2013

Target Tracking with Binary Sensor Networks

Mengmei Liu
University of Kentucky, mengmei.liu@uky.edu

Mengmei Liu, Student

Dr. Laurence G. Hassebrook, Major Professor

Dr. Zhi Chen, Director of Graduate Studies

Target Tracking with Binary Sensor Networks

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Electrical Engineering
in the College of Engineering
at the University of Kentucky

By

Mengmei Liu

Lexington, Kentucky

Director: Dr. Laurence G. Hassebrook, Department of Electrical Engineering

Lexington, Kentucky

2013

Abstract of Thesis

Target Tracking with Binary Sensor Networks

Binary Sensor Networks are widely used in target tracking and target parameter estimation. It is more computationally and financially efficient than surveillance camera systems. According to the sensing area, binary sensors are divided into disk shaped sensors and line segmented sensors. Different mathematical methods of target trajectory estimation and characterization are applied.

In this thesis, we present a mathematical model of target tracking including parameter estimation (size, intrusion velocity, trajectory, etc.) with line segmented sensor networks. Software simulation and hardware experiments are built based on the model. And we further analyze how the quantization noise affects the results.

KEYWORDS: Binary Sensor Network, Target tracking, Trajectory, Parameter estimation, Quantization

Mengmei Liu

Feb, 7th, 2013

Target Tracking with Binary Sensor Networks

By

Mengmei Liu

Acknowledgements

First of all, I would like to thank my academic advisor, Dr. Laurence G. Hassebrook, who directed me through the masters thesis research, this is a very interesting project, and I am really grateful for all his guidance and help. This thesis would not be possible without his outstanding knowledge and innovative ideas.

Special thanks to an excellent researcher: Dr. Charles Casey, who developed much of the target tracking and parameter estimation and made remarkable contribution to the mathematical modeling of tracking technique.

I would also like to thank Dr. Robert W. Cohn, Prashant, and Sukanta from University of Lousivelle, who developed the hardware side of the tracking system. They had done a great job in hardware development and provide useful pictures to illustrate the experimentally system. It was a nice time co-working with them and thanks for all the help they provided me during my research.

Thanks to National Institute for Hometown Security, for providing the funding for the optical intrusion project.

Finally, I would like to thank my parents and grandparents for raising me and supporting whatever decision I made out of their love for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Binary Sensor Networks (BSNs) are widely used in target motion tracking and shape recognition. There are BSN systems that can sense the intrusion of objects and characterize their position, trajectory, size and shape. These BSN systems could be used in broad area surveillance because of the low cost and large sensor range of binary sensor compared to camera surveillance while at the same time the set-up complexity and computation cost is reduced significantly.

The basic idea of binary sensor is to output a signal 1 which stands for target present when detecting the presence of an object and to output 0 otherwise. Binary sensors are classified into different types by their sensing area and detection probability. Traditionally, we can divide binary sensor into two categories based on sensing area: (1) disk shaped sensing area and (2) line-segmented sensing area. The former one would have a wide sensor area range which contains the local area of the binary sensor. The shape of the sensor range may be circular or other shape determine on the specific sensor type. And the detection probability may vary with the distance between target and binary sensor. In some BSNs are configured to have a transmitter and receiver, which are linked to detect intrusion between them.

These different types of sensors are used and modelled in papers to enable target tracking, trajectory estimation and target characterization. In this thesis we focus our attention on the line-segmented sensor which is defined by the line path or link between transmitter and receiver.

1.1 Historical background of BSN

Detection of intrusion has long been an important issue not only in military, but also in people's daily lives. With the emergence of BSN, researchers have developed mathematical models and corresponding computing methods to detect and track moving objects. There are two mathematical methods for BSN detection: (1) coded-aperture system, and (2) heavy-ion reconstruction. In a coded-aperture BSN system, a two dimensional mask is used to modulate projections from source points to a detector array, and the aperture is equal to the impulse response [2][3].While heavy-ion methods use strong scattering center as holographic references for phase retrieval in multi-dimensional modulated tomography[2]. The first method is mostly used when lens-fabrication is difficult while the second method approaches in x-ray crystallography. These two methods are the precursor to Reference Structure Tomography (RST). This RST method was first proposed by David J. Brady from Duke University and is used for imaging and tracking objects in the following works.

In [2], D. J. Brady, N. P. Pitsianis, and X. Sun introduced the concept of RST. They introduced a detailed mathematics model of RST and provided the simulation results of applying RST to 2D image reconstruction using MATLAB a propagating field model described by Daubechies wavelet basis.

D.J. Brady et al [2][4] extended RST to 3D image reconstruction and tracking moving objects by changing the structure geometry, spatial segmentation[5] and using different propagation fields to enable different mathematical analysis.

P. Potuluri et al [6] proposed a method by realizing the reference structure with a Hadamard matrix to track the motion of an object. The source, also the moving object is self illuminating which provides the propagating field.

A Sinha and D.J. Brady [1] introduced a statistical model and applied it to random 3D reference structures[7]. They were able to show a mathematical relationship between the measurement statistic state and the size of object. Also, based on the geometry of the projections and using the ideal of convex hull, the RST method using measurement statistics can recover 3D objects shape information.

By choosing different reference structures and propagation field, the RST can be applied for different applications. Thus, U. Gopinathan and D. J. Brady uses coded apertures as reference structure and uses pyroelectric sensor to track the motion of hot object.

Another research group located at the University of Alabama lead by Dr. Q. Hao also uses RST. They constructed a wireless distributed BSN with reference structure by using Fresnel lens arrays and coded masks to realize multiple human tracking and recognition [8]. Their algorithms use hidden Markov models (HMMs) and maximum-likelihood criterion to identify a single humans walking [9][10][11]. They extended their approach to multiple human tracking by forming a more sophisticated tracking algorithm [12][13]. Furthermore, Haos group used the ideal of context feature extraction to improve the data analysis within region of interest [14]. They have also performed a simulation of multi-target tracking in a rectangle area which contains distributed line-segmented active sensor networks [15]. The particle filter technique is used to estimate the trajectories of the moving target in [15].

There are also some groups interested in the use of disk-shaped sensor range of binary sensor networks[16]. In [17], Xuezhi Wang and Bill Moran from University of Melbourne present a Virtual Measurement (VM) approach for multi target tracking using BSN. The sensor simulated in this paper has a sensing range of R and detection probability of 1.

In [18], Hiroshi Saito et al developed a unified equation which can use only the binary information to work out general sensing areas, and then based on target-object shapes to enable the estimation of the parameters of the target object, including the perimeter length, the size and the object angle.

Our group presents a geometry based link network where we solve the link geometry to extract the target trajectory and key parameters.

1.2 Thesis Outline

This thesis consists of 6 chapters. The first chapter introduces wireless BSNs and different kinds of reference structure tomography on binary sensor networks to realize target tracking and estimation. The second chapter describes the background on RST method and wireless binary sensor networks on tracking objects and estimating the object parameters, including shape, trajectory, size and position. Chapter three describes the mathematical model for the deployment of sensor network and the calculation geometry. Chapter four gives the simulation results using the mathematical model described in the previous chapter. Chapter five gives the systematic simulation results and informations about hardware experimental set-up. Chapter six includes conclusion and future work of the research.

Chapter 2

Background on BSN and its application

Binary sensor networks (BSN) are systems which use large numbers of binary sensors to be deployed over a certain area in order to sense events within that particular area. With data processing of BSN signals, we are able to predict many features of the events (intruders), such as size, shape, intrusion velocity, position, etc. An ideal binary sensor would not miss any detection of events or intrusion data, while in reality, because of the limited capacities of the binary sensors in range and precision, there may have misses and false alarms. However, because of the large number of sensors deployed over a BSN, the detection errors can be reduced to an acceptable level.

There are several methods to realize visual tracking and size,space shape recognition from BSN signals. A statistically measured approach can result with an accurate estimate of the size of the intruder object in 2D without forming a physical image [1].

This stochastic method is more robust than deterministic method while it is more time consuming. An expectation-maximization-Bayesian tracking scheme is used to update the current information in contrast to the previous information based on maximum Bayesian criterion [8]. A hidden Markov model (HMM) which uses its emission matrix to represent the objects shape information and its transition matrix to describes the dynamics information can be used as a learning process to estimate the objects shape information [9] thus can be used in human recognition. Particle filtering method which is a non-parametric implementation of Bayesian filter method is also widely used in target tracking algorithm. It can be used in both single target tracking and multi-target tracking in BSN [15]. This method has the advantage of not only avoids the process of data-to-target association and also provides robust performance.

2.1 Introduction to Reference Structure Tomography (RST)

BSN signal processing can be numerically intensive. Thus, the methodology of collecting experimental data becomes an important issue. If we collect the least amount of data in the most efficient way, it may reduce the load of calculation. By using reference structure tomography (RST)[19][20], which uses a reference structure (can be designed in different ways for different functions) to set up a modulation between object space and sensor, the object space is then modulated into regions which can be described by vector or matrix, thus, it becomes easier to track the intruding object into the modulated regions instead of the entire object space. Fig.2-1 is an example of a typical RST set-up.

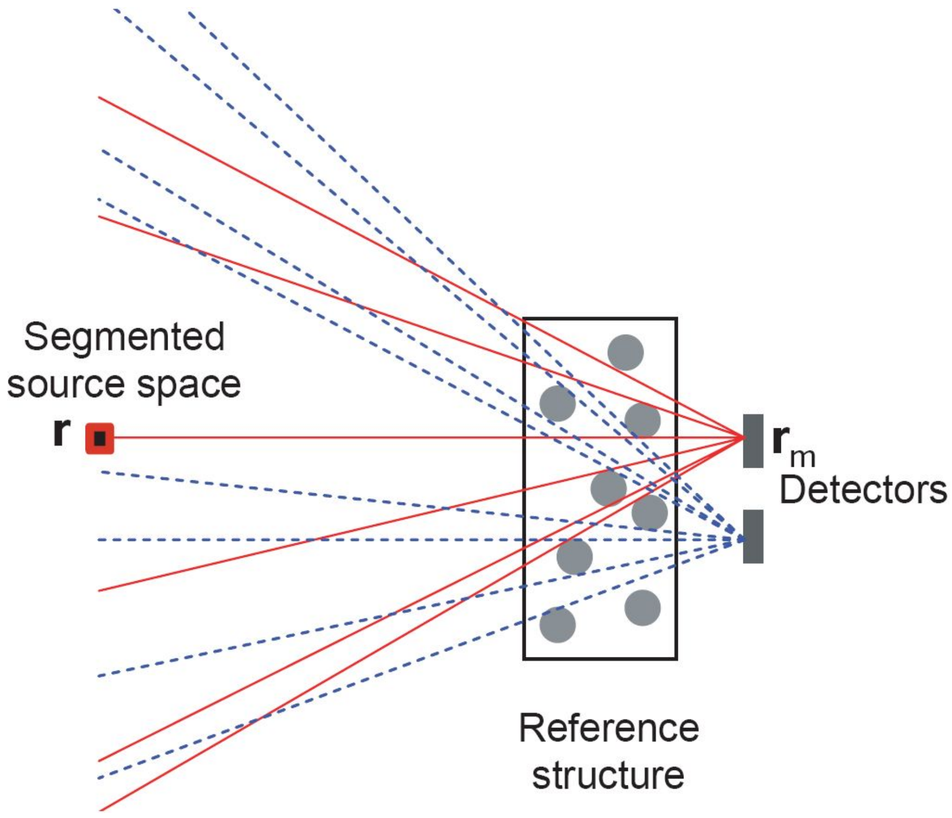


FIGURE 2.1: RST configuration for segmenting the source space([1])

The reference structure in RST can be designed in different ways to enable advanced analysis. For example, it can be designed to implement a Hadamard transformation to track a simple 1D pyroelectric motion[6]. By combining the RST method and different data analysis methodology, it is possible to enable target tracking and feature estimation in BSN networks.

2.2 Optical Intrusion Alarm

As presented in the previous chapters, BSN systems can have many applications in real life scenarios. They can be used for human tracking and identification [9][12] for instance.

In this optical intrusion alarm (OIA) project, BSN system is used to enable broad area surveillance. This application is very critical for homeland security usage. For example, the system could be set up on the perimeter surrounding of critical infrastructures like nuclear power plants, reservoirs, even the U.S. borders to detect unwanted human intrusions.

2.2.1 Goal of OIA

The ultimate goal of OIA is to set up a surveillance system to be able to sense the intrusion by humans and vehicles including their position, trajectory, and size under conditions of low visibility, like fog, smoke or fire. Because the area to be used is usually a broad area with fixed infrastructure, it would be too expensive to set up camera surveillance systems everywhere. The BSN system is an ideal solution which is a low cost network with highly sensitive optical transmitters and receivers.

2.2.2 Laboratory scale model

Before approaching the final goal, the preliminary laboratory simulation is set up to demonstrate the concept and verify the feasibility of the project.[cite University of Louisville, Robert W.Cohn et al] The laboratory scale model is scaled by a factor of 1/13.16 to a real world scenario in the parameters of test range, dimensions and velocities. The shortest path length between one receiver and transmitter is 15m and the object length is 0.5m in the real world scenario while the laboratory chamber size is

114cm*100cm*20cm approximately. The intrusion object width is scaled from 0.5m to 4cm. The chamber is shown in Fig 2.2:

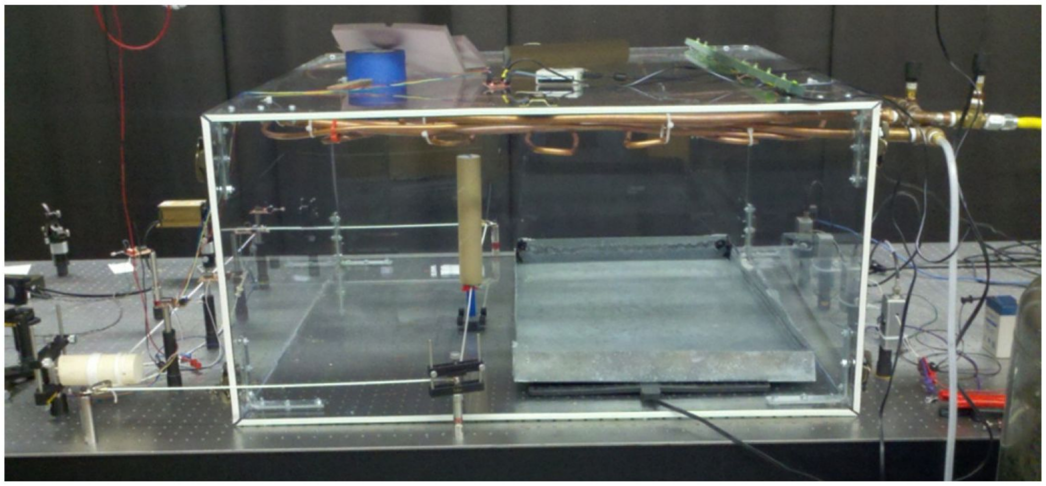


Figure 2.2: Scale model and fog chamber by University of Louisville

In the right side and left side of the chamber are the receivers and transmitters. The plexiglass chamber is an exclusive chamber which can contain fog to ensure different level of signal testing. There is a laser in the back to provide an sequence estimate of visibility. The next figure is a closer look of transmitters (LED emitters) and receivers (photo detector).

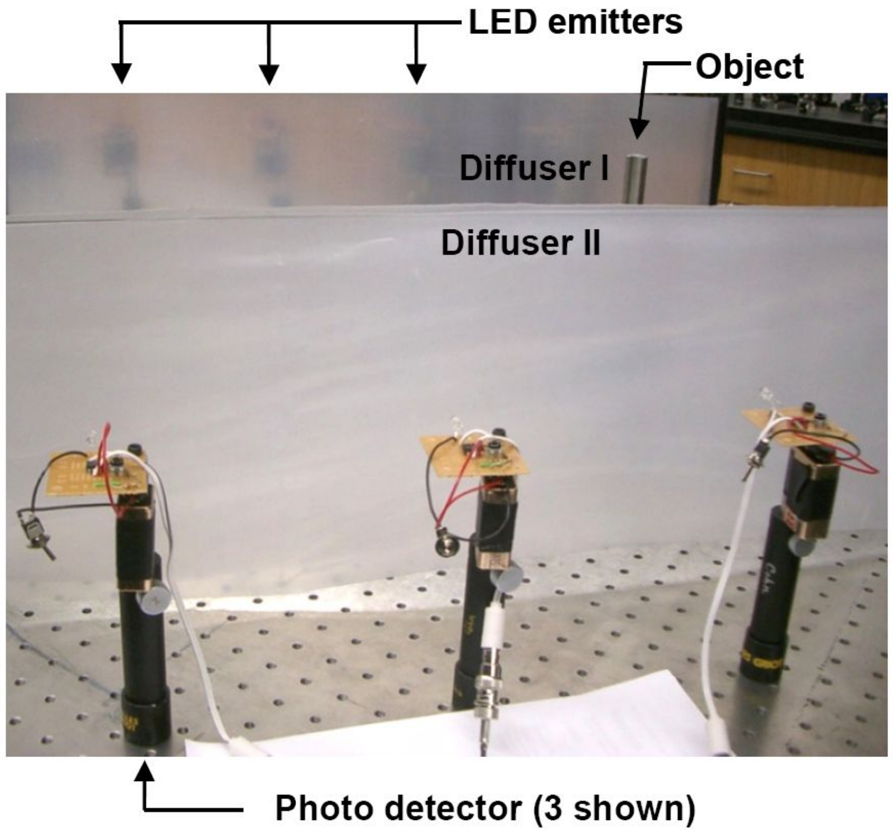


Figure 2.3: LED emitter and Photo detector

For each of the four LED emitters is modulated by voltage at four distinct frequencies and each detector is able to detect all frequencies using digital filters. Once the object intrudes through the line of sight, the strong intensity changes among the links would locate the object in 2D space. A simple illustration of the detection process is shown in the following Fig 2.4:

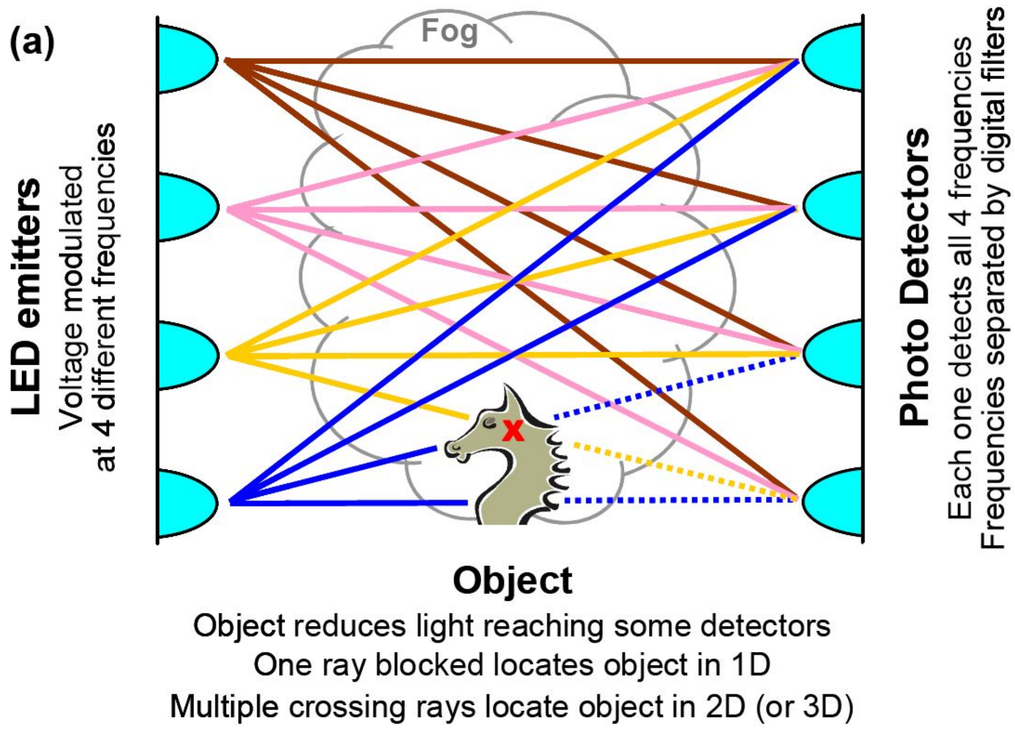

(a)
Fog
LED emitters
Voltage modulated at 4 different frequencies
Photo Detectors
Each one detects all 4 frequencies
Frequencies separated by digital filters
x
Object
Object reduces light reaching some detectors
One ray blocked locates object in 1D
Multiple crossing rays locate object in 2D (or 3D)

FIGURE 2.4: A simple illustration of 2D detection

As is the same for the laboratory model during the detection process, once the object breaks the line of sight between the emitter and detector pairs, one detection is made and data is sent back to the microprocessor. As is shown in Fig 2.5:

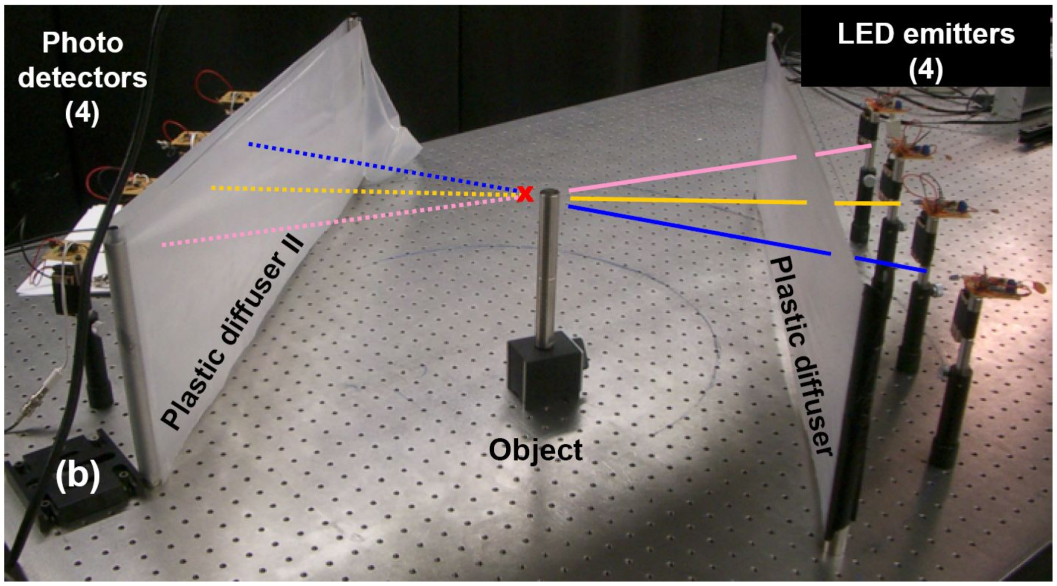


FIGURE 2.5: One detection of object

A view of detection displays from the labview computer interface can be seen in Fig 2.6, the two white lines are the ones shown change of intensity which locates the object (marked by red circle) at the intersection. The blue lines show the reference intensity.

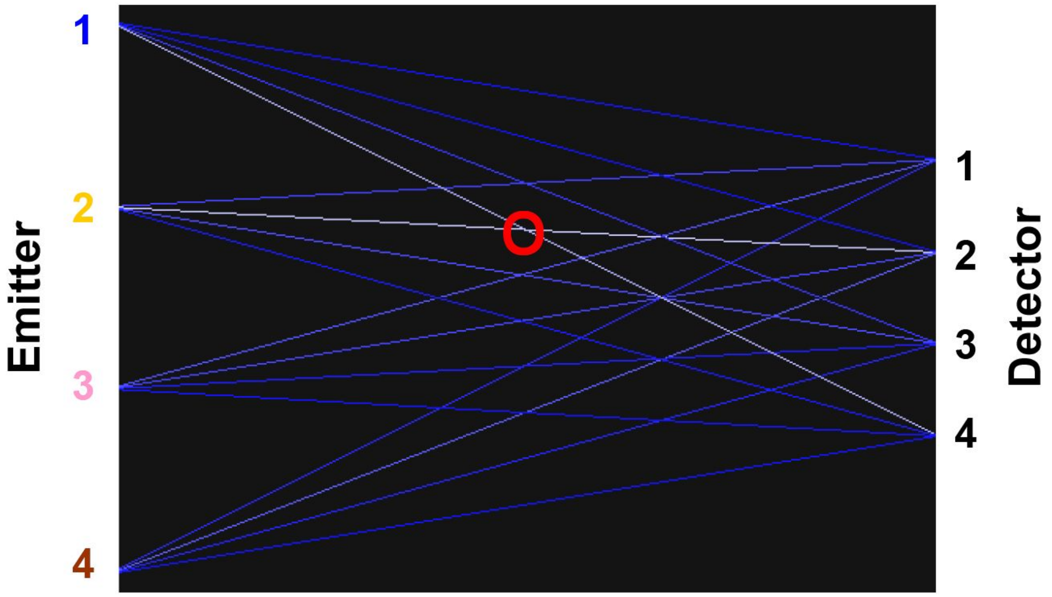


FIGURE 2.6: Display of detection

The above is a brief introduction of what OIA is and what it does. A full project requires system hardware setup, software setup, calculation of geometries and signal processing under the fog condition. In this thesis we focus our study on processing the temporal link breaks in order to estimate target size and first order trajectory.

Chapter 3

Calculation Process of Target Characteristics

In the OIA system, the OIA sensor monitoring software will generate an intrusion detect signal when the links (which is the line paths between the transmitter and the receiver) are interrupted by an intruder. The signal is depicted as inverted such that a high-level (or 1) when intruded versus a low-level (or 0) when not intruded with a width of the time period of intrusion. A depiction of the system can be shown in the cited figures. In all the following figures, small blue circles represent the sensors while the large purple circle represents the object. The blue line represents the object's trajectory with the arrow indicating direction. The dark lines connecting each sensor pair represents the links.

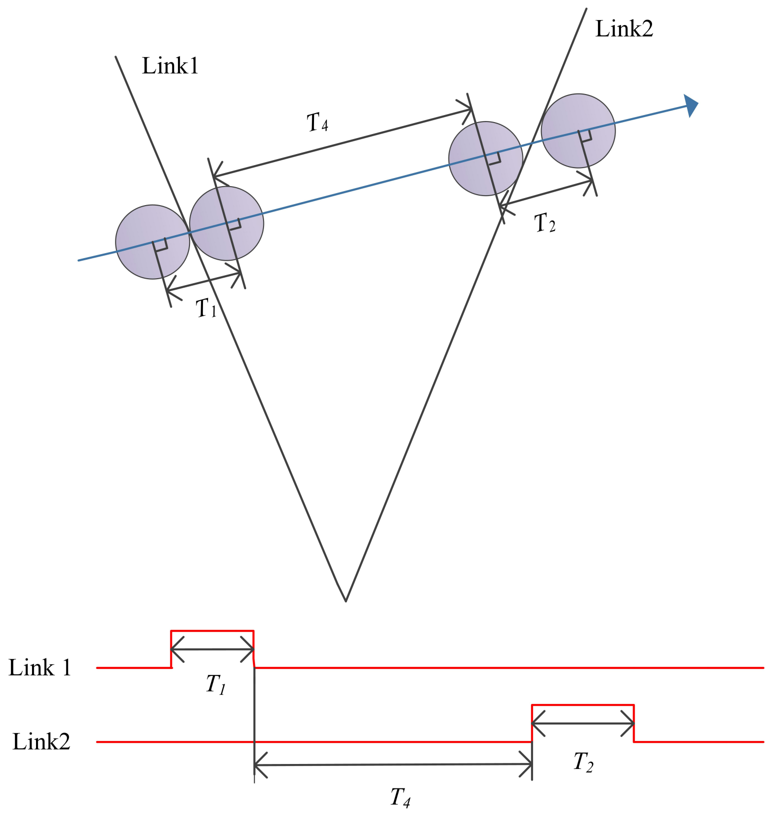


FIGURE 3.1: Depiction of intrusion signals

From Fig 3.1, we model the intruder with a circular shape with speed, diameter, position and direction parameters. The parameter calculation begin when the intrusion time T_1 and T_2 and the vacancy time T_4 are known. It will be shown in the following calculations that, under sufficient intrusions and with the knowledge of the link pathway geometries, we will be able to calculate and continually update the intruder diameter and trajectory.

3.1 Calculation of intrusion intersection angle

First, consider the following geometry and intruder path shown in Fig 3.2:

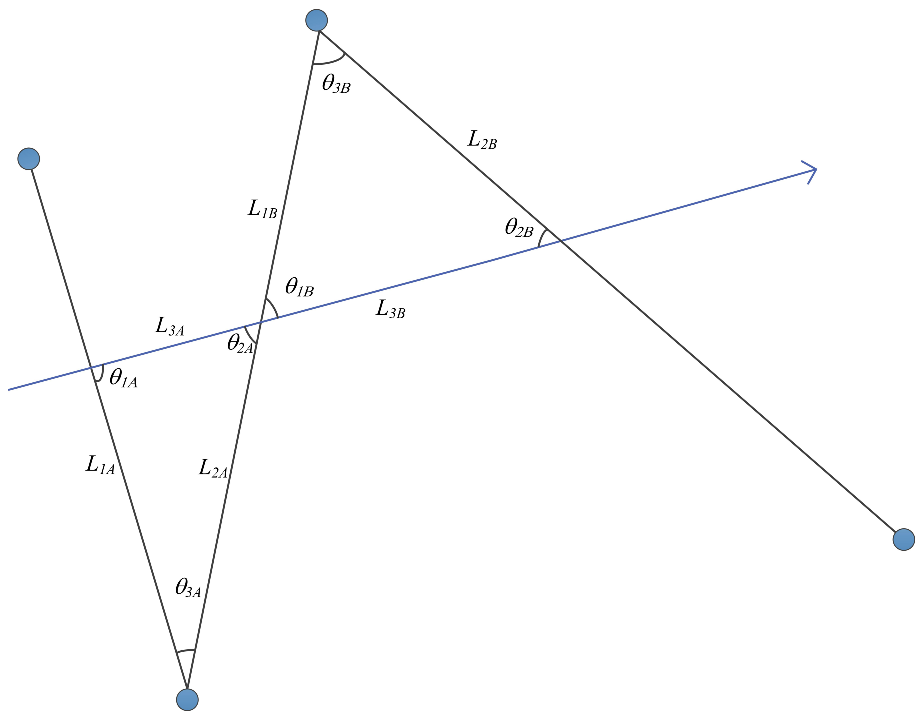


Figure 3.2: Triangles formed by 3 links and 1 intruder paths

A subset of Fig 3.2 geometry (i.e. 2 links)is shown in Fig 3.3:

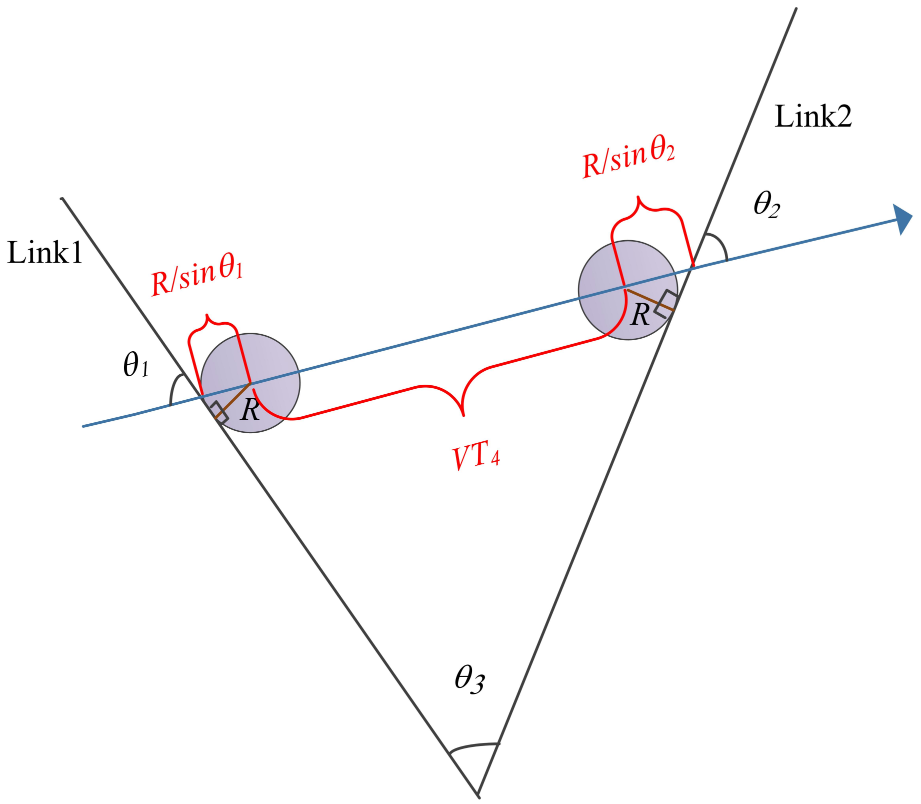


FIGURE 3.3: Intruder path length relationship

From Fig 3.3, we can determine the following relationships:

$$\frac{2R}{\sin\theta_1} = VT_1 \tag{3.1}$$

$$\frac{2R}{\sin\theta_2} = VT_2 \tag{3.2}$$

where V represents the intruder speed, R represents the intruder radius and θ is the intersect angle of the trajectory path and the sensor link. We solve for θ_1 using these two equations along with the following equation set:

$$\frac{T_1}{T_2} = \frac{\sin\theta_2}{\sin\theta_1} \tag{3.3}$$

$$\theta_2 = (\pi - \theta_3) - \theta_1 = \theta_x - \theta_1 \tag{3.4}$$

$$\theta_1 = \tan^{-1} \frac{\sin \theta_x}{\frac{T_1}{T_2} + \cos \theta_x} \tag{3.5}$$

Because θ_3 is a known parameter once the sensor set-up is completed, we can calculate the intruder path intersection angles each time by the arrangement of the links and detection period.

3.2 Calculation of intrusion speed

After knowing the intruder path intersection angle, we can further solve for the intrusion speed V. From Fig 3.3, we can form the following equation if we denote the intrusion trajectory length between Link1 and Link2 as L_3:

$$L_3 - R(\frac{1}{\sin \theta_1} + \frac{1}{\sin \theta_2}) = VT_4 \tag{3.6}$$

$$\frac{L_3}{R} = (\frac{1}{\sin \theta_1} + \frac{1}{\sin \theta_2}) + (\frac{V}{R})T_4 \tag{3.7}$$

From Eq.(3.1):

$$R = VT_1(\frac{\sin \theta_1}{2}) \tag{3.8}$$

$$\frac{V}{R} = \frac{2}{T_1 \sin \theta_1} \tag{3.9}$$

Substituting Eq.(3.8) into Eq.(3.6):

$$L_3 - V(\frac{T_1 \sin\theta_1}{2})(\frac{1}{\sin\theta_1} + \frac{1}{\sin\theta_2}) = VT_4 \tag{3.10}$$

This allows one to solve for the speed V as

$$V = L_3/(T_4 + (\frac{T_1 \sin\theta_1}{2})(\frac{1}{\sin\theta_1} + \frac{1}{\sin\theta_2})) \tag{3.11}$$

Substituting Eq.(3.9) into Eq.(3.7):

$$\frac{L_3}{R} = (\frac{1}{\sin\theta_1} + \frac{1}{\sin\theta_2}) + (\frac{2}{T_1 \sin\theta_1})T_4 \tag{3.12}$$

Eq.(3.12) gives a critical relationship we can used for further analysis and Eq.(3.11) yields the intrusion speed.

3.3 Calculation of link-path intersection length

Using Eq.(3.11), we can solve for intrusion speed V. However, this equation requires the detection period T_1, the vacancy period T_4, the angle formed by the BSN geometry θ_1,θ_2, and the link-path intersection length L_3. The first four parameters are known based on the BSN geometry and sensor detection. However, L_3 is not a known parameter because we do not know where the object is going to intrude. Thus we need to calculate L_3 to solve for V. To calculate L_3 we need the timing signals for three continuous link intrusions. The calculation can be processed by using the geometric sine law of triangle as in the following: Applying Eq.(3.12) to Fig 3.2, we can form the following relationship, denoted as G.

$$G = \frac{L_{3B}}{L_{3A}} = \frac{(\frac{1}{\sin\theta_{1B}} + \frac{1}{\sin\theta_{2B}}) + (\frac{2}{T_{1B}\sin\theta_{1B}})T_{4B}}{(\frac{1}{\sin\theta_{1A}} + \frac{1}{\sin\theta_{2A}}) + (\frac{2}{T_{1A}\sin\theta_{1A}})T_{4A}} \tag{3.13}$$

Also, from the triangle geometric law of sine, we have:

$$\frac{L_{3B}}{L_{3A}} = \frac{(\frac{\sin\theta_{3B}}{\sin\theta_{2B}})L_{1B}}{(\frac{\sin\theta_{3A}}{\sin\theta_{1A}})L_{2A}} \tag{3.14}$$

Denote the ratio $(\frac{\sin\theta_{3B}}{\sin\theta_{2B}})/(\frac{\sin\theta_{3A}}{\sin\theta_{1A}})$ as $1/F$, thus the multiplication of G and F becomes:

$$GF = \frac{L_{1B}}{L_{2A}} \tag{3.15}$$

Because $L_{1B} + L_{1A}$ is equal to the link distance D, which is determined by the sensor geometry, as in Fig 3.4

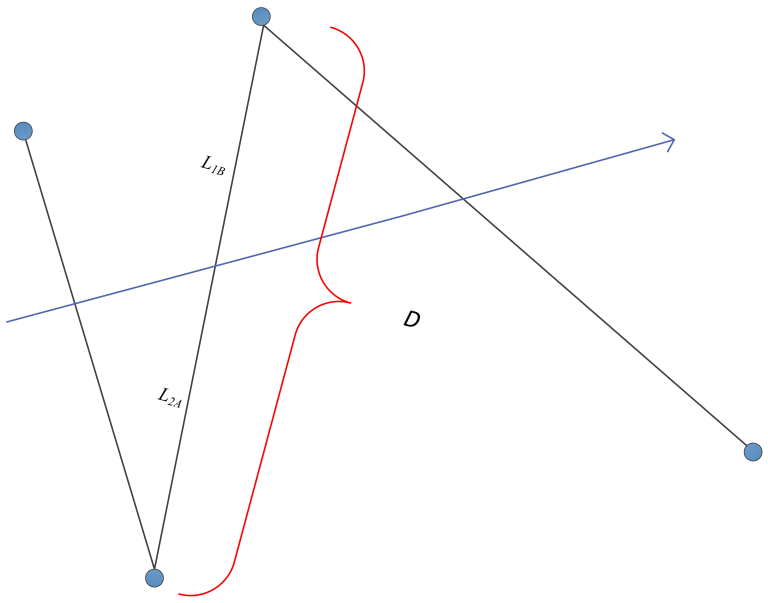


FIGURE 3.4: Link intersection distance

$$D = L_{2A} + L_{1B} \tag{3.16}$$

G and F are known ratio based on the BSN network, thus we can solve for L_{1B} by combining equation Eqs.(3.15) and (3.16):

$$L_{1B} = D(\frac{GF}{1+GF}) \tag{3.17}$$

$$L_{2A} = D(\frac{1}{1+GF}) \tag{3.18}$$

Using the triangle law of sine combined with L_{1B} and L_{2A}:

$$L_{3B} = (\frac{\sin\theta_{3B}}{\sin\theta_{2B}})L_{1B} \tag{3.19}$$

$$L_{3A} = (\frac{\sin\theta_{3A}}{\sin\theta_{1A}})L_{2A} \tag{3.20}$$

As shown in Fig 3.5, wherever the object intrudes in to the BSN network, after we calculated the intrusion intersection angle (see section 3.1), we determine the adjacent path length ratio G. Then knowing the network geometry, we could further get F. D is known for every link, thus, we could solve for L_{1B} and subsequently L_{2A}. Then we can get the L_{3B} and L_{3A} (see red and green line in Fig 3.5), thus we can substitute in to Eq.(3.11) to get V for either the first to second link segment or the second to third segment.

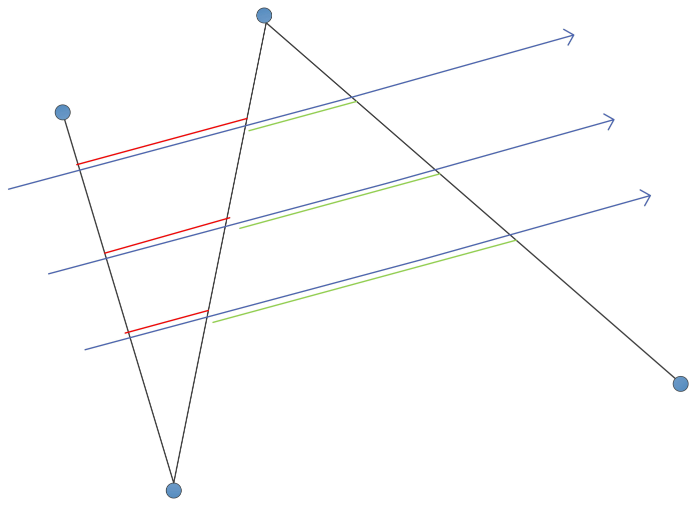


FIGURE 3.5: Significance of length ratio

3.4 Calculation of final position

In the previous sections 3.1, 3.2 and 3.3, we demonstrate and develop a method to calculate size R, intersection angle θ and speed V based on the BSN network. We also calculate the final object position by using vector equations.

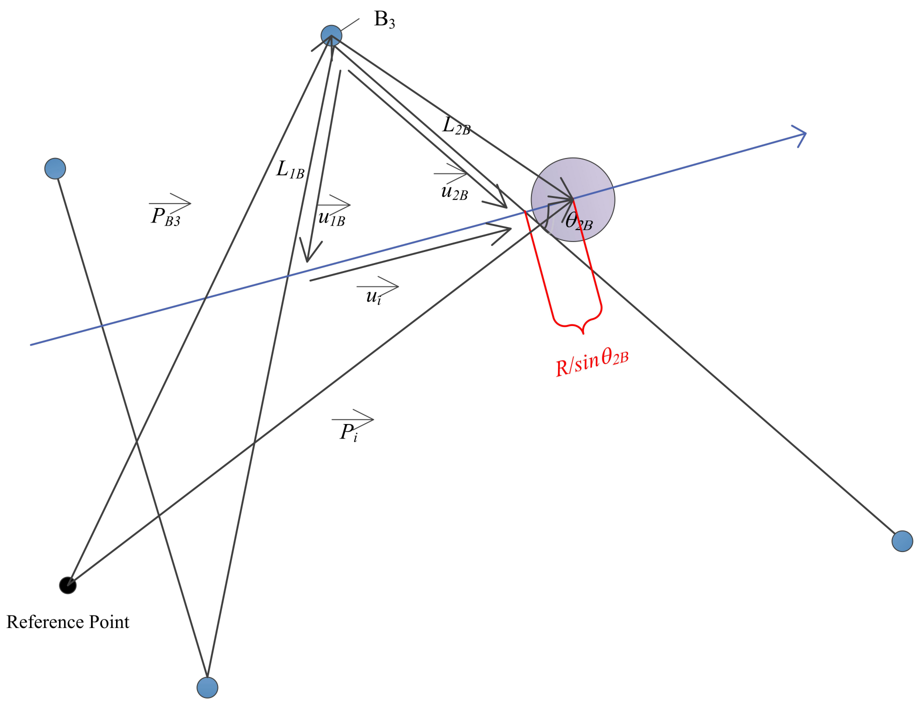


FIGURE 3.6: Final intruder position

From Fig 3.6, if we define $\overrightarrow{u_{1B}}$ and $\overrightarrow{u_{2B}}$ as the unit vector in the direction of $\overrightarrow{L_{1B}}$ and $\overrightarrow{L_{2B}}$, we have:

$$\overrightarrow{L_{2B}} = L_{2B}\overrightarrow{u_{2B}} \tag{3.21}$$

$$\overrightarrow{L_{1B}} = L_{1B}\overrightarrow{u_{1B}} \tag{3.22}$$

Then the intruder vector is:

$$\overrightarrow{I} = \overrightarrow{L_{2B}} - \overrightarrow{L_{1B}} \tag{3.23}$$

Define $\overrightarrow{u_i}$ as intruder direction unit vector,

$$\overrightarrow{u_i} = \frac{\overrightarrow{I}}{\left|\overrightarrow{I}\right|} \tag{3.24}$$

Then the final position which is defined by the centre of the intruder is:

$$\overrightarrow{P_i} = \overrightarrow{L_{2B}} + (\frac{R}{\sin\theta_{2B}})\overrightarrow{u_i} + \overrightarrow{P_{B3}} \tag{3.25}$$

$\overrightarrow{P_{B3}}$ is the position of B3 with relative to the reference point. The reference point can be defined as any fixed point. We randomly choose a reference point on Fig 3.6. Thus, $\overrightarrow{P_i}$ would represent the final position to the reference position.

3.5 Summarize

To summarize this chapter, we build a graph which presents a sequence of four sensors, three links, and three times of intrusions to calculate the demanding parameters, shown in Fig 3.7.

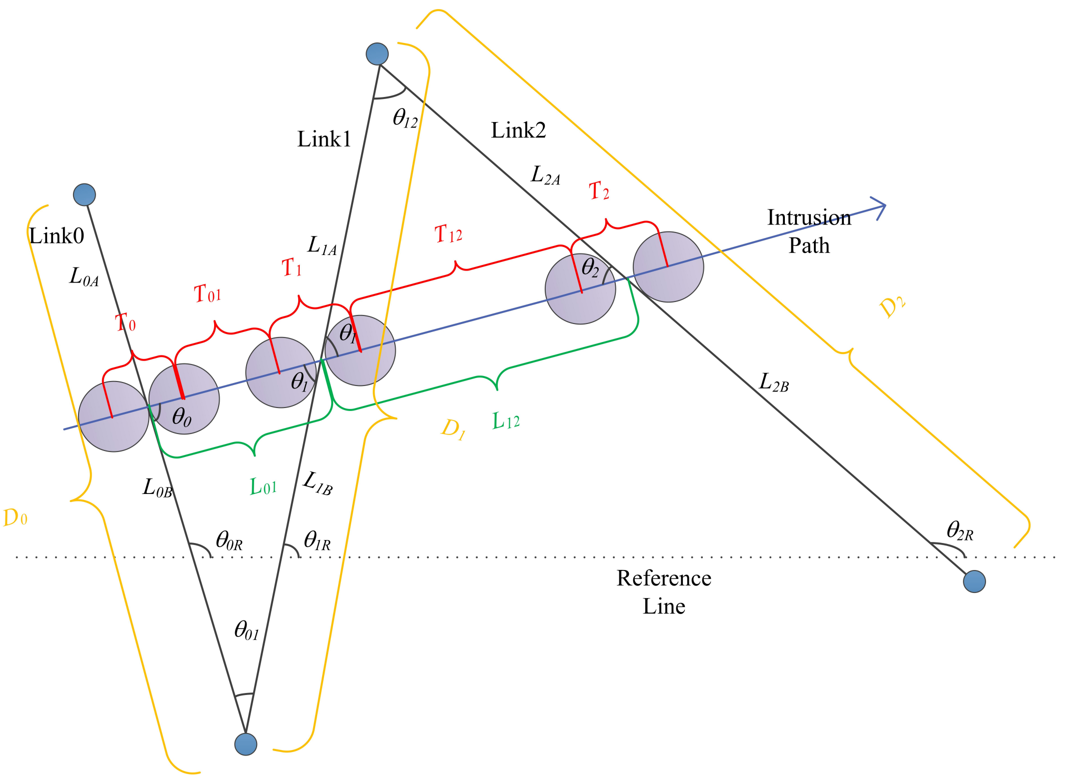


FIGURE 3.7: Summarize of calculation process

In this figure, the links are defined as a sequence of Link0, Link1, Link2, the intrusion time period subsequently are T_0, T_1, T_2. The object times between the links are T_{01} and T_{12}. The links are segmented by the intrusion path into a two line segments which are numbered as L_{0A}, L_{0B}, L_{1A}, L_{1B}, L_{2A}, L_{2B} in sequence. And the intrusion path length segmented by the links are numbered as L_{01}, L_{02}. The internal angle formed by the links are marked as θ_{01}, θ_{12} while the internal angle formed by the intersection of links and intrusion path are marked as θ_0, θ_1, θ_2. We also create a reference line which is horizontal as to define the absolute angle of the links as the upper-right angle of the intersection of the links and the reference line which are marked as θ_{0R}, θ_{1R}, θ_{2R}. Using this we can locate the value of θ_{01}, θ_{12}. And the total length of each links are marked as D_0, D_1, D_2.

Using the equations developed from the previous sections, we can summarize the calculations:

The link geometry angles and reference angles are known such that:

$$\theta_{01} = |\theta_{0R} - \theta_{1R}| \tag{3.26}$$

$$\theta_{12} = |\theta_{1R} - \theta_{2R}| \tag{3.27}$$

The intersection angles are:

$$\theta_0 = \tan^{-1} \frac{\sin\theta_{01}}{\frac{T_0}{T_1} - \cos\theta_{01}} \tag{3.28}$$

$$\theta_1 = \tan^{-1} \frac{\sin\theta_{01}}{\frac{T_1}{T_0} - \cos\theta_{01}} = \tan^{-1} \frac{\sin\theta_{12}}{\frac{T_1}{T_2} - \cos\theta_{12}} \tag{3.29}$$

$$\theta_2 = \tan^{-1} \frac{\sin\theta_{12}}{\frac{T_2}{T_1} - \cos\theta_{12}} \tag{3.30}$$

where T_0, T_1, and T_2 are known. The segmented intrusion path ratio is:

$$G = \frac{L_{12}}{L_{01}} = \frac{(\frac{1}{\sin\theta_1} + \frac{1}{\sin\theta_2}) + (\frac{2}{T_1 \sin\theta_1})T_{12}}{(\frac{1}{\sin\theta_0} + \frac{1}{\sin\theta_1}) + (\frac{2}{T_0 \sin\theta_0})T_{01}} \tag{3.31}$$

The segmented intrusion path lengths are:

$$L_{01} = D_1 \frac{1}{\frac{\sin\theta_0}{\sin\theta_{01}} + G\frac{\sin\theta_2}{\sin\theta_{12}}} \tag{3.32}$$

$$L_{12} = D_1 \frac{G}{\frac{\sin\theta_0}{\sin\theta_{01}} + G\frac{\sin\theta_2}{\sin\theta_{12}}} \tag{3.33}$$

The speed inside each segmented intrusion path are:

$$V_{01} = \frac{L_{01}}{T_{01} + (\frac{T_0}{2} + \frac{T_0}{2}\frac{\sin\theta_0}{\sin\theta_1})} \tag{3.34}$$

$$V_{12} = \frac{L_{12}}{T_{12} + (\frac{T_1}{2} + \frac{T_1}{2}\frac{\sin\theta_1}{\sin\theta_2})} \tag{3.35}$$

The object diameter is calculated as:

$$R_0 = \frac{VT_0\sin\theta_0}{2} \tag{3.36}$$

$$R_1 = \frac{VT_1\sin\theta_1}{2} \tag{3.37}$$

$$R_2 = \frac{VT_2\sin\theta_2}{2} \tag{3.38}$$

The following figure is the corresponding graph to determine the positions:

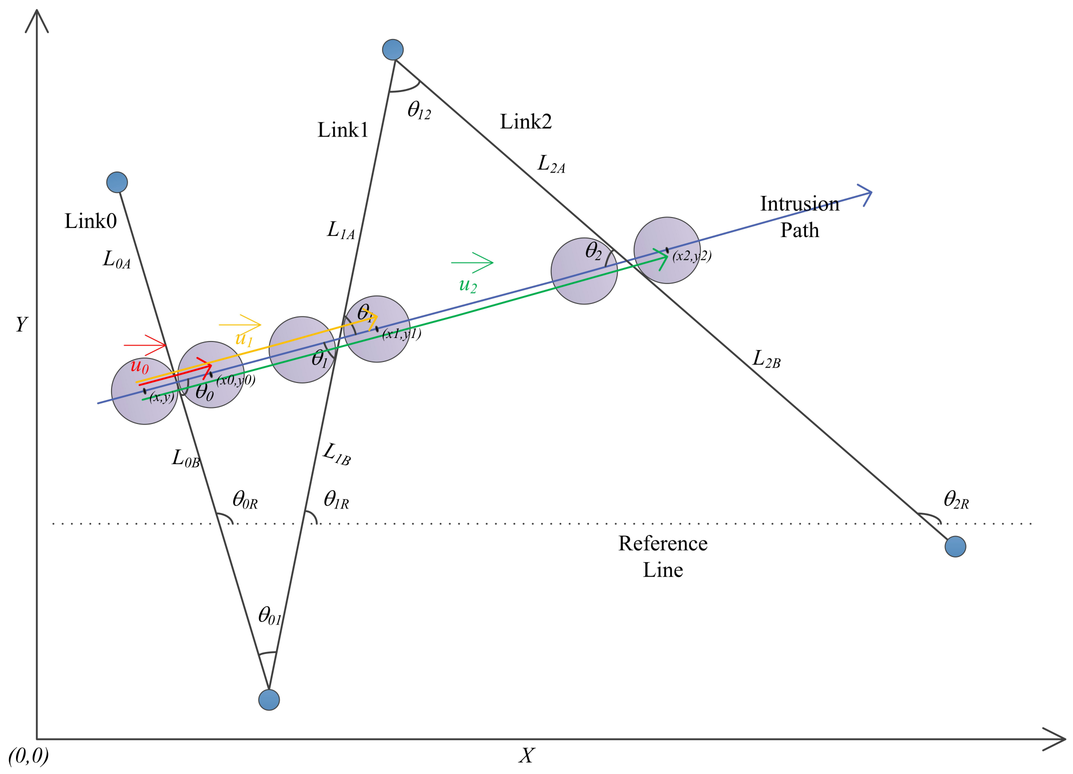


FIGURE 3.8: Positions of the object after each detection

Define $\overrightarrow{u_i}$ as intruder direction unit vector, $\overrightarrow{u_0}$, $\overrightarrow{u_1}$, $\overrightarrow{u_2}$ as the position vector after the first, second, third detection of intrusion with reference to the initial position.

$$\overrightarrow{u_0} = \frac{2R}{\sin\theta_0}\overrightarrow{u_i} \tag{3.39}$$

$$\overrightarrow{u_1} = (\frac{R}{\sin\theta_0} + L_{01} + \frac{R}{\sin\theta_1})\overrightarrow{u_i} \tag{3.40}$$

$$\overrightarrow{u_2} = (\frac{R}{\sin\theta_0} + L_{01} + L_{12} + \frac{R}{\sin\theta_2})\overrightarrow{u_i} \tag{3.41}$$

Then we define a rectangular coordinate system as shown in Fig 3.8 and with the knowing of initial position (a, b), we could locate all the coordinates after each intrusion. The position of object is defined as the central point of the object in this thesis.

After break Link0, the coordinates of the object position is (x_0, y_0):

$$x_0 = \frac{2R}{\sin\theta_0}\cos(\theta_0 + \theta_{0R} - \pi) + a \tag{3.42}$$

$$y_0 = \frac{2R}{\sin\theta_0}\sin(\theta_0 + \theta_{0R} - \pi) + b \tag{3.43}$$

After break Link1, the coordinates of the object position is (x_1, y_1):

$$x_1 = (\frac{R}{\sin\theta_0} + L_{01} + \frac{R}{\sin\theta_1})\cos(\theta_0 + \theta_{0R} - \pi) + a \tag{3.44}$$

$$y_1 = (\frac{R}{\sin\theta_0} + L_{01} + \frac{R}{\sin\theta_1})\sin(\theta_0 + \theta_{0R} - \pi) + b \tag{3.45}$$

After break Link2, the coordinates of the object position is (x_2, y_2):

$$x_2 = (\frac{R}{\sin\theta_0} + L_{01} + L_{12} + \frac{R}{\sin\theta_2})\cos(\theta_0 + \theta_{0R} - \pi) + a \tag{3.46}$$

$$y_2 = (\frac{R}{\sin\theta_0} + L_{01} + L_{12} + \frac{R}{\sin\theta_2})\sin(\theta_0 + \theta_{0R} - \pi) + b \tag{3.47}$$

We can also estimate the target trajectory with the initial position (a, b) after knowing object speed V. First, we decompose V into V_x and V_y:

$$V_x = V\cos(\theta_0 + \theta_{0R} - \pi) \tag{3.48}$$

$$V_y = V\sin(\theta_0 + \theta_{0R} - \pi) \tag{3.49}$$

Then the target trajectory is:

$$x(t) = a + V_x t \tag{3.50}$$

$$y(t) = b + V_y t \tag{3.51}$$

The above is the summary of this chapter with intrusion of three links. We can expand it into n links and update the information of the object in sequence which will be illustrated in the following chapters.

Chapter 4

Simulation

In Chapter 3, we described the calculation process of target characteristics in detail, in which we are mainly interested in object velocity, diameter and trajectory. In this chapter, we develop MATLAB program to simulate the object intrusion and calculate the characteristics using the equations given in the previous chapter and compare the results of the simulation at different parameter resolutions. There are two approaches that we can take to simulate the OIA system. One approach is to determine the continuous space link boundaries, padded by the radius of the object. There by using the continuous time trajectory of a 0 radius point we can determine precisely the time of each occlusion. However, this approach does not allow for more complicated network shapes and is complicated to implement. So a second approach is a pixel based simulation where the OIA space is a discrete space and time whose resolutions are variable. There are two binary markers representing the link geometry and the object at a specific time in its trajectory. The two binary markers are the same dimensionality and when 'Anded' together pixel wise will only be non zero when there is an occlusion. This 'Anding' is performed at small enough time increments and high enough spatial sampling as to approximate the continuous space/time model. We will refer to this as pseudo-space/time. In the ideal case, if the sample we took is sufficiently large, the error should be neglected. However, that requires a large amount of memory. So we evaluate how much space and time resolution is needed to simulate the OIA system effectively.

4.1 System set-up of modelling

From Chapter 3, we know that at least four sensors with three links are needed to ensure one calculation. Thus, in the simulation model, we deploy four sensors in a square with 1 (m) width and 1 (m) length. The two transmitters are deployed in one side of the square with locations of 0.162 (m) and 0.8 (m) while the two receivers are deployed in the other side with locations of 0.162 (m) and 0.8 (m). The three links are formed as seen in Fig 4.1. Note that the arrangements of the links form two rectangular triangle with height r equals 1 (m) and width d equals 0.638 (m) while the simulation can be done as long as links form triangle as stated in Chapter 3.

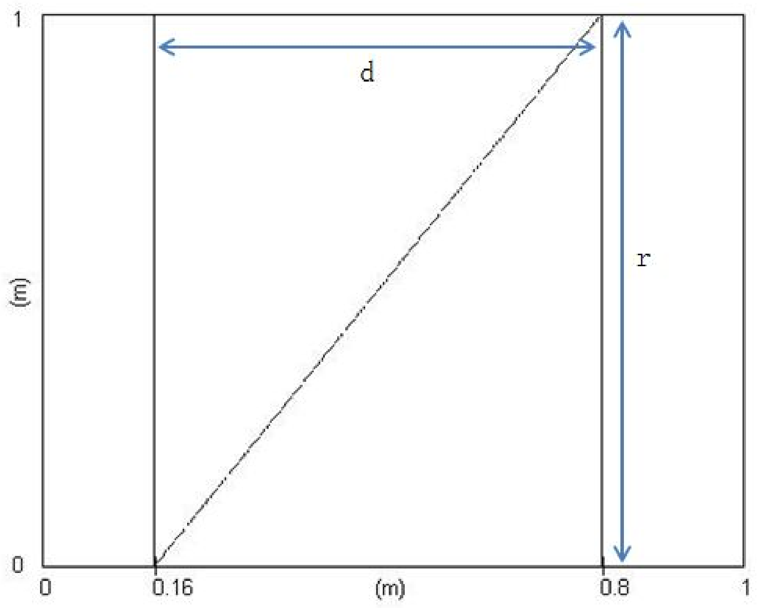


Figure 4.1: Simulation model of sensor deployment

The intrusion object is modelled as a dark circle with diameter of 0.02 (m) and it intrudes with a velocity of 0.04 (m/s) in the direction of sensor deployment and 0.02(m/s) perpendicular to the direction of sensor deployment. From Fig 4.1, if we define these two directions as X and Y with the original point in the lower left of the figure and with X axis points to the right and Y axis points to the upside, then the initial position of the object is (0,0.4) (m). The initial position and the trajectory of the object is shown in Fig 4.2:

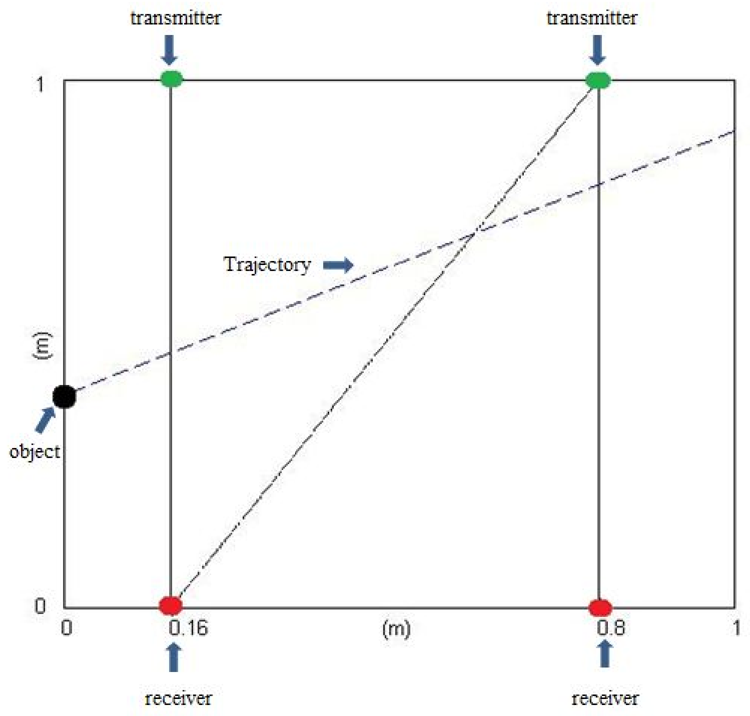


FIGURE 4.2: Initial position of the intrusion object

4.2 Simulation analysis with pseudo-continuous case

As we have discussed before, if our spatial and temporal sampling periods are small enough then the simulation parameter estimation converges to the correct values. By running the simulation, we find that if we sample 2000 times per 1 (m) in space and sample 100 times per 1 (s) in time, which means 0.05 cm per sample in space domain and 0.01 sec per sample in time domain, the result is "good enough": the measurement of R is scaled to 0.0202 (m); the speed in X is 0.04 (m/s) which is the same as scaled actual speed, and the speed in Y is 0.0198 (m/s). By saying that the result is "good enough", we mean the estimation error is very small, which will be show on Table 4.1. In this case the average error is less than 1% for V_s, V_x, V_y and R.

The intrusion detection signal is as follows, in which we document the sampling period as the follows: second per sample in time domain as $sec/sample_t$ and centimetre per sample in space domain as $cm/sample_s$.

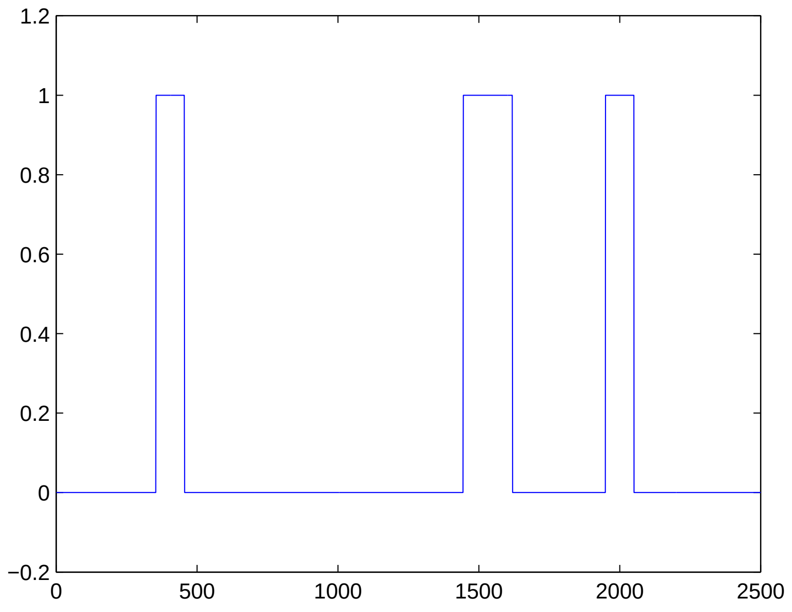


FIGURE 4.3: Intrusion detected with 0.01 $sec/sample_t$ and 0.05 $cm/sample_s$

The estimation results is given in the following table:

TABLE 4.1: Estimation results with best case scenario

	$PseudoContinuous$	$Actual$	$Error$
$Vs(m/s)$	0.0446	0.0447	0.0022
$Vx(m/s)$	0.0400	0.0400	0.0000
$Vy(m/s)$	0.0198	0.0200	0.0113
$R(m)$	0.0202	0.0200	0.0100

In this table, Vs represents the speed of the object, Vx and Vy represents the value of object speed decomposed in X direction and Y direction, from which we are able to calculate the angle of the object. Pseudo-continuous values refers to the results from the experiment and actual values refers to the actual parameter setting values. Because this experiment gives very accurate results (better than 1% accuracy), all the ideal values mentioned later refer to this set of results. And the error is calculated using the following equation:

$$Error = \left| \frac{PseudoContinuousValue - ActualValue}{ActualValue} \right| \tag{4.1}$$

And we define the total average error as:

$$AverageError = \frac{(ErrorVs + ErrorVx + ErrorVy + ErrorR)}{4} \tag{4.2}$$

Thus in this case, total average error is 0.0059, which is smaller than 1%.

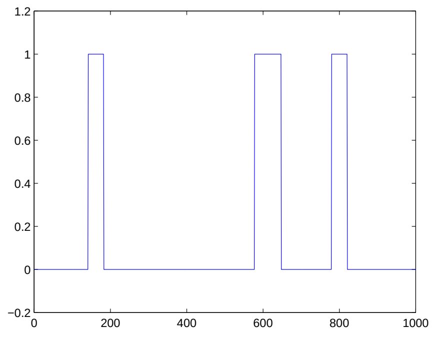
4.3 Simulation analysis with different quantization level

In the previous section, we showed the result of simulation with sufficient sample density would lead to better than 1% accuracy, while in this section, we want to discuss what happens when we decrease sample density. We discuss the case in time domain and space domain separately in the following experiments.

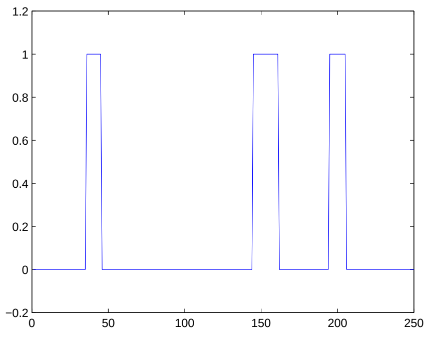
4.3.1 Simulation of decreased sample rate in time domain

In the first set of experiments, we want to see how the different sample rates of time would affect the results, thus we keep the space in high resolution, which is to take one sample every 0.05 (cm) in space domain, and we decrease the number of samples taken in the time domain. The time samples we take are 1000, 250, 50, 25 per 25 seconds, which gives sample periods of 0.025 (s), 0.1 (s), 0.5 (s), and 1 (s) in time domain.

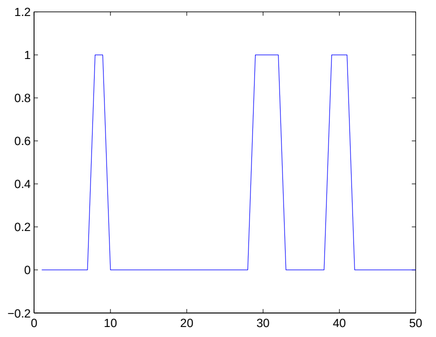
The intrusion detected timing graph is as follows:



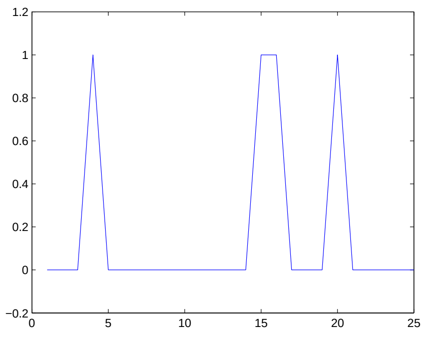
(a) Intrusion detected with 0.025 $sec/sample_t$



(b) Intrusion detected with 0.1 $sec/sample_t$



(c) Intrusion detected with 0.5 $sec/sample_t$



(d) Intrusion detected with 1 $sec/sample_t$

FIGURE 4.4: Intrusion detected timing graph

The result is shown in the following table and with comparison to the most accurate case scenario (Note that the first row of results are the ideal values of all experiments) :

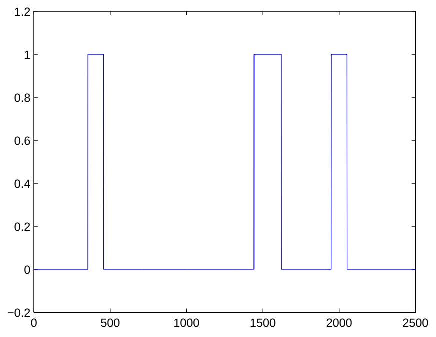
Table 4.2: Estimation results with different quantization level in time

$SamplePeriod(s)$	0.01	0.025	0.1	0.5	1
$Vs(m/s)$	0.0446	0.0443	0.0439	0.0468	0.0473
Vs_{Error}	0.0022	0.0085	0.0173	0.0461	0.0575
$Vx(m/s)$	0.0400	0.0400	0.0397	0.0394	0.0399
Vx_{Error}	0.0000	0.0000	0.0071	0.0139	0.0031
$Vy(m/s)$	0.0198	0.0191	0.0188	0.0252	0.0254
Vy_{Error}	0.0113	0.0432	0.0593	0.2578	0.2716
$R(m)$	0.0202	0.0205	0.0199	0.0197	0.0199
R_{Error}	0.0100	0.0250	0.0071	0.0139	0.0031
$Average_{Error}$	0.0059	0.0192	0.0227	0.0829	0.0838

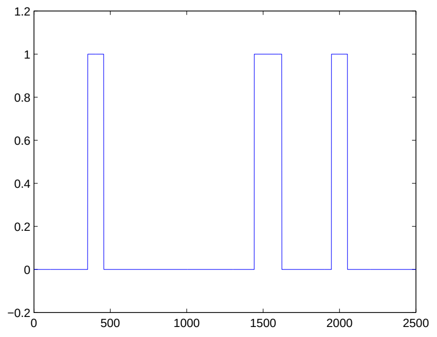
4.3.2 Simulation of decreased sample rate in space domain

In the second set of experiments, we keep the temporal sampling resolution as the best case scenario, which is when sample period is 0.01 (s), but decrease the resolution in space domain to compare with the previous results. The specific sample periods in space domain we took are 0.1 (cm), 0.2 (cm), 0.4 (cm), 4 (cm) per pixel.

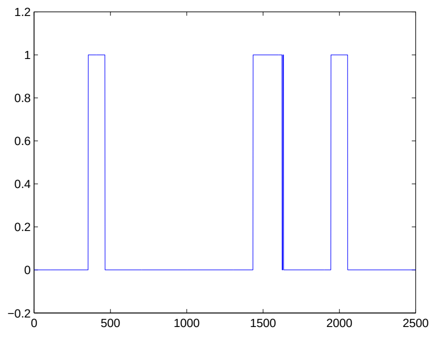
The intrusion detected timing graph is as follows:



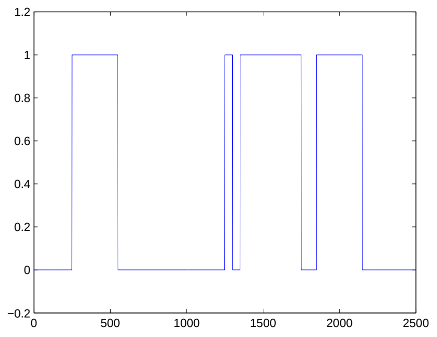
(a) Intrusion detected with 0.1 $cm/sample_s$



(b) Intrusion detected with 0.2 $cm/sample_s$



(c) Intrusion detected with 0.4 $cm/sample_s$



(d) Intrusion detected with 4 $cm/sample_s$

FIGURE 4.5: Intrusion detected timing graph

The result is shown in the following table, includes the best case scenario which is the case we took a sample period of 0.05 (cm) per pixel (Note that the first row of results are the ideal values of all experiments):

TABLE 4.3: Estimation result with different quantization level in space

$SamplePeriod(cm)$	0.050	0.100	0.200	0.400	4.000
$Vs(m/s)$	0.0446	0.0448	0.0444	0.0457	0.0438
V_{Error}	0.0022	0.0022	0.0068	0.0218	0.0216
$Vx(m/s)$	0.0400	0.0400	0.0400	0.0401	0.0399
Vx_{Error}	0.0000	0.0000	0.0000	0.0031	0.0031
$Vy(m/s)$	0.0198	0.0201	0.0193	0.0219	0.0180
Vy_{Error}	0.0113	0.0070	0.0343	0.0931	0.0991
$R(m)$	0.0202	0.0206	0.0210	0.0221	0.0598
R_{Error}	0.0100	0.0300	0.0500	0.1035	1.9906
$Average_{Error}$	0.0059	0.0098	0.0228	0.0554	0.5286

From comparing the data in Table 4.1, 4.2 and 4.3, we can sum up that the average simulation error converges to less than 1% when the resolution in space domain and in time domain are both high, thus the result is the most accurate. With the decreasing of quantization level in either time domain or space domain, the quantization error would generally go up; and with the increasing of quantization level in both time and space domain, the average quantization error should converge to less than 1%. From table 4.3 we can see that 0.1 cm/sample is almost as accurate as the maximum 0.05 cm/sample.

4.4 Analysis with temporal quantization noise

Among all the simulation process discussed in the last section, we are most interested in quantization noise in time domain, due to sample rate and its impact on the target parameter. This is because the real system is limited by sampling rate. In order do some further analysis, we look through Table 4.2 and find that the quantization error in the best case scenario which is when time frame is 0.01 (s), the quantization error is very small. We took this result as representing pseudo-continuous space and time. While keeping the pseudo-continuous time at high resolution, we propose studying temporal sampling error by using a stochastic model of temporal sampling. The quantitative noise model is add additive model based on where the noise is uniformly distributed white noise, shown in the following figure:

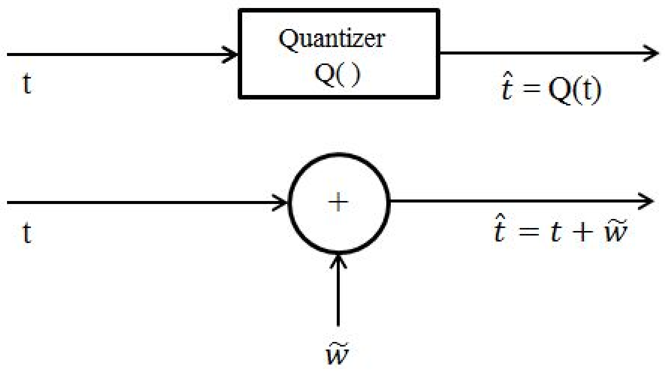


FIGURE 4.6: Additive noise model for quantizer

Now the mathematical model is

$$Q(t) = t + \widetilde{w} \tag{4.3}$$

Where $\widetilde{w}$ is the additive noise, then if we run simulation under this prerequisite, the estimation parameter result should simulate the quantization error as if we decreased the quantization level. By doing this, we would be able to get a large number of estimation parameter results with different errors if we run simulation sufficient times, then we can do some further analysis with these results.

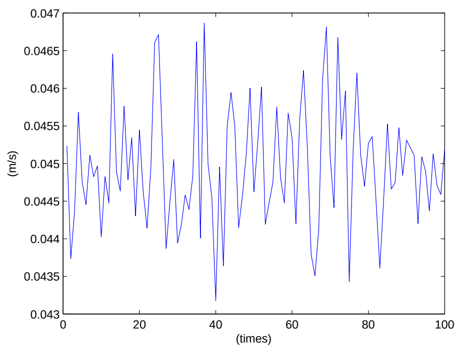
The equation to calculate expected value of quantization noise with quantization level known is as follows:

$$\delta_e{}^2 = \frac{\Delta^2}{12} \tag{4.4}$$

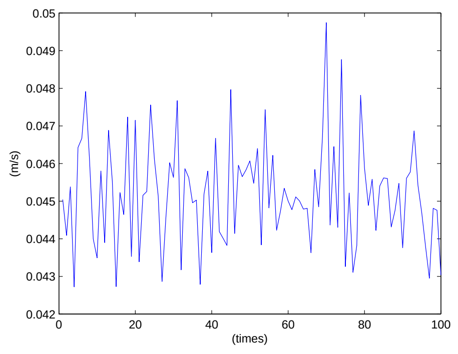
In which δ_e is the expected value of quantization noise and Δ is the quantization level or quantized frame period.[21]

4.5 Simulation with temporal quantization noise

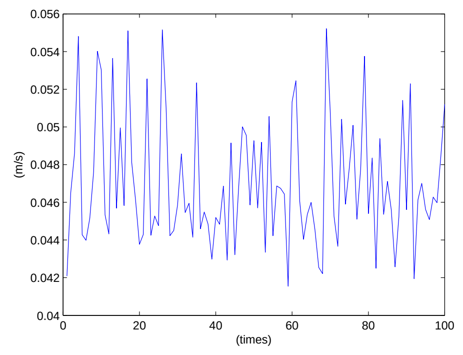
From Section 4.4, we develop a stochastic model of temporal sampling. To obtain the results of stochastic model, we only need to run the simulation under the pseudo-continuous case, which is when the space sampling rate is 0.05 cm/sample and the temporal sampling rate is 0.01 sec/sample, and then to add the corresponding uniformly distributed noise of lower quantized temporal case to the pseudo-continuous case. In this method, it is easier to get a large amount of 'simulation results' without actually run the simulation. The stochastic results which represents 100 times of different simulation results of four de-quantized temporal case discussed in Section 4.3.1 are shown in the following figures and the conclusion table:



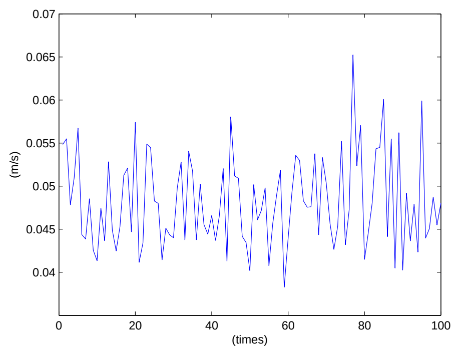
(a) Vs reprensts simulation of 0.025 $sec/sample_t$



(b) Vs reprensts simulation of 0.1 $sec/sample_t$



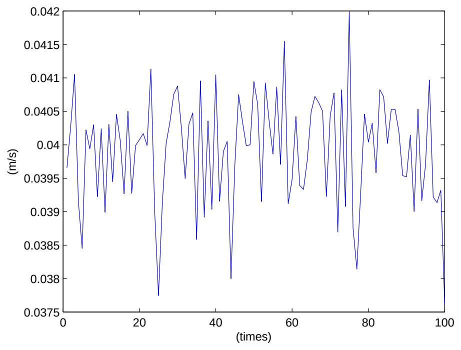
(c) Vs reprensts simulation of 0.5 $sec/sample_t$



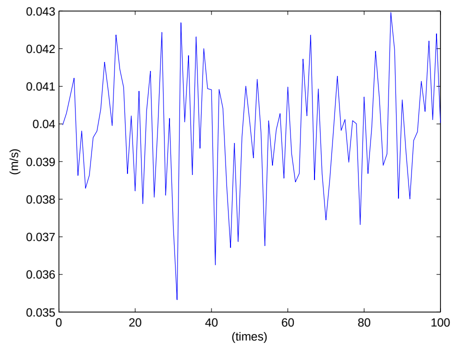
(d) Vs reprensts simulation of 1 $sec/sample_t$

FIGURE 4.7: Stochastic result of Vs

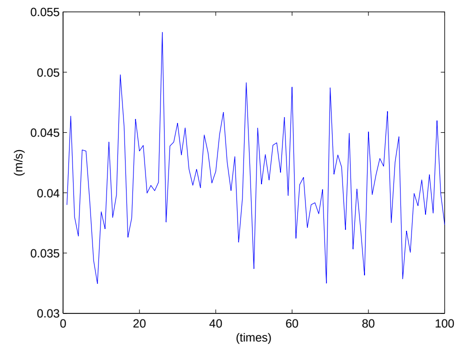
Fig 4.7 shows the stochastic result of Vs under the four different sampling rate. From this figure, we can see that with the decrease of quantization level in time domain, the deviation of Vs increases dramatically.



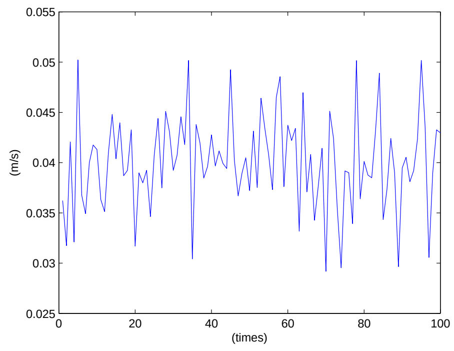
(a) Vx reprensts simulation of 0.025 $sec/sample_t$



(b) Vx reprensts simulation of 0.1 $sec/sample_t$



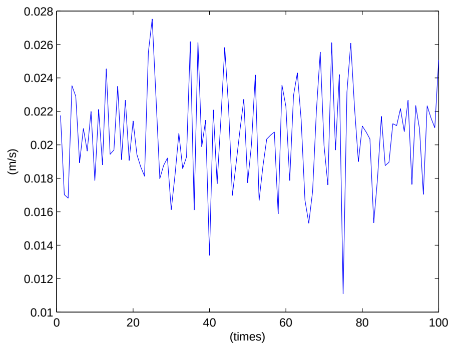
(c) Vx reprensts simulation of 0.5 $sec/sample_t$



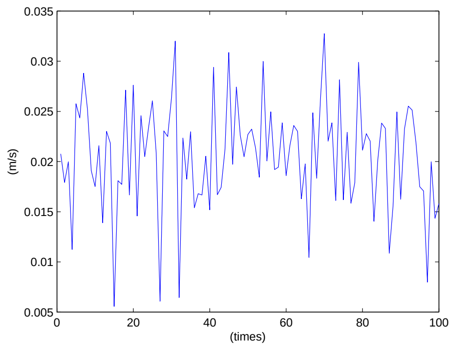
(d) Vx reprensts simulation of 1 $sec/sample_t$

FIGURE 4.8: Stochastic result of Vx

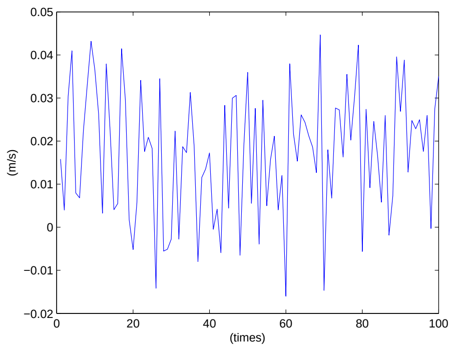
Fig 4.8 shows the stochastic result of Vx under the four different sampling rate. From this figure, we can see that with the decrease of quantization level in time domain, the deviation of Vx also increases dramatically.



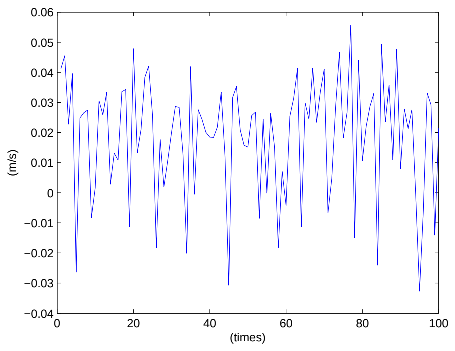
(a) Vy reprensts simulation of 0.025 $sec/sample_t$



(b) Vy reprensts simulation of 0.1 $sec/sample_t$



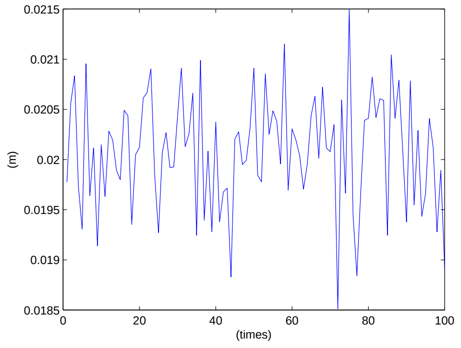
(c) Vy reprensts simulation of 0.5 $sec/sample_t$



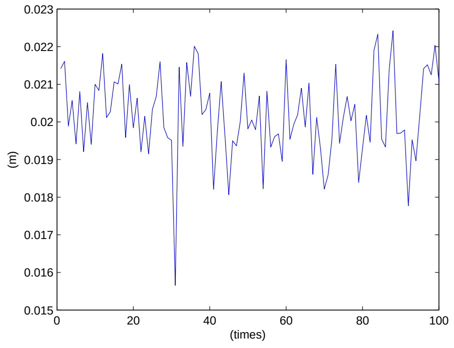
(d) Vy reprensts simulation of 1 $sec/sample_t$

FIGURE 4.9: Stochastic result of Vy

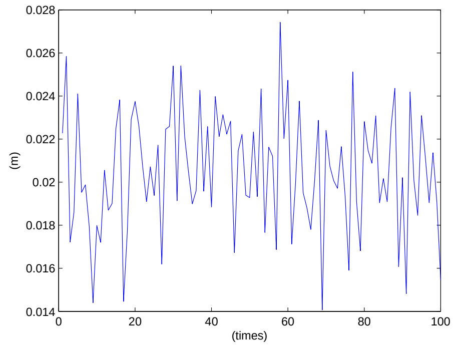
Fig 4.9 shows the stochastic result of Vy under the four different sampling rate. From this figure, we can see that with the decrease of quantization level in time domain, the deviation of Vy also increases dramatically.



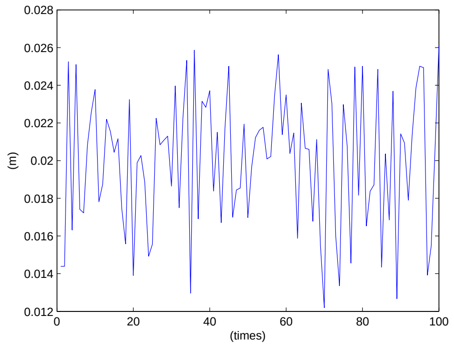
(a) R reprensts simulation of 0.025 $sec/sample_t$



(b) R reprensts simulation of 0.1 $sec/sample_t$



(c) R reprensts simulation of 0.5 $sec/sample_t$



(d) R reprensts simulation of 1 $sec/sample_t$

FIGURE 4.10: Stochastic result of R

Fig 4.10 shows the stochastic result of R under the four different sampling rate. From this figure, we can see that with the decrease of quantization level in time domain, the deviation of R also increases dramatically.

Generally, with the decrease of quantization level, the deviation generally goes up. The more detailed information with specific results will be shown in the following table:

Table 4.4: Stochastic results with different quantization level in time

$QuantiLevel(s)$	0.01	0.025	0.1	0.5	1
$QuantiNoise(s)$	0	0.0072	0.0289	0.1443	0.2887
$SimulatedVs(m/s)$	0.0446	0.0443	0.0439	0.0468	0.0473
$MeanVs(m/s)$	$-$	0.0450	0.0452	0.0470	0.0482
$STDVs(m/s)$	$-$	$7.924e-4$	0.0014	0.0034	0.0053
$SimulatedVx(m/s)$	0.0400	0.0400	0.0397	0.0394	0.0399
$MeanVx(m/s)$	$-$	0.0399	0.0399	0.0411	0.0400
$STDVx(m/s)$	$-$	$8.1962e-4$	0.0015	0.0040	0.0047
$SimulatedVy(m/s)$	0.0198	0.0191	0.0188	0.0252	0.0254
$MeanVy(m/s)$	$-$	0.0205	0.0206	0.0173	0.0187
$STDVy(m/s)$	$-$	0.0030	0.0054	0.0148	0.0195
$SimulatedR(m)$	0.0202	0.0205	0.0199	0.0197	0.0199
$MeanR(m)$	$-$	0.0201	0.0202	0.0205	0.0200
$STDR(m)$	$-$	$5.6876e-4$	0.0011	0.0028	0.0036

In Table 4.4, because we consider the quantization case of 0.01 as pseudo-continuous, so the quantization noise is assumed to be zero. The simulated results are from the simulation done in Chapter 4 and the statistical results are from the stochastic results shown from Fig 4.7 to Fig 4.10. We can tell from the table that the simulation results fall in the range of stochastic results and with the decrease of quantization level, or increase of quantization noise, the statistical standard deviation of the stochastic results would increase. For example, if we take Vs at 0.025 $sec/sample_t$, the simulation result of 0.0443 falls in range of (0.0450 - 7.924e-4, 0.0450 + 7.924e-4). With the decrease of quantization level, the statistical mean result of Vs deviates larger from 0.0446 and the standard deviation increase from 7.924e-4 to 0.0053, which means the result is less accurate with larger quantization noise. A graphical representation is shown in the following figure:

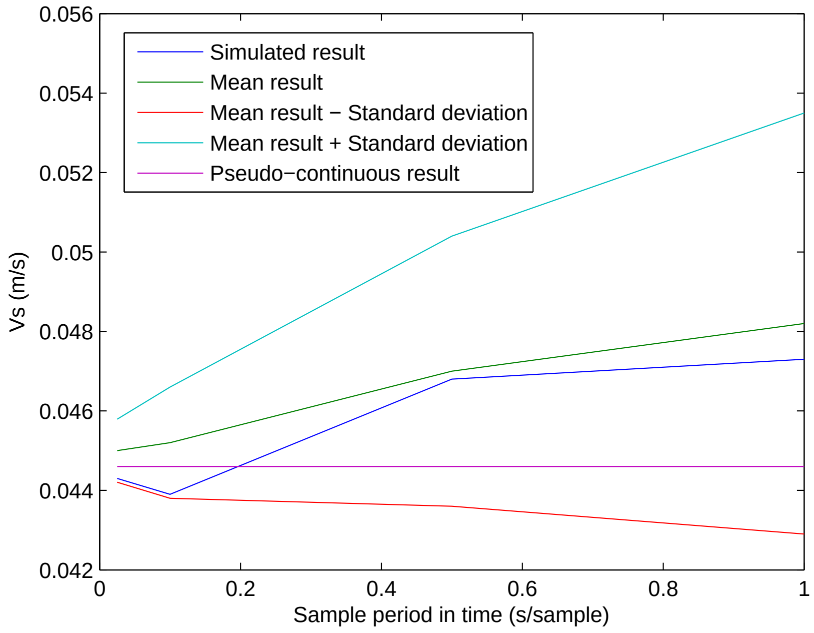


FIGURE 4.11: Analysis result of Vs

Chapter 5

Systematic Simulation and Experiments

In Chapter 4, we run the simulation with a simplified butterfly set-up of four sensors and analysed the result with quantization noise. In this chapter, we will run the systematic simulation based on the realistic system and compare to the experimental results.

5.1 Systematic Simulation

The set-up of experiment chamber scale model is shown on Fig 2.2, based on the chamber, the coordinate geometry of the model can be constructed as in the following figure, the left side are the transmitters and the right side are the receivers, the object is located 46 cm away from transmitters and moving at a constant speed on the line.

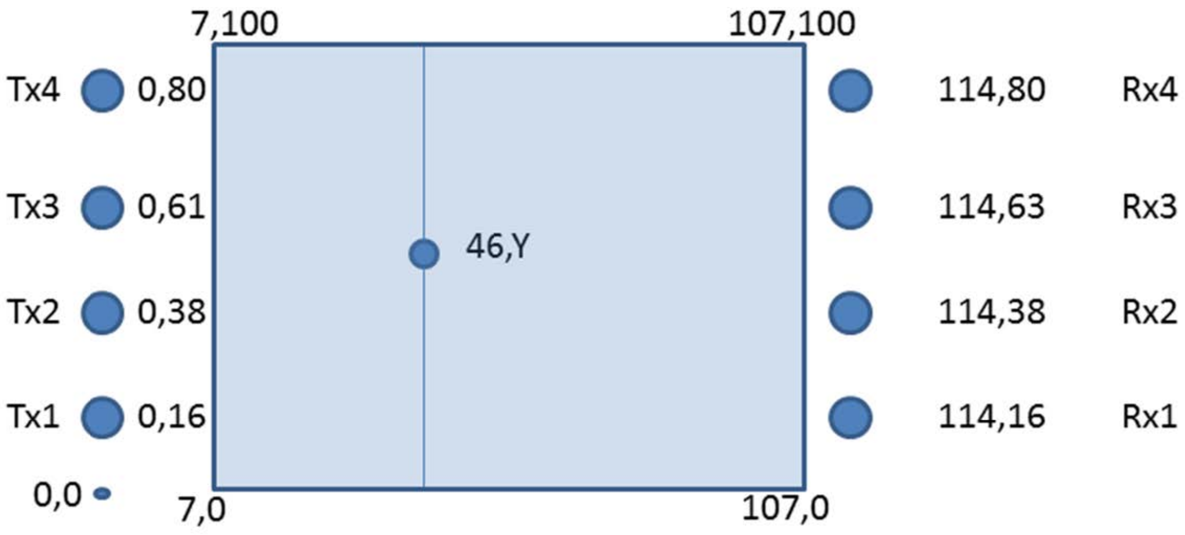


FIGURE 5.1: Coordinate geometry of scale model (values in cm)(University of Louisville)

Following this coordinate geometry, the simulated construction of the links is as follows:

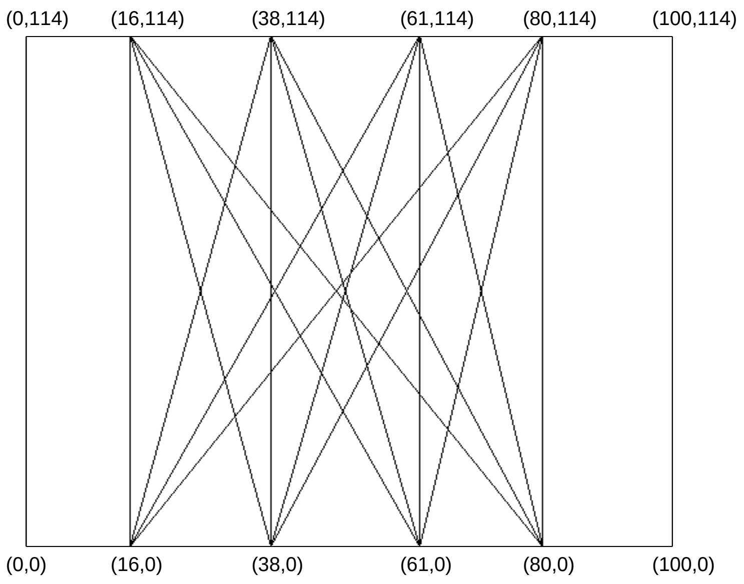


FIGURE 5.2: Simulated coordinate geometry model (values in cm)

Note that the simulation flipped 90 degree clockwise. And we denote the four sensors on the upside as Transmitter 1 to 4 and downside as Receiver 1 to 4. In this way, we can denote the 16 links from L_{11} to L_{44}, in the order of the first subscript stands for transmitter and second stands for receiver. Based on the calculation model proposed on Chapter 3, all the effective pairs of links that will be break without redundancy to enable the calculation are ① (L_{11},L_{21},L_{22}), ② (L_{11},L_{21},L_{23}), ③ (L_{11},L_{21},L_{24}), ④ (L_{11},L_{31},L_{33}), ⑤ (L_{11},L_{31},L_{34}), ⑥ (L_{11},L_{41},L_{44}), ⑦ (L_{22},L_{32},L_{33}), ⑧ (L_{22},L_{32},L_{34}), ⑨ (L_{22},L_{42},L_{44}), ⑩ (L_{33},L_{43},L_{44}). To generalize, we find that with n pairs of transmitters and receivers, the effective number of estimation is $\sum_{i=1}^{n-1} i(n-i)$.

Given the object parameter diameter as 2 (cm), initial position as 46 (cm) away from transmitters, and moving left to right at a speed of 4 (cm/s), using the highest sampling rate as 0.01 second per sample in time domain and 0.05 cm per sample in space domain, the intrusion timing graph can be shown in the following figure. Note that there are 16 links in total, so 16 intrusions are depicted by different colours in the lower value, but there are times that the object may break multiple links at one time, so the higher blue line depict whenever intrusion is detected.

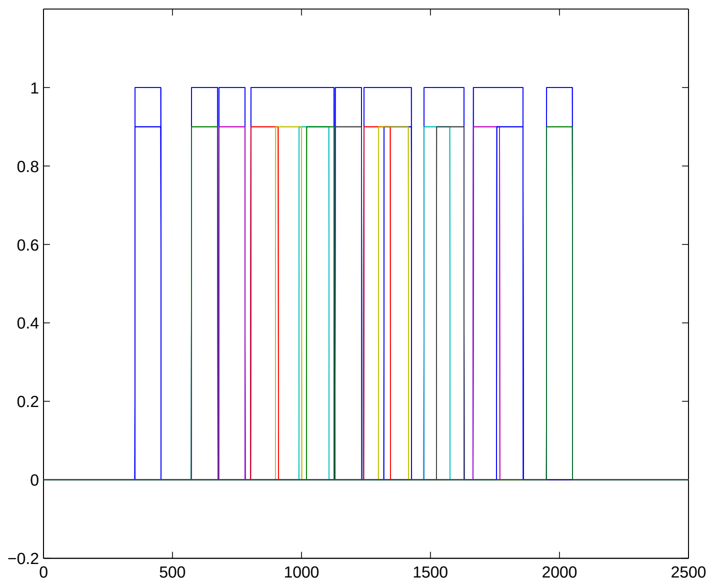
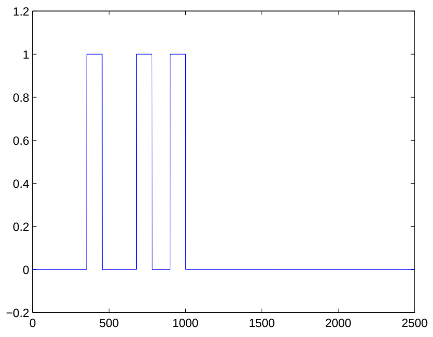
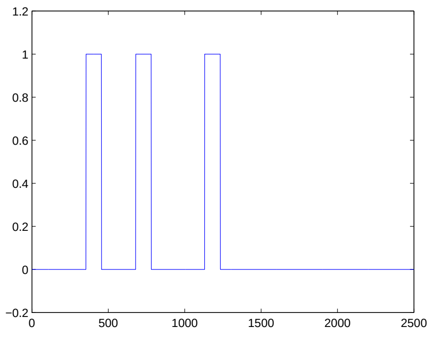


FIGURE 5.3: Systematic intrusion timing graph

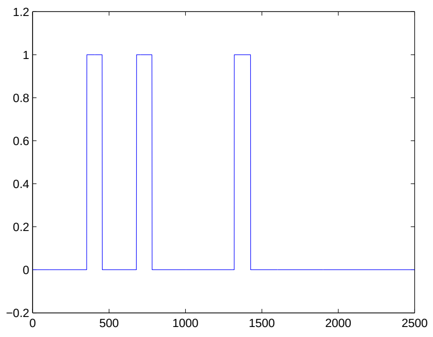
And below is the intrusion timing graph for each pair of links:



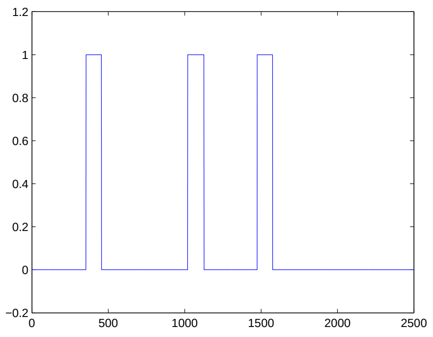
(a) Intrusion detected with link pair ①



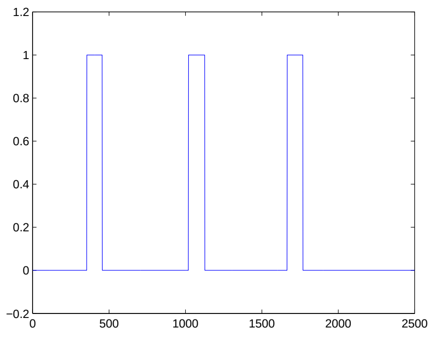
(b) Intrusion detected with link pair ②



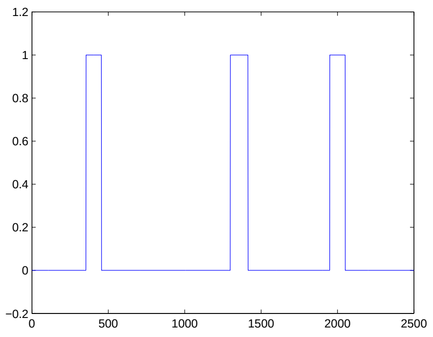
(c) Intrusion detected with link pair ③



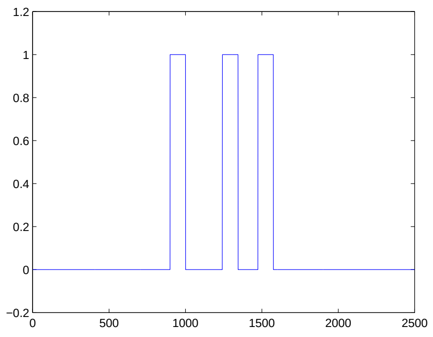
(d) Intrusion detected with link pair ④

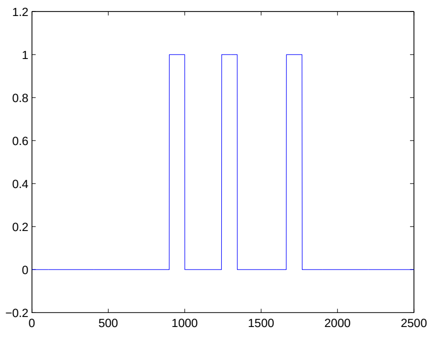


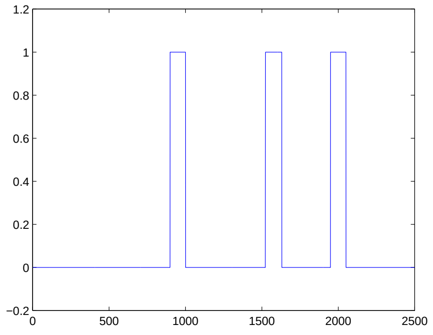
(e) Intrusion detected with link pair ⑤



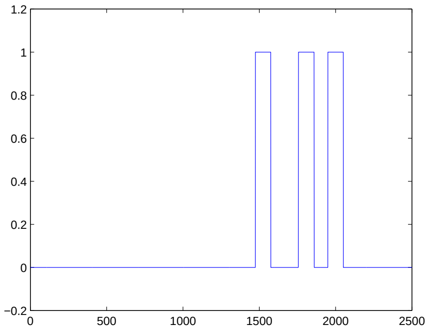
(f) Intrusion detected with link pair ⑥







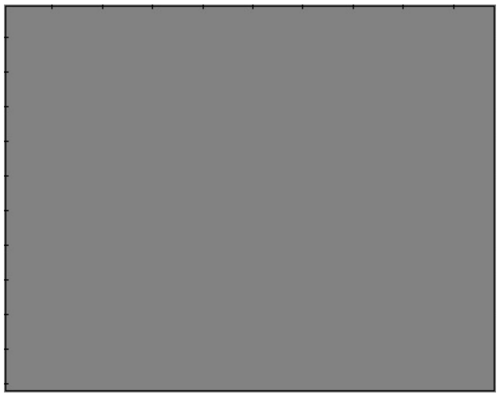
(i) Intrusion detected with link pair ⑨

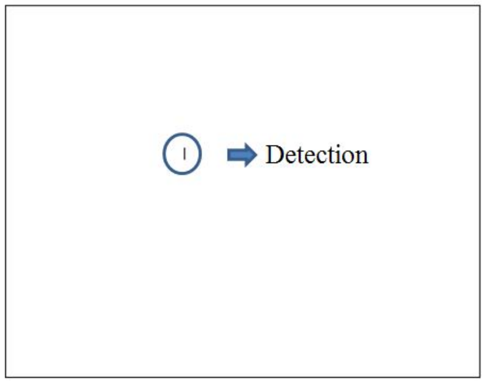


(j) Intrusion detected with link pair ⑩

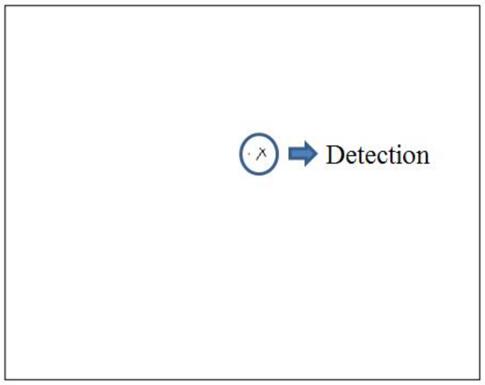
FIGURE 5.4: Intrusion detected timing graph with different pair of links

The detected of intrusion is shown by enlightening the intersection of the links that is being intruded, in which can be shown in the following figure:

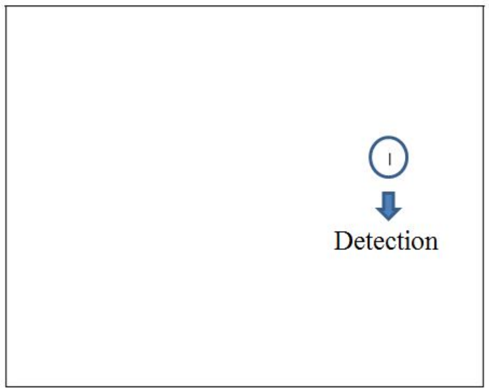

(a) Intrusion detection inteface at no detection



(b) Intrusion detection inteface at detection



(c) Intrusion detection inteface at detection



(d) Intrusion detection inteface at detection

FIGURE 5.5: Intrusion detection interface

The parameter estimation results of all the link pairs are shown in the following table:

Table 5.1: Estimation results with different pair of links

$\#LinkPair$	$Vs(m/s)$	$Vx(m/s)$	$Vy(m/s)$	$R(m)$
①	0.0400	0.0400	-0.0017	0.0202
②	0.0401	0.0400	-0.0017	0.0202
③	0.0400	0.0400	-0.0017	0.0202
④	0.0400	0.0400	-0.0014	0.0202
⑤	0.0401	0.0401	-0.0014	0.0202
⑥	0.0400	0.0400	0.0002	0.0202
⑦	0.0400	0.0400	-0.0001	0.0202
⑧	0.0400	0.0400	-0.0001	0.0202
⑨	0.0400	0.0400	-0.0006	0.0202
⑩	0.0400	0.0400	0.0014	0.0202

The trajectory is shown in the following figure, it is recalculated when each link pair is detected with the mean estimation results of the previous detection during the intrusion:

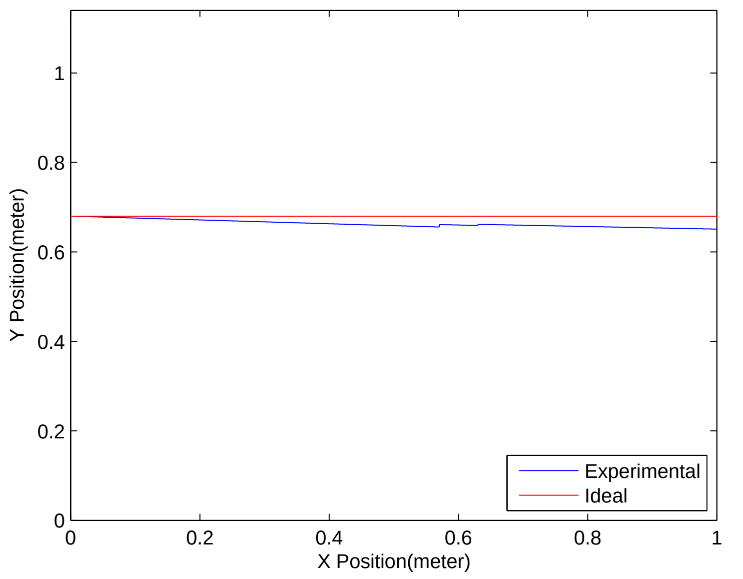


FIGURE 5.6: Systematic Estimation Trajectory

The trajectory error is shown in the following figure, it is calculated as the distance from the estimated trajectory to the real trajectory during the intrusion detection:

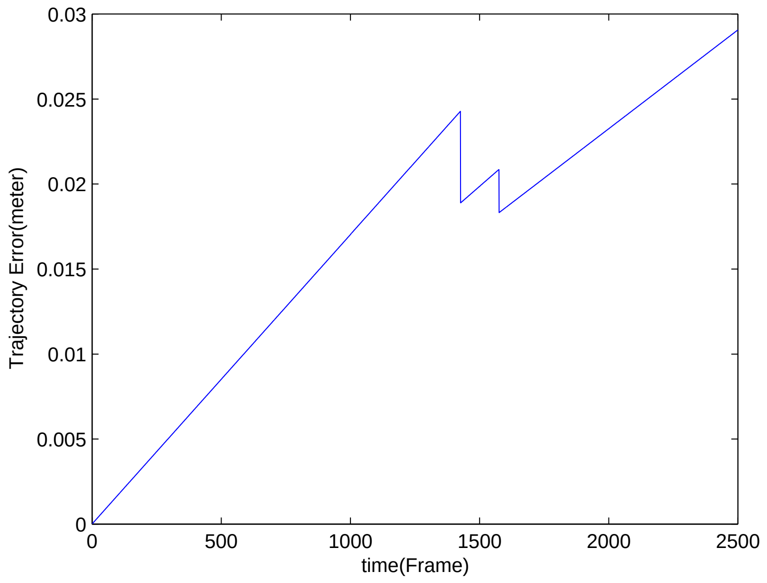
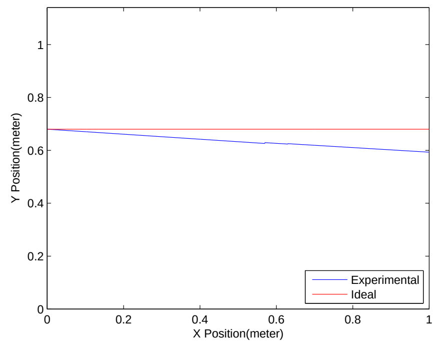
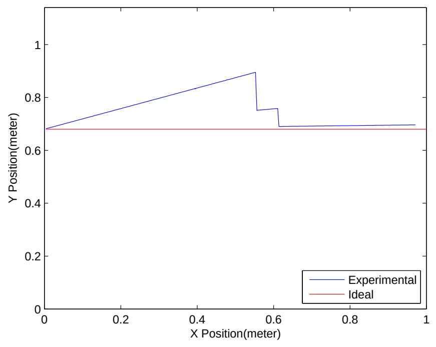


FIGURE 5.7: Systematic Trajectory Error

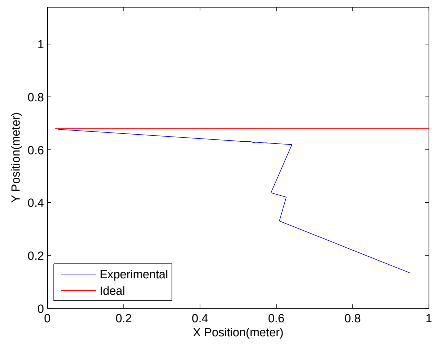
The estimation result is very accurate because we take the highest sampling rate. If we decrease the sample rate, the estimation result would be less accurate, which means the trajectory would be more deviate from the actual one and the error would generally increase. The following figures show the trajectory and trajectory error when we reduce the sample rate with an increased sample period in time to 0.025 (s), 0.1 (s), 0.5 (s), 1(s) per sample. The space sampling rate is kept at the highest rate of 0.05 cm per sample because the real system is limited by temporal resolution.



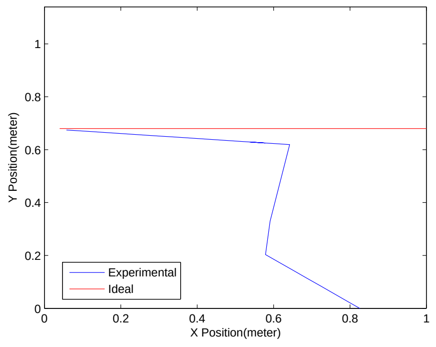
(a) Systematic Trajectory at 0.025 $sec/sample_t$



(b) Systematic Trajectory at 0.1 $sec/sample_t$

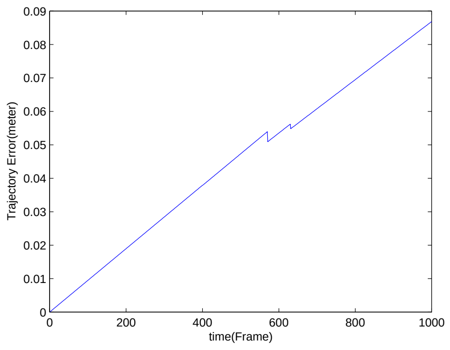


(c) Systematic Trajectory at 0.5 $sec/sample_t$

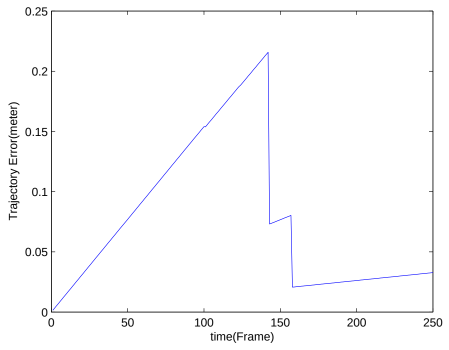


(d) Systematic Trajectory at 1 $sec/sample_t$

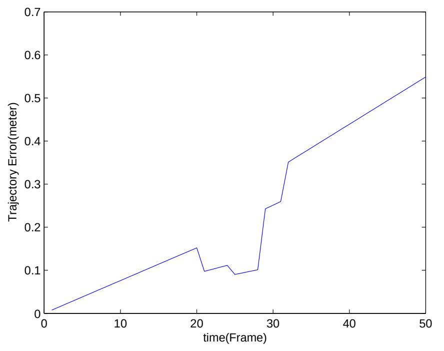
FIGURE 5.8: Systematic Trajectory at decreased sampling rate



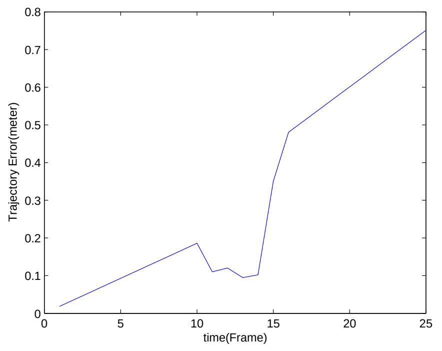
(a) Systematic Trajectory Error at 0.025 $sec/sample_t$



(b) Systematic Trajectory Error at 0.1 $sec/sample_t$



(c) Systematic Trajectory Error at 0.5 $sec/sample_t$



(d) Systematic Trajectory Error at 1 $sec/sample_t$

FIGURE 5.9: Systematic Trajectory Error at decreased sampling rate

5.2 Experiments

The hardware implementation of binary sensor system had been developed and constructed by Dr.Robert Cohn's research group in University of Louisville. The system includes a prototype transceiver module (shown in Fig 5.10) and an embedded micro-controller system (shown in Fig 5.11). The sensor module is able to transmit and receive LED pulsed signal and is tested in the chamber shown in Fig 2.2 as well as in the field.

The transmitter is a 4.3 mW infrared LED that is coupled to the output via a 0.5 inch lens for beam shaping. The receiver is a silicon photodiode with an infrared filter for the removal of the solar background and other visible light sources[cite University of Louisville, Robert W.Cohn et al].

The embedded micro-controller system consists of a dsPIC33F digital signal micro-controller, a Texas Instruments ADS1251 24-bit analog-to-digital(A/D) converter, a Microchip ENC28J60 stand-alone Ethernet controller, a National Semiconductor LMP7721 op-amp and Hamamastu S5821 photodiode. The signal from the sensor module is first converted to voltage and amplified by photodiode amplifier, and then converts to digital data through A/D converter, and passed to the micro-controller, the micro-controller collects a cycle of 512 data samples and perform FFT on the stored data, sent to Ethernet interface afterwards, then via TCP/IP protocol, the data would be transmitted to PC through Ethernet, a LabVIEW program was written to display, record and analysis the data. The data flow chart and a sample LabVIEW result is shown in Fig 5.12 and 5.13. Figures in this section are cite from University of Louisville, Robert W. Cohn, et al.

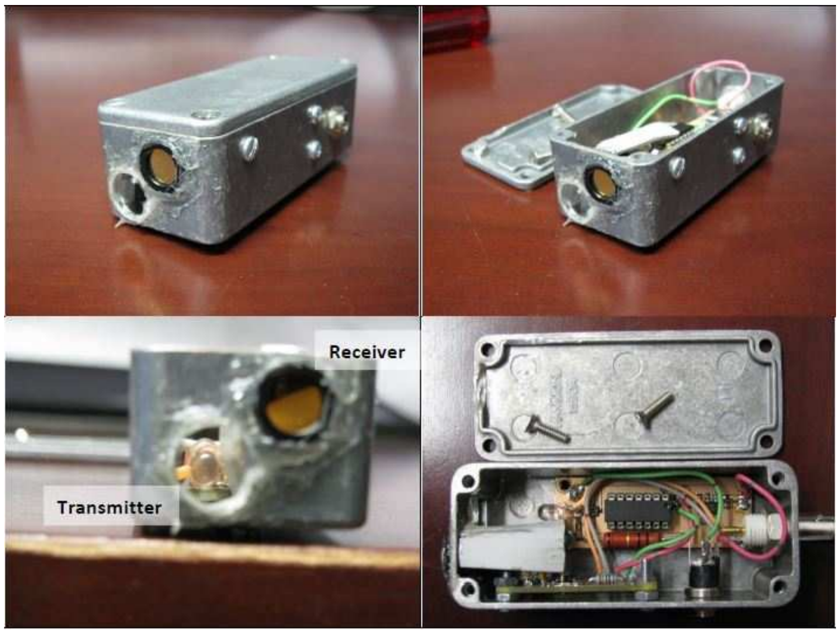


Figure 5.10: Dual transmit/receive prototype sensor module

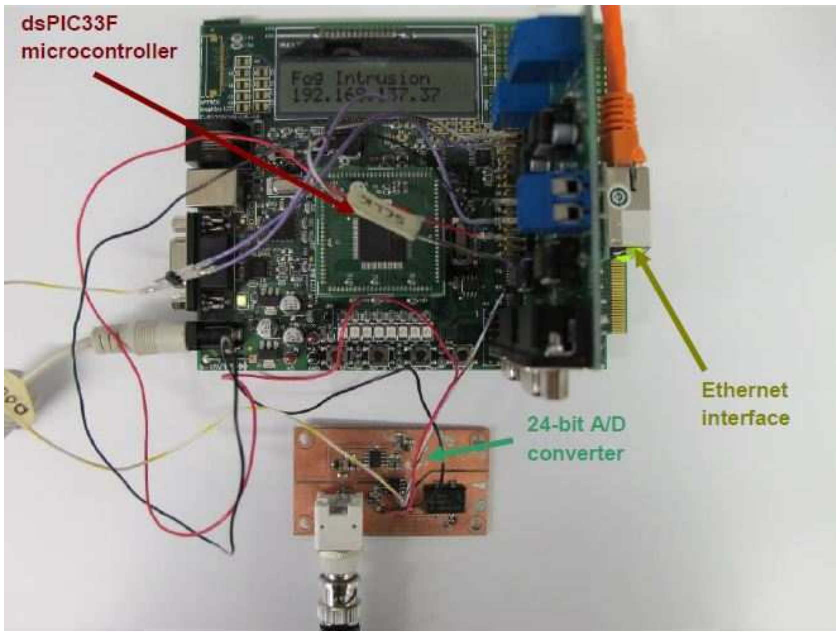


Figure 5.11: Embedded micro-controller system to process and transmit data

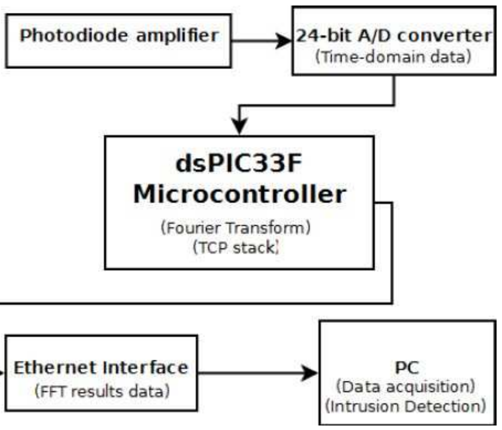
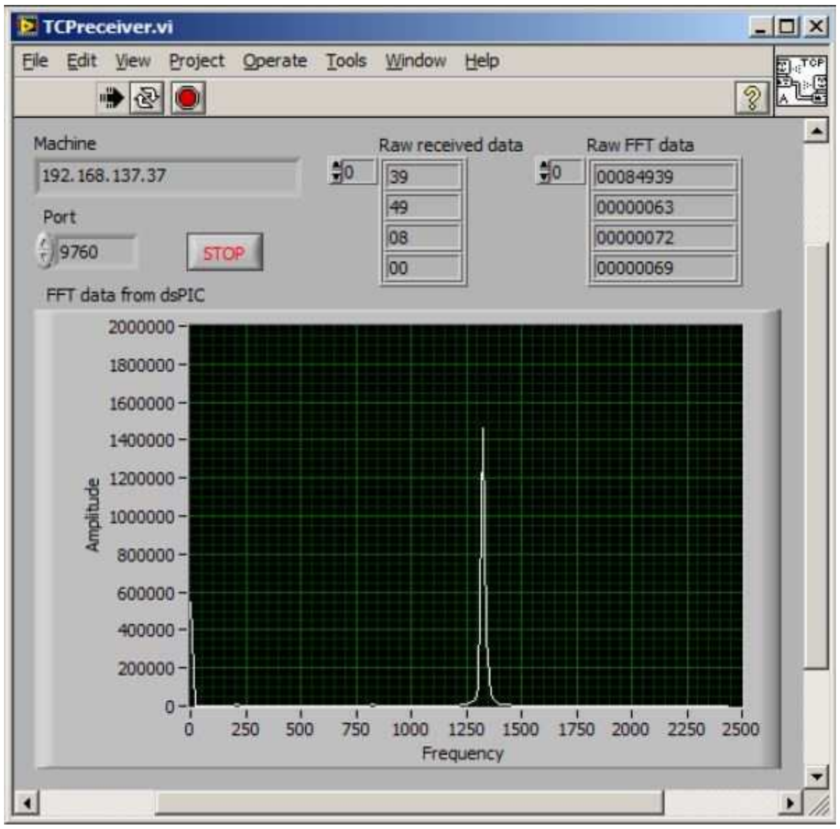


Figure 5.12: Illustration of data flow through the system



Machine: 192.168.137.37

Port: 9760

Raw received data	Raw FFT data
39	00084939
49	00000063
08	00000072
00	00000069

Figure 5.13: FFT of a 1.3kHz sine wave signal received in LabVIEW from the dsPIC micro-controller via Ethernet

In order to be commercialize, the micro-controller board is integrated with sensor module and the size is diminished to be packaged. The final version of the module is shown in the following figure:

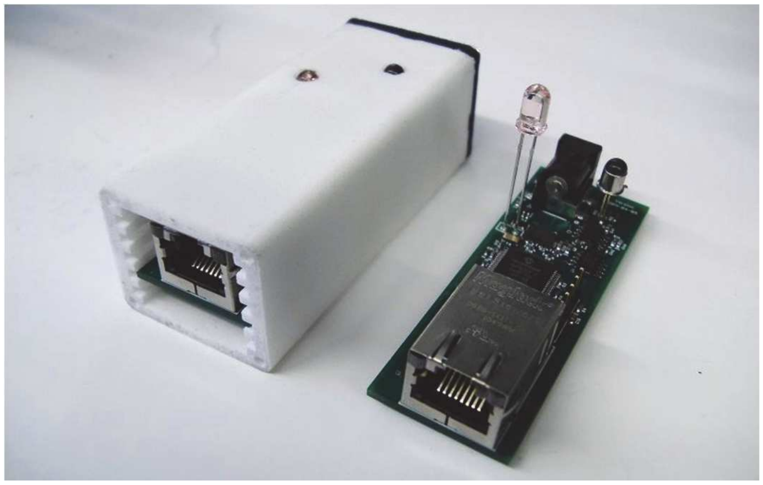


Figure 5.14: Packaged sensor module

One possible real life example is Fort Knox, shown in the following figure, 80 sensor modules are deployed outside the building, established an security grid such that each receiver sees at least 3 transmitter lines-of-sight. Transmitters are stationed around the perimeter of the building with the deployment of 10 transmitters on each edge, and receivers are positioned on the security fence correspondingly.

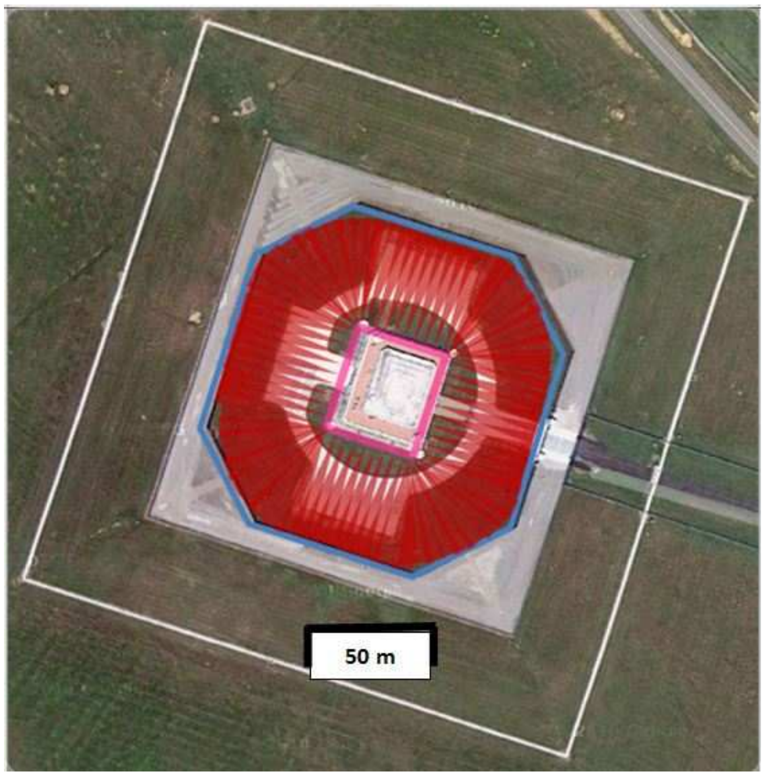


FIGURE 5.15: Aerial view of Fort Knox with possible deployment of sensor modules

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We have developed a mathematical model for tracking target characteristics and trajectory in binary sensor system has been achieved. By utilizing the geometric alignment of links formed by binary sensors, with merely the information of intrusion timing sequence and idle timing sequence collected by the binary sensors, we are able to calculate the speed, size, and trajectory of the tracking target. Note that the model is under the assumption of the speed being constant within three link intrusions and object being circular shape.

We presented simulation illustrating the validation of the mathematical model. In the simulation model, with evaluated the minimum sampling rates in space and time, estimation results are close to the realistic values with errors of no greater than one percent.

We further analyse how the quantization noise in temporal domain would affect the estimation results based on the simulation model.

A systematic simulation and hardware implementation of the binary sensor network tracking system have been attained. The binary sensor network system can be used in real world surveillance at a relative low cost.

6.2 Future Work

As the requirements for tracking and surveillance increased, the exiting algorithm may be not enough. One direction of future work is to develop the calculation model of other kinds of link deployment, such that less amount of binary sensors are needed for characteristic calculation and the deployment of sensors can be more scatter and random. Formulate a linear algebraic representation of the mathematics so we can apply to larger arrays and higher order trajectories with more shape characteristics. Another research direction is to enable multi-object tracking and identification. This is an essential area in surveillance system and different paper proposed different algorithms including the hidden Markov model, etc. The binary sensor surveillance system may have a extended market with the improvement of tracking algorithm.

Appendix A

MATLAB CODE

```
%simulation
clear all;
R=0.02; % radius meters
Tx=1; % meters
Ty=1.14; % meters
Nx=100*5; % pixels
My=114*5;  % pixels
x0=0;
y0=0.46;
vx=.04; % m/s
vy=0; % m/s
dt=0.01; % sample period seconds
OTheta=atan2(abs(vy),vx);% object intrusion angle
OPath=Tx/abs(cos(OTheta));% path length
V=sqrt(vx.^2+vy.^2);
Tw=OPath/V;
Nframes=floor(Tw/dt);

dmdy=My/Ty;dndx=Nx/Tx;
A1=ones(My,Nx);
A2=ones(My,Nx);
dash1=0;
dash0=0;

ix1=81; ix2=190; ix3=(305); ix4=400;% position of sensors in pixels
```

```
iy1=0; iy2=114*5;
B11 = DrawDashLine(A2,ix1,iy1,ix1,iy2,dash1,dash0);
B12 = DrawDashLine(A2,ix1,iy1,ix2,iy2,dash1,dash0);
B13 = DrawDashLine(A2,ix1,iy1,ix3,iy2,dash1,dash0);
B14 = DrawDashLine(A2,ix1,iy1,ix4,iy2,dash1,dash0);
B21 = DrawDashLine(A2,ix2,iy1,ix1,iy2,dash1,dash0);
B22 = DrawDashLine(A2,ix2,iy1,ix2,iy2,dash1,dash0);
B23 = DrawDashLine(A2,ix2,iy1,ix3,iy2,dash1,dash0);
B24 = DrawDashLine(A2,ix2,iy1,ix4,iy2,dash1,dash0);
B31 = DrawDashLine(A2,ix3,iy1,ix1,iy2,dash1,dash0);
B32 = DrawDashLine(A2,ix3,iy1,ix2,iy2,dash1,dash0);
B33 = DrawDashLine(A2,ix3,iy1,ix3,iy2,dash1,dash0);
B34 = DrawDashLine(A2,ix3,iy1,ix4,iy2,dash1,dash0);
B41 = DrawDashLine(A2,ix4,iy1,ix1,iy2,dash1,dash0);
B42 = DrawDashLine(A2,ix4,iy1,ix2,iy2,dash1,dash0);
B43 = DrawDashLine(A2,ix4,iy1,ix3,iy2,dash1,dash0);
B44 = DrawDashLine(A2,ix4,iy1,ix4,iy2,dash1,dash0);
A2=B11.*B12.*B13.*B14.*B21.*B22.*B23.*...
B24.*B31.*B32.*B33.*B34.*B41.*B42.*B43.*B44;
% A2=B11.*B14.*B44;
A3=ones(My,Nx);
T=zeros(1,Nframes);
D11=zeros(1,Nframes);
D12=zeros(1,Nframes);
D13=zeros(1,Nframes);
D14=zeros(1,Nframes);
D21=zeros(1,Nframes);
D22=zeros(1,Nframes);
D23=zeros(1,Nframes);
D24=zeros(1,Nframes);
D31=zeros(1,Nframes);
D32=zeros(1,Nframes);
D33=zeros(1,Nframes);
D34=zeros(1,Nframes);
D41=zeros(1,Nframes);
D42=zeros(1,Nframes);
D43=zeros(1,Nframes);
D44=zeros(1,Nframes);
P1=zeros(1,Nframes);
P2=zeros(1,Nframes);
```

```
P3=zeros(1,Nframes);
P4=zeros(1,Nframes);
P5=zeros(1,Nframes);
P6=zeros(1,Nframes);
P7=zeros(1,Nframes);
P8=zeros(1,Nframes);
P9=zeros(1,Nframes);
P10=zeros(1,Nframes);
for it=1:Nframes
    t=it*dt; % seconds
    x=vx*t+x0;
    y=vy*t+y0;
    ix=floor(dndx*x+0.5);
    iy=floor(dmdy*y+0.5);
    A11=A1;
    iR=floor(dndx*R+0.5);
    A = DrawCircle(A11,ix,iy,iR,0);
    figure(1);
    imagesc(A);
    colormap gray;
    figure(2);
    colormap gray;
    imagesc(A2);
    A3=(1-A).*(1-A2);
    figure(3);
    colormap gray;
    imagesc(A3);
    if max(max(A3))==1
        T(it)=1;
    end
    figure(4);
    plot(T);
    axis([0 Nframes -0.2 1.2]);
    C11=(1-A).*(1-B11);
    if max(max(C11))==1
        D11(it)=1;
    end
    C12=(1-A).*(1-B12);
    if max(max(C12))==1
        D12(it)=1;
    end
    C13=(1-A).*(1-B13);
```

```
if max(max(C13))==1
    D13(it)=1;
end
C14=(1-A).*(1-B14);
if max(max(C14))==1
    D14(it)=1;
end
C21=(1-A).*(1-B21);
if max(max(C21))==1
    D21(it)=1;
end
C22=(1-A).*(1-B22);
if max(max(C22))==1
    D22(it)=1;
end
C23=(1-A).*(1-B23);
if max(max(C23))==1
    D23(it)=1;
end
C24=(1-A).*(1-B24);
if max(max(C24))==1
    D24(it)=1;
end
C31=(1-A).*(1-B31);
if max(max(C31))==1
    D31(it)=1;
end
C32=(1-A).*(1-B32);
if max(max(C32))==1
    D32(it)=1;
end
C33=(1-A).*(1-B33);
if max(max(C33))==1
    D33(it)=1;
end
C34=(1-A).*(1-B34);
if max(max(C34))==1
    D34(it)=1;
end
C41=(1-A).*(1-B41);
if max(max(C41))==1
```

```
    D41(it)=1;
end
C42=(1-A).*(1-B42);
if max(max(C42))==1
    D42(it)=1;
end
C43=(1-A).*(1-B43);
if max(max(C43))==1
    D43(it)=1;
end
C44=(1-A).*(1-B44);
if max(max(C44))==1
    D44(it)=1;
end
E1=(1-A).*(1-B11.*B21.*B22);
if max(max(E1))==1
    P1(it)=1;
end
E2=(1-A).*(1-B11.*B21.*B23);
if max(max(E2))==1
    P2(it)=1;
end
E3=(1-A).*(1-B11.*B21.*B24);
if max(max(E3))==1
    P3(it)=1;
end
E4=(1-A).*(1-B11.*B31.*B33);
if max(max(E4))==1
    P4(it)=1;
end
E5=(1-A).*(1-B11.*B31.*B34);
if max(max(E5))==1
    P5(it)=1;
end
E6=(1-A).*(1-B11.*B41.*B44);
if max(max(E6))==1
    P6(it)=1;
end
E7=(1-A).*(1-B22.*B32.*B33);
if max(max(E7))==1
    P7(it)=1;
end
```

```
    E8=(1-A).*(1-B22.*B32.*B34);
    if max(max(E8))==1
        P8(it)=1;
    end
    E9=(1-A).*(1-B22.*B42.*B44);
    if max(max(E9))==1
        P9(it)=1;
    end
    E10=(1-A).*(1-B33.*B43.*B44);
    if max(max(E10))==1
        P10(it)=1;
    end
end; % for t

i=1:Nframes;
figure(5)
plot(i,T);
axis([0 Nframes -0.2 1.2]);
hold on
plot(i,0.9*[D11;D12;D13;D14;D21;D22;D23;...
D24;D31;D32;D33;D34;D41;D42;D43;D44])
figure(6);
plot(P1);
axis([0 Nframes -0.2 1.2]);
figure(7);
plot(P2);
axis([0 Nframes -0.2 1.2]);
figure(8);
plot(P3);
axis([0 Nframes -0.2 1.2]);
figure(9);
plot(P4);
axis([0 Nframes -0.2 1.2]);
figure(10);
plot(P5);
axis([0 Nframes -0.2 1.2]);
figure(11);
plot(P6);
axis([0 Nframes -0.2 1.2]);
figure(12);
plot(P7);
```

```
axis([0 Nframes -0.2 1.2]);
figure(13);
plot(P8);
axis([0 Nframes -0.2 1.2]);
figure(14);
plot(P9);
axis([0 Nframes -0.2 1.2]);
figure(15);
plot(P10);
axis([0 Nframes -0.2 1.2]);

%calculation
Vs = zeros(1,10);
Rs = zeros(1,10);
VX = zeros(1,10);
VY = zeros(1,10);
%1
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P1(j)==0
        d0(i)=d0(i)+1;
    elseif P1(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P1(j)==1
        d1(i)=d1(i)+1;
    elseif P1(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
```

```
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix2; Ix4=ix2;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(1)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(1)=Vs(1).*T0.*sin(Theta0)./2;
VX(1)=Vs(1).*cos(Theta0+Theta0R-pi);
VY(1)=Vs(1).*sin(Theta0+Theta0R-pi);

%2
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
```

```
    for j=a:Nframes
    if P2(j)==0
        d0(i)=d0(i)+1;
    elseif P2(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P2(j)==1
        d1(i)=d1(i)+1;
    elseif P2(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix2; Ix4=ix3;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);
```

```
Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(2)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(2)=Vs(2).*T0.*sin(Theta0)./2;
VX(2)=Vs(2).*cos(Theta0+Theta0R-pi);
VY(2)=Vs(2).*sin(Theta0+Theta0R-pi);

%3
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P3(j)==0
        d0(i)=d0(i)+1;
    elseif P3(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P3(j)==1
        d1(i)=d1(i)+1;
    elseif P3(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
```

```
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix2; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(3)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(3)=Vs(3).*T0.*sin(Theta0)./2;
VX(3)=Vs(3).*cos(Theta0+Theta0R-pi);
VY(3)=Vs(3).*sin(Theta0+Theta0R-pi);
%4
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P4(j)==0
```

```
        d0(i)=d0(i)+1;
    elseif P4(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P4(j)==1
        d1(i)=d1(i)+1;
    elseif P4(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix3; Ix4=ix3;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
```

```
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(4)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(4)=Vs(4).*T0.*sin(Theta0)./2;
VX(4)=Vs(4).*cos(Theta0+Theta0R-pi);
VY(4)=Vs(4).*sin(Theta0+Theta0R-pi);

%5
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P5(j)==0
        d0(i)=d0(i)+1;
    elseif P5(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P5(j)==1
        d1(i)=d1(i)+1;
    elseif P5(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
```

```
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix3; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(5)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(5)=Vs(5).*T0.*sin(Theta0)./2;
VX(5)=Vs(5).*cos(Theta0+Theta0R-pi);
VY(5)=Vs(5).*sin(Theta0+Theta0R-pi);
%6
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P6(j)==0
        d0(i)=d0(i)+1;
    elseif P6(j)==1
```

```
        a=j;
    break
    end
    end

for j=a:Nframes
    if P6(j)==1
        d1(i)=d1(i)+1;
    elseif P6(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix1; Ix2=ix1; Ix3=ix4; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
```

```
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(6)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(6)=Vs(6).*T0.*sin(Theta0)./2;
VX(6)=Vs(6).*cos(Theta0+Theta0R-pi);
VY(6)=Vs(6).*sin(Theta0+Theta0R-pi);
%7
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P7(j)==0
        d0(i)=d0(i)+1;
    elseif P7(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P7(j)==1
        d1(i)=d1(i)+1;
    elseif P7(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);
```

```
T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix2; Ix2=ix2; Ix3=ix3; Ix4=ix3;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(7)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(7)=Vs(7).*T0.*sin(Theta0)./2;
VX(7)=Vs(7).*cos(Theta0+Theta0R-pi);
VY(7)=Vs(7).*sin(Theta0+Theta0R-pi);
%8
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P8(j)==0
        d0(i)=d0(i)+1;
    elseif P8(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P8(j)==1
        d1(i)=d1(i)+1;
    elseif P8(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix2; Ix2=ix2; Ix3=ix3; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
```

```
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(8)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(8)=Vs(8).*T0.*sin(Theta0)./2;
VX(8)=Vs(8).*cos(Theta0+Theta0R-pi);
VY(8)=Vs(8).*sin(Theta0+Theta0R-pi);
%9
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P9(j)==0
        d0(i)=d0(i)+1;
    elseif P9(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P9(j)==1
        d1(i)=d1(i)+1;
    elseif P9(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
```

```
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix2; Ix2=ix2; Ix3=ix4; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

Vs(9)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(9)=Vs(9).*T0.*sin(Theta0)./2;
VX(9)=Vs(9).*cos(Theta0+Theta0R-pi);
VY(9)=Vs(9).*sin(Theta0+Theta0R-pi);
%10
d0 = zeros(1,3);
d1 = zeros(1,3);
a = 1;
for i = 1:1:3
    for j=a:Nframes
    if P10(j)==0
        d0(i)=d0(i)+1;
    elseif P10(j)==1
        a=j;
    break
    end
    end

for j=a:Nframes
    if P10(j)==1
```

```
        d1(i)=d1(i)+1;
    elseif P10(j)==0
        a=j;
        break
    end
end

end
t00 = d0(1);
t0 = d1(1);
t01 = d0(2);
t1 = d1(2);
t12 = d0(3);
t2 = d1(3);



T00=t00.*dt;
T0=t0.*dt;
T01=t01.*dt;
T1=t1.*dt;
T12=t12.*dt;
T2=t2.*dt;

Ix1=ix3; Ix2=ix3; Ix3=ix4; Ix4=ix4;%
Theta0R=atan2(iy2,Ix1-Ix2);
Theta1R=atan2(iy2,Ix3-Ix2);
Theta2R=atan2(iy2,Ix3-Ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
```

```
Vs(10)=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
Rs(10)=Vs(10).*T0.*sin(Theta0)./2;
VX(10)=Vs(10).*cos(Theta0+Theta0R-pi);
VY(10)=Vs(10).*sin(Theta0+Theta0R-pi);

%Trajectory
ET = [find(P1,1,'last') find(P2,1,'last') find(P3,1,'last') find(P4,1,'last')...
    find(P5,1,'last') find(P6,1,'last') find(P7,1,'last') find(P8,1,'last')...
    find(P9,1,'last') find(P10,1,'last')];
[s ind] = sort(ET);

%SpeedX
SpeedX=zeros(1,Nframes);
SpeedX(1:s(1))=VX(ind(1));
SpeedX(s(1)+1:s(2))=mean([VX(ind(1)) VX(ind(2))]);
SpeedX(s(2)+1:s(3))=mean([VX(ind(1)) VX(ind(2)) VX(ind(3))]);
SpeedX(s(3)+1:s(4))=mean([VX(ind(1)) VX(ind(2)) VX(ind(3)) VX(ind(4)) VX(ind(5))]);
SpeedX(s(5)+1:s(6))=mean([VX(ind(1)) VX(ind(2)) VX(ind(3)) VX(ind(4)) VX(ind(5))...
    VX(ind(6)) VX(ind(7))]);
SpeedX(s(6)+1:s(7))=mean([VX(ind(1)) VX(ind(2)) VX(ind(3)) VX(ind(4)) VX(ind(5))...
    VX(ind(6)) VX(ind(7)) VX(ind(8)) VX(ind(9)) VX(ind(10))]);
SpeedX(s(7)+1:Nframes)=SpeedX(s(7));
IdealSpeedX=vx;

%SpeedY
SpeedY=zeros(1,Nframes);
SpeedY(1:s(1))=VY(ind(1));
SpeedY(s(1)+1:s(2))=mean([VY(ind(1)) VY(ind(2))]);
SpeedY(s(2)+1:s(3))=mean([VY(ind(1)) VY(ind(2)) VY(ind(3))]);
SpeedY(s(3)+1:s(4))=mean([VY(ind(1)) VY(ind(2)) VY(ind(3)) VY(ind(4)) VY(ind(5))]);
SpeedY(s(5)+1:s(6))=mean([VY(ind(1)) VY(ind(2)) VY(ind(3)) VY(ind(4)) VY(ind(5))...
    VY(ind(6)) VY(ind(7))]);
SpeedY(s(6)+1:s(7))=mean([VY(ind(1)) VY(ind(2)) VY(ind(3)) VY(ind(4)) VY(ind(5))...
    VY(ind(6)) VY(ind(7)) VY(ind(8)) VY(ind(9)) VY(ind(10))]);
SpeedY(s(7)+1:Nframes)=SpeedY(s(7));

%PositionX
PositionX=zeros(1,Nframes);
t=1:1:Nframes;
PositionX(1:Nframes)=(x0+SpeedX.*t.*dt);
IdealPositionX(1:Nframes)=(x0+vx*t(1:Nframes)*dt);
```

```
t=1:1:Nframes;
figure(16);
plot(t,PositionX,'-',t,IdealPositionX,':');
legend('Experimental','Ideal',4);
axis([0 Nframes 0 1]);
xlabel('time(Frame)');
ylabel('X Position(meter)');

%PositionY
PositionY=zeros(1,Nframes);
t=1:1:Nframes;
PositionY(1:Nframes)=(1.14-y0+SpeedY.*t.*dt);
IdealPositionY(1:Nframes)=(1.14-y0+abs(vy)*t(1:Nframes)*dt);
t=1:1:Nframes;
figure(17);
plot(t,PositionY,'-',t,IdealPositionY,':');
legend('Experimental','Ideal',4);
axis([0 Nframes 0 1.14]);
xlabel('time(Frame)');
ylabel('Y Position(meter)');

%Trajectory
figure(18);
plot(PositionX,PositionY,'-',IdealPositionX,IdealPositionY,'r');
legend('Experimental','Ideal',4);
axis([0 1 0 1.14]);
xlabel('X Position(meter)');
ylabel('Y Position(meter)');

%Position Error
PosError=zeros(1,Nframes);
PosError(1:Nframes)=sqrt((PositionX(1:Nframes)-IdealPositionX(1:Nframes))...
.^2+(PositionY(1:Nframes)-IdealPositionY(1:Nframes)).^2);
t=1:1:Nframes;
figure(19);
plot(t,PosError);
xlabel('time(Frame)');
ylabel('Trajectory Error(meter)');
%quantization noise model
N=100;
RV01=zeros(1,N);
```

```
RVX01=zeros(1,N);
RVY01=zeros(1,N);
RR0=zeros(1,N);
for i=1:1:N
DeltaT=1;
deltaE=DeltaT/sqrt(12);
FrameN=25/DeltaT;
Dt=deltaE.*(2*rand(1,FrameN)-1);

T00=t00.*dt+sum(Dt(1:round(t00*FrameN/2500)));
T0=t0.*dt+sum(Dt((round(t00*FrameN/2500)+1):round((t00+t0)*FrameN/2500)));
T01=t01.*dt+sum(Dt((round((t00+t0)*FrameN/2500)+1)...
:round((t00+t0+t01)*FrameN/2500)));
T1=t1.*dt+sum(Dt((round((t00+t0+t01)*FrameN/2500)+1)...
:round((t00+t0+t01+t1)*FrameN/2500)));
T12=t12.*dt+sum(Dt((round((t00+t0+t01+t1)*FrameN/2500)+1)...
:round((t00+t0+t01+t1+t12)*FrameN/2500)));
T2=t2.*dt+sum(Dt((round((t00+t0+t01+t1+t12)*FrameN/2500)+1)...
:round((t00+t0+t01+t1+t12+t2)*FrameN/2500)));


Theta0R=atan2(iy2,ix1-ix2);
Theta1R=atan2(iy2,ix3-ix2);
Theta2R=atan2(iy2,ix3-ix4);
Theta01=abs(Theta0R-Theta1R);
Theta12=abs(Theta1R-Theta2R);


Theta0=atan2(sin(Theta01),(T0./T1-cos(Theta01)));
Theta1=atan2(sin(Theta01),(T1./T0-cos(Theta01)));
Theta1_1=atan2(sin(Theta12),(T1./T2-cos(Theta12)));
Theta2=atan2(sin(Theta12),(T2./T1-cos(Theta12)));
G=(1./sin(Theta1)+1./sin(Theta2)+2*T12./(T1.*sin(Theta1)))...
/(1./sin(Theta0)+1./sin(Theta1)+2.*T01/(T0.*sin(Theta0)));
D1=Ty/sin(Theta1R);
L01=D1./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));
L12=D1.*G./(sin(Theta0)./sin(Theta01)+G.*sin(Theta2)./sin(Theta12));

V01=L01./(T01+T0./2+T0.*sin(Theta0)./(2.*sin(Theta1)));
R0=V01.*T0.*sin(Theta0)./2;

VX01=V01.*cos(Theta0+Theta0R-pi);
```

```
VY01=V01.*sin(Theta0+Theta0R-pi);

RV01(i)=V01;
RVX01(i)=VX01;
RVY01(i)=VY01;
RR0(i)=R0;
end
figure(5)
plot(RV01)
xlabel('(times)')
ylabel('(m/s)')
figure(6)
plot(RVX01)
xlabel('(times)')
ylabel('(m/s)')
figure(7)
plot(RVY01)
xlabel('(times)')
ylabel('(m/s)')
figure(8)
plot(RR0)
xlabel('(times)')
ylabel('(m)')

%function DrawDashLine cite http://www.engr.uky.edu/~lgh/soft/matlab/DrawDashLine.m
function [B] = DrawDashLine(A,x1,y1,x2,y2,dash1,dash0)
B=A;
[My,Nx]=size(B);
% find out which dimension is the domain
dx=(x2-x1);
dy=(y2-y1);
ihorizvert=1; % dy > dx
if abs(dx)>abs(dy)
    ihorizvert=0;
end;
if ihorizvert==0 %horizontal domain
    toggle=0;
    a=dy/dx;b=y1-a*x1;
    for x=x1:x2
        y=a*x+b;
        y=floor(y+0.5);
        if x>0
            if y>0
                if x<=Nx
                    if y<=My
                        if toggle==0
                            B(y,x)=dash0;
                            toggle=1;
                        else
                            B(y,x)=dash1;
                            toggle=0;
                        end;
                    end; % if y<=My
                end; % if x<=Nx
            end; % if y>0
        end; % if x>0
    end; % for x
else % vertical domain
    toggle=0;
    a=dx/dy;b=x1-a*y1;
    for y=y1:y2
        x=a*y+b;
        x=floor(x+0.5);
        if x>0
            if y>0
                if x<=Nx
                    if y<=My
                        if toggle==0
                            B(y,x)=dash0;
                            toggle=1;
                        else
                            B(y,x)=dash1;
                            toggle=0;
                        end;
                    end; % if y<=My
                end; % if x<=Nx
            end; % if y>0
        end; % if x>0
    end; % for x
end; % end of vertical domain
end

%function DrawCircle cite http://www.engr.uky.edu/~lgh
function [B] = DrawCircle(A,xc,yc,iR,ivalue)
```

```
B=A;
[My,Nx]=size(B);
iR0=-floor(iR);
iR1=iR0+2*iR;
for x=iR0:iR1
for y=iR0:iR1
    r=(x)*(x)+(y)*(y);
    r=sqrt(r);
    if(r<=iR)
        ix=floor(x+xc);iy=floor(y+yc);
        if ix>=1
             if ix<=Nx
                  if iy>=1
                      if iy<=My
                        %
                         B(iy,ix)=ivalue;
                       end; % y<=My
                    end; %y>=1
                end; %x<=Nx
          end; % x>=1
        end; % if r<R
end; % for y
end; % for x
end
```

Bibliography

[1] A. Sinha and D. Brady. Size and shape recognition using measurement statistics and random 3d reference structures. *Optics Express*, 11(20):2606–2618, 2003.

[2] D.J. Brady, N.P. Pitsianis, and X. Sun. Reference structure tomography. *JOSA A*, 21(7):1140–1147, 2004.

[3] U. Gopinathan, DJ Brady, and NP Pitsianis. Coded apertures for efficient pyroelectric motion tracking. *Optics Express*, 11(18):2142–2152, 2003.

[4] Y. Zheng, D.J. Brady, and P.K. Agarwal. Localization using boundary sensors: An analysis based on graph theory. *ACM Transactions on Sensor Networks (TOSN)*, 3(4):21, 2007.

[5] Y. Zheng. *Efficient object localization and tracking with discrete spatial mapping*. PhD thesis, Duke University, 2005.

[6] P. Potuluri, U. Gopinathan, J. Adleman, and D. Brady. Lensless sensor system using a reference structure. *Optics Express*, 11(8):965–974, 2003.

[7] P. Potuluri, M. Xu, and D. Brady. Imaging with random 3d reference structures. *Optics Express*, 11(18):2134–2141, 2003.

[8] Q. Hao, F. Hu, and Y. Xiao. Multiple human tracking and identification with wireless distributed pyroelectric sensor systems. *Systems Journal, IEEE*, 3(4):428–439, 2009.

[9] X. Zhou, Q. Hao, and H. Fei. 1-bit walker recognition with distributed binary pyroelectric sensors. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 168–173. IEEE, 2010.

[10] Q. Hao, D.J. Brady, B.D. Guenther, J.B. Burchett, M. Shankar, and S. Feller. Human tracking with wireless distributed pyroelectric sensors. *Sensors Journal, IEEE*, 6(6):1683–1696, 2006.

[11] J.S. Fang, Q. Hao, D.J. Brady, B.D. Guenther, and K.Y. Hsu. Real-time human identification using a pyroelectric infrared detector array and hidden markov models. *Optics express*, 14(15):6643–6658, 2006.

[12] N. Li and Q. Hao. Multiple human tracking with wireless distributed pyro-electric sensors. In *SPIE Defense and Security Symposium*, pages 694033–694033. International Society for Optics and Photonics, 2008.

[13] N. Li and Q. Hao. Multiple walker recognition with wireless distributed pyroelectric sensors. *Proc. of SPIE Defense and Security*, page 694034, 2008.

[14] S. Qingquan, H. Fei, and Q. Hao. Context awareness emergence for distributed binary pyroelectric sensors. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 162–167. IEEE, 2010.

[15] X. Wang and B. Moran. Multitarget tracking using virtual measurement of binary sensor networks. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. IEEE, 2006.

[16] M. Shankar, J. Burchett, S.D. Feller, B. Jones, R. Swagart, B.D. Guenther, and D.J. Brady. Biometric tracking with coded pyroelectric sensor clusters. In *Proc. of SPIE Vol*, volume 5796, page 175, 2005.

[17] H. Saito, S. Tanaka, and S. Shioda. Estimating parameters of non-convex target object using networked binary sensors. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pages 146–154. IEEE, 2010.

[18] Y. Zheng, N.P. Pitsianis, and D.J. Brady. Nonadaptive group testing based fiber sensor deployment for multiperson tracking. *Sensors Journal, IEEE*, 6(2):490–494, 2006.

[19] M. Peng and Y. Xiao. A survey of reference structure for sensor systems. *Communications Surveys & Tutorials, IEEE*, 14(3):897–910, 2011.

[20] P.K. Agarwal, D. Brady, and J. Matoušek. Segmenting object space by geometric reference structures. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):455–465, 2006.

[21] Ronald W. Schafer Alan V. Oppenheim. *Discrete-time Signal Processing.* Prentice-Hall, Inc., 1989.

Vita

Mengmei Liu was born in Jinzhou, China. She was awarded the Bachelor of Engineering in Electrical Engineering from Wuhan University of Technology in 2010. Currently she is a research assistant at University of Kentucky.