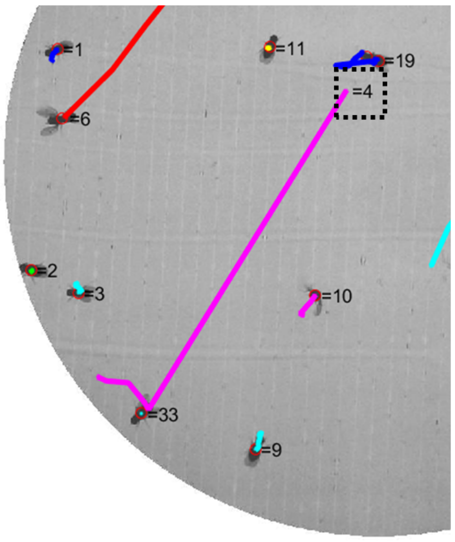


Figure 4.34: Trajectories images showing the result from identity assignment at time step k+3. The colored line represents each trajectory from last 6 time steps. At this time step, the flying fly (fly number 16 enclosed by black dotted rectangle) finally flew back to the middle of the arena.

Fig. 4.35 and Fig. 4.36 show examples of identity loss and identity swapping. Identity loss usually occurs when detection failed. This causes the tracker to incorrectly track the fly with noise from failed detection. For identity swapping, it sometimes occurs when the fly flies over other fly with failed heading direction.

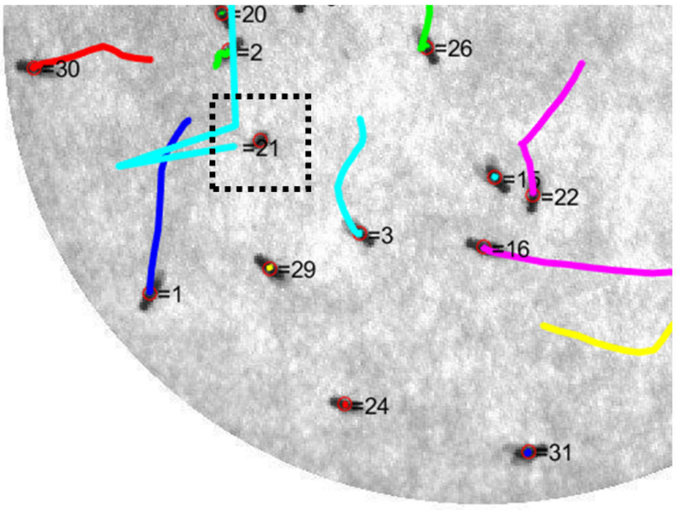


(a) Identity loss 1st frame

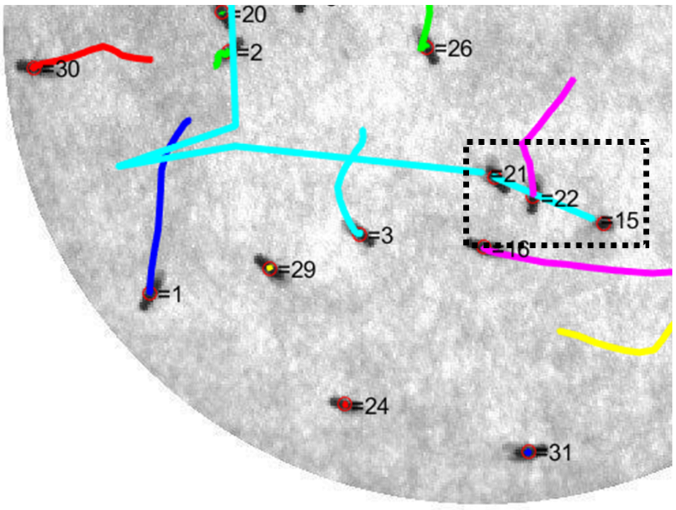


(b) Identity loss 2nd frame

Figure 4.35: Examples of identity loss scene. Identity loss occurred at fly number 4 in (a) and it lost in (b). New tracker number 33 was assigned to the fly instead.



(a) Identity swapping 1st frame



(b) Identity swapping 2nd frame

Figure 4.36: Examples of identity swapping scene. Identity swapping is shown in (a) and (b). Fly number 21 was flying (a) and passing fly number 15 in the next frame. Their identities swapped as seen in (b).

Tab. 4.1 shows detail of videos used for efficiency analysis of this tracking system. There are total of five videos shown in the table. The provided video "Video_1" has resolution of 827 × 818 and was taken at 19 fps. Other video (Video_2 to Video_5) were taken at 20 fps with resolution of 593×592, 593×592, 980×980, and 1031×1031, respectively. For Video_1 which is the main example in this part, total number of frames is 1184 with 32 flies in a closed arena. Average velocity in each video is represented by $v_{\mathbf{av}}$. This average velocity shows how active the flies are in each video.

Table 4.1: Detail of each video. The first column is video name. The second column shows number of frames with number of fly per frame. The third column shows average velocity.

Name	**Frames (flies/frame)**	$v_{\mathbf{av}}$**[pixel/second]**
Video_1	1184 (32)	93.78
Video_2	1500 (26)	13.34
Video_3	1043 (25)	22.63
Video_4	987 (27)	37.48
Video_5	1269 (31)	34.07

Tab. 4.2 shows efficiency of this tracking system. The "Crossing" column from the table contains the number of crossing event that occurred in each video. The crossing event is an event that two or more flies overlap, causing one detection. All errors were observed by eyes for all frames. Swapping error obtained from counting the number of identity swapping occurred throughout all the frames. The number in the parentheses is swapping error caused by the crossing event. Identity loss error is the cumulative number of loss of identity events. It counts when an identity leaves

its current fly. The last column of the table represents the number of false positive (FP), detection of non-fly object and additional detection due to the error of blob splitting. For this tracking system, there are 10 flies swapping. No identity loss and FP throughout all the frames in Video_1 to Video_4. The identity loss in Video_5 occurred because of failure in fly detection stage. It can be seen from Fig. 4.35 that additional detecting result was added to the fly number 4 and the identity number 4 was paired with another point in Fig. 4.36. This is a problem to this program since it uses a certain value of fly size to split a single blob of multiple flies. There will be no problem in a scene of multiple flies whose size are about the same, but it will introduce errors in a scene of flies varying in size.

Table 4.2: Error of flies swapping. The first column is video name. The second column is the number of crossing event that occurred in each video. The third column shows the total number of swapping occurrence with the number of swapping from crossing event in the parentheses. The fourth column represents the number of loss of identity events. The last column is the total number of false positive.

Name	**Crossing**	**Swapping (Cross-swap)**	**Loss**	**FP**
Video_1	5	10 (4)	0	0
Video_2	7	2 (0)	0	0
Video_3	14	7 (7)	0	0
Video_4	16	3 (3)	0	0
Video_5	16	13 (6)	4	472

All videos were also tested with C-Trax with automatic setting offered by the program. The tracking results failed in every video input. All the resulting videos by

C-Trax are available online at [69]. Software of the proposed system was written in MATLAB programming language. The software is called "TPro", running on Windows OS. The requirement for using Tpro is just installation of MATLAB Runtime that is a standalone set of MATLAB shared libraries. The user can define specific range of processing frames. Only a threshold value is required from the user to set. The program also provides an option for the user to try various threshold values and choose a proper value to use in fly detection step. All supplementary videos, software, and examples of the proposed *Drosophila* tracking system including resulting videos are also available online at [69].

4.5 Conclusion and Discussion

In this work, a system for detecting and tracking a group of *Drosophila Melanogaster* is introduced. The system can get rid of unwanted object besides the flies themselves. This system can deal with situation where flies are within close proximity of one another resulting in merging of their position while having the ability to separate them apart. It also provides accurate result of posture modeling in terms of position and heading direction of each individual fly. The system uses Kalman filter in identity assignment and tracking part to refine measured position and to obtain velocity. It uses predicted position along with the Hungarian assignment algorithm to get the match of each individual fly for the next frame. Moreover, the closest neighbor assignment algorithm is used for the case of heading direction mismatch resulting in more proper matching.

Instead of doing the detection and the tracking instantaneously at each frame, the system performs detection to all the frames first then the tracking. This makes it easy for the user to check and to refine the detection result. Furthermore, same detection result can be used with different methods in the tracking part. This saves the user a lot of time when working with many video files.

Chapter 5

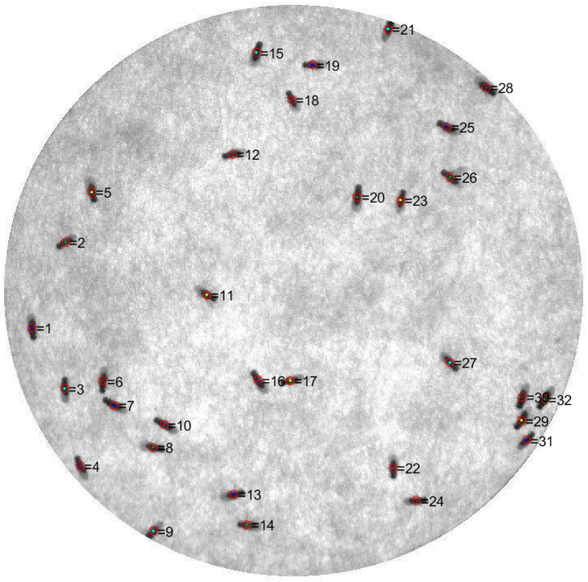
Behavior analysis of fruit fly

In this chapter, the output data from fly tracking in chapter 4 is used to classify simple behavior of fly such as stopping, walking, and jumping. All 1,184 frames of Video_1 in chapter 4 were chosen to be analyzed. To show the efficiency of data provided by the tracking program, manual tracking and classification performed by human is used to compare with random forest classification of tracked data by the program. The random forest classification is one of the classification techniques in machine learning tasks. The method consists of a number of decision trees that are constructed based on random data from a set of training data.

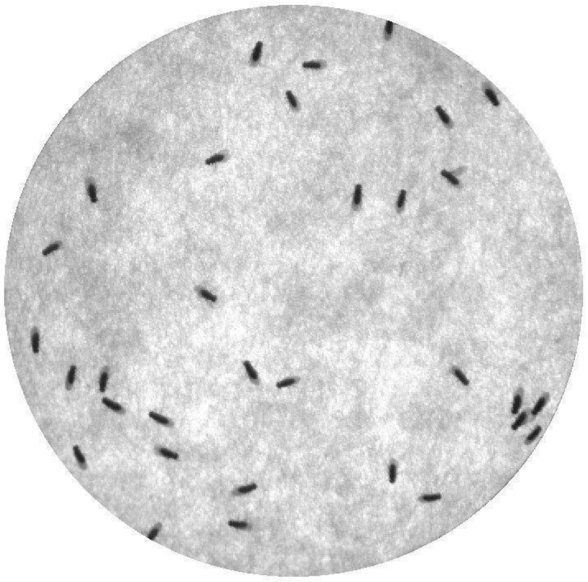
5.1 Manual tracking and classification

In this part, tracking and classification result done by human are shown. Fig. 5.1 illustrates the first and the second frames seen by the human. At each frame, the human has to decide the behavior of a fly that he/she is tracking. Fig. 5.2 to Fig. 5.9 show the result of classification done by human of fly number 1 to 32, respectively

The vertical axis in Fig. 5.2 to Fig. 5.9 represents the behaviors as 1: stopping 2: walking and 3: jumping.

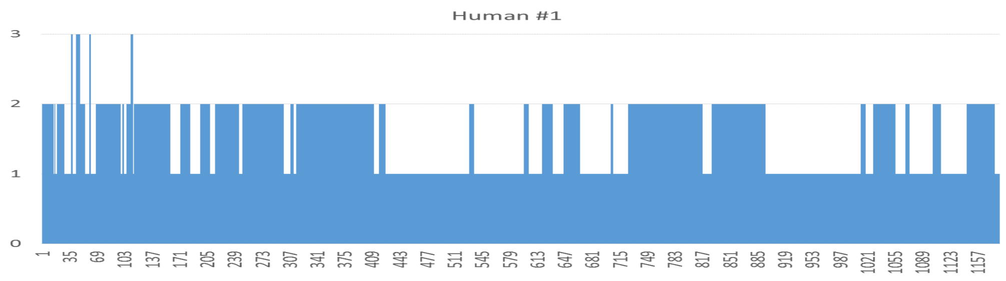


(a)

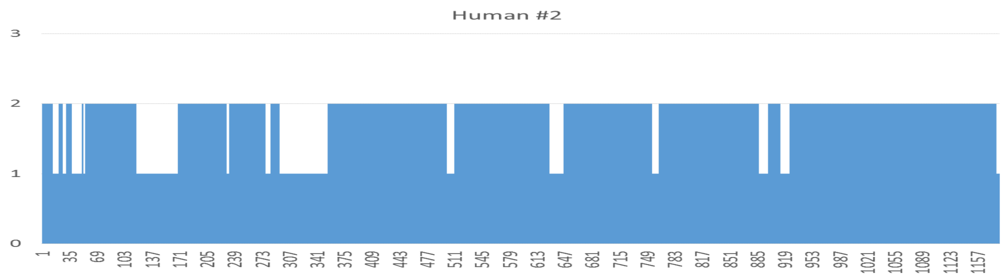


(b)

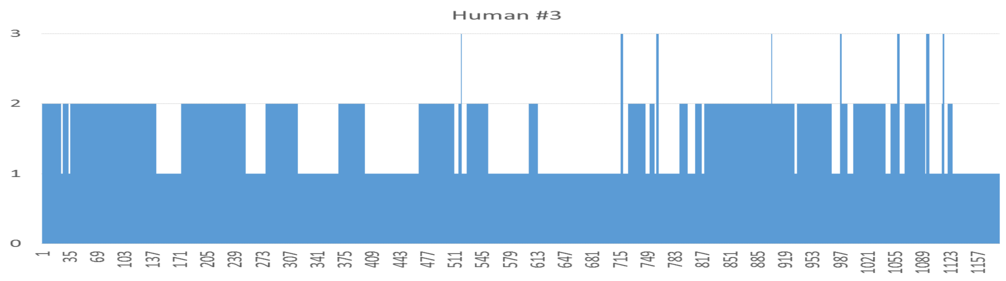
Figure 5.1: (a) The first and (b) the second frame seen by human. All flies have number label in order to identify them. From the second frame to the last frame, there are no labels and trajectories.



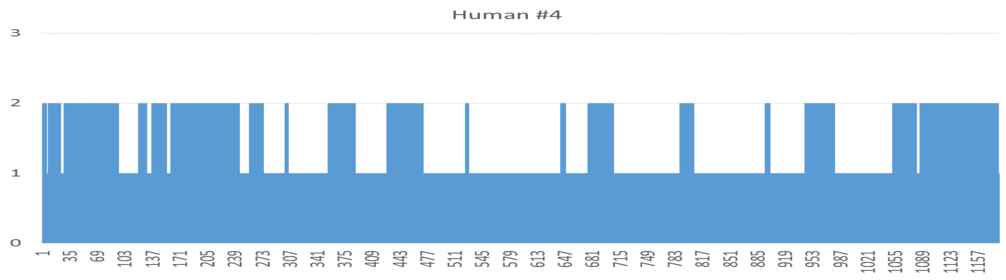
(a)



(b)

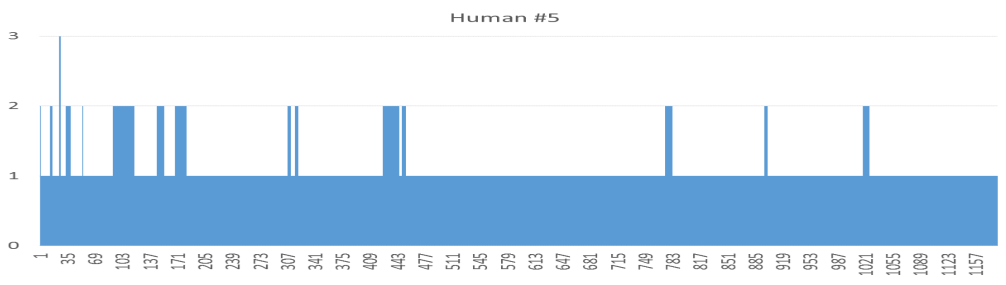


(c)

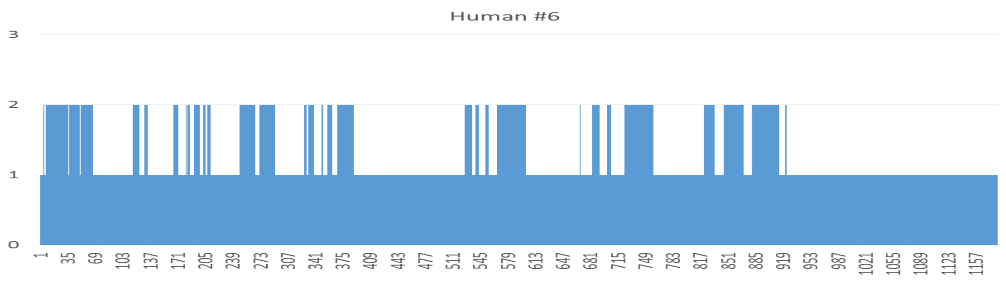


(d)

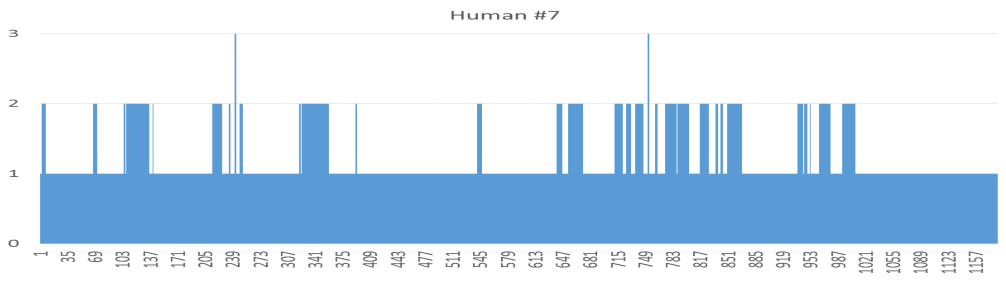
Figure 5.2: Result of behavior classification performed by human for fly number 1 to 4.



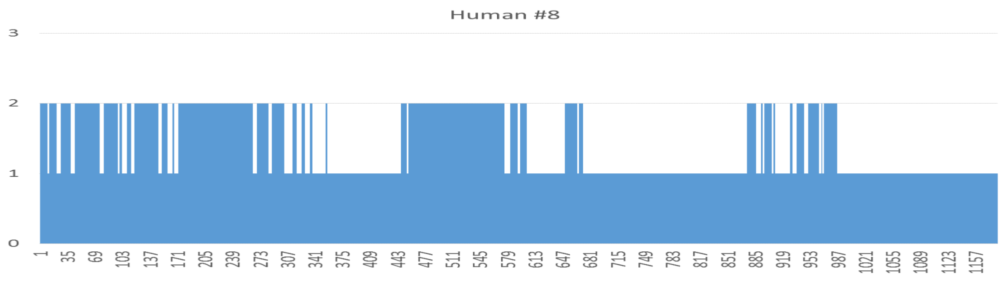
(a)



(b)

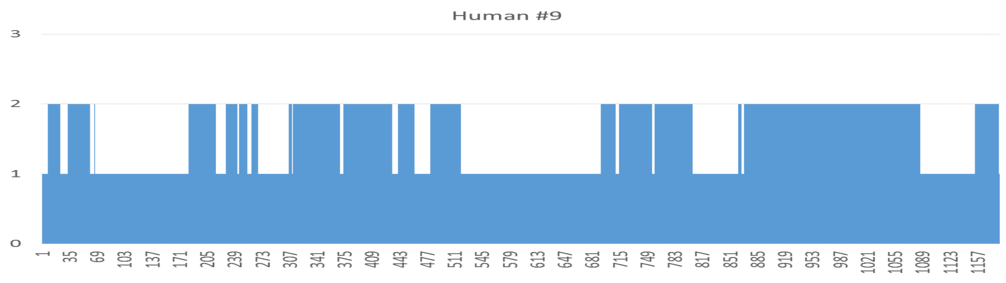


(c)

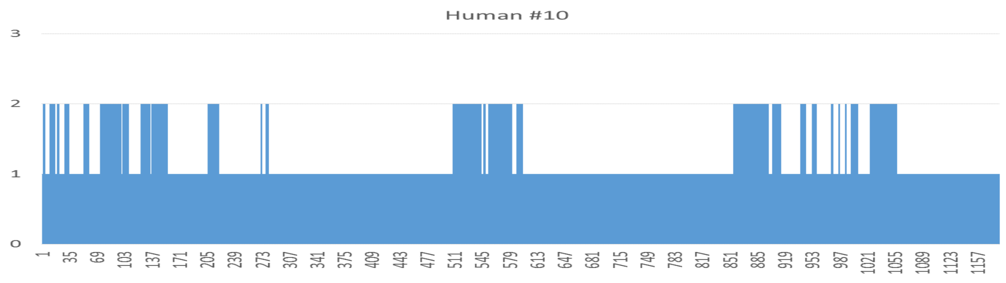


(d)

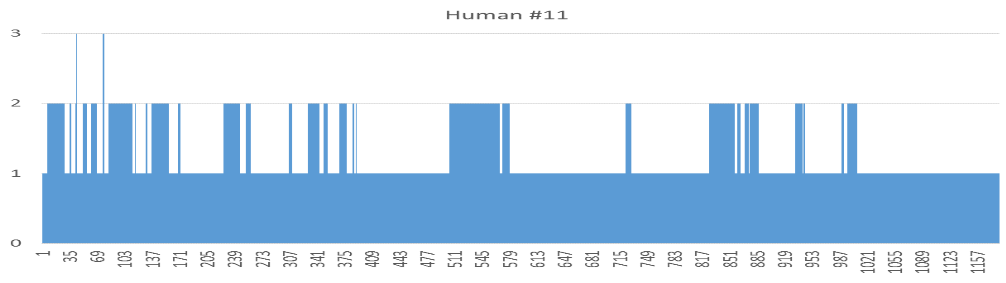
Figure 5.3: Result of behavior classification performed by human for fly number 5 to 8.



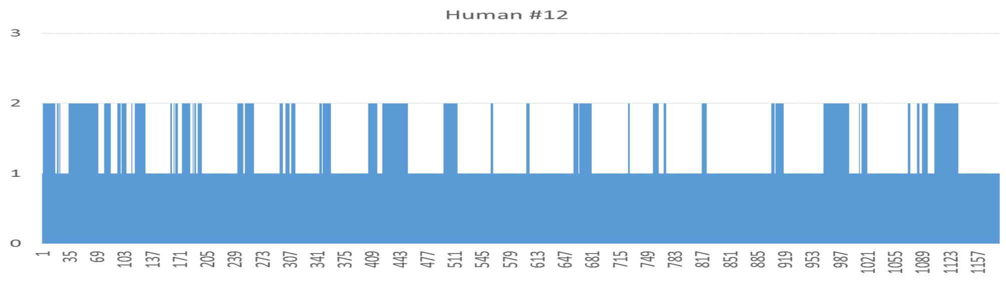
(a)



(b)

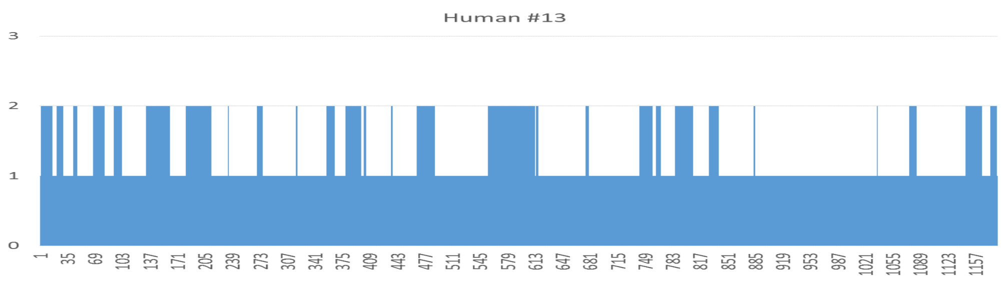


(c)

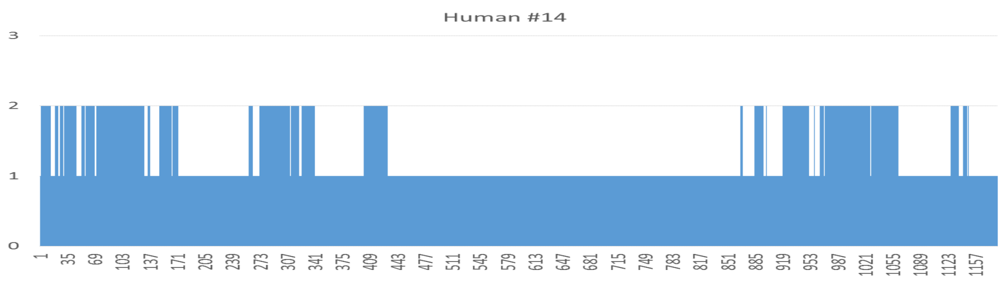


(d)

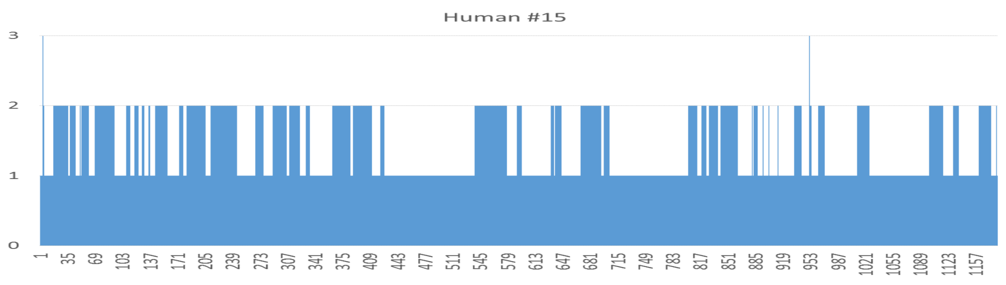
Figure 5.4: Result of behavior classification performed by human for fly number 9 to 12.



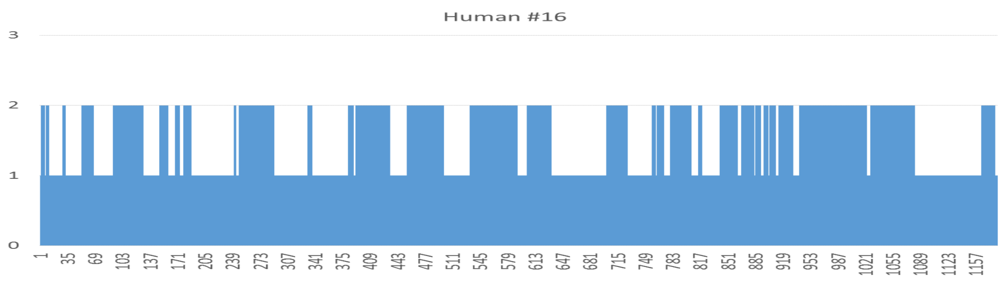
(a)



(b)

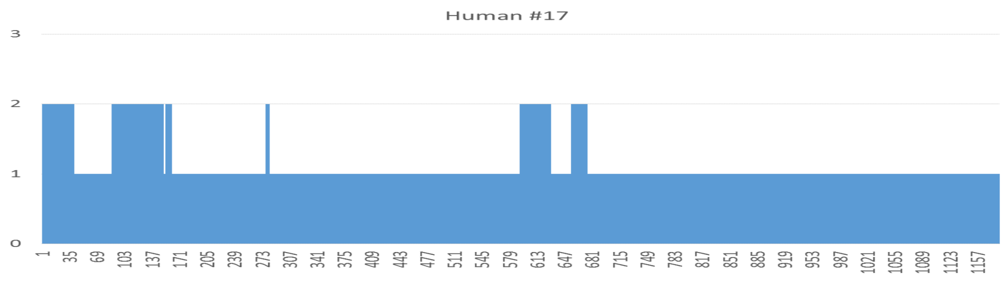


(c)

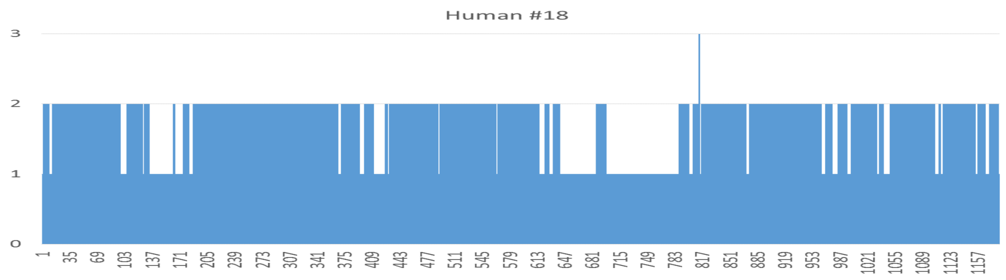


(d)

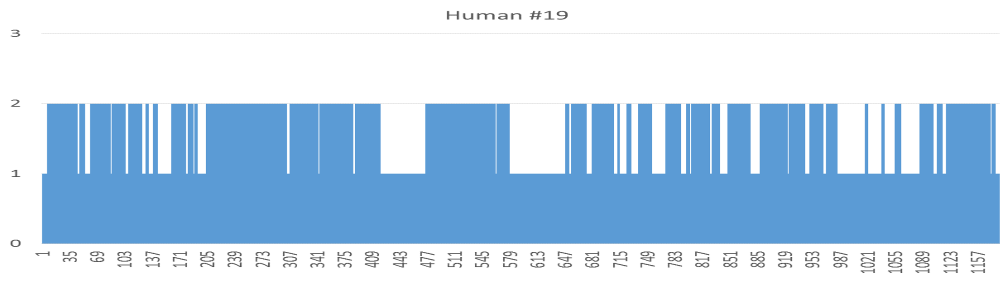
Figure 5.5: Result of behavior classification performed by human for fly number 13 to 16.



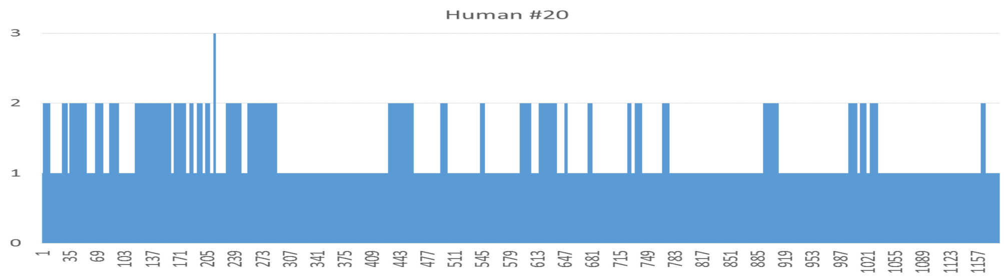
(a)



(b)

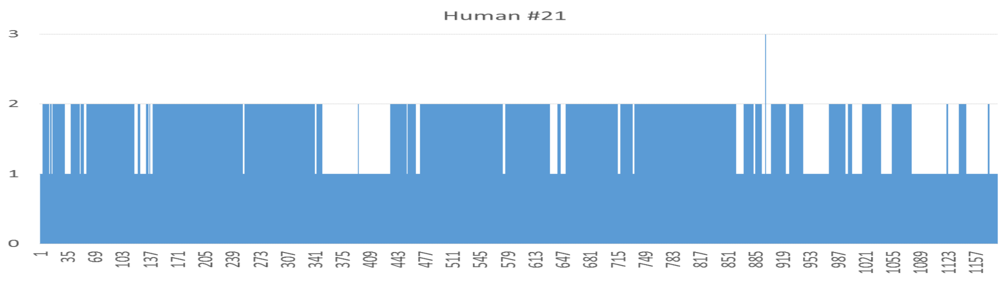


(c)

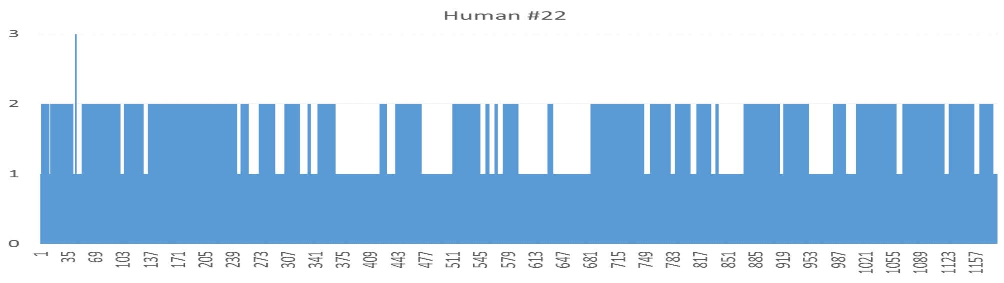


(d)

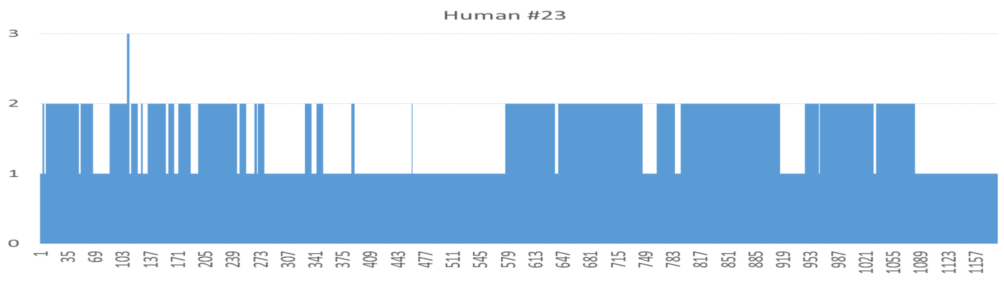
Figure 5.6: Result of behavior classification performed by human for fly number 17 to 20.



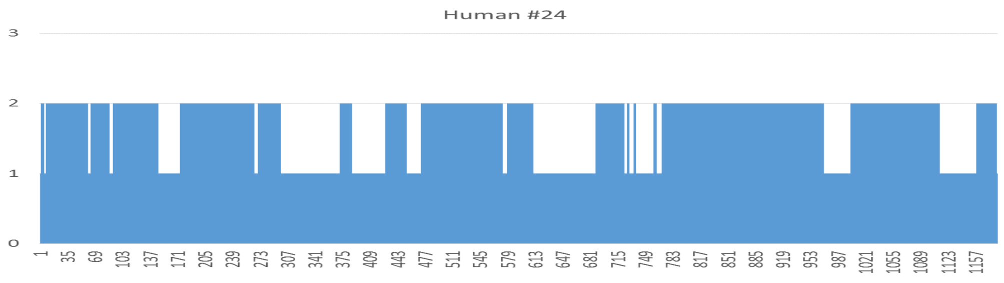
(a)



(b)

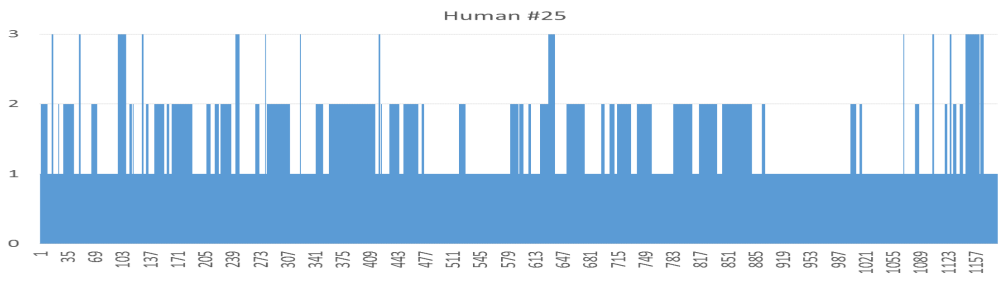


(c)

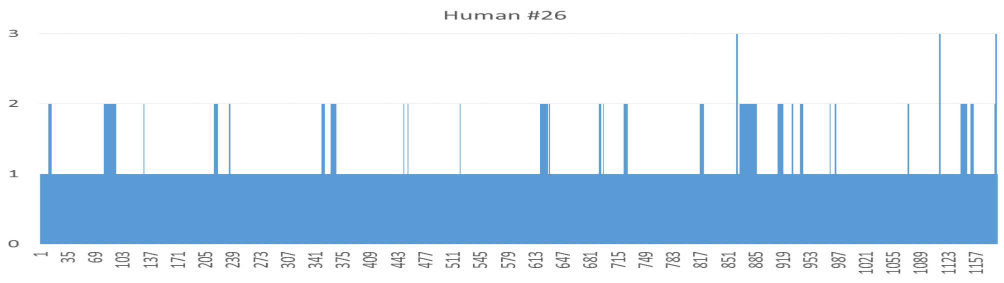


(d)

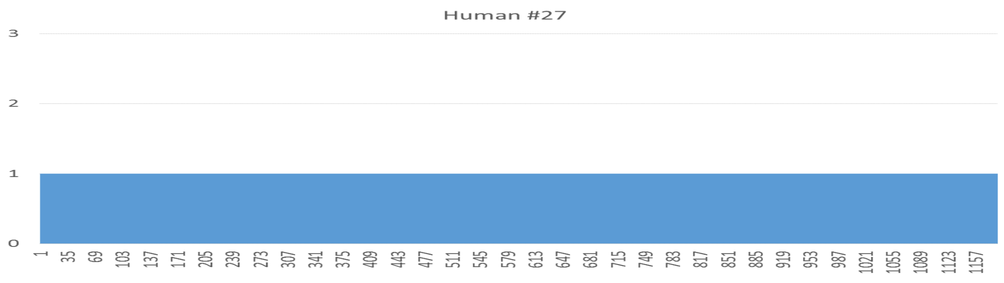
Figure 5.7: Result of behavior classification performed by human for fly number 21 to 24.



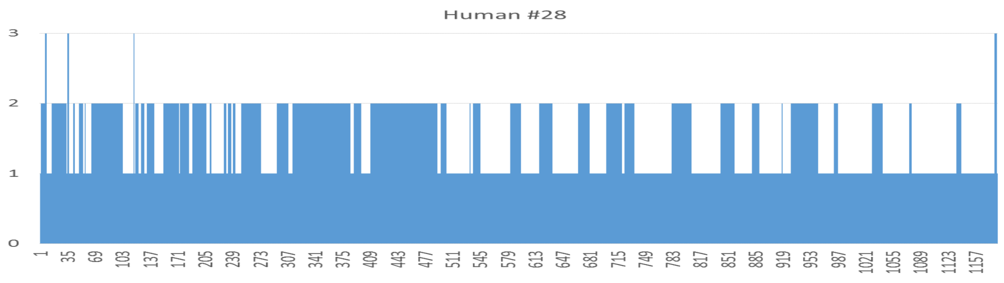
(a)



(b)

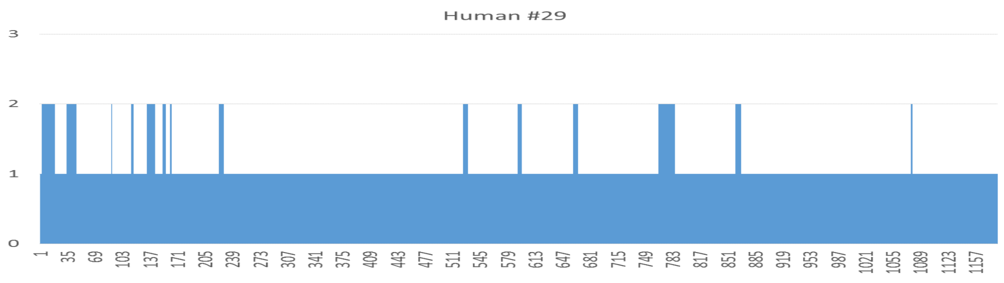


(c)

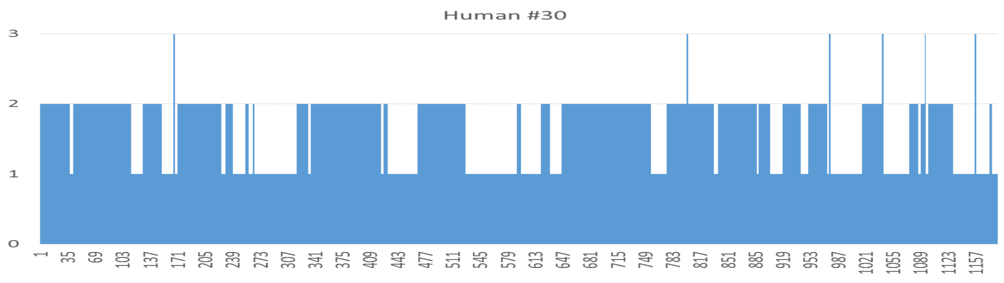


(d)

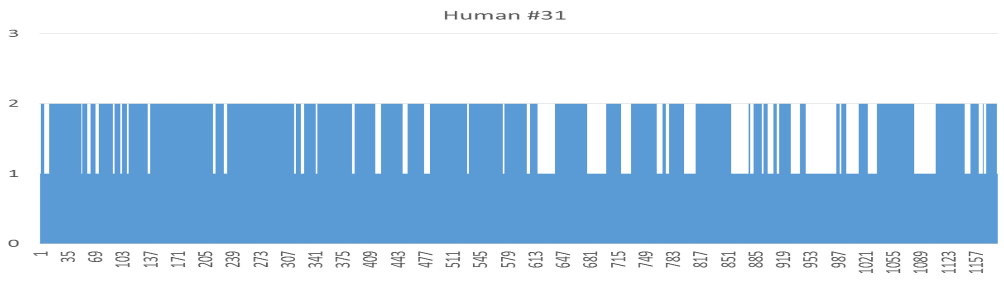
Figure 5.8: Result of behavior classification performed by human for fly number 25 to 28.



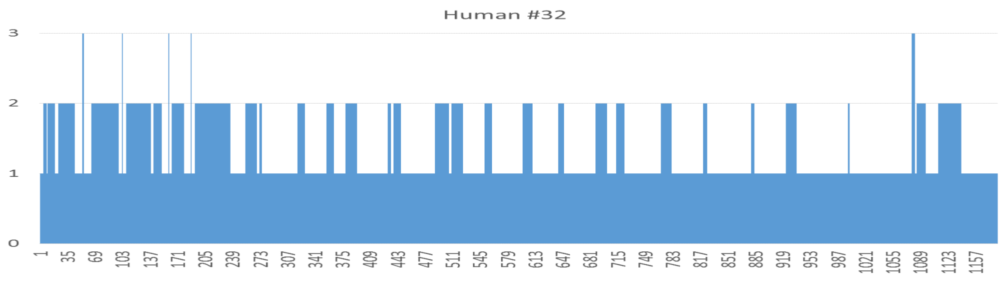
(a)



(b)



(c)



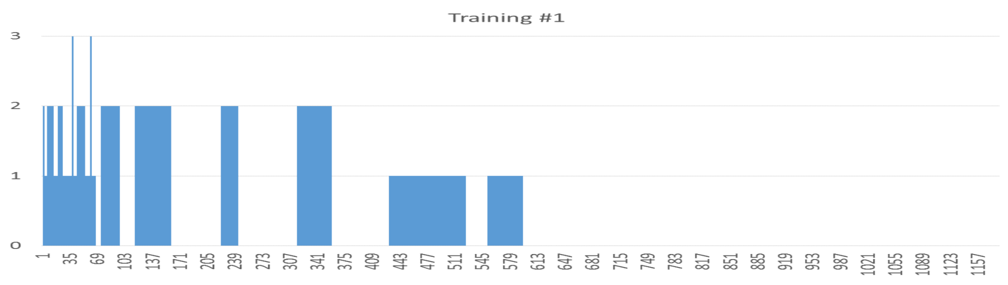
(d)

Figure 5.9: Result of behavior classification performed by human for fly number 29 to 32.

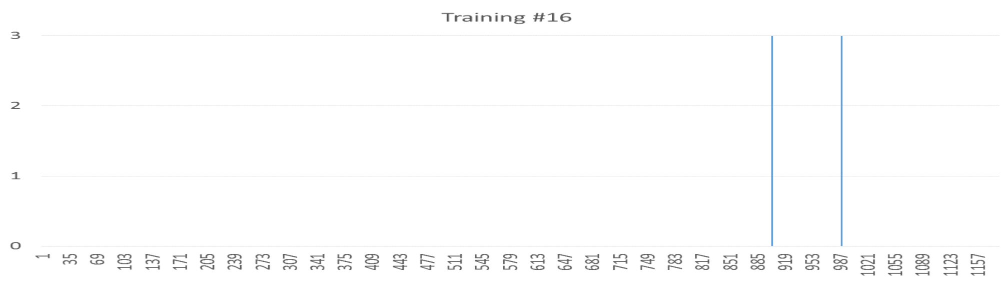
5.2 Random Forest Classification

To analyze the behavior of the Drosophila in each frame, random forest or random decision forest is chosen as the classification tool. Only the velocity in x and y direction are considered as the features. Behavior classes are the same as those in previous section as 1: stopping 2: walking and 3: jumping. To create a classification forest, training data is required. The training data was obtained by manually observe the fly behavior carefully and the data was chosen only from the frame that the fly performs obvious behavior. The training data used in this section is shown in Fig. 5.10. According to the figure, blank data means there is no training data at that frame. All the training data was taken from fly number 1, 16 , 23, and 32.

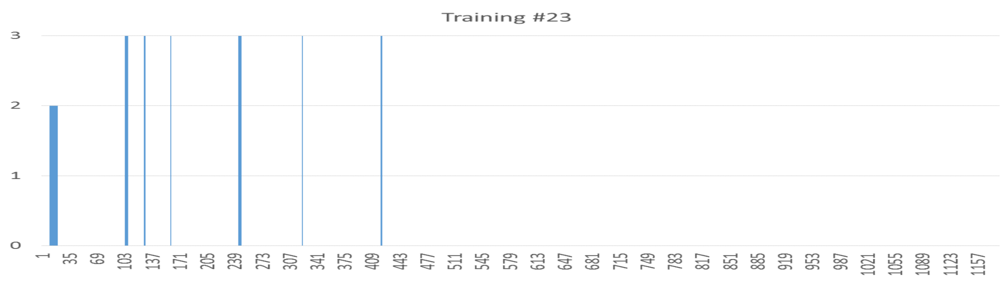
Total of 149 data is used to create a classification forest that consists of 20 decision trees. Fig. 5.11 and Fig. 5.12 show example of two decision trees created from the training data. Fig. 5.13 to Fig. 5.20 show the result of behavior classification obtained from random forest for fly number 1 to 32.



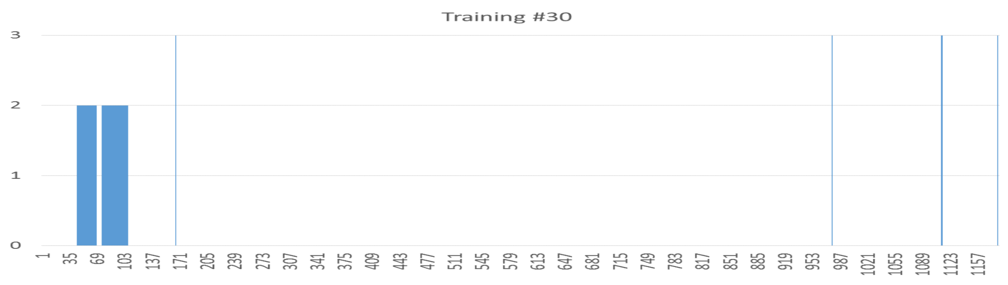
(a)



(b)



(c)



(d)

Figure 5.10: (a) (d) Training data from fly number 1, 16, 23, and 32, respectively. The blank data means there is no training data at that frame.

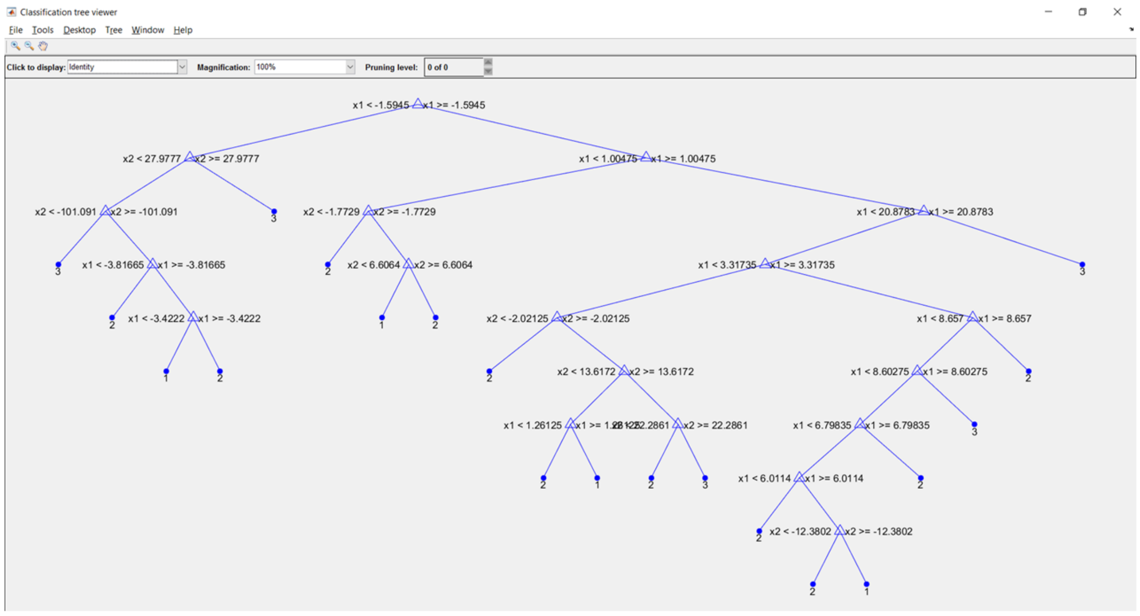

Classification tree viewer
File Tools Desktop Tree Window Help
Click to display: Identity
Magnification: 100%
Pruning level: 0 of 0
x1 < -1.5945 x1 >= -1.5945
x2 < 27.9777 x2 >= 27.9777
x1 < 1.00475 x1 >= 1.00475
x2 < -101.091 x2 >= -101.091
x2 < -1.7729 x2 >= -1.7729
x1 < 20.8783 x1 >= 20.8783
x1 < -3.81665 x1 >= -3.81665
x2 < 6.6064 x2 >= 6.6064
x1 < 3.31735 x1 >= 3.31735
x1 < -3.4222 x1 >= -3.4222
x2 < -2.02125 x2 >= -2.02125
x1 < 8.657 x1 >= 8.657
x2 < 13.6172 x2 >= 13.6172
x1 < 8.60275 x1 >= 8.60275
x1 < 1.26125 x1 >= 1.26125
x2 < 22.2861 x2 >= 22.2861
x1 < 6.79835 x1 >= 6.79835
x1 < 6.0114 x1 >= 6.0114
x2 < -12.3802 x2 >= -12.3802

Figure 5.11: The first decision tree of the classification forest.

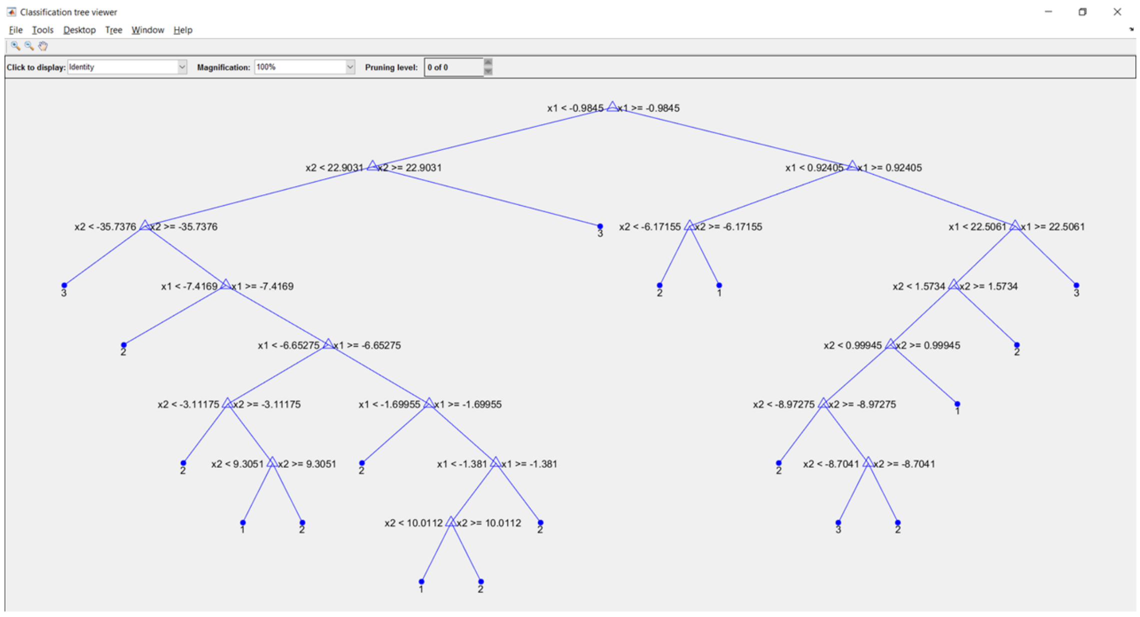
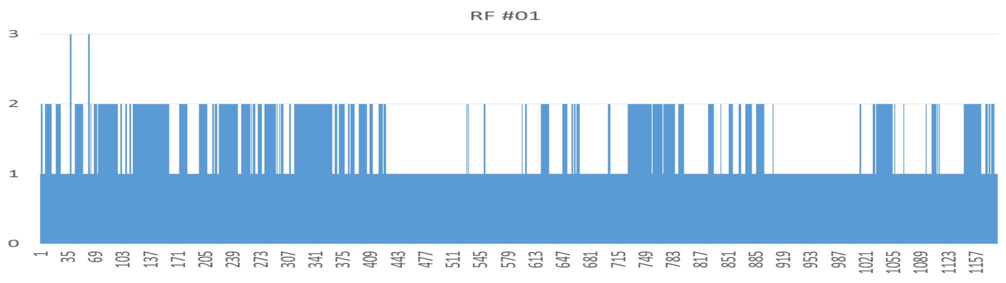
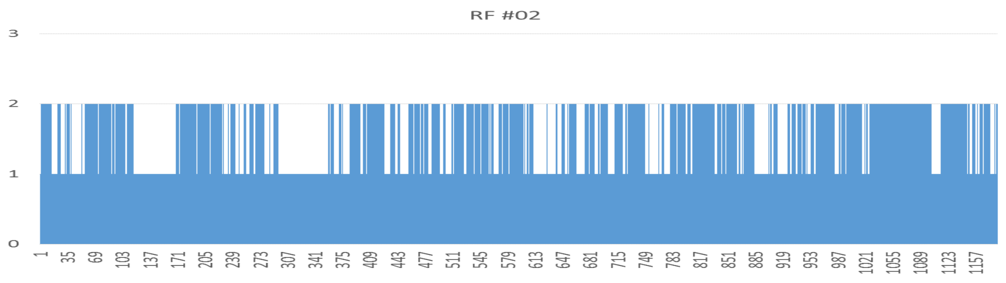

Classification tree viewer
File Tools Desktop Tree Window Help
Click to display: Identity
Magnification: 100%
Pruning level: 0 of 0
x1 < -0.9845 x1 >= -0.9845
x2 < 22.9031 x2 >= 22.9031
x1 < 0.92405 x1 >= 0.92405
x2 < -35.7376 x2 >= -35.7376
x2 < -6.17155 x2 >= -6.17155
x1 < 22.5061 x1 >= 22.5061
x1 < -7.4169 x1 >= -7.4169
x2 < 1.5734 x2 >= 1.5734
x1 < -6.65275 x1 >= -6.65275
x2 < 0.99945 x2 >= 0.99945
x2 < -3.11175 x2 >= -3.11175
x1 < -1.69955 x1 >= -1.69955
x2 < -8.97275 x2 >= -8.97275
x2 < 9.3051 x2 >= 9.3051
x1 < -1.381 x1 >= -1.381
x2 < -8.7041 x2 >= -8.7041
x2 < 10.0112 x2 >= 10.0112

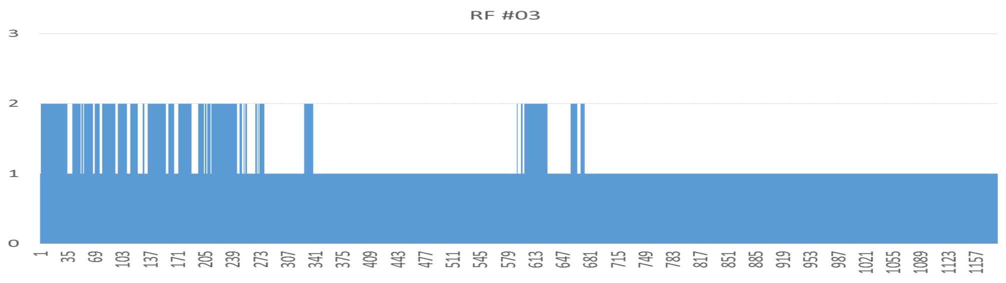
Figure 5.12: The second decision tree of the classification forest.



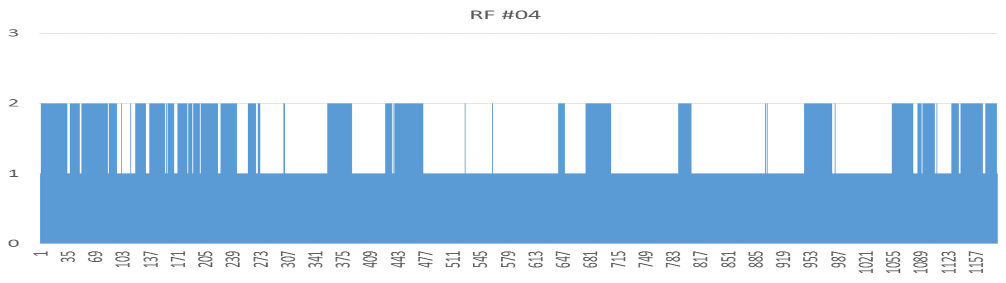
(a)



(b)

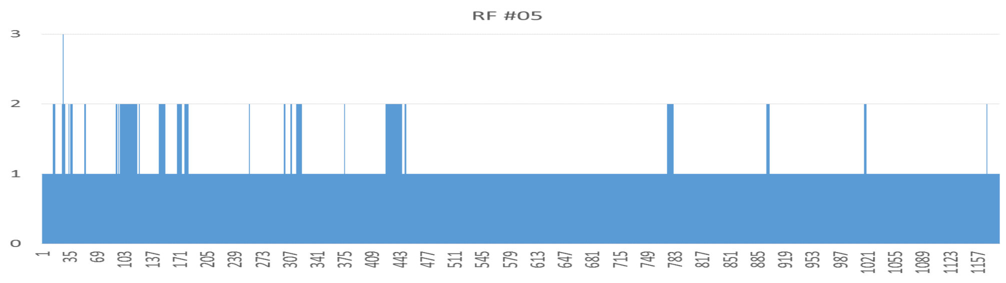


(c)

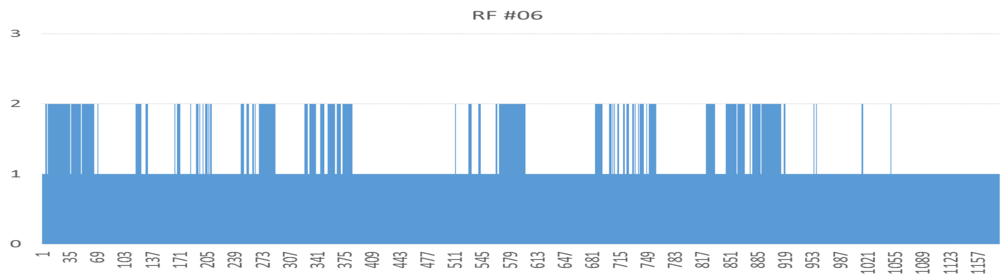


(d)

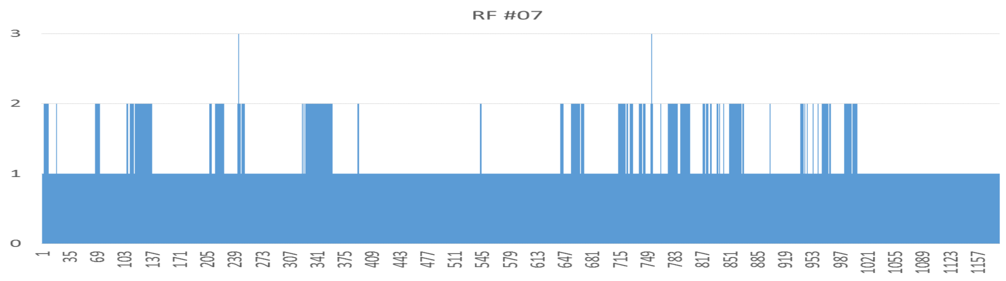
Figure 5.13: Result of behavior classification performed by random forest for fly number 1 to 4.



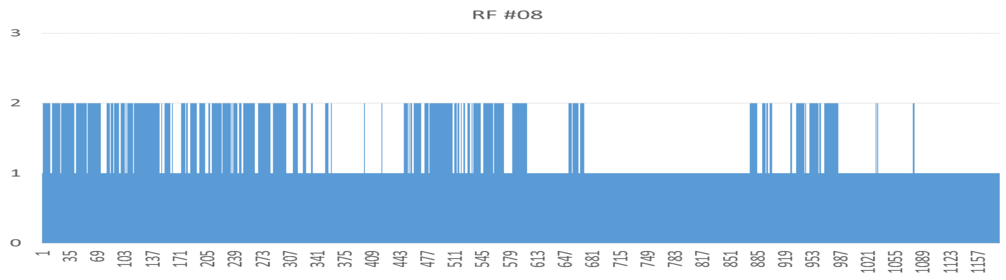
(a)



(b)

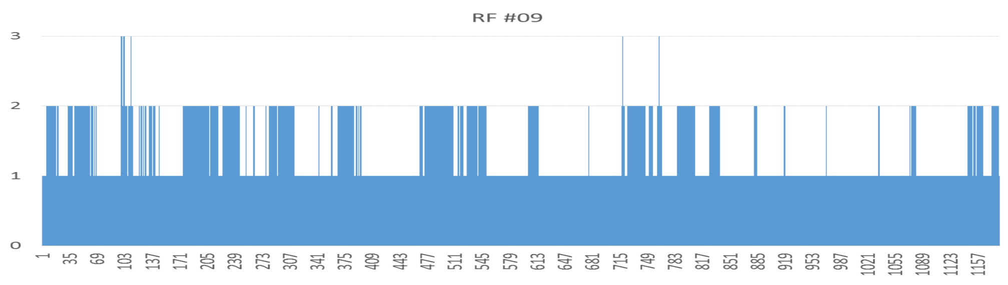


(c)

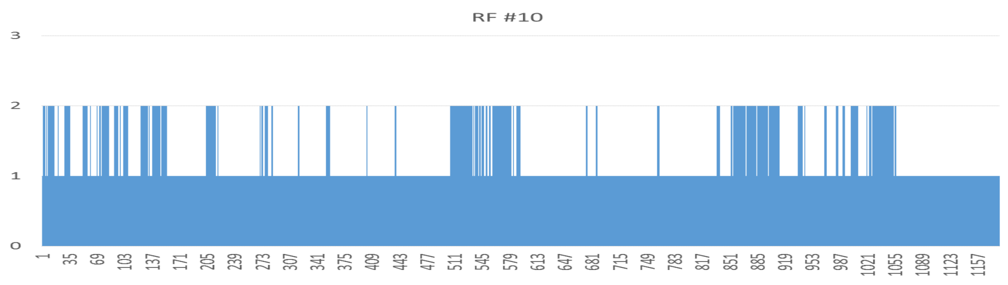


(d)

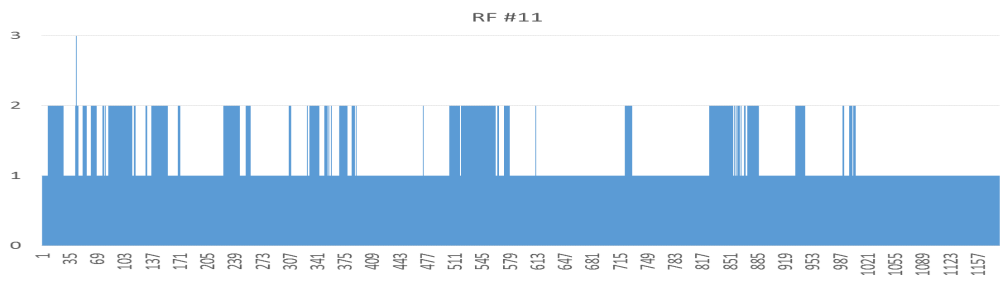
Figure 5.14: Result of behavior classification performed by random forest for fly number 5 to 8.



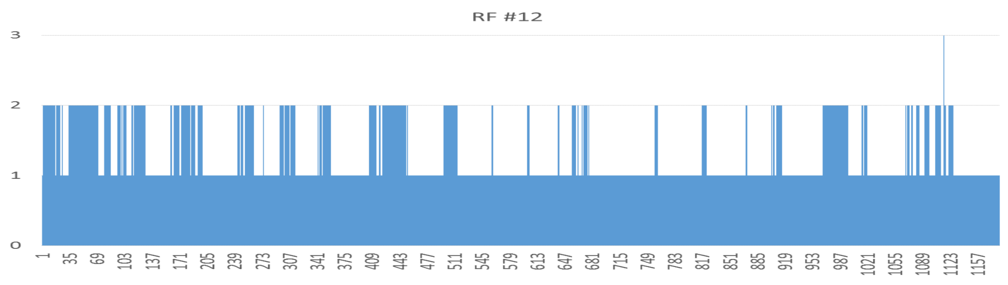
(a)



(b)

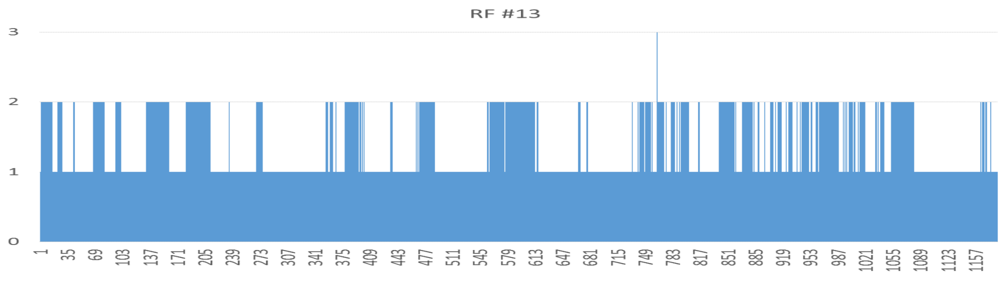


(c)

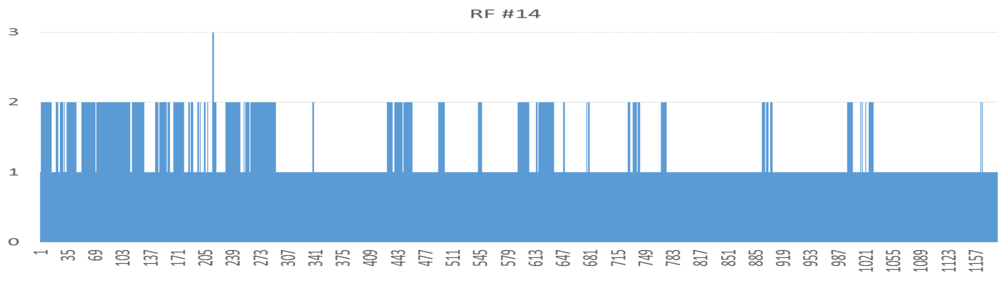


(d)

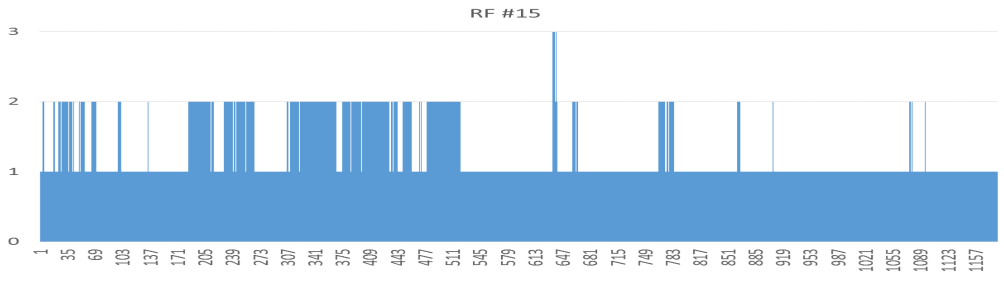
Figure 5.15: Result of behavior classification performed by random forest for fly number 9 to 12.



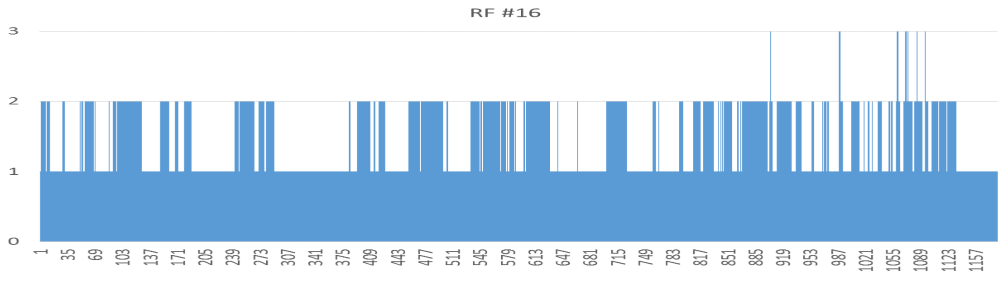
(a)



(b)

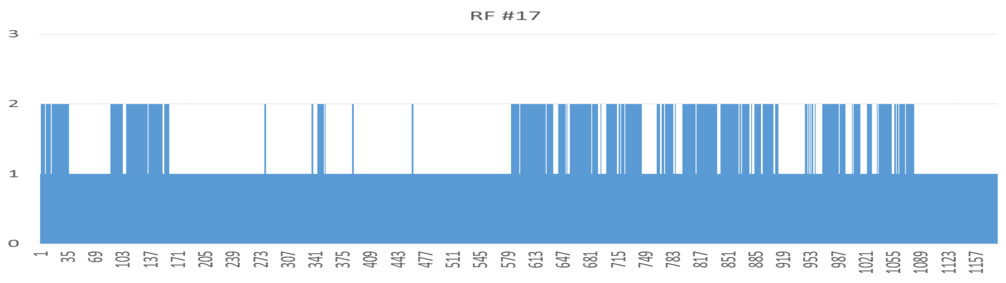


(c)

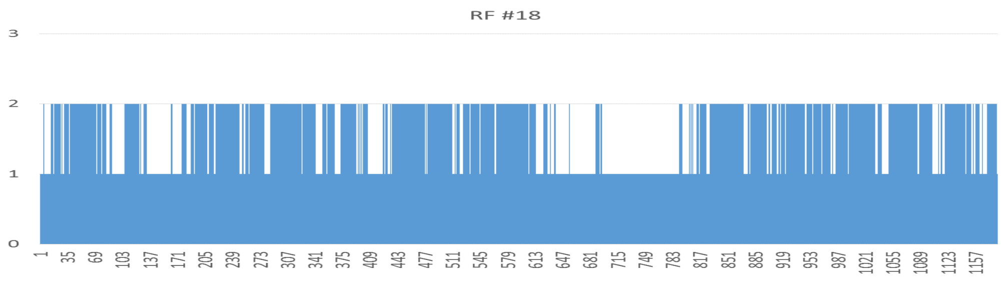


(d)

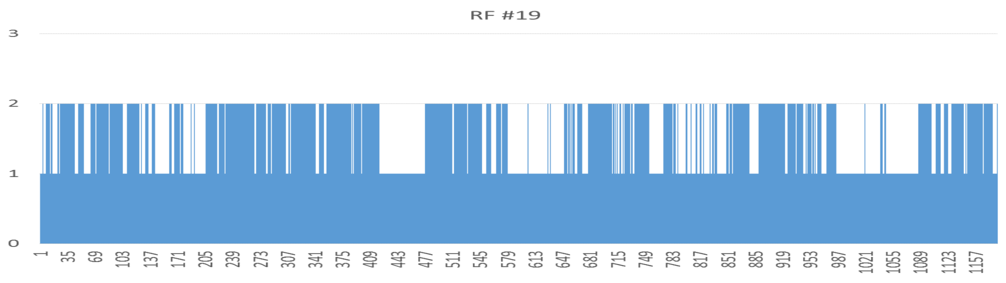
Figure 5.16: Result of behavior classification performed by random forest for fly number 13 to 16.



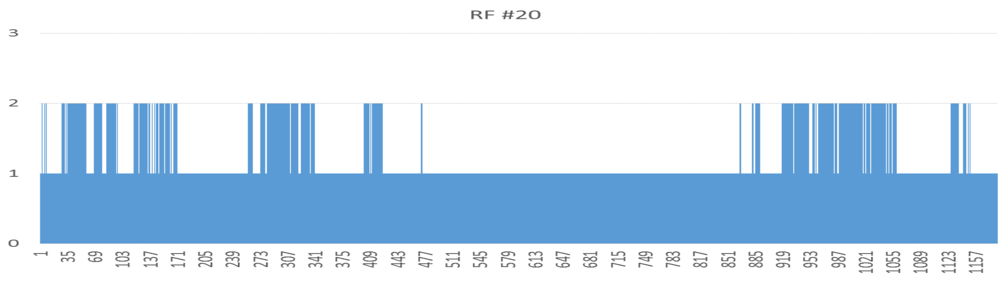
(a)



(b)

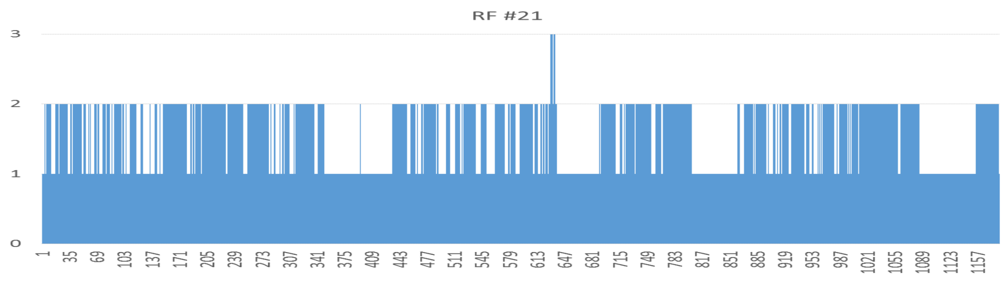


(c)

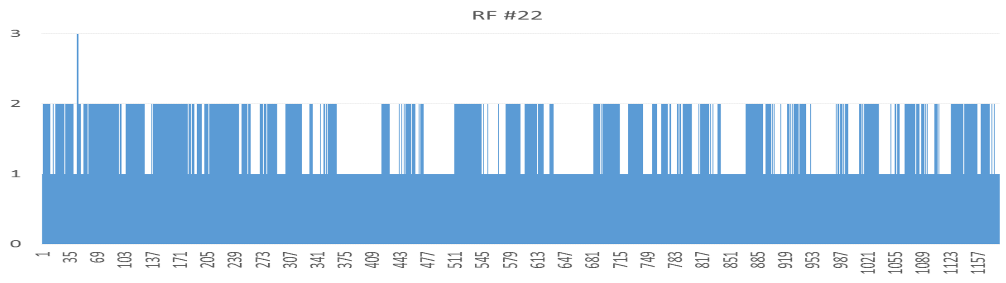


(d)

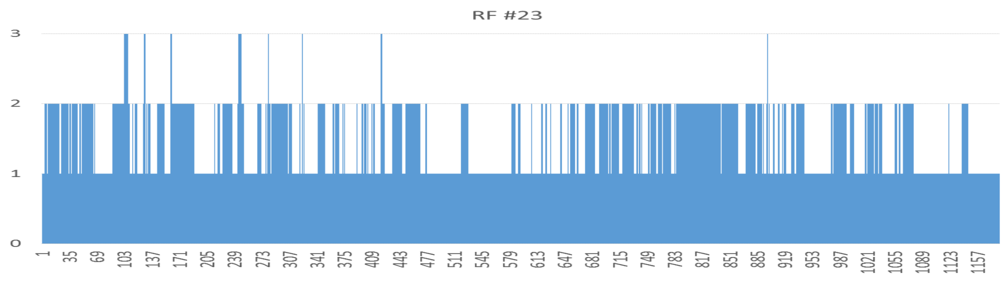
Figure 5.17: Result of behavior classification performed by random forest for fly number 17 to 20.



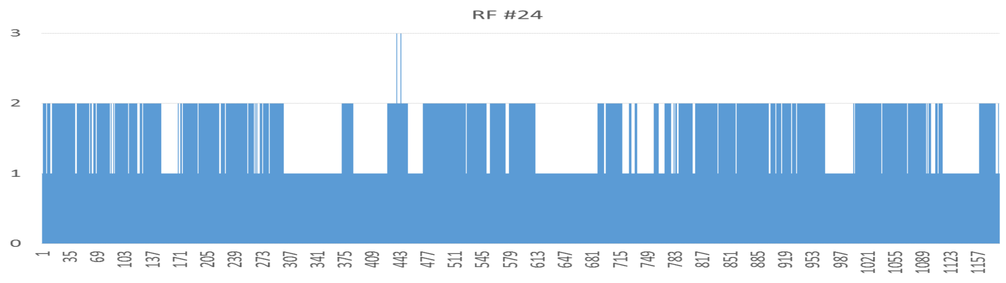
(a)



(b)

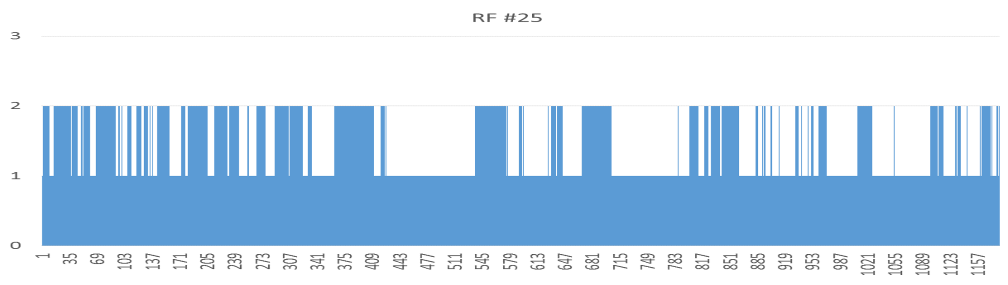


(c)

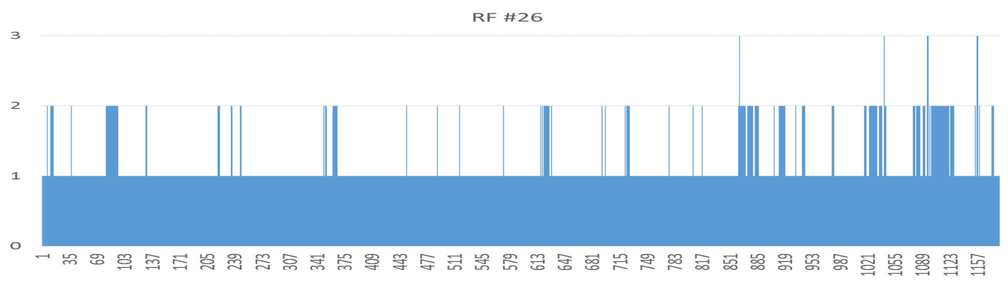


(d)

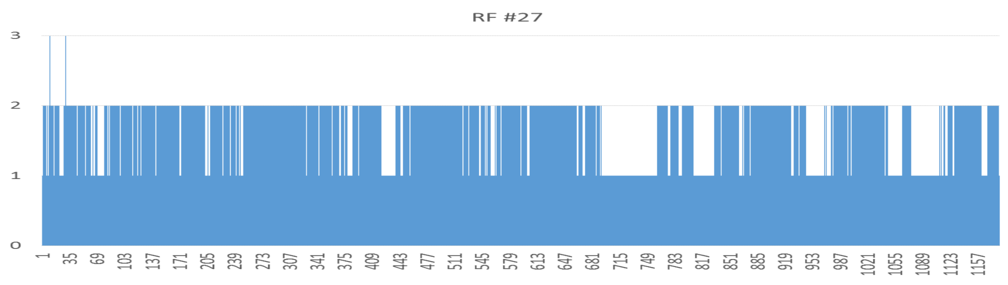
Figure 5.18: Result of behavior classification performed by random forest for fly number 21 to 24.



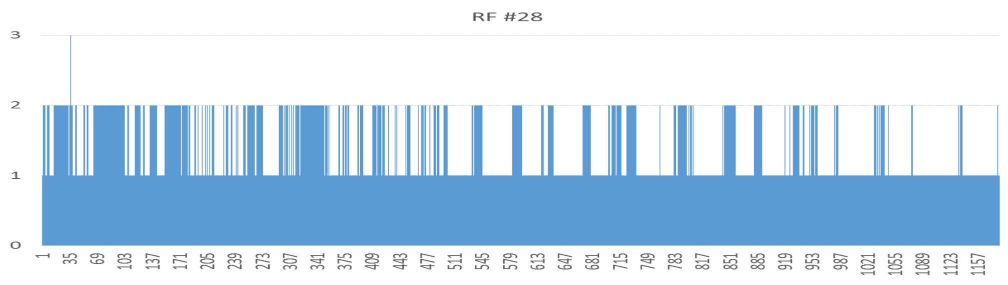
(a)



(b)

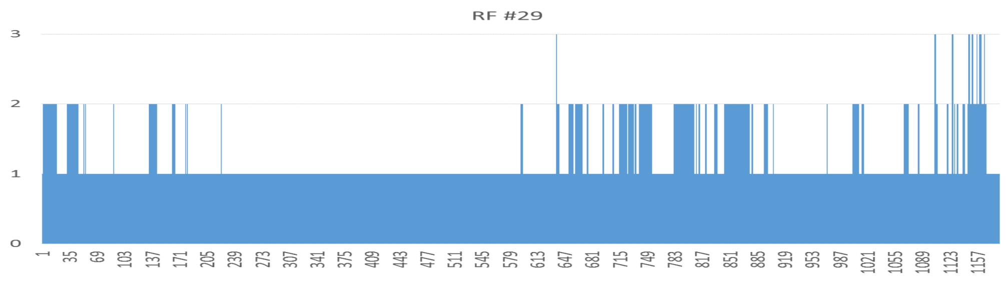


(c)

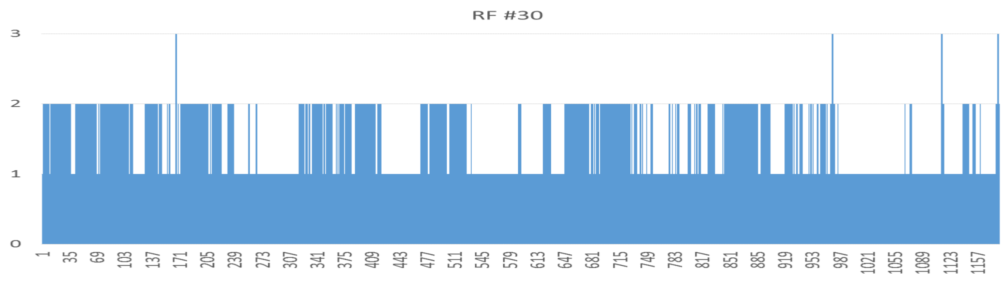


(d)

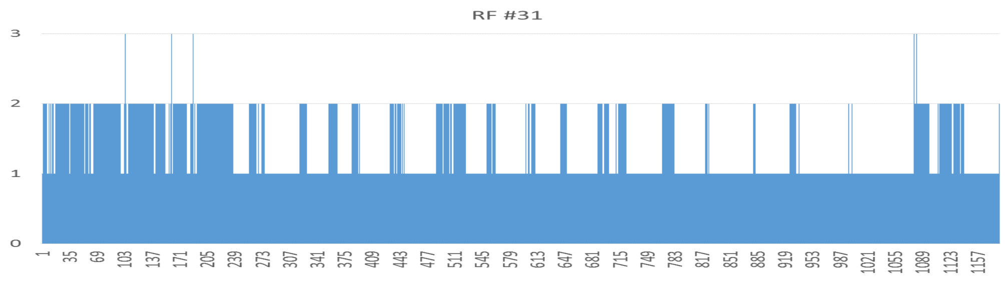
Figure 5.19: Result of behavior classification performed by random forest for fly number 25 to 28.



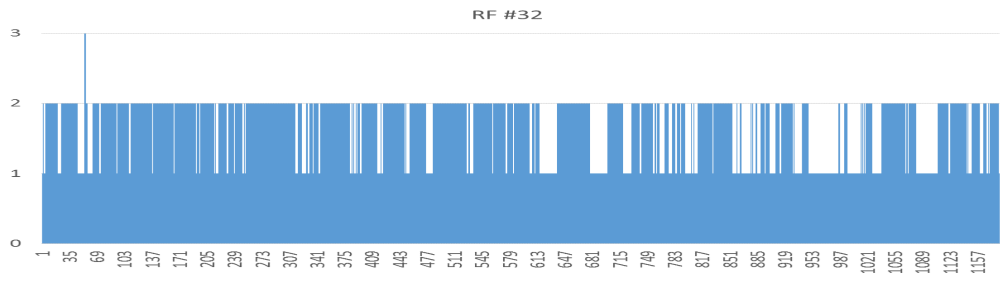
(a)



(b)



(c)



(d)

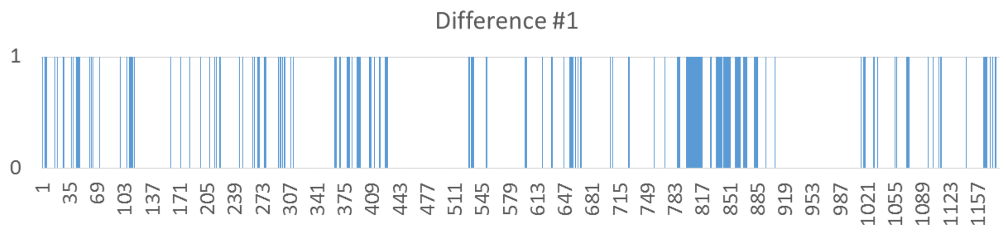
Figure 5.20: Result of behavior classification performed by random forest for fly number 28 to 32.

5.3 Results

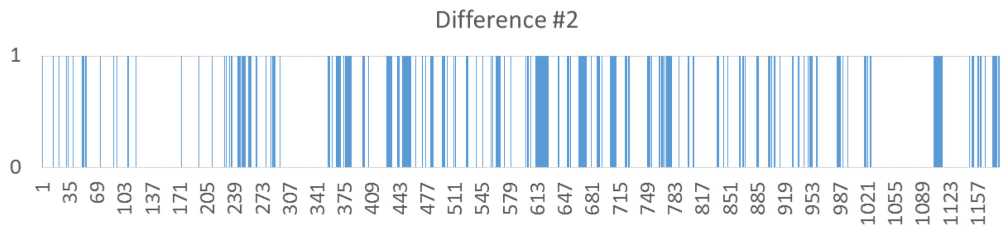
The classification results from human and the random forest are compared in this section. Fig. 5.21 to Fig. 5.28 show the error at each frame of each individual fly. The value of 1 in the vertical axis indicates that the behavior determined by the random forest is different from the one determined by human.

5.4 Discussion

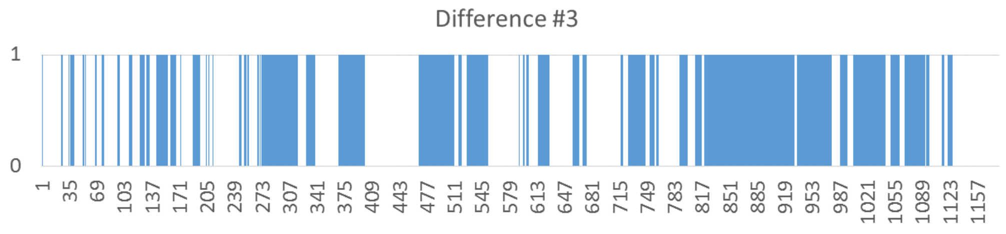
By observing the difference of behavior between using manual classification and random forest classification (Fig. 5.21 to Fig. 5.28), most matching results are the classification of fly number 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 18, 19, 22, 24, 26, and 30. For other flies, there were some failures of the tracking program that caused the identity assignment to fail and resulted in identity swapping. One example of this failure occurred at frame 337 of the fly number 3. After observed carefully, at the frame 337, identity swapping occurred between fly number 3 and fly number 17 because fly detection failed to separate those flies. The result of matching in classification show that the usage of the tracking program together with classification by the random forest can provide the similar results to the works done by human. However, manual tracking and classifying took about 15 minutes in average for each fly. This means it takes about 8 hours to work with all 32 flies without resting. On the other hand, using the program took less than 10 minutes to do the task for all flies. This can save a lot of time and the error can be easily found by using the tracked trajectories.



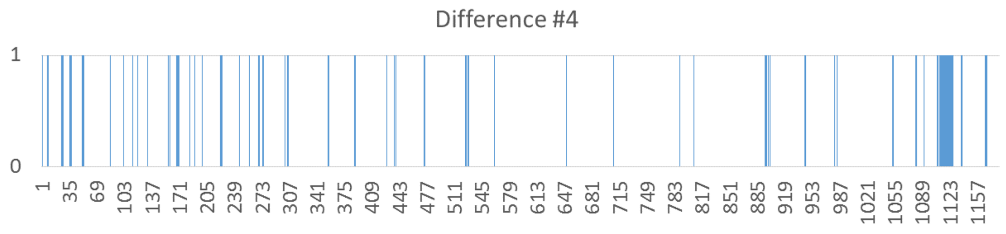
(a)



(b)

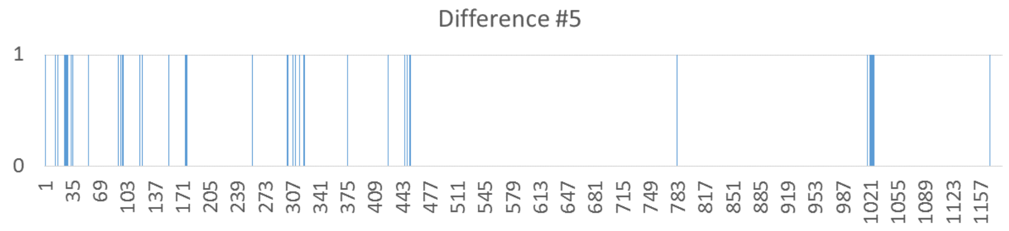


(c)

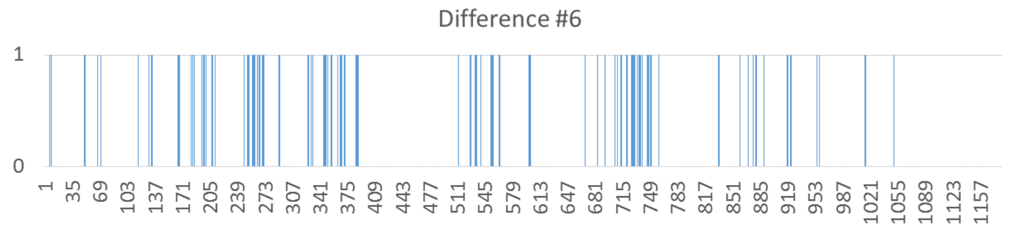


(d)

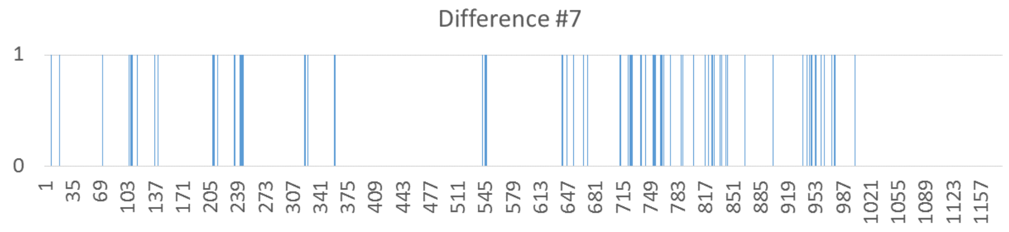
Figure 5.21: Difference of behavior between using manual classification and random forest classification of fly number 1 to 4.



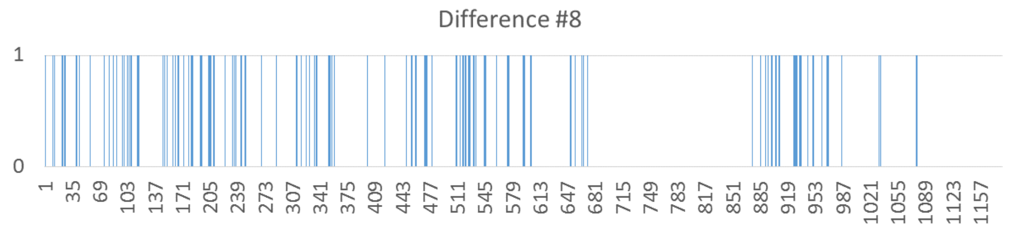
(a)



(b)

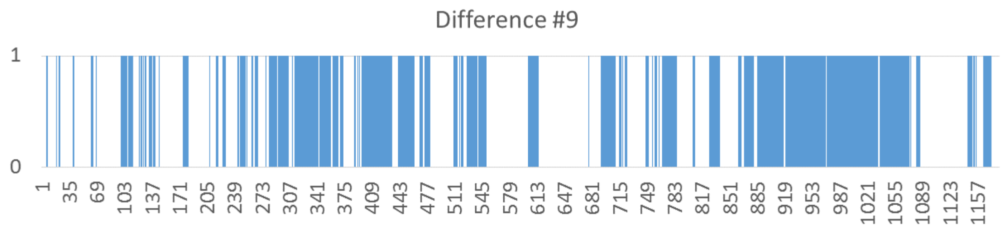


(c)

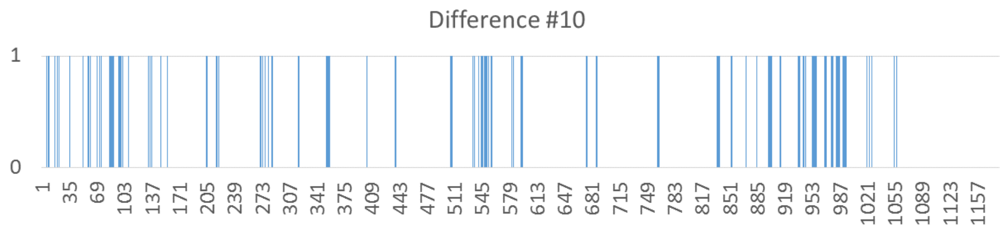


(d)

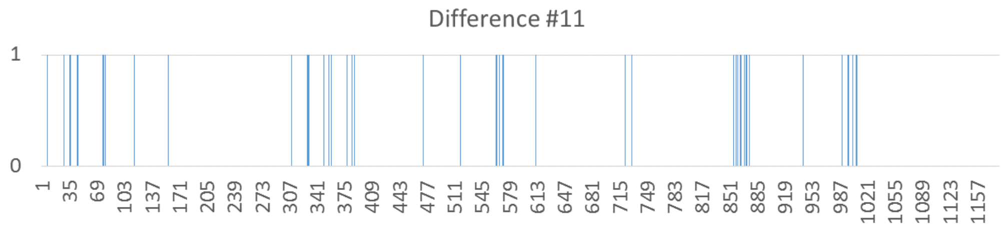
Figure 5.22: Difference of behavior between using manual classification and random forest classification of fly number 5 to 8.



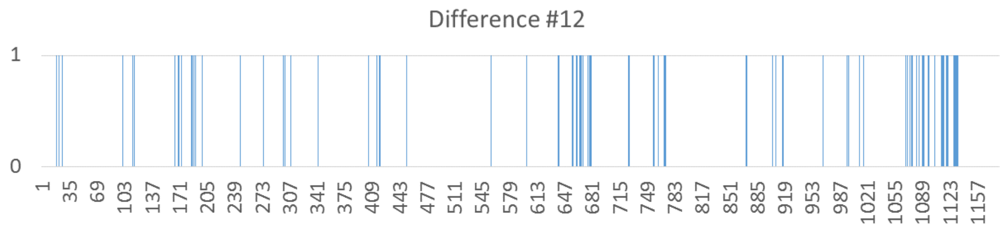
(a)



(b)

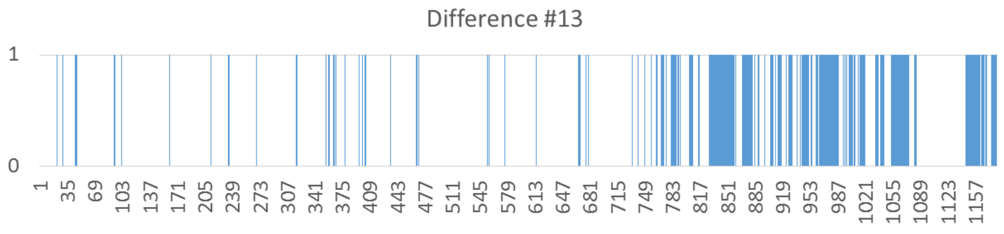


(c)

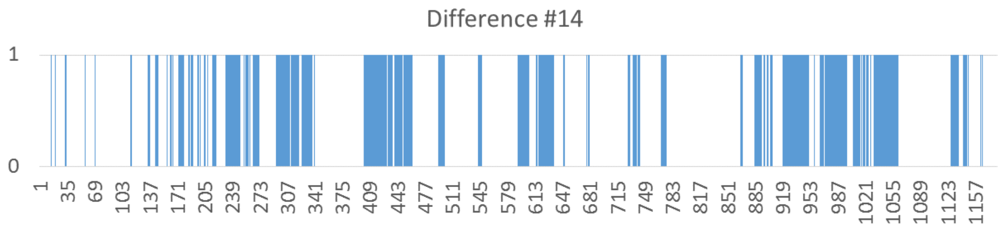


(d)

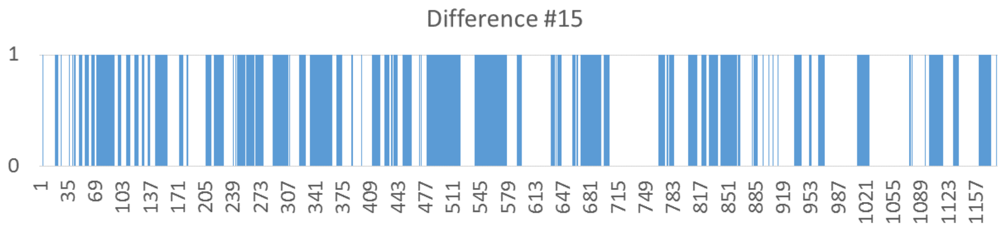
Figure 5.23: Difference of behavior between using manual classification and random forest classification of fly number 9 to 12.



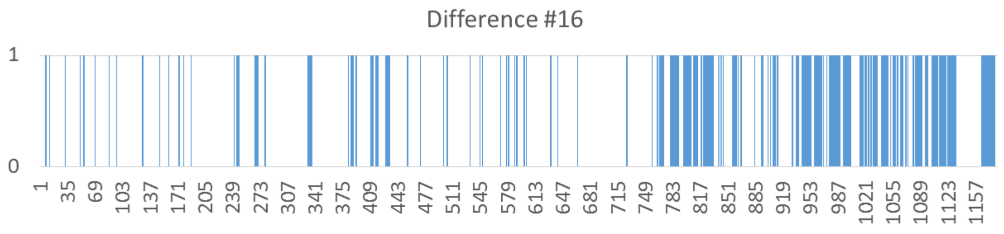
(a)



(b)

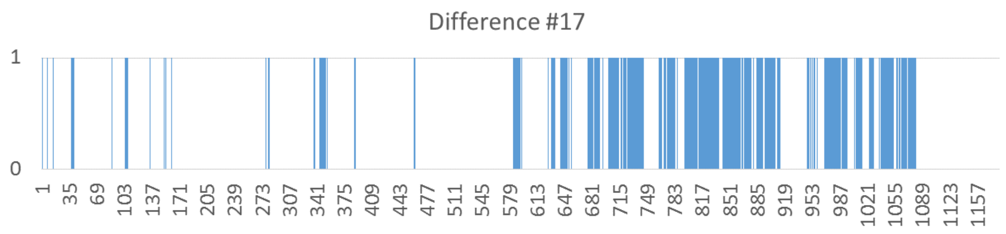


(c)

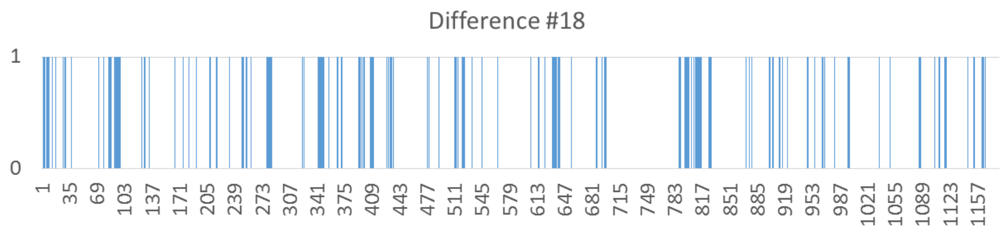


(d)

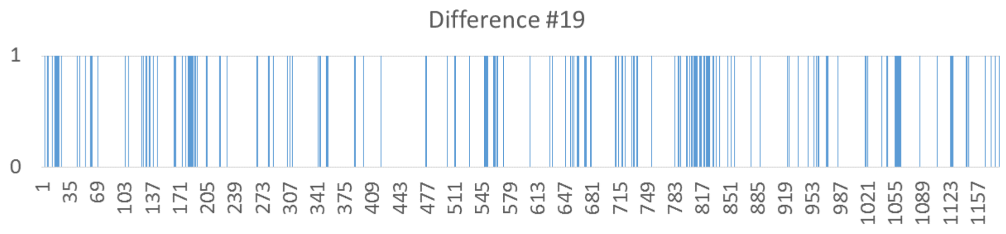
Figure 5.24: Difference of behavior between using manual classification and random forest classification of fly number 13 to 16.



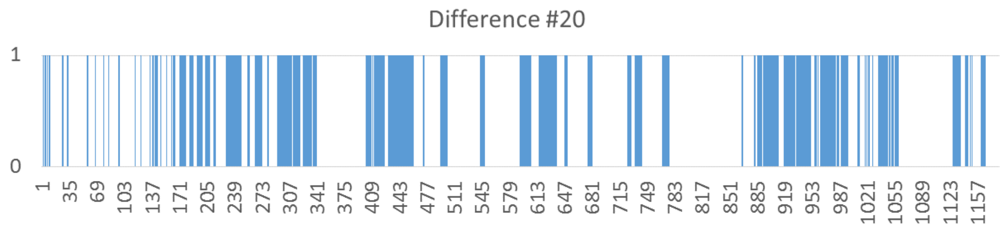
(a)



(b)

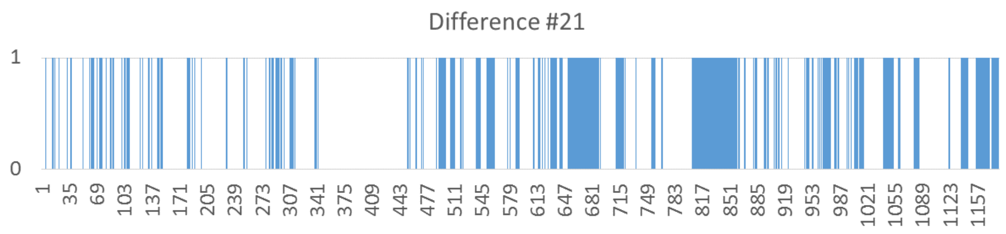


(c)

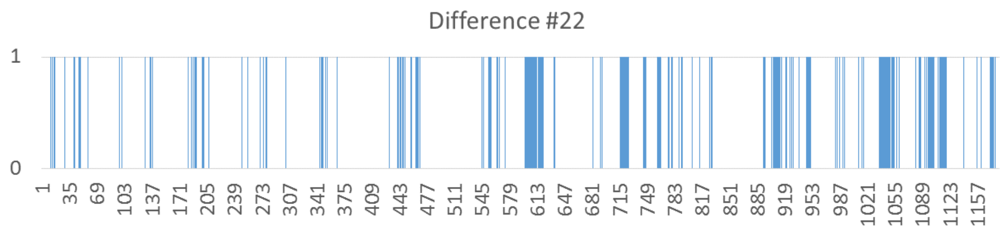


(d)

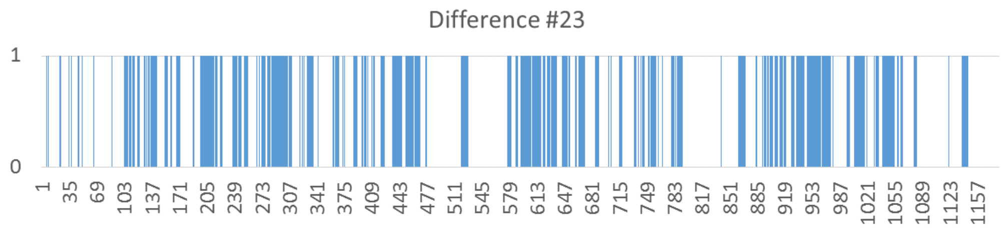
Figure 5.25: Difference of behavior between using manual classification and random forest classification of fly number 17 to 20.



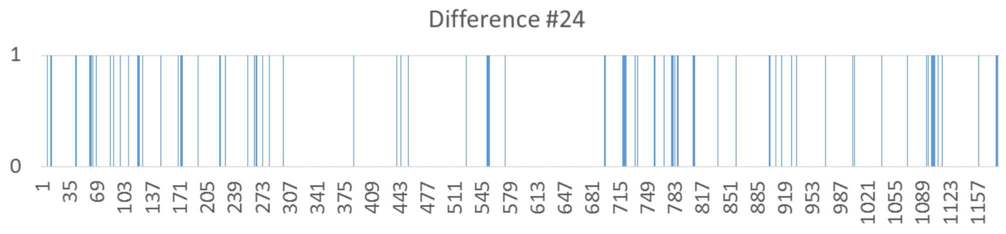
(a)



(b)

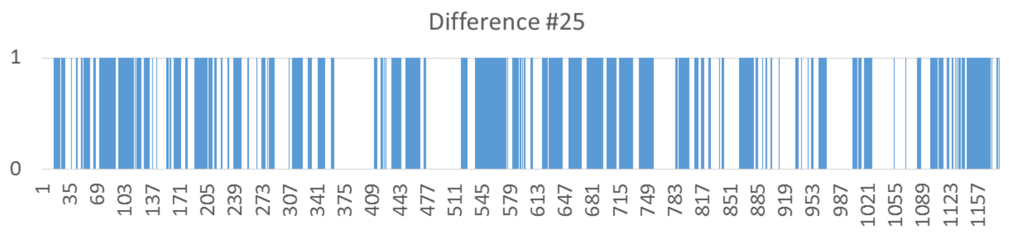


(c)

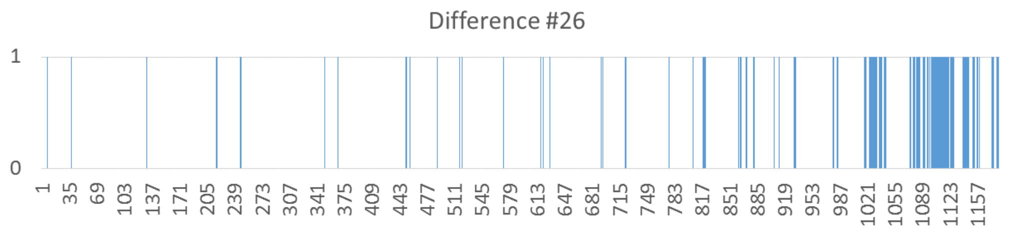


(d)

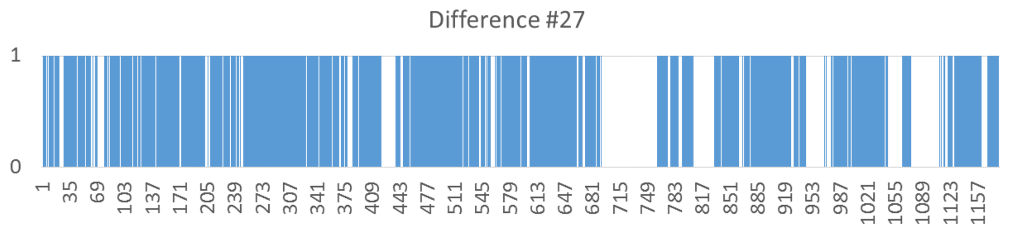
Figure 5.26: Difference of behavior between using manual classification and random forest classification of fly number 21 to 24.



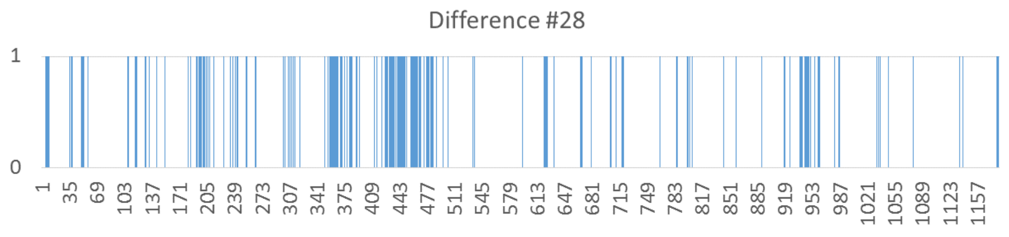
(a)



(b)

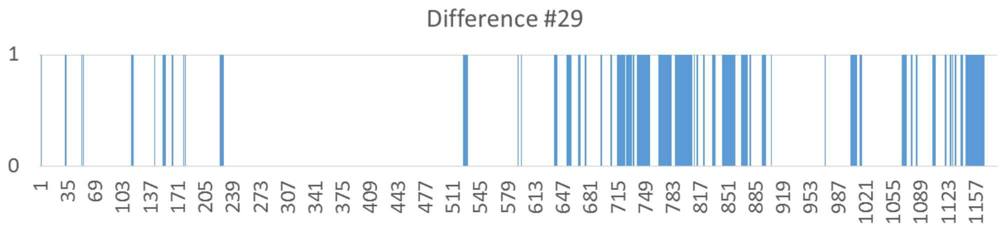


(c)

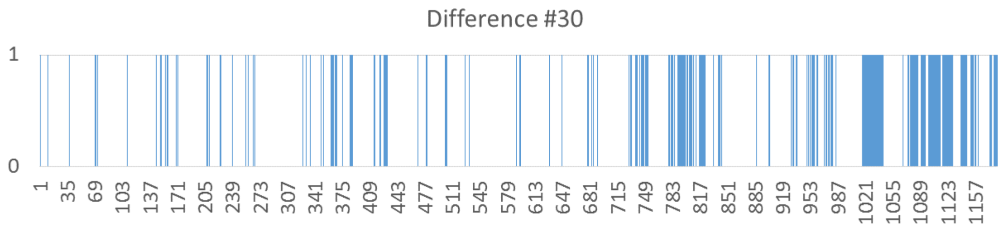


(d)

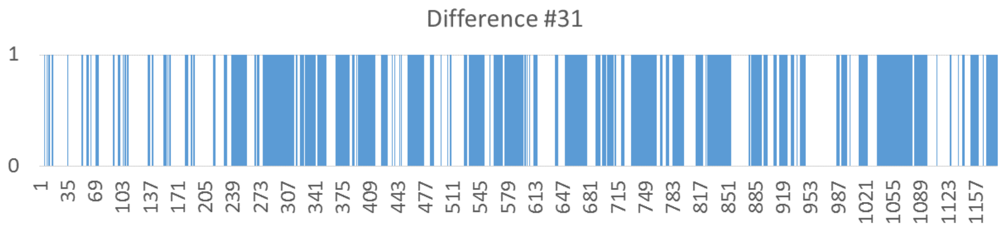
Figure 5.27: Difference of behavior between using manual classification and random forest classification of fly number 25 to 28.



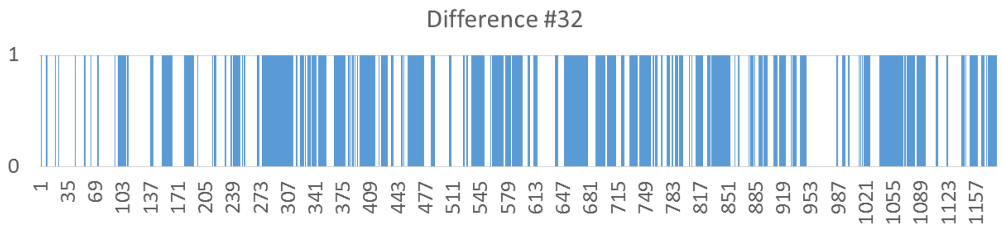
(a)



(b)



(c)



(d)

Figure 5.28: Difference of behavior between using manual classification and random forest classification of fly number 29 to 32.

Chapter 6

Conclusion and Future Perspectives

According to the need of locomotion analysis for small model organism such as *Drosophila Melanogaster* or fruit fly, a machine vision system that can process data in form of video file and give correct tracking data is considered. In order to have a successful fly tracking program, many techniques such as fly detection, merging blob splitting, wing detection, identity assignment, and prediction of motion are the requirement of the program. In this dissertation, two automatic fly tracking systems are proposed and discussed as multiple-object tracking problem with indistinguishable features. The data from the tracking program is input to behavior classification of *Drosophila* by using a machine learning technique, random forest. In addition, a system of controlling multiple lake monitoring sensor nodes is also discussed as multiple-object problem with distinguishable features. All the simulation methods are provided in each chapter and the tracking results of using both fly tracking systems are mentioned in this dissertation.

In chapter 2, to achieve long time monitoring of water resource and collect measurement of water resource effectively, we have proposed the concept of mobile monitoring sensor and control method of the sensors. The proposed mobile monitoring sensor has actuator and thus can move by itself. To reduce cost of proposed sensor and to achieve long time operation, it is necessary to prolong battery life. For this purpose, the proposed control method for distribution of sensors utilize the drag force from the flow field effectively for sensor's movement. The distributions are performed in both uniform constant water flow field and complex steady flow field. Numerical examples show that the proposed controller provides less energy consumption comparing with a method using simple position-velocity control that is considered to be the conventional control. Furthermore, the modified version of the proposed controller gives more effective group control with lower in energy consumption than the original proposed controller. This makes the proposed lake monitoring sensors system be able to achieve long time monitoring of water resource. The lower in energy consumption for the monitoring sensors distribution saves the amount of limited energy inside the battery attached to each sensor node. Therefore, with low energy consumption, the monitoring system operating time will last longer and the system efficiency will be increased. In the future, by taking control of the sensors group based on high accuracy whole flow field of the lake, more effective multiple-object control can be achieved.

In chapter 3, we present an algorithm for analyzing the locomotion behavior of the fruit fly *Drosophila melanogaster*, a popular model organism in neurobiology. Our proposed algorithm detects all flies inside a circular arena, determines their position and orientation and assigns fly identities between consecutive frame pairs. Position and orientation of the flies are accurately estimated with average errors of 0.108 ± 0.006

mm (approximately 5% of fly body length) and $2.2 \pm 0.2°$, respectively. Importantly, fly identity is correctly assigned in 99.5% of the cases. The algorithm can be used to quantify the linear and angular velocities of walking flies in the presence or absence of various behaviorally important stimuli. The next goal is to apply the system to more challenging fly experiment condition like applying electric shock to the fly. With the presence of electric shock, the fly will jump or fly with high speed and this makes the fly detection and the identity assignment problem to be more challenging.

In chapter 4, an improved system for detecting and tracking a group of *Drosophila* is introduced. It can deal with situation where flies are within close proximity of one another resulting in merging of their position while having the ability to separate them apart. This system provides a solution to crossover and touching identity swapping problem in tracking process. It uses heading direction for evaluating tracking result from prediction strategy and refines with the closest neighbor method to determine correct identity of each fly in the consecutive image frame. The closest neighbor assignment algorithm is used for the case of heading direction mismatch resulting in more proper matching. Suggested follow up works are improving accuracy of the system, reducing process time, and wing position of a fly. The latter is beneficial for extracting more information about fly behavior.

In chapter 5, fly behavior analysis is shown. The output data from the tracking program from chapter 4 is used for simple behavior classification using one of the classification techniques called random forest. The efficiency of the usage of the data is emphasized by comparing the classified result from the program with the behavior classification manually done by human. As a result, the program can provide a good behavior result that is the same performance as human labeled classification with more time efficiency than human.

In summary, the latest improved *Drosophila* tracking system provides successful locomotion data of each individual fly by using a combination of predict-matching and closest neighbor approach assignment methods in the tracking process. The proposed method of using heading direction to check the pairing result can reduce the identity swapping occurred from touching and crossing over scenarios. For the future work, with the raw data output from the proposed tracking program we would like to expand the ability of video analysis in a scope of more complex behavioral classification. Moreover, the detection part in the system can be improved in order to obtain more detailed data. This includes precise wing detection, leg detection, and accurate shape detection.

Bibliography

[1] Ball tracking with opencv, 2015. [Online; https://www.youtube.com/watch?v=dKpRsdYSCLQ; accessed December 01, 2016].

[2] Sony's playstation move controllers working with its virtual reality headset., 2015. [Online; https://www.cnet.com/news/sony-gives-playstation-vr-a-hand-by-acquiring-gesture-tracking-tech/; accessed December 01, 2016].

[3] Major experiment and public demonstration of the floating sensor fleet, 2012. [Online; http://float.berkeley.edu/project/experiments/2012-may-walnut/floating-sensor-fleet-walnut-grove-may-09-2012-part-2; accessed November 27, 2016].

[4] Evolution of drifter. [Online; http://float.berkeley.edu/drifters/evolution; accessed November 27, 2016].

[5] Floating sensor network for real-time response to levee breach, 2009. [Online; http://float.berkeley.edu/fsn/?q=webfm_send/91; accessed November 27, 2016].

[6] Hardware specification of floating sensor network. [Online; http://float.berkeley.edu/drifters/hardware; accessed November 27, 2016].

[7] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–1090–I–1097 vol.1, 2001.

[8] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 1, pages 943–948 vol.1, Sept 2004.

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[11] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *BMVC*, volume 1, page 6, 2006.

[12] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[13] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[14] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 2756–2759. IEEE, 2010.

[15] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[16] Vladimiros Thoma, Stephan Knapek, Shogo Arai, Marion Hartl, Hiroshi Kohsaka, Pudith Sirigrivatanawong, Ayako Abe, Koichi Hashimoto, and Hiromu Tanimoto. Functional dissociation in sweet taste receptor neurons between and within taste organs of drosophila. *Nat Commun*, 7, 2016.

[17] Kristin Branson, Alice A. Robie, John Bender, Pietro Perona, and Michael H. Dickinson. High-throughput ethomics in large groups of drosophila. *Nat Meth*, 6(6):451–457, 2009.

[18] Alfonso Perez-Escudero, Julian Vicente-Page, Robert C. Hinz, Sara Arganda, and Gonzalo G. de Polavieja. idtracker: tracking individuals in a group by automatic identification of unmarked animals. *Nat Meth*, 11(7):743–748, 2014.

[19] Ilker T. Telci, Kijin Nam, Jiabao Guan, and Mustafa M. Aral. Optimal water quality monitoring network design for river systems. *Journal of Environmental Management*, 90(10):2987 – 2998, 2009.

[20] Kijin Nam and Mustafa Mehmet Aral. Optimal placement of monitoring sensors in lakes. 2007.

[21] Calvin C Teague, Donald E Barrick, Peter M Lilleboe, and T Ralph. Two-dimensional surface river flow patterns measured with paired riversondes. In *2007 IEEE International Geoscience and Remote Sensing Symposium*, pages 2491–2494. IEEE, 2007.

[22] John F Vesecky, Lorelle A Meadows, Calvin C Teague, Peter Hansen, Michal Plume, and Yolanda Fernandez. Surface current observations by hf radar during eegle 2000. In *Geoscience and Remote Sensing Symposium, 2001. IGARSS'01. IEEE 2001 International*, volume 1, pages 272–274. IEEE, 2001.

[23] Calvin C Teague, Donald E Barrick, Peter M Lilleboe, and Ralph T Cheng. Extended uhf radar observations of river flow velocity and comparisons with in-situ measurements. In *Ninth International Symposium on River Sedimentation, Minist. of Water Resour., Yichang, China*, 2004.

[24] H. Guo and Y. Zhang. Notice of retraction application of optical flow method in inversion of ocean surface flow field. In *2010 Second IITA International Conference on Geoscience and Remote Sensing*, volume 1, pages 31–35, Aug 2010.

[25] Calvin C Teague. Two-dimensional flow patterns observed at threemile slough using two riversondes. In *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 4, pages IV–890. IEEE, 2008.

[26] Takaharu Kitamura, Takao Tsuchiya, Yoshiaki Watanabe, and Shin-ichi Sakamoto. Direct in situ measurements of sound speed and water temperature profiles the example of north lake biwa. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 457–460. IEEE, 2011.

[27] TR Consi, G Barske, H Bootsma, T Hansen, J Janssen, V Klump, R Paddock, and JT Waples. Glucos: The great lakes urban coastal observing system. In *OCEANS 2006*, pages 1–5. IEEE, 2006.

[28] LC Shen, JC Juang, and CL Tsai. Remote sensing coastal sea level and ocean tide by using reflected gps l1 and l2 observation for integrated gps receiver. In *OCEANS 2008-MTS/IEEE Kobe Techno-Ocean*, pages 1–8. IEEE, 2008.

[29] Wei Chen. Surface velocity estimation from satellite imagery using displaced frame central difference equation. *IEEE Transactions on Geoscience and Remote Sensing*, 50(7):2791–2801, 2012.

[30] Remko de Lange, Adrian Luckman, and Tavi Murray. Improvement of satellite radar feature tracking for ice velocity derivation by spatial frequency filtering. *IEEE transactions on geoscience and remote sensing*, 45(7):2309–2318, 2007.

[31] Andrew Tinka, Mohammad Rafiee, and Alexandre M Bayen. Floating sensor networks for river studies. *IEEE Systems Journal*, 7(1):36–49, 2013.

[32] Andy Schneider. Near shore wireless communication system for sensor buoys. In *OCEANS 2006*, pages 1–5. IEEE, 2006.

[33] Deepak Bhadauria, Volkan Isler, Andrew Studenski, and Pratap Tokekar. A robotic sensor network for monitoring carp in minnesota lakes. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3837–3842. IEEE, 2010.

[34] Giancarlo Ferrari-Trecate, Luca Galbusera, Marco Pietro Enrico Marciandi, and Riccardo Scattolini. Model predictive control schemes for consensus in multi-

agent systems with single-and double-integrator dynamics. *IEEE Transactions on Automatic Control*, 54(11):2560–2572, 2009.

[35] Erfu Yang, Dongbing Gu, and Huosheng Hu. Nonsingular formation control of cooperative mobile robots via feedback linearization. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 826–831. IEEE, 2005.

[36] Tansel Yucelen and Magnus Egerstedt. Control of multiagent systems under persistent disturbances. In *2012 American Control Conference (ACC)*, pages 5264–5269. IEEE, 2012.

[37] Peter C Chu, Leonid M Ivanov, Tatiana P Korzhova, Tatiana M Margolina, and Oleg V Melnichenko. Analysis of sparse and noisy ocean current data using flow decomposition. part i: Theory. *Journal of Atmospheric and Oceanic Technology*, 20(4):478–491, 2003.

[38] Feng Li, Liwei Xu, Philippe Guyenne, and Jingyi Yu. Recovering fluid-type motions using navier-stokes potential flow. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2448–2455. IEEE, 2010.

[39] Tuncer Cebeci, Jian P Shao, Fassi Kafyeke, and Eric Laurendeau. *Computational fluid dynamics for engineers.* Springer Berlin Heidelberg, 2005.

[40] Benjamin Seibold. A compact and fast matlab code solving the incompressible navier-stokes equations on rectangular domains mit18086 navierstokes. m. *Massachusetts Institute of Technology*, 2008.

[41] Gianluca Antonelli. Interconnected dynamic systems: An overview on distributed control. *IEEE Control Systems*, 33(1):76–88, 2013.

[42] A. Letsou and D. Bohmann. Small flies-big discoveries: Nearly a century of drosophila genetics and development. *Developmental Dynamics*, 232(3):526 – 528, 2005.

[43] A. Veeraraghavan, R. Chellappa, and M. Srinivasan. Shape-and-behavior encoded tracking of bee dances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):463–476, 2008.

[44] Christian Wehrhahn, Tomaso Poggio, and Heinrich Blthoff. Tracking and chasing in houseflies (musca). *Biological Cybernetics*, 45(2):123–130, 1982.

[45] Zinan Zhao and M. Kumar. An mcmc-based particle filter for multiple target tracking. *Information Fusion*, pages 1676–1682, 2012.

[46] Steven N. Fry, Nicola Rohrseitz, Andrew D. Straw, and Michael H. Dickinson. Trackfly: Virtual reality for a behavioral system analysis in free-flying fruit flies. *Journal of Neuroscience Methods*, 171(1):110–117, 2008.

[47] F.A. Zabala. Directionality control and flight stability of takeoff. *Robotics and Automation*, pages 4213–4218, 2009.

[48] F.A. Zabala, G.M. Card, E.I Fontaine, M.H. Dickinson, and R.M. Murray. Flight dynamics and control of evasive maneuvers: The fruit fly's takeoff. *Biomedical Engineering*, 56(9):2295–2298, 2009.

[49] M. Mronz and R. Strauss. Visual motion integration controls attractiveness of

objects in walking flies and a mobile robot. *Intelligent Robots and Systems*, pages 3559–3564, 2008.

[50] F. A. Zabala. Directionality control and flight stability of takeoff. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4213–4218.

[51] F. A. Zabala, G. M. Card, E. I. Fontaine, M. H. Dickinson, and R. M. Murray. Flight dynamics and control of evasive maneuvers: The fruit fly's takeoff. *IEEE Transactions on Biomedical Engineering*, 56(9):2295–2298, 2009.

[52] J. Fan, N. Jiang, and Y. Wu. Automatic video-based analysis of animal behaviors. In *2010 IEEE International Conference on Image Processing*, pages 1513–1516.

[53] Kieran F. Mulchrone and Kingshuk Roy Choudhury. Fitting an ellipse to an arbitrary shape: implications for strain analysis. *Journal of Structural Geology*, 26(1):143–153, 2004.

[54] S. Arai, P. Sirigrivatanawong, and K. Hashimoto. Multiple drosophila tracking and posture estimation algorithm. In *Informatics, Electronics & Vision (ICIEV), 2015 International Conference on*, pages 1–6.

[55] Zia Khan, Tucker Balch, and Frank Dellaert. *An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets*, pages 279–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[56] M. Li, Y. Zhu, and J. Huang. Video background extraction based on improved mode algorithm. In *Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on*, pages 331–334.

[57] Ahmed Elgammal. *Background Subtraction: Theory and Practice*, pages 1–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[58] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 1–252 Vol. 2.

[59] P Wayne Power and Johann A Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings image and vision computing New Zealand*, volume 2002, pages 10–11.

[60] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 126–133 vol.2.

[61] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[62] H. Deng, X. Tian, K. Yamazaki, and M. Mori. Line extraction with composite background subtract. In *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, pages 69–69.

[63] M. Haralick Robert. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550, 1987.

[64] H. Kong, H. C. Akakin, and S. E. Sarma. A generalized laplacian of gaussian filter for blob detection and its applications. *IEEE Transactions on Cybernetics*, 43(6):1719–1733, 2013.

[65] Mayank Kabra, Alice A. Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nat Meth*, 10(1):64–67, 2013.

[66] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[67] Markus buehre (2014), functions for the rectangular assignment problem (https://www.mathworks.com/matlabcentral/fileexchange/6543), matlab central file exchange. Retrieved: 2016-04-06.

[68] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice with MATLAB*. Wiley-IEEE Press, 2014.

[69] Tpro. https://drive.google.com/drive/folders/0B2FvzGSXPe3pNEZNcktPbERwSzQ?usp=sharing. Accessed: 2016-12-02.

List of Author's Publications

- **Journal**

1. P. Sirigrivatanawong, S. Arai, V. Thoma, and K. Hashimoto. Multiple Drosophila Tracking System with Heading Direction, *SENSORS*, 2017-01-05.

2. V. Thoma, S. Knapek, S. Arai, M. Hartl, H. Kohsaka, P. Sirigrivatanawong, A. Abe, K. Hashimoto, and H. Tanimoto. Functional dissociation in sweet taste receptor neurons between and within taste organs of Drosophila, *Nature Communications 7*, 2016.

- **Conference**

1. P. Sirigrivatanawong and K. Hashimoto. Multiple Drosophila Tracking with Heading Direction in Crossover and Touching Scenarios. *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016.

2. S. Arai, P. Sirigrivatanawong and K. Hashimoto. Multiple Drosophila tracking and posture estimation algorithm. *2015 International Conference on Informatics, Electronics & Vision (ICIEV)*, 2015.

3. S. Arai, P. Sirigrivatanawong and K. Hashimoto. Control of water resource monitoring sensors with flow field estimation for low energy consumption. *11th IEEE International Conference on Control & Automation (ICCA)*, 2014.

4. P. Sirigrivatanawong and K. Hashimoto. Lake monitoring sensors distribution with low energy consumption. *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013.

Acknowledgments

It would not have been possible to write this dissertation without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to express my special appreciation and thanks to my supervisor *Professor Koichi Hashimoto* who always gives me valuable advises not only the scientific knowledge but also the fundamental of life for living in Japan while doing scientific research. I feel very honored to have the opportunity to participate as part of his laboratory. Being able to do the research with him is a very precious experience for me.

The good advice, support and friendship of *Associate Professor Shingo W. Kagami* has been invaluable on both an academic and a personal level, for which I am extremely grateful.

I am most grateful to *Associate Professor Shogo Arai* for providing me valuable advises and his kindness.

I am deeply thankful to *Professor Hiromu Tanimoto* for his advice and guidance throughout the research. This research would not have been possible without the help and support from him.

For the members of graduate school of life sciences, I would like to thank *Mr. Vladimiros Thoma* for his kind support and advice. Moreover, I would like to show my appreciation to *Mr. Masayoshi Ikarashi* for his hard work and cooperation.

I would also like to thank all of the laboratory members, especially *Mr. Chen Min, Mr. Cherdsak Kingkan, Mr. Li Mingyu, Mr. Owoyemi Toluwaleke Joshua, Mr. Ilya Ardakani, Mr. Naoya Chiba, Mr. Nobuaki Fukuchi, Mr. Zhang Zhenyu* and former members *Mr. Andreas Lennart Pettersson, Mr. Shogo Ito, Mr. Toshiki*

Boshu, and *Mr. Erik O'Riley* for having precious time in the lab together, helping everything when I asked for help. Thank you very much for your kindness, friendship and support.

For the secretary of the laboratory, *Mrs. Kaori Ichinohe*, I would like to express my special thank for helping me with every kind of problems since the time I contacted for entering this laboratory.

A special thanks to my family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that you have made on my behalf.

Last, but by no means least, I would like express appreciation to my friends from Japan, Thailand, China and elsewhere for their support and encouragement.

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.