OPTIMIZED ESTIMATES BASED ON MULTIPLE SENSOR CONFIGURATION KNOWLEDGE

by

ROOCHI MISHRA

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2013

To my family and friends.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Victoria Chen and co-supervising professor Dr. Brian Huff for constantly motivating and encouraging me, and also for their invaluable advice during the course of my doctoral studies. I wish to thank my academic advisor Dr. Jay Rosenberger for his interest in my research and for taking time to serve in my dissertation committee.

COMMITTEE

Supervising Professor : Dr. Victoria C. Chen

Co-Supervising Professor : Dr. Brian Huff

Committee Member : Dr. Jay Rosenberger

Graduate Advisor : Dr. Sheik Imrhan

July 19, 2013

ABSTRACT

OPTIMIZED ESTIMATES BASED ON MULTIPLE SENSOR CONFIGURATION KNOWLEDGE

Roochi Mishra, Ph.D.

The University of Texas at Arlington, 2013

Supervising Professors: Victoria C. P. Chen, Brian L. Huff

There has been an increase in the usage of sensor technology as they are adaptable for use in different environments, many of which are hostile to direct human observation, such as regions affected by land mines or forest fires. The sensors used to monitor developing situations are small inexpensive computing devices with limited processing capabilities, limited power supply and may be destroyed in the event that they are monitoring, without suffering a great financial cost. Also, since these devices are inexpensive, multiple sensors can be deployed for the same application where the sensors are placed or attached to a mobile platform in a particular configuration or shape. However, because these devices are inexpensive, they do not possess all the software and hardware necessary to produce accurate observed data from the environment they are deployed in. Moreover, there are additional factors such as atmospheric conditions, network delays, sensor characteristics etc. that may affect the measures being monitored and therefore, the data observed and produced as output by the sensors may be inaccurate.

The issue of obtaining accurate estimates of location and orientation from inaccurate observed sensor data has been subjected to thorough investigation in the literature. However, in the application environment of multiple Global Positioning Satellite (GPS) sensors attached to a mobile robot platform, these previous methods do not take advantage of the sensor configuration information to produce more accurate estimates of the measures being observed. In this dissertation the authors will demonstrate that in fact, with the use of the sensor configuration and the inaccurate observed sensor data, it is possible to obtain accurate estimates of location and orientation of the deployed sensors.

In this dissertation, we propose several concrete issues and their respective solutions for the framework of the mobile robot platform with GPS sensors attached in a particular configuration to be operational. We use optimization techniques to fit estimates of locations and orientations of the sensors on the mobile platform given observed data that is highly unstable when the platform is stationary or in motion, while taking advantage of the known configuration knowledge. To deal with outliers and missing data, we use statistical and heuristic weighting techniques to favor accurate observed sensor data over inaccurate data. Moreover, the production of estimates through the use of optimization has to conform to the real-time constraints when the platform is in motion and therefore, we introduce sliding windows to be able to generate updated estimates. Furthermore, depending on the size of the sliding window, the task of correcting the lag between the estimate over the sliding window and the estimate with respect to the current time-step is also considered to produce more accurate results.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

There are many areas of the world where constant monitoring is needed to learn about an environment or observe developing situations. Sensor devices allow us to remotely supervise potentially inhospitable environments and provide the data needed to make decisions as a result of changes in these locations. Depending on the application, more than one type of sensor may be used with wide-varying reliability. To be able to make a coherent decisions based on newly received data, we must develop a method to aggregate different types of sensor information, while determining the precision and accuracy of the data being received. This dissertation aims to provide a framework for processing sensor data from multiple sources, in order to as increase the reliability of the sensor data system, which in turn increases the quality of our decision-making process.

1.1 Sensor Devices and Data

Sensor devices are small computing devices used to measure attributes in a given local setting for the purposes of relaying that data for further processing to other computing devices with more resources.

Sensor devices generally possess low processing and memory capabilities, limited power supply and are low in cost. Due to this low cost, they are generally spread in bulk around the territory from which we want to obtain data. The low cost indirectly implies that these sensor devices are not rigorously verified and validated

and therefore, the data that is obtained from such devices is subject to high variance and bias.

1.2 Data Fusion

Depending on the type of application, different sensors are used to collect different types of data. Global Positioning System (GPS) sensors are capable of providing location and time information in different atmospheric conditions. Accelerometers measure speed and direction. Other sensors such as gyroscopic sensors, temperature sensors etc. also exist. In any given application, it is possible that one or more different types of sensors will be deployed for the purposes of collecting data. In these situations, it is important to combine the information not only across different sensors measuring the same information, but also across the different types of sensors to provide one effective measure for making a decision.

1.3 Motivating Examples

The size and cost effectiveness of sensors make it a versatile tool in a variety of environmental conditions. The small form factor of these devices allow for deployment in areas where they are mostly imperceptible, while the low cost allows multiple redundant sensors to be set up for the purposes of collecting data.

1.3.1 Monitoring Robot Location

One possible application is for the monitoring of mobile robot platforms in areas to search for hidden landmines in war-torn regions. This type of mobile robot requires a multitude of sensors attached to a moving platform such as GPS sensor units for ascertaining the location of the platform and accelerometers to obtain the speed and direction as the platform moves over uncertain and possibly rugged terrain.

Other sensors are attached to the mobile platform to determine whether the landmine exists in current location of the mobile platform. This is a very useful task for which sensors are vital, since human lives need not be risked in the very dangerous task of retrieving unexploded landmines. This application also demonstrates the different types of sensors used, the number of each type of sensor used and the necessity of position of the robot to be precise and accurate to assist in the decision of whether or not a landmine is located at the current location of the platform.

The application space under which our framework is relevant is a sub-section of the application of monitoring hazardous environments using mobile robot platforms. In this case, multiple GPS and accelerometer sensors are attached to a mobile robot platform. With the GPS information received by a base station, the location of mobile platform must be estimated so as to track and direct the path of the mobile platform. This dictates that the information received should be accurate and precise. However, in the initial studies, it has been noted that there is a high degree of variance and bias in the GPS data from the sensors. To combat this situation, we propose multiple tasks for obtaining the best estimate for the location of the mobile platform.

1.3.2 Monitoring Movement of Forest Line of Fire

Another application of sensor technology is in determining fire conditions in a forest, where sensors spread across an area can provide data about temperature, location and the speed with which the line of fire is moving. Again here, the data from multiple sensors and multiple types of sensors need to be combined with high precision and accuracy so as to detect fires and keep them from spreading further and causing harm. Additionally, with the possibility of a sensor being consumed in the fire, the lack of data is as important as the actual data.

1.4 Problem Statement

As discussed in the previous sections, data from sensor deployments can be used for various applications. As per our problem space, we attempt to estimate the location and track the movement of a mobile robot platform given minimal initial time-invariant knowledge. To resolve issues relating to high variance and potential bias in the received GPS coordinate data from the sensors, we propose the following tasks :

1. Obtaining An Estimate using Quadratic Optimization and Known Sensor Configuration : Given the configuration of multiple sensors arranged on the mobile platform in addition to the observed measurements from all the sensors, an estimate of the location of the sensors of the platform can be obtained using quadratic optimization. For the case when the mobile platform is stationary, all the observed measurements may be considered. For calculating the estimate while when the mobile platform is in motion, the estimates are continually updated.
2. Position and Orientation Estimation : In this task, the problem of estimating position and orientation of a mobile platform is further explored in the context of GPS sensors and using the quadratic optimization solution for estimating the location of the mobile platform for known sensor configurations.
3. Outlier Analysis and Weighting : The accuracy of the data received from the GPS sensors is subject to atmospheric conditions, satellite constellations, individual sensor characteristics and noise. Additionally, some of these data may be erroneous. These issues lead to the data having low reliability, making it difficult to yield accurate estimates for location and orientation. Moreover, depending on our analysis of historical data of the sensors, we may be able to identify sensors which have been closer to the true values and thus, should be

assigned higher weights, hence contributing more to the resulting estimate of the location. The outlier analysis and the weighting schemes will provide the framework for favoring accurate data with respect to the sensor configuration for the purposes of increasing the accuracy of the estimates of location of the sensors and platform.

4. Sliding Windows and Correction for Lag : Besides the impact of missing data on accuracy of the estimates, the estimates are also subject to real-time utility constraints when the platform is in motion. It is also not feasible to store all observed data that has been collected over long period of time. It is necessary to strike a balance between the amount of historical data stored along with the new observed data and the accuracy of the estimates produced. The amount of historical data stored also depends on the path information and the rate of change in the position and orientation of the mobile platform. In the cases of using a large amount of historical data, the calculation of the estimates may not be able to respond quickly to sudden changes such as turns, leading to lag. This needs to be accounted for and corrected to produce more accurate estimates of position.

For the purposes of producing a demonstration of the efficacy of these solutions, we are limiting the application space to that of GPS sensors on a mobile robot platform. We will be using minimal domain information for estimating the location and orientation of the GPS sensors. One may imply that this solution would not be applicable under general conditions with different data characteristics than GPS sensors, but we intend to show that our framework is indeed adaptable as the application requires it to be. Regardless of the application space and type of sensors used, the framework of improving the quality of information from these sensors will remain the same.

1.5 Outline of the Dissertation

This dissertation is divided into the following chapters: (*a*) The necessary background information is discussed in Chapter 2; (*b*) Chapter 3 features a discussion of our approach and methodology to solving the above detailed issues; (*c*) Chapter 4 shows the results of our approaches; (*d*) Chapter 5.2 provides a summary of our results and an outline for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter of the dissertation reviews the different approaches our research took for the tasks listed in Section 1.4. However, not all of these methods were utilized in the current solutions to the aforementioned tasks.

2.1 Quadratic Programming in MATLAB®

Quadratic programming [1] is a type of mathematical optimization problem that involves a quadratic objective function and linear constraints. The standard form a quadratic programming problem is shown in Equation 2.1:

$$\begin{aligned} MIN \quad f(x) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ st. A\mathbf{x} \leq \mathbf{b} \\ E\mathbf{x} = \mathbf{d} \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}$, $\mathbf{x}$ and $\mathbf{c}$ are column vectors with n elements and Q is a symmetric $n \times n$ matrix, $\mathbf{x}^T$ is a transpose of $\mathbf{x}$. Additionally, the area of quadratically constrained quadratic programming allows a generalization of not only having a quadratic objective but also having quadratic constraints. Quadratic programs can be solved using a variety of algorithms, some of which are [2]:

1. Trust Region Reflective Algorithms : Trust regions [2] follow the idea of approximating the objective with a simpler function that reflects the behavior of the objective at a particular point x in a neighborhood N, thus creating a

trust-region subproblem. The point x is updated if the objective function value decreases; otherwise the neighborhood N is shrunk and all steps are repeated.

2. Active Set Algorithm : From [3], in quadratic programming, the active set algorithm initializes by finding a feasible solution after which the constraints are divided into active and inactive. Equality constraints are always active and the algorithm can continue searching for a solution within the reduced subset of active constraints. Whether the constraint is active or inactive is decided by computing Lagrange multipliers of the current active set and reducing the active set to contain only non-negative Lagrange multiplier results and then repeat the search for infeasible constraints.
3. Sequential Quadratic Programming (SQP) Algorithm : As developed in [2], the active set algorithm and the SQP algorithm are similar except that the SQP algorithm takes every iterative step in a region constrained by bounds. Additionally, it may take an iterative step that returns an invalid objective value, which is corrected in the next iteration by taking a smaller step. In case constraints are not met, a merit function is calculated that combines the objective and constraint functions. This merit function is then minimized with relaxed constraints. The SQP algorithm is also capable of using second-order approximation on the constraints, if a solution is still not found.
4. Interior Point Algorithm : As implemented in [2], the interior point approach consists of solving a sequence of approximate minimization problems with a conversion of inequality constraints to equality constraints using slack variables and an added logarithmic term called a barrier function. Solving the approximate problem requires two steps, a direct step (solving Karush-Kuhn-Tucker equations) via a linear approximation and then a conjugate gradient step using a trust-region.

2.2 Time Series Analysis

As per [4] and [5], time series analysis is one of the methods to obtain a statistical model of data when there is a correlation introduced by the sampling of adjacent points in time. The time domain approach is generally motivated by the assumption that correlation between adjacent points in time is accounted for by a dependence of the current value on the past values and therefore, modeling of future values of that time series contains a parametric function of the current and past values. One of the seminal approaches to analyzing time series data was developed by Box and Jenkins [6] which involved a class of models called autoregressive integrated moving average (ARIMA) models to handle the modeling and forecasting of time series data. The advantage that ARIMA models have over other statistical models is the use of multiplicative models, such that the observed data are assumed to result from the products of factors involving differential or difference equation operators responding to a white noise input. Initially, classical regression methods are tested to check if they are sufficient for modeling the trends and seasonal effects generally noted in time series data.

For time series analysis to be an effective tool to model time series data, it is imperative to conduct initial tests to find out if the data in hand follows a time series model or not. By obtaining a time series plot, one can look for no consistent trends and sharp jagged-ness, all of which are hallmarks of time series data. The first model to be tested against the data is the autoregressive model of order 1 : AR(1) as seen in Equation 2.2 with assumptions that errors are i.i.d and $N(0, \sigma_w^2)$ and properties of errors ω_t are independent of x. The AR(1) model

$$x_t = \delta + \phi_1 x_{t-1} + \omega_t \tag{2.2}$$

works when lag 1 values of the series show a linear association, positive or negative. Then, residual analysis is conducted to note serious issues such as non-constant variance and outliers. It is also interesting to check the sample autocorrelation function (ACF) plot where the lag (time steps between observations) is plotted on the x-axis and the autocorrelation function value is plotted on the y-axis. Lines indicating statistical significance bounds are also plotted and to obtain a good ACF for residuals, the autocorrelation should not exceed the statistical significance boundaries. This is to make sure that there is no significant autocorrelation in the residuals. If significant autocorrelation is present in the residuals, a different model should be tested.

2.3 Handling Missing Values

Working with unreliable sensors may lead to data not being received from the sensors. There are various reasons for that to happen, such as atmospheric conditions, delays from GPS satellites, sensor characteristics etc. Missing data is a phenomenon that cannot simply be ignored and [7] reviews the different methods available currently to account for missing data. Initially, the type of missing data must be identified out of four possible types :

1. Missing by Definition of the subpopulation : If a sample was not collected from a particular subpopulation due to omission, then these should be eliminated/noted before handling the true missing values.
2. Missing completely at random (MCAR) : Here the missing data may form a random distribution throughout the data.
3. Missing at random (MAR) : Here the missing data for a variable are MAR if the likelihood of missing data on the variable is not related to the response variable, after controlling for other variables in the data set.
4. NI missing values : These are missing values that are neither MAR or MCAR

Traditional approaches to handling missing values are listwise or case deletion (works well if sample is large and values are MCAR), pairwise deletion (it may produce a covariance matrix that is not positive definite), mean substitution or mean substitution for subgroups (attenuates variance and can produce inconsistent bias), all of which may result in biases in a positive or a negative direction, increase in Type II errors and underestimate correlations and β weights. Newer methods use expectation maximization (EM) or imputations [8]. Imputation is the direct replacement of missing subjects by new subjects from an identifiable source population based on observed subject characteristics. If the missing data follows MAR, the subject is randomly chosen from the same source population or from the estimation of the distribution of the test result in the source population. Using the estimation of the distribution for replacement purposes requires in complex situations with multivariate factors, a multivariate regression model to estimate the underlying distribution of the test result. This method is known is known as single imputation procedure. Multiple imputation takes into account that the initial estimate of the distribution had missing values. Therefore, several or multiple imputed data sets based on random draw from different estimated underlying distributions are used. An average is calculated to get pooled estimates of the association which results in lowering the variance of the combined estimate. [7] also makes mention of full information maximum likelihood estimation algorithms such as structural equation modeling and hierarchical modeling. In conclusion, both multiple imputation and full information maximum likelihood approaches work well on large samples.

2.4 Statistical Process Control

In [9], statistical process control (SPC) is defined as a powerful collection of problem-solving tools useful in achieving process stability and improving capability

through the reduction of variability. One of the tools of performing statistical process control is through the use of control charts. A control chart is a graphical display of a quality characteristics that has been measured or computed from a sample versus the sample number or time. A control chart consists of a center line, an upper control limit (UCL) and a lower control limit (LCL). A center line represents the average value of the quality characteristic corresponding to the in-control state. The purpose of UCL and LCL is to define the region within which the process is in control; if a sample point selected falls between the UCL and LCL, the process is in control. If the data are beyond the control limits, then the process is out of control. A control chart is analogous to hypothesis testing which defines a rejection region and then tests the sample point whether the data falls in the rejection region or not. The general model for a control chart is given in Equation 2.3 :

$$\begin{aligned} UCL &= \mu_w + k\sigma_w \\ CenterLine &= \mu_w \\ LCL &= \mu_w - k\sigma_w \end{aligned} \tag{2.3}$$

where w is a sample statistic, μ_w is the mean of w and the standard deviation of w is σ_w. There are as many control charts as there are sample statistics that characterize some quality attribute of interest. There are variable control charts such as the Shewhart $\bar{X}$, R, S, CUSUM, Exponentially Weighted Moving Average (EWMA) charts; attribute control charts such as P, NP, C and U charts; and multivariate control charts such as Hotelling T^2, Multivariate EWMA charts and Principal Components Control charts [10]. For the purposes of deriving a model for the location of GPS sensors, multivariate SPC charts were deemed more appropriate as it was believed that multiple quality characteristics would affect the output of a GPS sensor, alongwith the

multiple quality characteristics being possibly correlated. The most versatile chart is the Hotelling T^2 control charts.

2.4.1 Multivariate SPC Charts : Hotelling T^2

The author of [9] establishes that the Hotelling T^2 control chart is analogous to the univariate Shewhart $\bar{x}$ chart. The Hotelling T^2 statistic is shown in Equation 2.4, where n = 1 for individual observations, m is the number of samples, p is the number of quality characteristics observed in each sample, $\bar{x}$ is the sample mean vector and S is the covariance matrix.

$$T^2 = (x - \bar{x})' S^{-1} (x - \bar{x}) \tag{2.4}$$

The corresponding control chart for Phase I is shown in Equation 2.5 where $\beta_{\alpha,\frac{p}{2},\frac{m-p-1}{2}}$ is the upper α percentage point of a beta distribution, with the other two elements being parameters to the β distribution. Since n = 1, it has been shown that approximations using F and χ^2 distribution are likely to be inaccurate.

$$\begin{aligned} UCL &= \frac{(m-1)^2}{m} \beta_{\alpha,\frac{p}{2},\frac{m-p-1}{2}} \\ LCL &= 0 \end{aligned} \tag{2.5}$$

The control chart for Phase II is shown in Equation 2.6.

$$\begin{aligned} UCL &= \frac{p(m+1)(m-1)}{(m^2 - mp)} F_{\alpha,p,m-p} \\ LCL &= 0 \end{aligned} \tag{2.6}$$

In multivariate SPC, the cause of a signal sensing out of control situation may be caused by one of p variables or between two or more of the variables. To obtain the variable that is causing the signal, orthogonal decomposition of T^2 into its principal components and then interpret the principal components is one approach. The

authors of [11] present Mason-Young-Tracy (MYT) decomposition technique to a signaling T^2 statistic where MYT decomposition is shown in Equation 2.7 where T_1^2 is an unconditional Hotelling's T^2 for the first variable of the observation vector $\mathbf{X}$. The other terms are conditional terms with their formulas given in Equation 2.8

$$T^2 = T_1^2 + T_{2\cdot1}^2 + \cdots + T_{p\cdot1,2,\ldots,p-1}^2 \tag{2.7}$$

$$\begin{aligned} T_1^2 &= \frac{(x_1 - \bar{x_1})^2}{s_1^2} \quad Unconditional \\ T_{j\cdot1,2,\ldots,j-1}^2 &= \frac{(x_j - \bar{x}_{j\cdot1,2,\ldots,j-1})^2}{s_{j\cdot1,2,\ldots,j-1}^2} \quad Conditional \end{aligned} \tag{2.8}$$

Through the application of the sequential algorithm for MYT decomposition as described in [11], it is possible to interpret unconditional signaling terms as they measure whether an individual observation is within control, while a signaling conditional term signifies that the observation of the corresponding set of variables is counter to the relationship established by the historical data. The authors of [12] used the MYT decomposition algorithm to study the consistency of impurity profiles of drug substances where signals in the conditional components were interpreted as being due to the fouling of the relationship among impurities, thus, discovering not just signals for individual impurities but also those between impurities and transition points.

2.5 Data Fusion

From [13], we can define data fusion as the combination of information from disparate sources that are aiming to measure the same quantity, attribute or characteristic to provide the best estimate of the attribute. There are three general approaches to performing data fusion:

1. Dempster-Shafer Theory: It is also known as the theory of belief functions and is considered to be a generalization of the Bayesian theory of subjective probability. Dempster-Shafer does not require exact probabilities for each question of interest; instead, it allows for belief functions to base degrees of belief for questions of interest, even though degrees of belief may or may not have mathematical properties. DS theory is based on two ideas:
   (a) The idea of obtaining degrees of belief for one question from subjective probabilities for a "related" question
   (b) When the degrees of belief are based on independent items of evidence, DS theory allows for them to be combined.

   One use of this theory was in [14], where ultrasonic sensors were returning data with large uncertainties due to the presence of highly shiny surfaces and Dempster-Shafer theory was used as a filtering factor on the sensor model to reduce uncertainty. Dempster Shafer theory was used as opposed to a Bayes theory since the requirement was to build a grid of surroundings of the robot and assign three values as a measure of belief/mass : occupied, empty and unknown. Using Dempster's rule of combination allows for a calculation of confidence by using the masses of those three states of a grid.
2. Bayes Theory : Bayes' theory of data fusion uses Bayes' rule to combine successive measurements of the state of a system from a single source, which involves obtaining a new estimate for the target state given the previous old estimate.
3. Fuzzy Logic : Fuzzy logic is the reasoning approach that is approximate where fuzzy logic variables have a truth value that ranges between 0 and 1 as opposed to binary variables that have truth values that are either 0 or 1. For example, in [15], the authors attempt to correlate and fuse information from sensor data obtained from maritime sources and attempts to track the paths of ships through

the use of Adapative Fuzzy Logic Correlation using messages containing only positional data. The use of fuzzy logic was in terms of the value of confidence that needed to be used based on which type of sensor or data source was providing the information. Moreover, different fuzzy logic membership functions need to be defined based on the different applications, but the authors ran into issues when different fuzzy logic memberships needed to be fused.

Closer to one of our goal applications, the authors in [16] aimed to produce less biased estimates of the performance measured in probability of detection and probability of false alarm through the use of Fuzzy probabilities, Bayes theory and Dempster-Shafer Theory in an grid-map area by dividing the mapped area into grid square each of which uses sensors to produce confidence values. These confidence values are then used in data fusion for a final indicator about the presence of a land-mine by using a voting mechanism, where the votes are summed together and divided by the number of sensors. Their results indicate that Bayes' theory and Dempster-Shafer theory are more robust than Fuzzy Logic, which gave unpredictable results.

CHAPTER 3

METHODOLOGY

This section will describe the methodology for the tasks discussed in Chapter 1. Briefly the issues are :

1. Obtaining An Estimate using Quadratic Optimization and Known Sensor Configuration.
2. Position and Orientation Estimation.
3. Outlier Analysis and Weighting.
4. Sliding Windows and Correction for Lag.

Section 3.1 clarifies and distills our approach for estimating a measure using the observed data from multiple sensors, the configuration of the multiple sensors and the cost based solution to reducing error using quadratic optimization. Section 3.2 outlines the application of the quadratic optimization for estimating the position and orientation of a mobile platform. Section 3.3 details our plan for handling the issue of assessing the reliability of the data that we are receiving from the sensors through the use of outlier analysis techniques and weighting. Section 3.4 shows our strategy for dealing with missing observations using sliding windows and corrections for lag in estimates due to inefficient sliding window size through a demonstration of the consequences of using improper window sizes when using a sliding window scheme for optimization resulting in a time lag. Subsequently, the results of these approaches and performance metrics are presented Chapter 4 in their respective sections.

3.1 Obtaining An Estimate Using Quadratic Optimization and Known Sensor Configuration

Sensors that measure some aspect of the physical world provide data at varying rates which are provided as input for analysis and estimation. As previously discussed, this data is subject to large variance and high bias, along with errors. These data or errors may or may not follow a particular distribution and may or may not be identically and independently distributed. For the purposes of estimating the actual value of the measure that is being recorded, we can use an optimization based technique to reduce the errors between the observed measures and the estimated measure, while constraining the estimated measure to follow the configuration of the sensor as provided. Concretely, for the purposes of obtaining an estimate for the location of the mobile robot platform, we have developed a method for estimating the location of the mobile platform in this section, with estimation of orientation discussed in Section 3.2. Quadratic optimization was used due to visual inspection techniques which showed the random structure of the data violating all classical statistical assumptions with results discussed in Chapter 4. Initially, we discuss the quadratic optimization solution for the stationary case where the mobile platform is not moving and extend the same principles to the case where the platform is in motion. Subsequently,

1. Case A : Platform is stationary.
2. Case B : Platform is in motion.

As we proceeded through the steps of analysis, we opted to use an optimization based technique for location and orientation estimation.

3.1.1 Case A: Mobile platform is stationary

In our current application, the components that are available for the purposes of building an optimization framework to determine an estimate for the location of the mobile robot platform are:

1. Distances between each sensor on the mobile robot platform: These are distance values that stay constant for a particular application instance and are defined by the user. The complete collection of all these distances define the "sensor configuration" of the platform with sensors attach at its various points. This is denoted by the notation a_{ij}, where i, j are the numbers of sensors out of the m available sensors in the system and listed for each $i < j$ for all possible number of sensors.
2. Number of sensors: Also defined by the user, denoted by m.
3. The observed values of locations of each sensor: These are GPS coordinates in meters on two dimensions represented as x and y values (x, y). This is denoted by the notation $(x_{i,t},\ y_{i,t})$, where i is the sensor identification number and t is the value of the time-step (in our application, the time-step used is seconds) during which the observed values were recorded.

Additionally, we denote the estimated location of the sensors in a fashion similar to the observed values of the locations of the sensors by using x and y coordinates, designated as $(\ \widehat{x_{i,t}},\ \widehat{y_{i,t}})$, where i is the sensor identification number. This estimate is calculated over all the data available in the stationary case. In Section 3.1.2 and 3.4, there is a discussion over the time-frame used in calculating the estimates when the platform is in motion.

The task we wish to solve is given the observed values of the locations of the sensors and the distances between the sensors, where do we estimate the sensors to be located ? The error/cost that we are minimizing is the distance between an observed

value of the location of the sensor and the estimated value of the location of the same sensor. More concretely, let us consider the case for 3 GPS sensors in the first time-step of operation of the stationary platform using the following notation where the starting time $t_{start} = 1$ and the ending time is $t_{end} = n$:

1. true known distance between sensor 1 and sensor 2 : a_{12}.
2. true known distance between sensor 2 and sensor 3 : a_{23}.
3. true known distance between sensor 1 and sensor 3 : a_{13}.
4. observed (x,y) location for sensor 1 at time-step 1: $(x_{1,1}, y_{1,1})$.
5. observed (x,y) location for sensor 1 at time-step n: $(x_{1,n}, y_{1,n})$.
6. observed (x,y) location for sensor 2 at time-step 1: $(x_{2,1}, y_{2,1})$.
7. observed (x,y) location for sensor 2 at time-step n: $(x_{2,n}, y_{2,n})$.
8. observed (x,y) location for sensor 3 at time-step 1: $(x_{3,1}, y_{3,1})$.
9. observed (x,y) location for sensor 3 at time-step n: $(x_{3,n}, y_{3,n})$.

Initially, estimates of the location of the variables will be calculated over all available GPS values, which means that the starting time of the estimating the locations. This is instituted for all 3 GPS sensors using the following notation:

1. estimated (x,y) location for sensor 1 at time-step 1: $(\widehat{x_{1,1}}\ ,\ \widehat{y_{1,1}})$.
2. estimated (x,y) location for sensor 1 at time-step n: $(\widehat{x_{1,n}}\ ,\ \widehat{y_{1,n}})$.
3. estimated (x,y) location for sensor 2 at time-step 1 : $(\widehat{x_{2,1}}\ ,\ \widehat{y_{2,1}})$.
4. estimated (x,y) location for sensor 2 at time-step n : $(\widehat{x_{2,n}}\ ,\ \widehat{y_{2,n}})$.
5. estimated (x,y) location for sensor 3 at time-step 1: $(\widehat{x_{3,1}}\ ,\ \widehat{y_{3,1}})$.
6. estimated (x,y) location for sensor 3 at time-step n: $(\widehat{x_{3,n}}\ ,\ \widehat{y_{3,n}})$.

For defining the cost between three sensors, we use the difference between Euclidean distance between the observed and the estimated *and* the true known distance, as shown for our example for three sensors using the notation above in Equation 3.1 for time-step 1.

$$
\begin{aligned}
COST &= \sqrt{(x_{1,1} - \widehat{x_{1,1}})^2 + (y_{1,1} - \widehat{y_{1,1}})^2} \\
&+ \sqrt{(x_{2,1} - \widehat{x_{2,1}})^2 + (y_{2,1} - \widehat{y_{2,1}})^2} \\
&+ \sqrt{(x_{3,1} - \widehat{x_{3,1}})^2 + (y_{3,1} - \widehat{y_{3,1}})^2}
\end{aligned} \tag{3.1}
$$

Constraints are expressed in the framework as Euclidean distances between estimated position of one sensor and the estimated position of second sensor which must be exactly equal to the data provided by the user regarding the "sensor configuration" of the sensors attached to the platform. The constraints for Sensor 1,Sensor 2 and Sensor 3 is demonstrated in Equation 3.2 for time-step 1.

$$
\begin{aligned}
(\widehat{x_{1,1}} - \widehat{x_{2,1}})^2 + (\widehat{y_{1,1}} - \widehat{y_{2,1}})^2 - a_{12}^2 = 0 \\
(\widehat{x_{1,1}} - \widehat{x_{2,1}})^2 + (\widehat{y_{1,1}} - \widehat{y_{2,1}})^2 - a_{23}^2 = 0 \\
(\widehat{x_{1,1}} - \widehat{x_{2,1}})^2 + (\widehat{y_{1,1}} - \widehat{y_{2,1}})^2 - a_{13}^2 = 0
\end{aligned} \tag{3.2}
$$

Here, it is not necessary to denote the time over which the estimates were calculated due to the fact that we compute only one set of estimate values as the result for the entire system of quadratic programming equations. Thus, Equations

3.1 and 3.2 will be reduced to Equation 3.3, over multiple time-steps $t = \{1 \dots n\}$ while dropping the t for the estimates being calculated.

$$\begin{aligned}
\textbf{min } Z &= \sum_{t=1}^{n} \left(\sqrt{(x_{1,t} - \widehat{x_1})^2 + (y_{1,t} - \widehat{y_1})^2} \right) \\
&+ \sum_{t=1}^{n} \left(\sqrt{(x_{2,t} - \widehat{x_2})^2 + (y_{2,t} - \widehat{y_2})^2} \right) \\
&+ \sum_{t=1}^{n} \left(\sqrt{(x_{3,t} - \widehat{x_3})^2 + (y_{3,t} - \widehat{y_3})^2} \right) \\
s.t & \\
& (\widehat{x_1} - \widehat{x_2})^2 + (\widehat{y_1} - \widehat{y_2})^2 - a_{12}^2 = 0 \\
& (\widehat{x_2} - \widehat{x_3})^2 + (\widehat{y_2} - \widehat{y_3})^2 - a_{23}^2 = 0 \\
& (\widehat{x_1} - \widehat{x_3})^2 + (\widehat{y_1} - \widehat{y_3})^2 - a_{13}^2 = 0
\end{aligned} \tag{3.3}$$

The t index will be necessary during the case where the mobile platform is in motion and will be re-introduced in Section 3.1.2. Additionally, each distance provided by the user is converted into a constraint for the relative positions of the estimates of locations of the sensors. If the relative locations of the sensors form a fully connected graph with vertices representing the sensors and the edges representing the distances between them, there can be only a maximum of $\frac{i(i-1)}{2}$ such edges and therefore, constraints for i sensors.

The objective function in Equation 3.3 can be alternately expressed to account for any number of sensors, or more generally, the objective and constraints can be expressed for any m sensors over time n using Equation 3.4 with $\forall i, j = \{1 \dots m\}, i < j$. This produces one set of estimates of locations of the sensors.

$$\begin{aligned}
\textbf{min } Z &= \sum_{i=1}^{m} \sum_{t=1}^{n} \left(\sqrt{(x_{i,t} - \widehat{x_i})^2 + (y_{i,t} - \widehat{y_i})^2} \right) \\
s.t & \\
& (\widehat{x_i} - \widehat{x_j})^2 + (\widehat{y_i} - \widehat{y_j})^2 - a_{ij}^2 = 0
\end{aligned} \tag{3.4}$$

For the purposes of reducing computation costs and time, certain simplifications of the problem statement were used. Recall that the objective statement of the quadratic programming optimization problem to estimate locations of the sensors, shown in Equation 3.4, involves the Euclidean distances between the observed (x, y) UTM coordinates and the estimated (x, y) UTM coordinates for each sensor. In linear and quadratic programming, the objective is used for the purposes of selecting values for the location estimates (or the unknowns) are minimum and conform to the constraints. Since the process of these selections involve comparison with only one objective value with the another objective value, the squared errors between the observed and the estimate locations is all that is needed, as seen in Equation 3.5 and implemented.

$$\textbf{min } Z = \sum_{i=1}^{m}\sum_{t=1}^{n}\left((x_{i,t} - \widehat{x_i})^2 + (y_{i,t} - \widehat{y_i})^2\right) \tag{3.5}$$

With the presence of a large number of equality constraints, obtaining the estimates was time consuming and inaccurate due to the processing of the possibilities by the Active-Set algorithm. To alleviate that issue, the equality constraints were converted to inequality constraints with a small tolerance ϵ added and subtracted to provide some elasticity to the values of the distances between the sensors as demonstrated in Equation 3.6 for Sensors 1 and 3, as shown generally in Equation 3.7 which presents the entire quadratic programming problem for $\forall i, j = \{1 \ldots m\}, i < j$. In this case, there are two constraints for each edge in the sensor configuration. Therefore, the maximum number of constraints is $i(i-1)$ for i sensors.

$$\begin{aligned}(\widehat{x_1} - \widehat{x_3})^2 + (\widehat{y_1} - \widehat{y_3})^2 - (a_{13} + \epsilon)^2 \leq 0 \\ (a_{13} - \epsilon)^2 - (\widehat{x_1} - \widehat{x_3})^2 - (\widehat{y_1} - \widehat{y_3})^2 \leq 0\end{aligned} \tag{3.6}$$

$$\begin{aligned} \textbf{min } Z &= \sum_{i=1}^{m}\sum_{t=1}^{n}\left((x_{i,t}-\widehat{x_i})^2+(y_{i,t}-\widehat{y_i})^2\right) \\ s.t & \\ & (\widehat{x_i}-\widehat{x_j})^2+(\widehat{y_i}-\widehat{y_j})^2-(a_{ij}+\epsilon)^2\leq 0 \\ & (a_{ij}-\epsilon)^2-(\widehat{x_i}-\widehat{x_j})^2-(\widehat{y_i}-\widehat{y_j})^2\leq 0 \end{aligned} \tag{3.7}$$

Additionally, computational time was further reduced through an enhanced method of selecting starting points. Recall, that in the quadratic optimization algorithms, such as Active-Set [3], selection of a starting point is provided as input and at times the key to the success of generating appropriate solutions to the quadratic programming problem. To ensure that the mobile platform receives updates to the estimates of the location of the sensors in initial and subsequent iterations, the estimates obtained as the result of the previous calculation are used as the starting point for the next calculation of estimates. More concretely, the estimates obtained in iteration j are used as the starting point in iteration $j+1$ to calculate the estimates of location of sensors. This addition reduced the computational time of quadratic programming problem significantly.

The estimates of locations of the sensors can be used to obtain the estimate of the position of center of gravity of the sensor configuration on the mobile platform. With the estimates of the locations of the actual sensors obtained directly from the non-stationary case using Equation 3.9 or the stationary case using Equation 3.7, the location of the center of gravity of the sensor configuration $(\widehat{x_t^{\mathrm{CG}}}, \widehat{y_t^{\mathrm{CG}}})$can be calculated using Equation 3.8 with $i = \{2, 3, 6, 7\}$ over time $t = \{1, \ldots, n\}$ based on the knowledge from observing the sensor configuration described in Section 4.1. These sensors were selected by taking advantage of the knowledge we possess about

the configuration as these sensors formed a configuration without completely missing sensors, such as Sensor 4.

$$\widehat{x_t^{\mathrm{CG}}} = \frac{\sum_{i=2,3,6,7} \widehat{x_{i,t}}}{m}$$
$$\widehat{y_t^{\mathrm{CG}}} = \frac{\sum_{i=2,3,6,7} \widehat{y_{i,t}}}{m} \tag{3.8}$$

It must be noted that the center of gravity of the platform is at a constant distance from the locations of the sensors attached to its corners and edges as described by the sensor configuration in Section 4.1. If the extra information regarding the missing sensors or configuration were *not* available, the center of gravity can be obtained by using a distance based formulation using the constant distance of the center of gravity and the location of the sensors.

3.1.2 Case B: Mobile platform is in motion

When the mobile platform is in motion, the data set of observed position values of the sensors available for estimation in the optimization framework changes from one time-step to the next time-step. One option is to calculate the estimate of the location and orientation of the mobile platform on a time-step by time-step basis using only the observed values in that one particular time step while the second option is to store the incoming data and update the estimate every time-step, using the expanding data set of observed values for the optimization framework. It is not possible in this case to wait and collect a large number of observed location values and only then, calculate an estimate for the location, since when the platform is in motion, older observed location values will no longer be relevant to determining the value of the estimate of the location, as the platform may have moved in the time-being to a completely different location. Also, the application subset under

consideration has real-time utility constraints over the estimates of locations that are obtained. An estimate obtained in time step t may not be relevant in time step $t+2$ or $t+3$. This also illustrates the storage issue of estimation since storing the observed location values is infeasible due to sensor memory constraints and the usefulness of the data after a certain amount of time. Therefore, in the second option, we propose the use of sliding windows, or a sufficient number of observed values that are updated at each time-step, to provide "fresh" estimates during motion. Sliding windows are explained in detail in Section 3.4.

The equation described in Equation 3.7 can be used to obtain an estimate of the location of the sensors. In the stationary case, it is customary to use all the data for the calculation of the estimate and therefore, the total size of the data under consideration is $t = \{1, \ldots, n\}$. When the platform is in motion, there are real-time constraints on when the estimate is useful. Therefore, it is not feasible to wait till the end of the platform's motion to provide the estimate. Depending on the constraints established by the user, these estimates may be provided at the end of a sliding window sized section of the data. The size of the window is defined by the number of time-steps used in the calculation of the estimate, denoted as s. We establish that although the total set of time-steps available to us is $t = \{1, \ldots, n\}$, in reality the available time-steps for use is $t = \{s, \ldots, n\}$ as estimates of location cannot be calculated without the requisite number of time-steps for the sliding window to be complete. Please refer to Section 3.4 for more details. Therefore, Equation 3.7 is transformed to Equation 3.9 with $\forall i, j = \{1, \ldots, m\}, i < j$ to account for the production of the estimate at the

end of the sliding window, where t time-steps can range from $\{s, \ldots, n\}$ with sensors $\forall i, j = \{1, \ldots, m\}, i < j$ and estimates are produced *every* time step $t = \{s, \ldots, n\}$.

$$\begin{aligned} \textbf{min } Z &= \sum_{i=1}^{m} \sum_{w=t-s+1}^{t} \left((x_{i,w} - \widehat{x_{i,t}})^2 + (y_{i,w} - \widehat{y_{i,t}})^2 \right) \\ s.t & \\ & (\widehat{x_{i,t}} - \widehat{x_{j,t}})^2 + (\widehat{y_{i,t}} - \widehat{y_{j,t}})^2 - (a_{ij} + \epsilon)^2 \leq 0 \\ & (a_{ij} - \epsilon)^2 - (\widehat{x_{i,t}} - \widehat{x_{j,t}})^2 - (\widehat{y_{i,t}} - \widehat{y_{j,t}})^2 \leq 0 \end{aligned} \tag{3.9}$$

In Equation 3.9, the limits for the inner sum delineate the observed data under consideration as a block of data of the size of the sliding window, s, starting at time-step $t - s + 1$ and ending at t. This is repeated for all values of $t = \{s, \ldots, n\}$, thus producing a set of estimates of locations of sensors for all t. More details are added in Section 3.4 with the discussion of sliding windows.

3.2 Position and Orientation Estimation

As described in Section 3.1, we can obtain the estimates of the positions of the all sensors and the center of gravity in the sensor configuration defined by using the quadratic programming problem described in Equations 3.7 for stationary and 3.9 for non-stationary cases. The results for these approaches are discussed in Chapter 4. For each estimate of location of the sensors, an orientation estimation can be calculated by using the initial estimate as a reference orientation and the current location estimation can be used to calculate the current orientation estimation. Additionally, orientation is computed with reference to the North direction as 0 radians or 0°. Refer to Figure 3.1 for common angles.

Orientation estimates are calculated using the N-direction (North direction) line as a reference and any side of the mobile platform as the opposing side and then applying the Law of Cosines in Equation 3.10 where a is the reference distance from

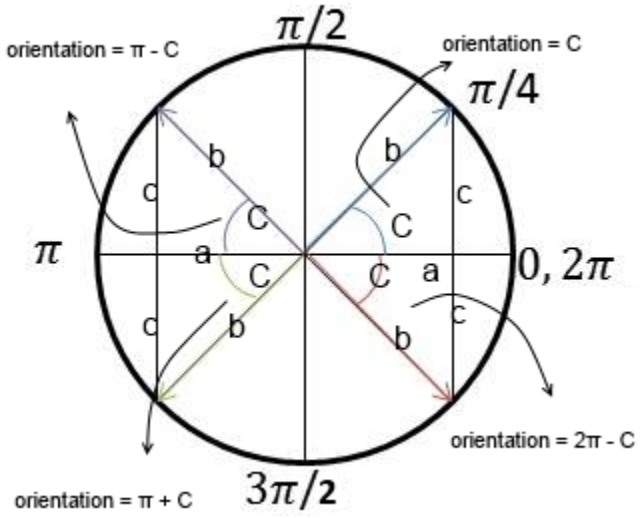


Figure 3.1: Calculation of orientation of mobile platform.

origin, b is the distance from origin for the estimated current location, c is the distance between the reference and the result to obtain an orientation estimate. To obtain the difference in orientation, this is subtracted from the original orientation estimate of the vehicle at the start of the observed data being recorded. This measure is used to further reduce the error in calculation of location estimates in the following time-steps through correction of lag when using large values of sliding windows. Additionally, the sides and locations of the estimates and references must be translated to the origin. Each estimate of orientation is calculated using estimates obtained for location of sensors and thus, calculated at either time-step $t = \{1, \ldots, n\}$or sliding window step $t = \{s, \ldots, n\}$ where s is the size of the sliding window.

$$C_t = \cos^{-1}\left[\frac{a_t^2 + b_t^2 - c_t^2}{2a_t b_t}\right] \tag{3.10}$$

Depending on the quadrant that the estimate of the current location of the platform, orientation can be calculated using the schemes shown in Figure 3.1. If the translated estimate lies :

1. Quadrant I : Quadrant I consists of the area between 0 and $\frac{\pi}{2}$, therefore, orientation = C_t calculated in Equation 3.10.
2. Quadrant II : Quadrant II consists of the area between $\frac{\pi}{2}$ and π, therefore, orientation = $\pi - C_t$.
3. Quadrant III : Quadrant III consists of the area between π and $\frac{3\pi}{2}$, therefore, orientation = $\pi + C_t$.
4. Quadrant IV : Quadrant IV consists of the area between $\frac{3\pi}{2}$ and 2π, therefore orientation = $2\pi - C_t$.

3.3 Outlier Analysis and Weighting

Due to the uncertainty regarding the precision and accuracy of the data, outlier analysis and removal of the outliers may provide greater precision and reduce the variance so as to provide a better estimate of the location of the mobile robot platform. There exist many methods of performing outlier analysis or selecting influential points. We use two popular methods of statistical outlier analysis for the stationary case : Huber weight function in Iteratively Re-weighted Least Squares Robust Regression method and Standardized Deleted Residuals to select points that are influential. In addition to those common statistical methods, we propose the Huff weight allocation algorithm for the stationary case and the motion case. Moreover, the Huff weight allocation system provides an intuitive method of allocating weights to a observed location value with respect to the relationships between the sensors and with respect to the reliability of the sensors. Subsequently, we attempt to leverage the use of sliding windows of different sizes and combining them with Huff weighting schemes for added benefits.

3.3.1 Huber Weight Function in Robust Regression

The Huber weight function in Robust Regression uses the Median Absolute Deviation (MAD) estimator instead of the Mean Square Error (MSE) to estimate the deviation from the median. This method suits our application framework even more so, due to the fact that we do not know the true value of the stationary location values and therefore, can only use a median as a substitute for the true value. The Huber weight function , $w_{i,t}^{\text{HUBER}}$, is calculated using the following steps for $i = \{1, \ldots, m\}$ sensors and $t = \{1, \ldots, n\}$ time-steps :

1. Calculate $\text{MAD}_{i,t} = \frac{1}{.6745} * median\{|e_{i,t} - median\{e_{i,t}\}|\}$, where $e_{i,t}$ is i^{th} residual for each sensor.
2. Calculate scaled residual, $u_{i,t} = \frac{e_{i,t}}{MAD_{i,t}}$.
3. Calculate weight value $w_{i,t}^{\text{HUBER}} = \begin{cases} 1, & |u_{i,t}| \leq 1.345 \\ \frac{1.345}{|u_{i,t}|}, & |u_{i,t}| > 1.345 \end{cases}$.

The optimization is altered slightly to account for the new weighting terms as described in Equation 3.12. Moreover, as the data is not analyzed using statistical methods, the definition of residual e_i is altered as well, as shown in Equation 3.11 where $e_{x_{i,t}}$ are the x-components residuals per sensor per time-step, $e_{y_{i,t}}$are the y-components residuals per sensor per time-step, $x_{i,t}$ are the x-components per sensor per time-step, $y_{i,t}$ are the y-components per sensor per time-step, $median\{x_i\}$ is the median of x-components, $median\{y_i\}$ is the median of y-components and i is the sensor number.

$$\begin{aligned} e_{x_{i,t}} &= x_{i,t} - median\{x_i\} \\ e_{y_{i,t}} &= y_{i,t} - median\{y_i\} \end{aligned} \tag{3.11}$$

The Huber weight function steps described above are repeated to produce Huber weights for x-components w_{i,t_x}^{HUBER} and y-components w_{i,t_y}^{HUBER} of locations sep-

arately but are combined in the optimization shown in Equation 3.12 with $\forall i, j = \{1, \ldots, m\}, i < j$.

$$\begin{aligned} \textbf{min } Z &= \sum_{i=1}^{m}\sum_{t=1}^{n}\left(w_{i,t_x}^{\text{HUBER}} * (x_{i,t} - \widehat{x_i})^2 + w_{i,t_y}^{\text{HUBER}} * (y_{i,t} - \widehat{y_i})^2\right) \\ s.t & \\ & (\widehat{x_i} - \widehat{x_j})^2 + (\widehat{y_i} - \widehat{y_j})^2 - (a_{ij} + \epsilon)^2 \leq 0 \\ & (a_{ij} - \epsilon)^2 - (\widehat{x_i} - \widehat{x_j})^2 - (\widehat{y_i} - \widehat{y_j})^2 \leq 0 \end{aligned} \tag{3.12}$$

3.3.2 Standardized Deleted Residuals

The use of deleted residuals for the purposes of making residuals more effective for detecting outliers is another suggested method. However, in our application space, the predicted response is denoted by the median and the steps of the algorithm are modified in the following manner :

1. Calculate median of the observed values for each sensor's x and y coordinate values and denote them as $median\{x_i\}$ and $median\{y_i\}$ $\forall i$ sensors.
2. Calculate $e_{x_{i,t}} = x_{i,t} - median\{x_i\}$ and $e_{y_{i,t}} = y_{i,t} - median\{y_i\}$ $\forall i$ sensors.
3. Calculate $d_{xi,t} = \dfrac{e_{x_{i,t}}}{1 - h_{tt,i}}$ and $d_{yi,t} = \dfrac{e_{y_{i,t}}}{1 - h_{tt,i}}$ $\forall i$ sensors where $h_{tt,i}$ is the hat matrix of sensor i.
4. Any values of $d_{xi,t}$ and $d_{yi,t} \leq 1.0$ will be considered as outliers and removed from estimation calculations.

The use of deleted residuals does not dictate the use of an altered form of the quadratic optimization for sensors and using Equation 3.9 is applicable.

3.3.3 Huff Weight Function

The Huff weight function is another method of calculating weights on data through the use of the mutual presence of sensor observed values. This algorithm is outlined further in the following steps :

1. For every timestep, calculate relationship i.e. every sensor that reported in that timestep for that sensor.
2. For every timestep, if that relationship exists, then calculate the Euclidean distance between those sensors.
3. Calculate error by subtracting the known given distance and Euclidean distance in Step 2 per timestep per sensor and take the absolute value.
4. Per timestep, take the sum all absolute errors divide by the number of relationships and denote it as average error.
5. If the average error $\geq$ threshold for that time-step, exclude all the sensor data obtained on that timestep for the purposes of generating the estimate.
6. To determine the relative reliability of a sensor works with respect to the other sensors reporting in at that particular timestep, take the sum of the absolute errors of all of the distance relationships of just that sensor in that timestep and denote it as sum of absolute error per sensor per timestep and divide them by the number of relationships per timestep per sensor to supply the relative error.
7. If the relative error $\geq$ threshold, for a given sensor exclude the observed values for calculating location estimates.
8. If the relative error $<$ threshold, calculate the weighting factor per sensor per timestep as $\frac{1}{\text{relative error}} * \frac{\text{total number of relationships in time-step}}{(m-1)}$ where $(m-1)$ is the maximum number of possible relationships between each sensor per sensor per timestep. The total number of sensors is m .

9. The weighting factor per sensor per timestep is used as the Huff Weight, denoted as $w_{i,t}^{\text{HUFF}}$ of sensor i and at time-step t.

This algorithm can be used regardless of statistical rigor on the sample of the data in consideration, so it is appropriate to use this either when the platform is stationary or in motion, which is not the case for Huber Weight Functions or Deleted Residuals. The subsequent quadratic optimization cost and constraints for use with Huff Weights is in Equation 3.13 with $\forall i, j = \{1, \ldots, m\}, i < j$.

$$\begin{aligned} \textbf{min } Z &= \sum_{i=1}^{m}\sum_{t=1}^{n} w_{i,t}^{\text{HUFF}} * \left((x_{i,t} - \widehat{x_i})^2 + (y_{i,t} - \widehat{y_i})^2\right) \\ s.t & \\ & (\widehat{x_i} - \widehat{x_j})^2 + (\widehat{y_i} - \widehat{y_j})^2 - (a_{ij} + \epsilon)^2 \leq 0 \\ & (a_{ij} - \epsilon)^2 - (\widehat{x_i} - \widehat{x_j})^2 - (\widehat{y_i} - \widehat{y_j})^2 \leq 0 \end{aligned} \tag{3.13}$$

.

Both Huber and Huff weight functions can be multiplied to produce another set of weights and the application of which is shown in Equation 3.14 with $\forall i, j = \{1, \ldots, m\}, i < j$.

$$\begin{aligned} \textbf{min } Z &= \sum_{i=1}^{m}\sum_{t=1}^{n} w_{i,t}^{\text{HUFF}} * \left(w_{i,t_x}^{\text{HUBER}} * (x_{i,t} - \widehat{x_i})^2 + w_{i,t_y}^{\text{HUBER}} * (y_{i,t} - \widehat{y_i})^2\right) \\ s.t & \\ & (\widehat{x_i} - \widehat{x_j})^2 + (\widehat{y_i} - \widehat{y_j})^2 - (a_{ij} + \epsilon)^2 \leq 0 \\ & (a_{ij} - \epsilon)^2 - (\widehat{x_i} - \widehat{x_j})^2 - (\widehat{y_i} - \widehat{y_j})^2 \leq 0 \end{aligned} \tag{3.14}$$

All of these methods of outlier analysis and reduction are compared with one another and additionally compared with random weight functions to show their effectiveness in improving the calculation of the estimates of location and orientation shown in the results of Chapter 4.

3.4 Sliding Windows and Correction for Lag

As the data from the GPS sensors is subject to variation depending on a number of very unpredictable factors, such as atmospheric, satellite constellation change, sensor characteristics etc., it is very likely that during a particular time step, some of the GPS sensors may not report the coordinate values. This leads to missing data. Missing data may be handled in a variety of methods and some common methods used are imputation, interpolation and others discussed in Section 2.3. However, these methods may not be applicable to the application environment of GPS sensors attached to a mobile platform because these methods assume that the data/output produced follows a distribution that can be ascertained.

Furthermore, the coordinate information from GPS sensors on a mobile platform may be limited by real-time processing time vs. utility constraints. When the estimate is being determined while the platform is in motion, the estimates must be generated in such a manner that it may be used by the kinematic system of the platform in motion. Therefore, the estimate of location and orientation cannot be generated all the way at the end of the platforms movement. Additionally, most modern GPS sensors are capable of sending more than one data reading per second, thus capable of providing a second by second accurate estimate, which is acceptable for a mobile robot platform performing some real-time processing. Moreover, according to the preliminary results we have collected, observed location and orientation vary widely from one time-step to the next. We expect that the use of sliding windows will reduce that variation and enact a smoothing effect on the output.

3.4.1 Sliding Window

A sliding window is a first-in-first-out queue of data of a size lower than the total size of the data and is refreshed with new information in the following unit of

time by deleting the oldest member of the queue and adding the oldest member of the newly observed data to the queue of data in consideration, which creates a "sliding" effect. A sliding window may add new members based on time of new data or based on rows of new data. This only is an issue if every row of new data does not represent a singular addition in number of time-steps. In our application, we assume that the GPS sensors generate observed values in a way where each row of new data represents an increase of value 1 to the number of time-steps.

Using the sliding window mechanism, an estimate can be generated by every time-unit by using a large portion of the historical observed data and the most recent observed data. Initially, till the requisite number of rows in the sliding windows is reached (from data point 0 to data point at the size of the sliding window), a sliding window estimate may not be produced. However, once the requisite number of rows (i.e. sliding window size) is reached, an estimate can be generated every singular time-step. As the system receives a new observed data point, the oldest data point is discarded and the newest data point is added. More concretely, for a sliding window of size s, sliding window estimates cannot begin being generated till s^{th} time-step, assuming that each new data point is generated every time-step. The sliding window data queue consists of all observed data $w = \{t - s + 1, \ldots, t\}$, where $t = \{s, \ldots, n\}$. On receiving the first s set of new observed data at time-step t_s, the sliding window shifts such that the queue consists of all observed data $w_1 = \{(t_s) - s + 1, \ldots, t_s\}$. The next sliding window data set if the time-step increment in units of 1 is defined by $w_2 = \{(t_s + 1) - s + 1, \ldots, t_s + 1\}$.

With the use of sliding windows, we can generate estimates in a real time fashion with more weight on the "older" data and less weight on the new and recent data. Specifically, for a window of size s, the "older" data consists of all data points $\{t-s+1, \ldots, t-1\}$ and the s^{th} element of data is occupied by the newly observed data.

All of these described characteristics of sliding windows will allow for the generated estimates to experience less variance from one estimate to the next, thus creating a smoother path of location and orientation estimates. Additionally, with a sliding window size $s > 1$, the effect of missing data can be decreased as the chances of a sensor missing all the data points in the the window are less. Therefore, missing data in the quadratic optimization equations in Equation 3.9 do not have to be adjusted for. Since, the cost of Equation 3.9 is additive for each present observed data, we do not face negative consequences of missing data in the stationary case.

The reasons for using sliding windows was to establish a method of iteratively obtaining up-to-date estimates of location and orientation of the mobile platform during its operation. It is not feasible to wait to calculate an estimate for location at a particular point (for example : every 1 min) in time by halting all motion to collect data over a much longer period of time (for example : after 3 hours) and then calculate the estimate. This is primarily the case due the possible application of the mobile robot platform in different hostile environments where there may be real-time constraints on the utility of data received. The selection of the size of the sliding window is however, specific to a particular application and may be determined empirically. Please refer to Section 3.1.2 to obtain the quadratic programming formulation to obtain location estimates of the sensors.

3.4.2 Correction For Lag

For sliding windows of size 1, essentially, there is no sliding window because to generate the estimate only the most recent data is being used and then discarded when the new observed data is input in the system. As the size of the sliding window increases, we face the issue of the estimates generated as lagging behind the actual location and orientation values. The results are in the preliminary stages and therefore

will be discussed in Chapter 5.2. To correct for this lagging, we calculate an estimate for the orientation and correct the estimate to location at the end of the sliding window as the corrected estimate. Orientation is calculated as discussed in Section 3.2. The correction for lag is obtained using the following steps :

1. Calculate the estimate generated for the particular sliding window.
2. Calculate the distance between the first complete set of observed data for a particular sensor in that window and the estimate. This generates the time position in that sliding window where the estimate occurs.
3. To calculate the time remaining by which the estimate has to be corrected, subtract the time position in Step 2 from the size of the sliding window.
4. Multiple the time remaining by the velocity to obtain the distance, d, the estimate should be corrected.
5. Using orientation value θ, the x-component and the y-component is obtained through $(d * cos(\theta))$ and $(d * \sin(\theta))$ respectively.
6. Add the x-components and y-components to the estimate to shift the estimate to correct for the lag.

Our results in terms of sliding window sizes shows that the issue of lag only becomes apparent in very high sliding window sizes, such as 100 and higher but is not very apparent for low sizes, such as $\{1, \ldots, 10\}$. The technique of lag correction has been implemented for different window sizes to show the benefits in terms of performance in Section 4.12.

3.4.2.1 Dynamic Sliding Window Size

Depending on the sensor configuration that the path that the mobile platform follows, there may be long sections of the path where the platform is moving steadily without large changes in orientation. Consequently, there are sections of the path that

are short and involve changes in orientation, such as curves and turns. It follows that when the platform is on a long steady path, the more historical information retained, the better the stability of the estimate of the locations of the sensors will be. To retain more historical information, larger sliding window sizes are appropriate. However, when the platform is negotiating curves and turns, retain historical information is not as vital as the orientation is changing from one time-step to the next. Therefore, a smaller sliding window size is recommended. To implement this, it is necessary to know that the platform is changing orientation or turning. This information can be obtained from the control information of the mobile platform. Moreover, this method of adapting window sizes according to the type of path that is being followed can be combined with the calculation for correction of lag for superior results as discussed in Section 4.12.

CHAPTER 4

RESULTS

4.1 Experimental Set-Up

To obtain a working prototype of the methodologies described in the previous chapters and to show working proof of our techniques, multiple experiments were conducted both in the stationary case and the case where the platform is in motion. Initially, eight GPS sensors reporting locations in latitudes and longitudes were installed in a known and static configuration on the mobile platform. Out of the eight sensors , only seven sensors actually reported values and there was one power failure during the initial recording of the data. All data collected before the power failure were discarded for the purposes of these experiments. The latitudes and longitudes were converted to the Universal Transverse Mercator (UTM) coordinate system.

4.1.1 Stationary Experimental Set-Up

To test the validity of our framework and to identify any potential problems with our approach, the optimization framework and subsequent outlier analysis, missing data analysis and weighting was performed on a stationary platform of eight GPS sensors that were arranged in an outdoor field. All sensor data collection was recorded onto a single laptop for approximately 4000 seconds. The laptop and the sensors were rested in a stationary position outdoors and the subsequent values were obtained. The locations of the arrangement of the sensors is shown in Figure 4.1.

There were a multitude of systemic and data errors during the collection of the data. Sensor 4 did not produce any results and the other sensors were deprived of

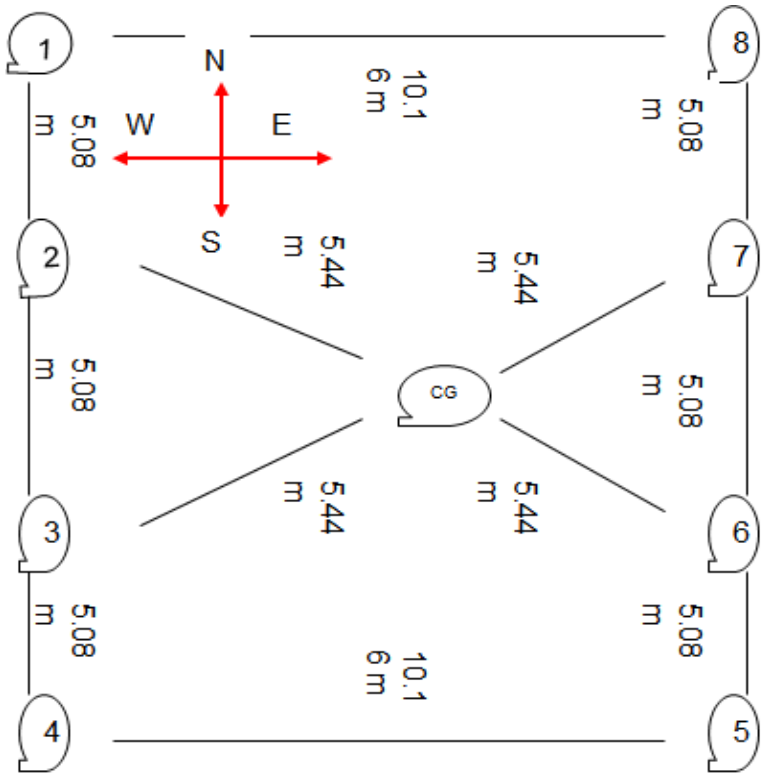


Figure 4.1: Location of Eight Sensors (CG = Center of Gravity) UTM Coordinates.

power for a short while, so there is a drop in the responses. For the purposes of this experiment, in the following optimization framework experiments, we have discarded the data before the power outage, using only the data after the power outage, and we do not consider Sensor 4.

4.1.2 Movement Experimental Set-Up

For the purposes of obtaining observed data from a mobile platform with eight GPS sensors attached from a simulation, a SimuLink® application developed in the Mechanical and Aerospace Engineering Department at the University of Texas at Arlington by Hakki Seval was used. The application simulates the actual movement of the described vehicles taking into account physical and kinematic effects. As input, the errors from the average location of each sensor in the stationary case are used to

create errors in the GPS sensor observed data, similar to the stationary case. The simulation also provides the "perfect" case where the mobile platform follows the path without any errors. This provides a mechanism of calculating the effectiveness of our methodologies for calculating the location and orientation estimate along with effects of outlier analysis and sliding windows.

4.1.3 Quadratic Optimization Set-Up

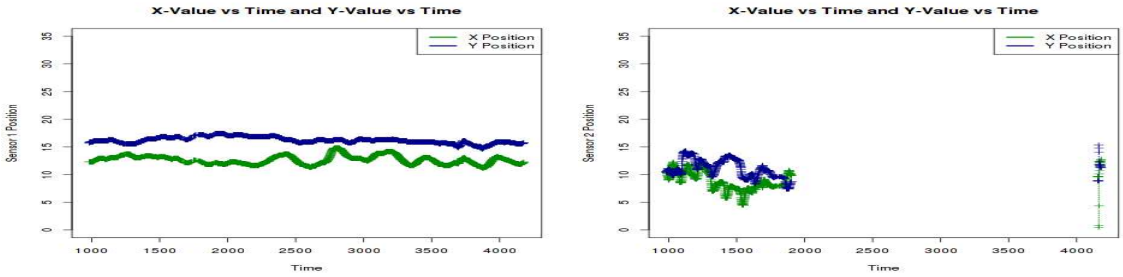
The quadratic optimization methods were programmed and run in MatLab® software tools using the Active Set Algorithm [3] with maximum number of functions as 20000 and maximum number of iterations as 50000. The estimates for location and orientation were calculated on an Intel® Core™ i5-3320M CPU @ 2.60GHz processor Latitude E6430 with 4.00 GB RAM using Windows 7 64-bit operating system. For calculating 3900 location estimates, the system takes 316.896 seconds with 42 non-linear inequality constraints to describe the distances between seven sensors, which implies that to generate one location estimate takes approximately 0.081255 s.

Initially, to ascertain the distributions and the behaviors of the sensors, statistical plots were used, such as the UTM x-values of the coordinates vs. time, y-values of the coordinates vs. time and normal probability plots followed by time-series analysis plot,s such as auto-correlation factor plots of the stationary data.

4.2 Graphical Analysis

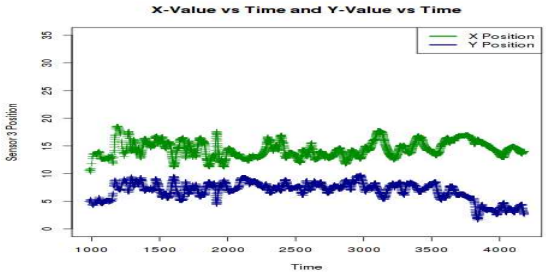
Figures 4.2a, 4.2b, 4.2c, 4.2d, 4.2e, 4.2g show the trend in the x-values and y-values over time for Sensors 1, 2, 3, 5, 6, 7, 8. Figures 4.3a, 4.3b, 4.3c, 4.3d, 4.3e, 4.3f, through4.3g show the scatter plots for each of the sensors with y-values vs. x-values. We can conclude that although, ideally, the sensor graphs over time should show a straight line, and the scatter plots should show a tight circle of data points,

such is not the case. Although all of these sensors were of the same make and model, they all show different trends of noise and variances. These results are also valid for the distances between each of sensors for each time-step as seen in Figure 4.4 and 4.5. Additionally, the NPP plots also show that non-normality is an issue as seen for Sensor 1 in Figures 4.7a and 4.8a, Sensor 2 in Figures 4.7b and 4.8b, Sensor 3 in Figures 4.7c and 4.8c, Sensor 5 in Figures 4.7d and 4.8d, Sensor 6 in Figures 4.7e and 4.8e, Sensor 7 in Figures 4.7f and 4.8f and Sensor 8 in Figures 4.7g and 4.8g.

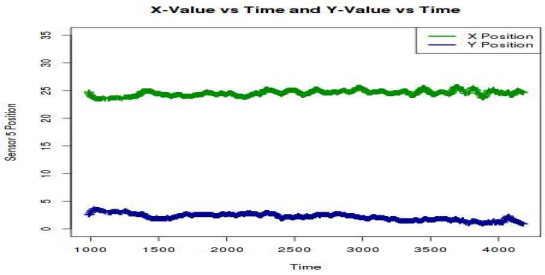


(a) Sensor 1.

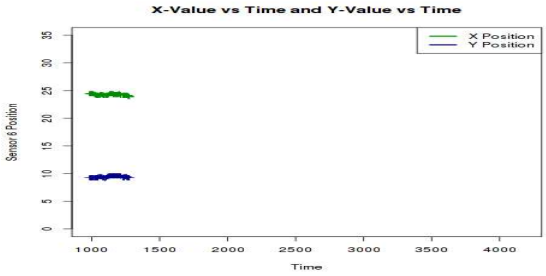
(b) Sensor 2.



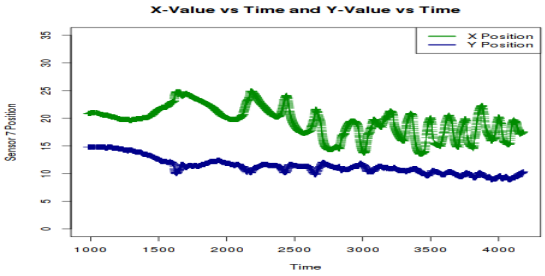
(c) Sensor 3.



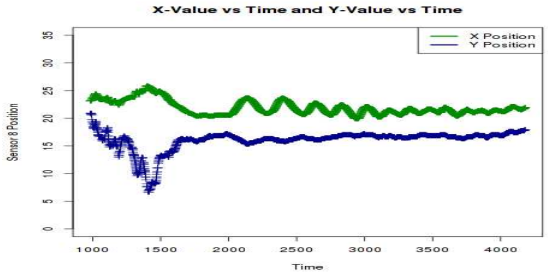
(d) Sensor 5.



(e) Sensor 6.

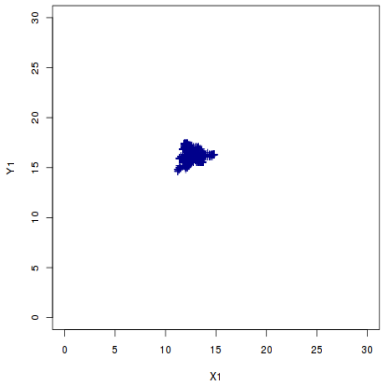


(f) Sensor 7.

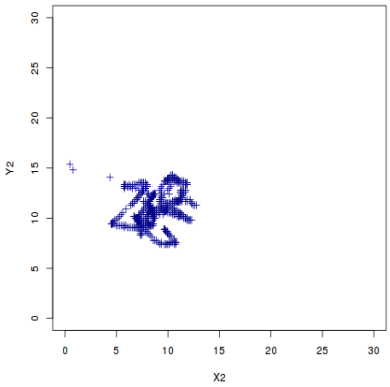


(g) Sensor 8.

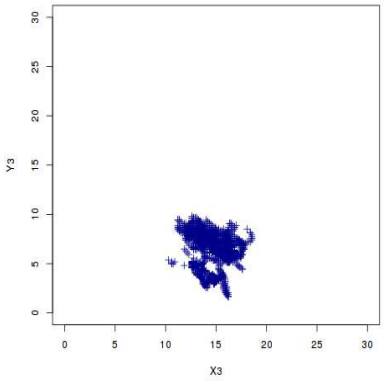
Figure 4.2: Plots of UTM X (Green) and Y (Blue) Positions of Sensors vs. Time.



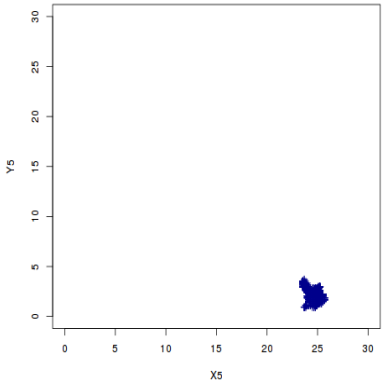
(a) Sensor 1 y_1 vs x_1.



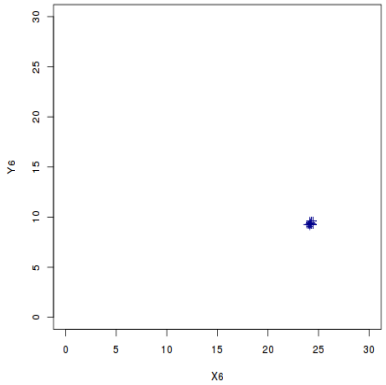
(b) Sensor 2 y_2 vs x_2.



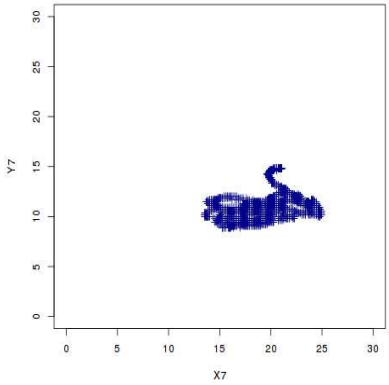
(c) Sensor 3 y_3 vs x_3.



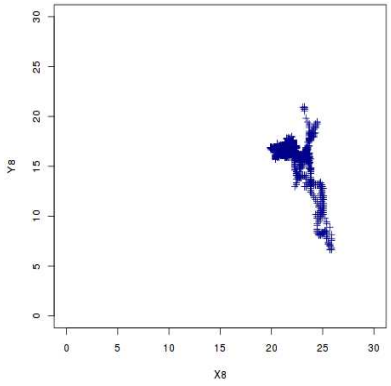
(d) Sensor 5 y_5 vs x_5.



(e) Sensor 6 y_6 vs x_6.

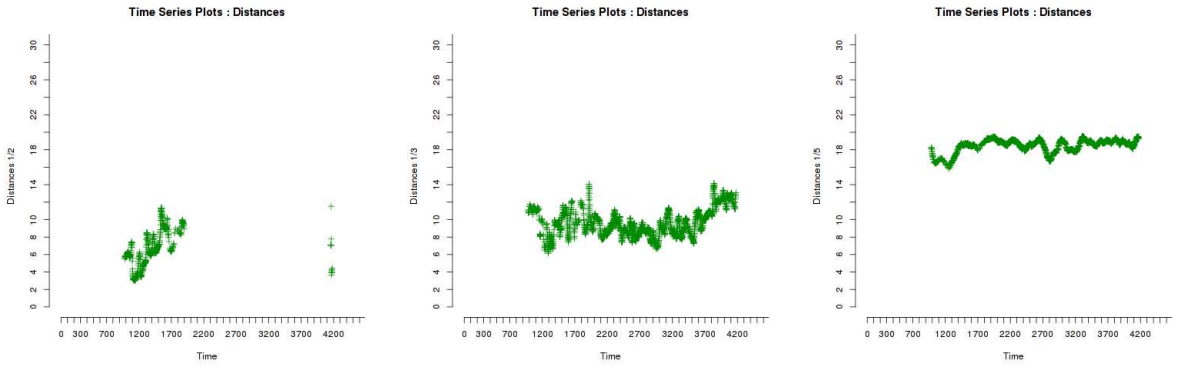


(f) Sensor 7 y_7 vs x_7.



(g) Sensor 8 y_8 vs x_8.

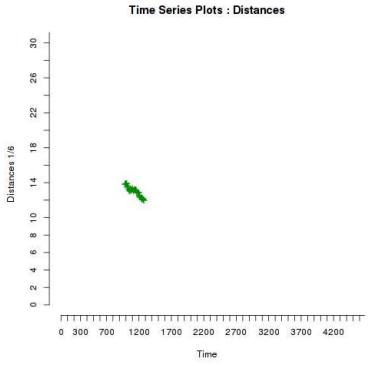
Figure 4.3: Scatter Plots of Sensors UTM Y values vs. X values.



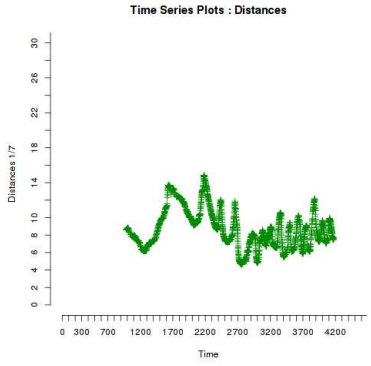
(a) Sensor 1 and Sensor 2.

(b) Sensor 1 and Sensor 3.

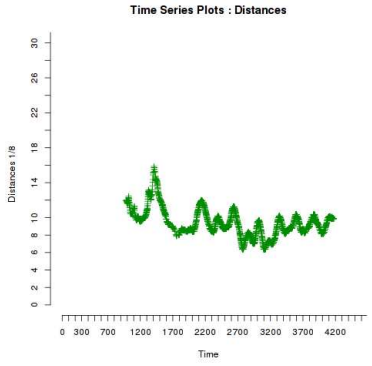
(c) Sensor 1 and Sensor 5.



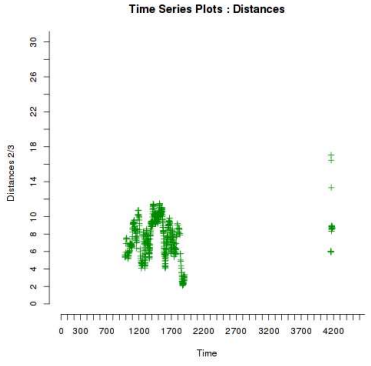
(d) Sensor 1 and Sensor 6.



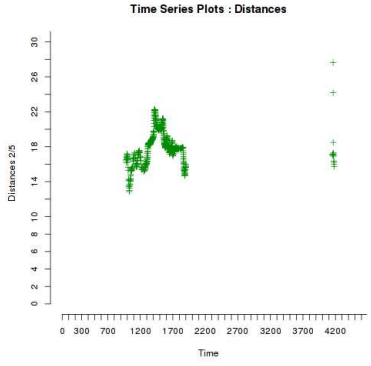
(e) Sensor 1 and Sensor 7.



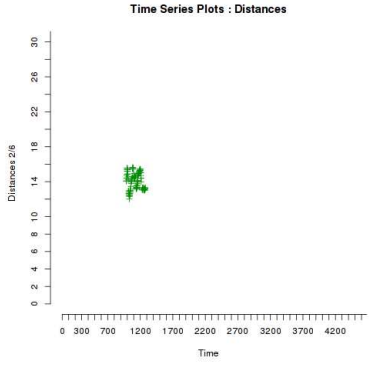
(f) Sensor 1 and Sensor 8.



(g) Sensor 2 and Sensor 3.

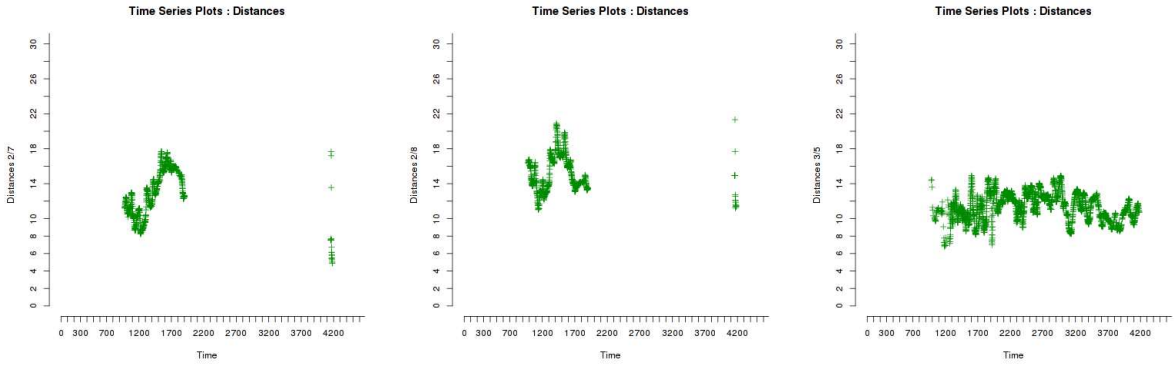


(h) Sensor 2 and Sensor 5.

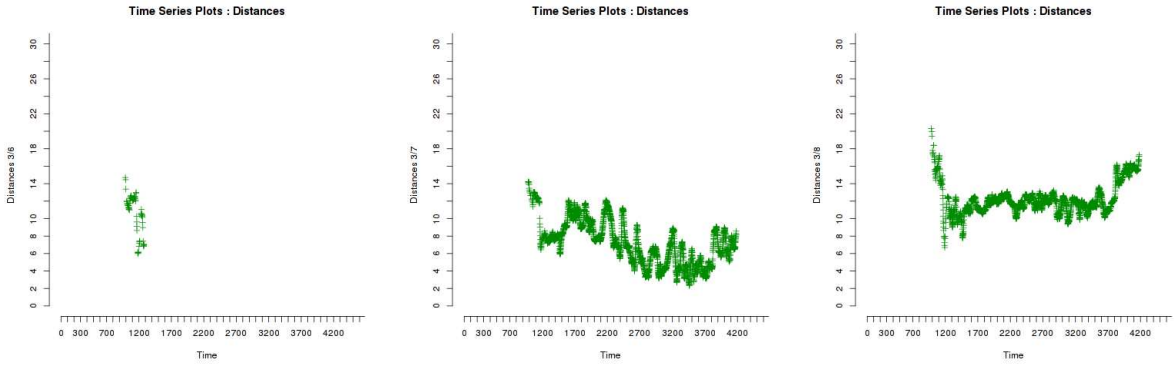


(i) Sensor 2 and Sensor 6 .

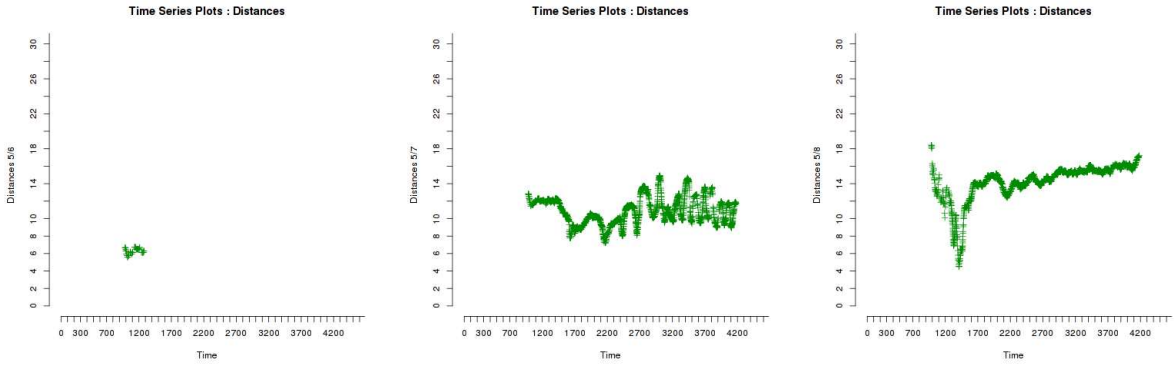
Figure 4.4: Plots of Distances between Two Sensors vs. Time.



(a) Sensor 2 and Sensor 7. (b) Sensor 2 and Sensor 8. (c) Sensor 3 and Sensor 5.

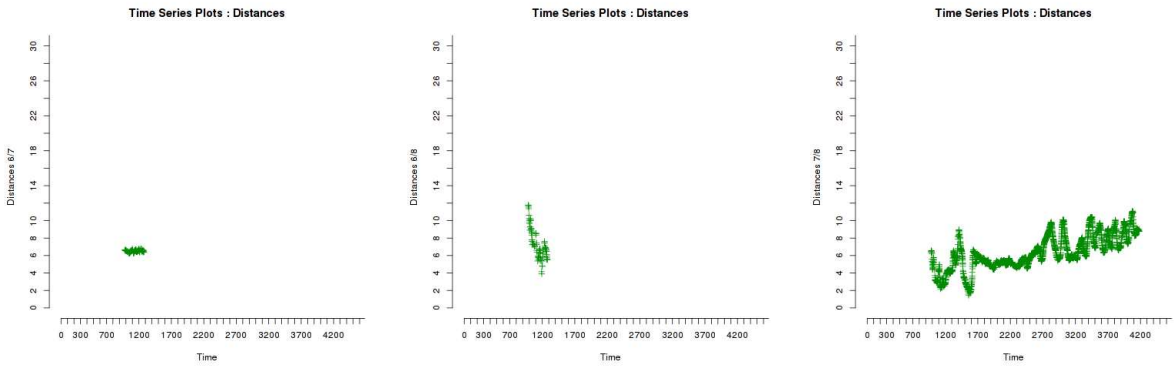


(d) Sensor 3 and Sensor 6. (e) Sensor 3 and Sensor 7. (f) Sensor 3 and Sensor 8 .



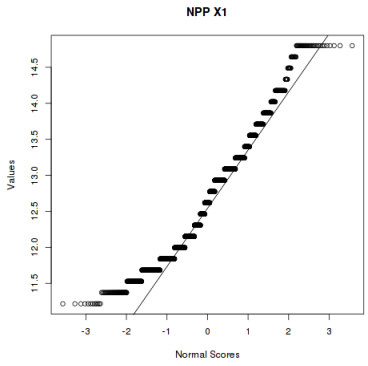
(g) Sensor 5 and Sensor 6. (h) Sensor 5 and Sensor 7. (i) Sensor 5 and Sensor 8.

Figure 4.5: Plots of Distances between Two Sensors vs. Time (contd.).

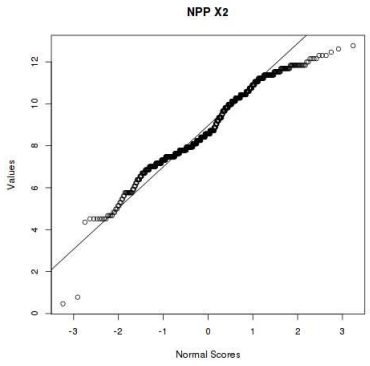


(a) Sensor 6 and Sensor 7. (b) Sensor 6 and Sensor 8. (c) Sensor 7 and Sensor 8.

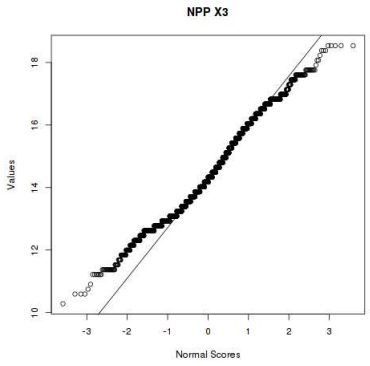
Figure 4.6: Plots of Distances between Two Sensors vs. Time (contd.).



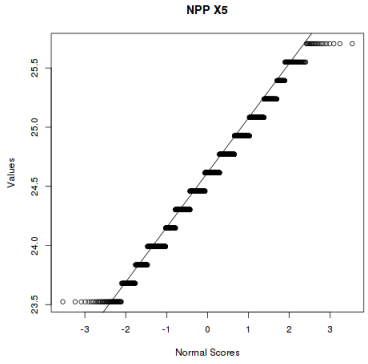
(a) Sensor 1.



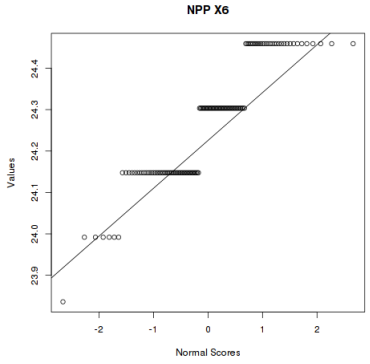
(b) Sensor 2.



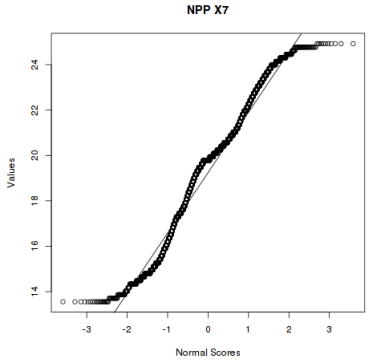
(c) Sensor 3.



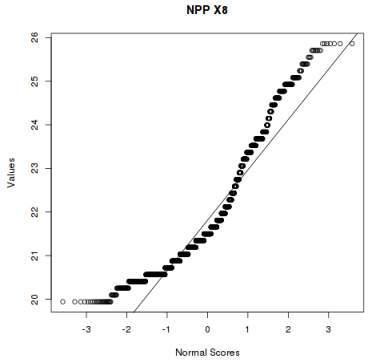
(d) Sensor 5.



(e) Sensor 6.

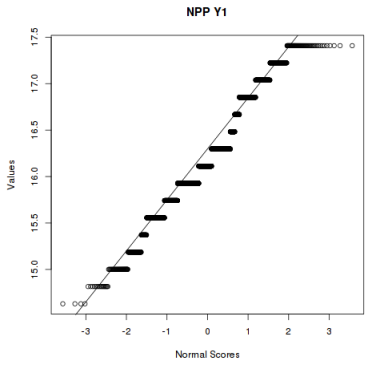


(f) Sensor 7.

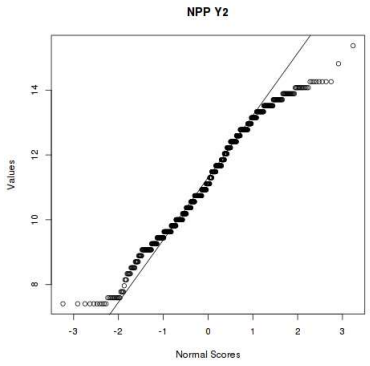


(g) Sensor 8.

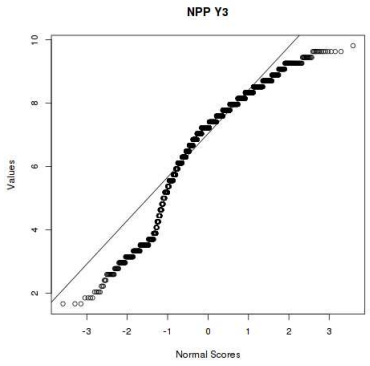
Figure 4.7: Normal Probability Plots UTM X Coordinates for All Sensors.



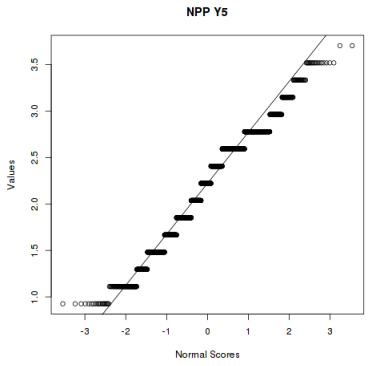
(a) Sensor 1.



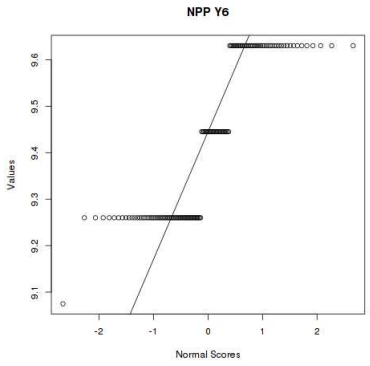
(b) Sensor 2.



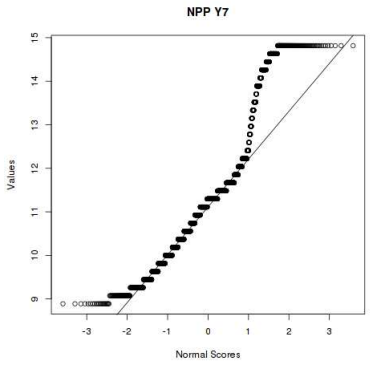
(c) Sensor 3.



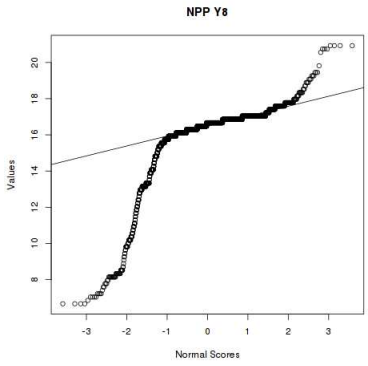
(d) Sensor 5.



(e) Sensor 6.



(f) Sensor 7.



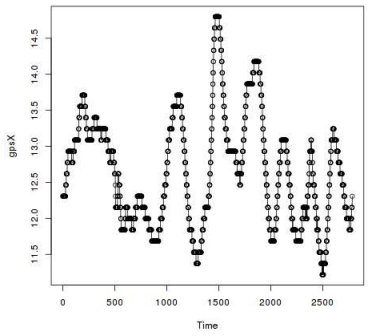
(g) Sensor 8.

Figure 4.8: Normal Probability Plots UTM Y Coordinates for All Sensors.

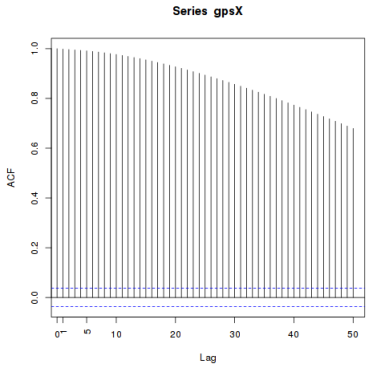
4.3 Time Series Analysis

Time series analysis was performed using the time series functionality in R, as the statistical plots in Section 4.2 was used to ascertain the level of autocorrelation for the observed location data. The aim of the time series analysis was to obtain a model of the behavior of the platform over time. The time series plot for Sensor 1 in Figures 4.9a and 4.9d show that although the coordinates of the sensors vary widely, they do possess a trend, and they do not have the sharp jagged-ness that is typically associated with time series data. Moreover, the time series plots show a distinct trend of the location values approaching a plateau followed by a distinct phase of transition.

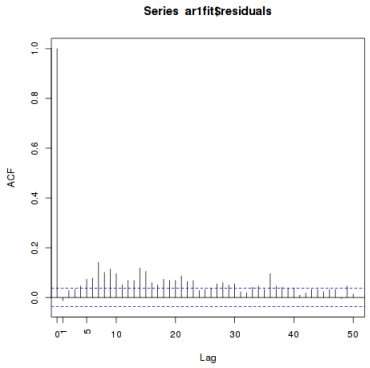
The sample autocorrelation function (ACF) plot in Figures 4.9b and 4.9e show that the autocorrelation function values exceed the bounds for statistical significance for all the sensors coordinate data. Additionally, the autocorrelation factors plot for the residuals in Figures 4.9c and 4.9f of the sensor coordinate data show that the factors mostly exceed the bounds for statistical significance, which implies that there is significant autocorrelation in the residuals. Similar results were obtained for the location coordinates of other sensors and distances between the sensors at each time-step. We can conclude that obtaining a time series model would not be suitable as this data does not comply with the basic requirements of time series data.



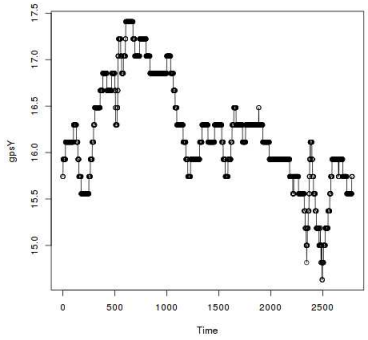
(a) Sensor 1 Time Series Plot of UTM X Values.



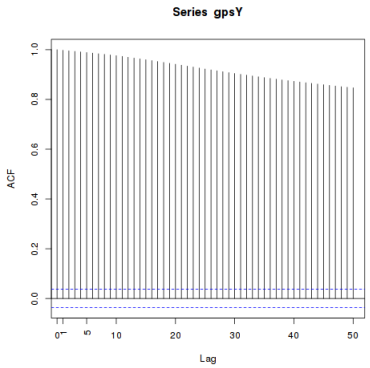
(b) Sensor 1 ACF Plot of UTM X Values.



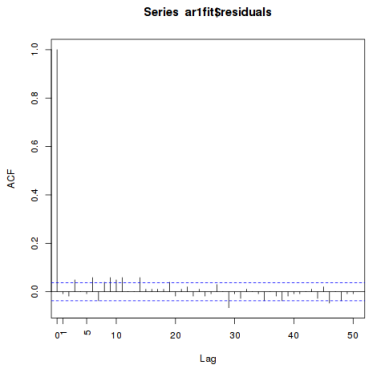
(c) Sensor 1 ACF Residuals Plot of UTM X Values.



(d) Sensor 1 Time Series Plot of UTM Y Values.

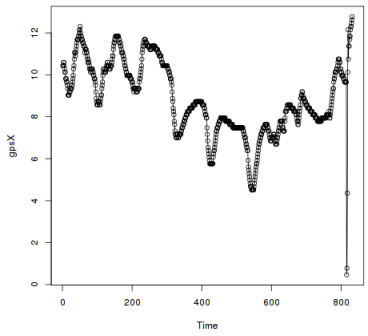


(e) Sensor 1 ACF Plot of UTM Y Values.

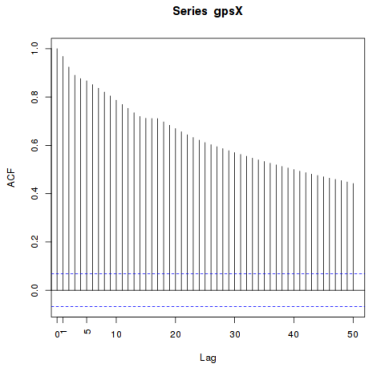


(f) Sensor 1 ACF Residuals Plot of UTM Y Values.

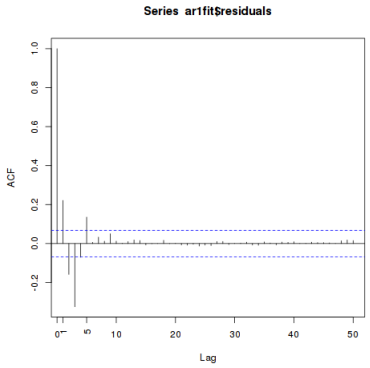
Figure 4.9: Time Series Analysis for Sensor 1.



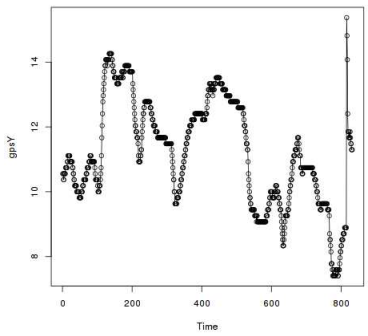
(a) Sensor 2 Time Series Plot of UTM X Values.



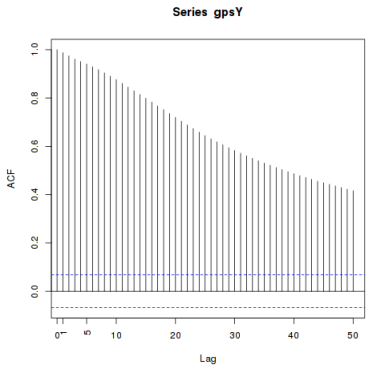
(b) Sensor 2 ACF Plot of UTM X Values.



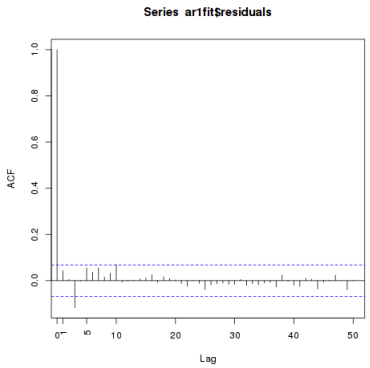
(c) Sensor 2 ACF Residuals Plot of UTM X Values.



(d) Sensor 2 Time Series Plot of UTM Y Values.

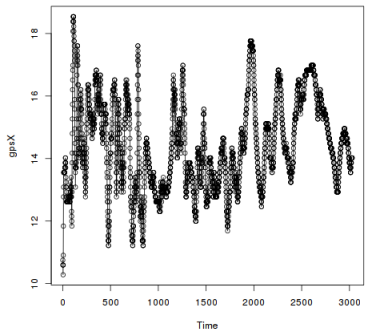


(e) Sensor 2 ACF Plot of UTM Y Values.

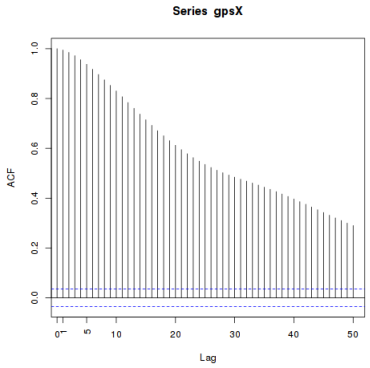


(f) Sensor 2 ACF Residuals Plot of UTM Y Values.

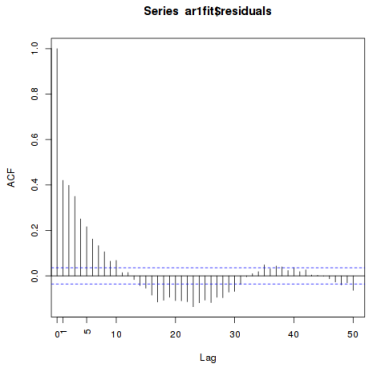
Figure 4.10: Time Series Analysis for Sensor 2.



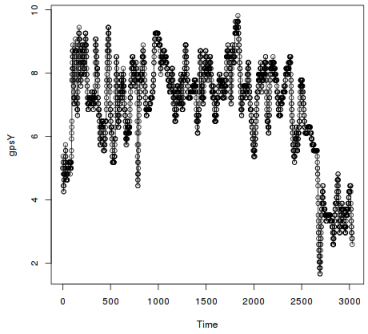
(a) Sensor 3 Time Series Plot of UTM X Values.



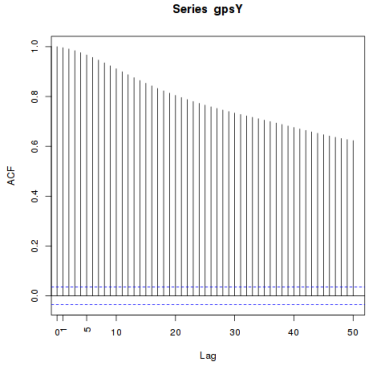
(b) Sensor 3 ACF Plot of UTM X Values.



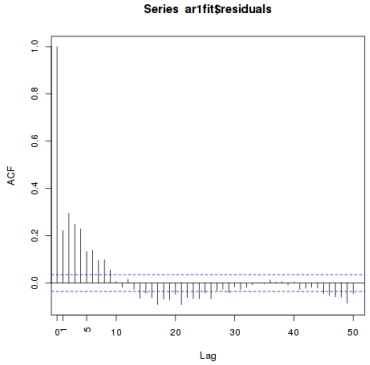
(c) Sensor 3 ACF Residuals Plot of UTM X Values.



(d) Sensor 3 Time Series Plot of UTM Y Values.

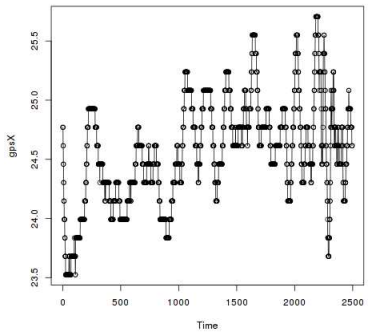


(e) Sensor 3 ACF Plot of UTM Y Values.

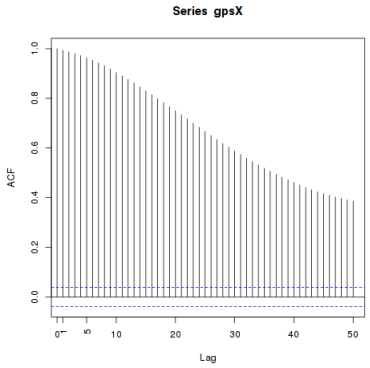


(f) Sensor 3 ACF Residuals Plot of UTM Y Values.

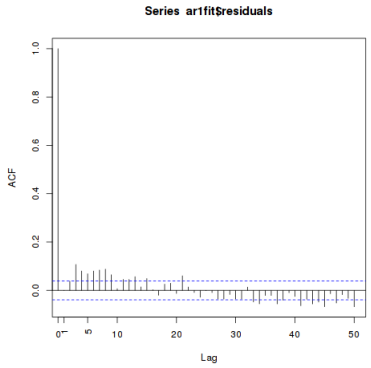
Figure 4.11: Time Series Analysis for Sensor 3.



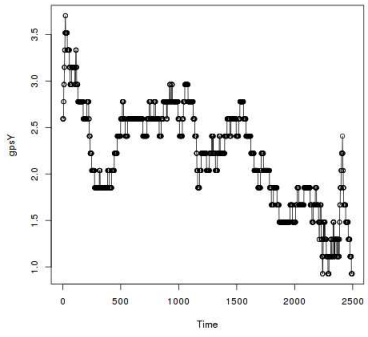
(a) Sensor 5 Time Series Plot of UTM X Values.



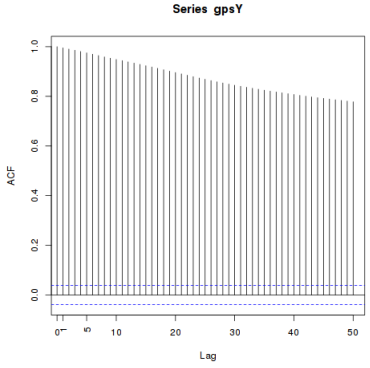
(b) Sensor 5 ACF Plot of UTM X Values.



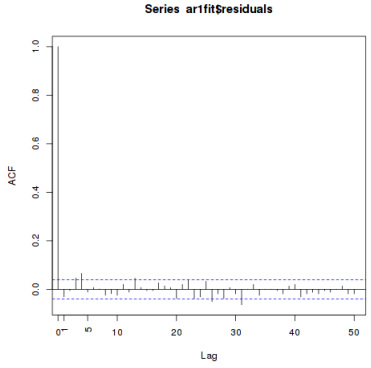
(c) Sensor 5 ACF Residuals Plot of UTM X Values.



(d) Sensor 5 Time Series Plot of UTM Y Values.

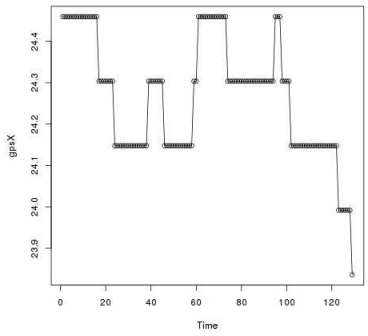


(e) Sensor 5 ACF Plot of UTM Y Values.

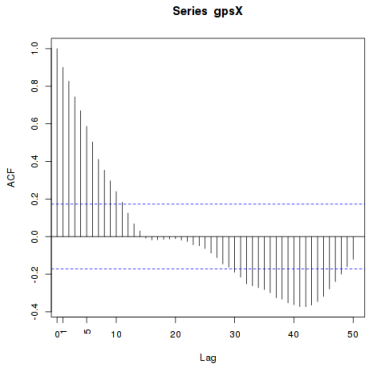


(f) Sensor 5 ACF Residuals Plot of UTM Y Values.

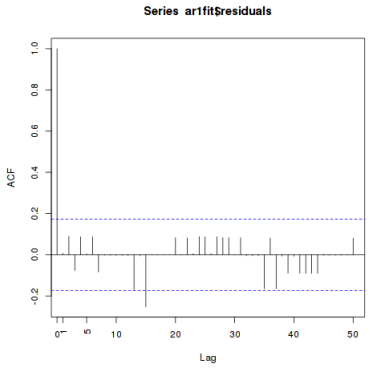
Figure 4.12: Time Series Analysis for Sensor 5.



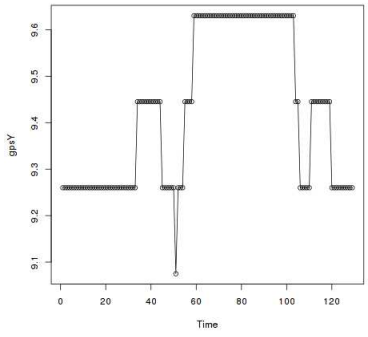
(a) Sensor 6 Time Series Plot of UTM X Values.



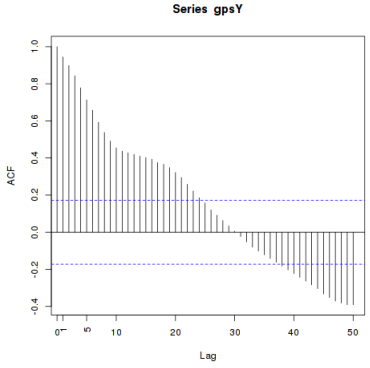
(b) Sensor 6 ACF Plot of UTM X Values.



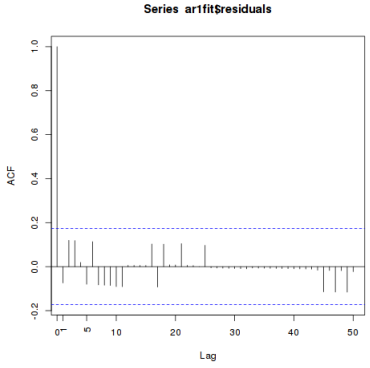
(c) Sensor 6 ACF Residuals Plot of UTM X Values.



(d) Sensor 6 Time Series Plot of UTM Y Values.

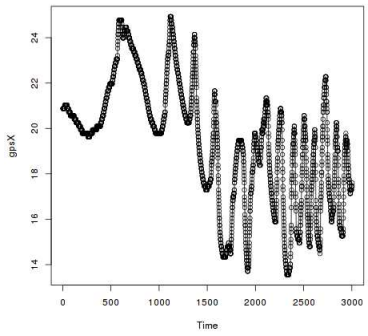


(e) Sensor 6 ACF Plot of UTM Y Values.

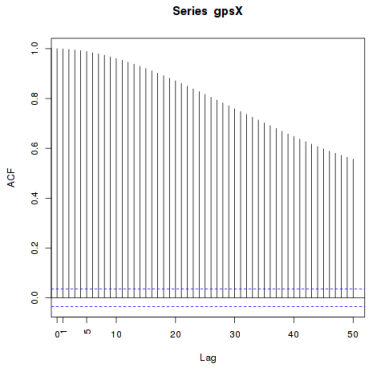


(f) Sensor 6 ACF Residuals Plot of UTM Y Values.

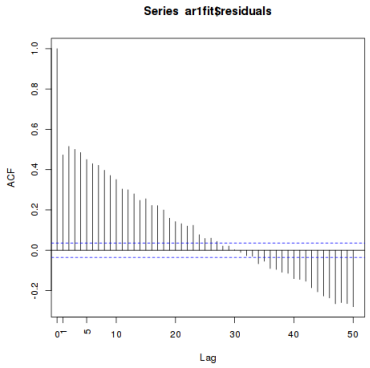
Figure 4.13: Time Series Analysis for Sensor 6.



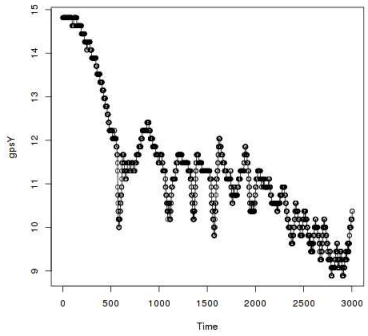
(a) Sensor 7 Time Series Plot of UTM X Values.



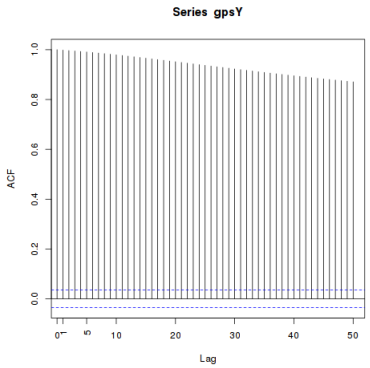
(b) Sensor 7 ACF Plot of UTM X Values.



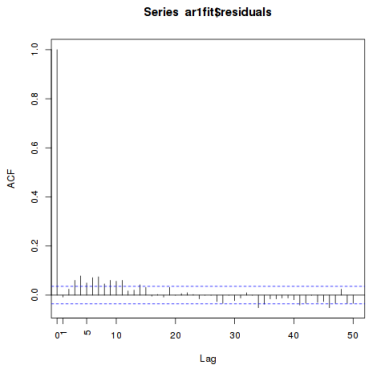
(c) Sensor 7 ACF Residuals Plot of UTM X Values.



(d) Sensor 7 Time Series Plot of UTM Y Values.

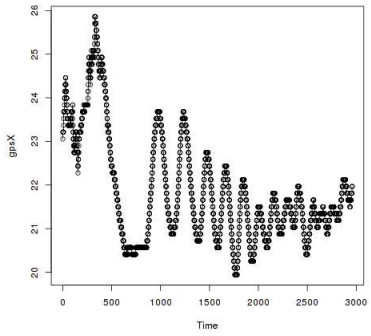


(e) Sensor 7 ACF Plot of UTM Y Values.

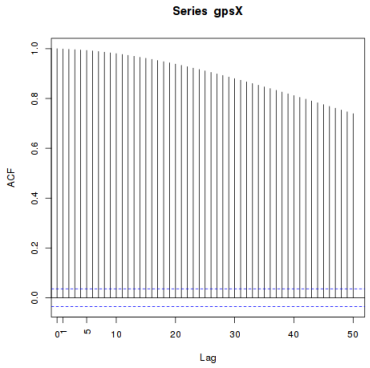


(f) Sensor 7 ACF Residuals Plot of UTM Y Values.

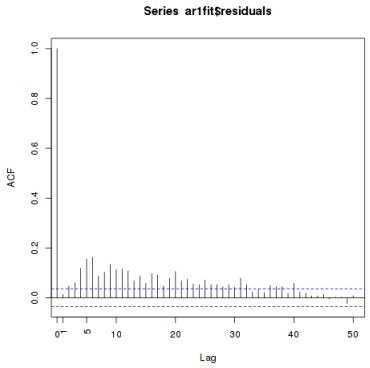
Figure 4.14: Time Series Analysis for Sensor 7.



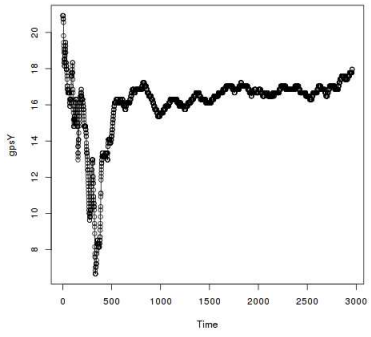
(a) Sensor 8 Time Series Plot of UTM X Values



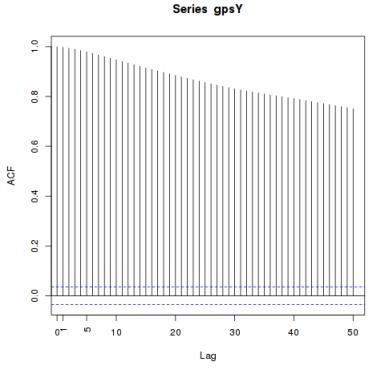
(b) Sensor 8 ACF Plot of UTM X Values.



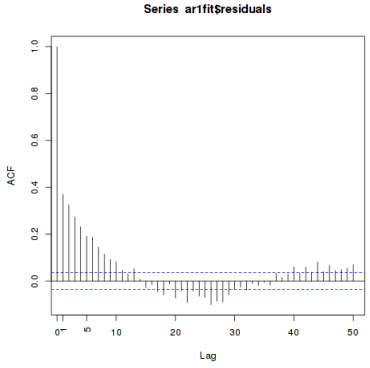
(c) Sensor 8 ACF Residuals Plot of UTM X Values.



(d) Sensor 8 Time Series Plot of UTM Y Values.



(e) Sensor 8 ACF Plot of UTM Y Values.



(f) Sensor 8 ACF Residuals Plot of UTM Y Values.

Figure 4.15: Time Series Analysis for Sensor 8.

4.4 Obtaining An Estimate Using Quadratic Optimization and Known Sensor Configuration

The observed values that were used to obtain estimates are shown in Figure 4.16. Estimates for location and orientation were produced for when the platform is stationary and when the platform is in motion. Furthermore, when the platform is stationary, different outlier analysis schemes are compared. When the platform is in motion, the only outlier analysis used is the Huff weighting function, which is compared with random weighting and includes various combinations with sliding windows of different sizes.

4.5 Unweighted Quadratic Optimization Estimates of Locations of GPS Sensors for Stationary Platform

Using Equation 3.4 in the optimization framework to calculate estimates for 7 out of the 8 deployed sensors, we obtained the estimated locations of each of the sensors shown in Figure 4.17. The graph shows that the estimates of location form the configuration shown in Figure 4.1 and the estimates lie approximately within the area of the observed values of location of the sensors.

The results shown in Figure 4.17 are based an optimization framework that does not account for outliers and different weights on different sensors and sensor observed values. Therefore, this is the simplest use case of the optimization framework showing the worst case answers for the stationary location of the seven out of eight sensors. The removal of outliers and preferential weighting can only improve the outcome of the objective and constraints. The next set of graphs will illustrate the results of introducing outlier removal, weighting and handling of missing data and highlight the stability of the proposed optimization framework.

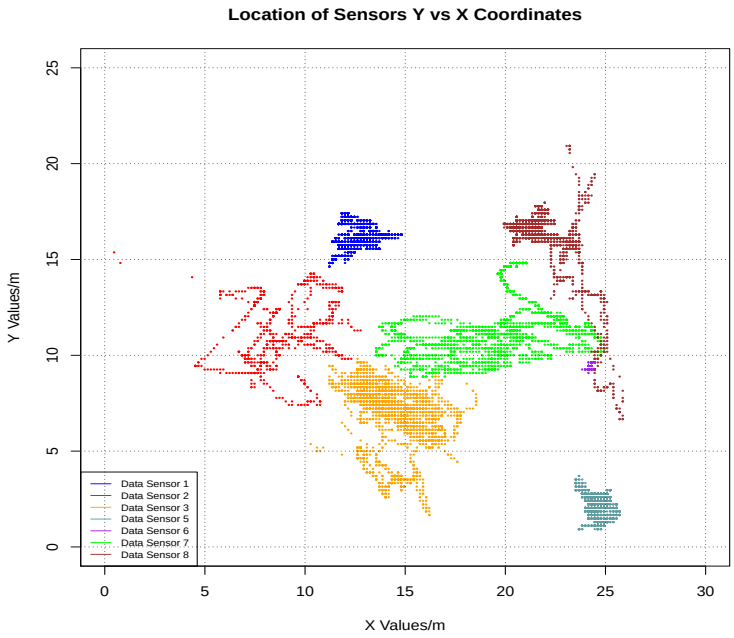


Figure 4.16: GPS Sensors Observed Locations UTM Y Coordinates vs. X Coordinates.

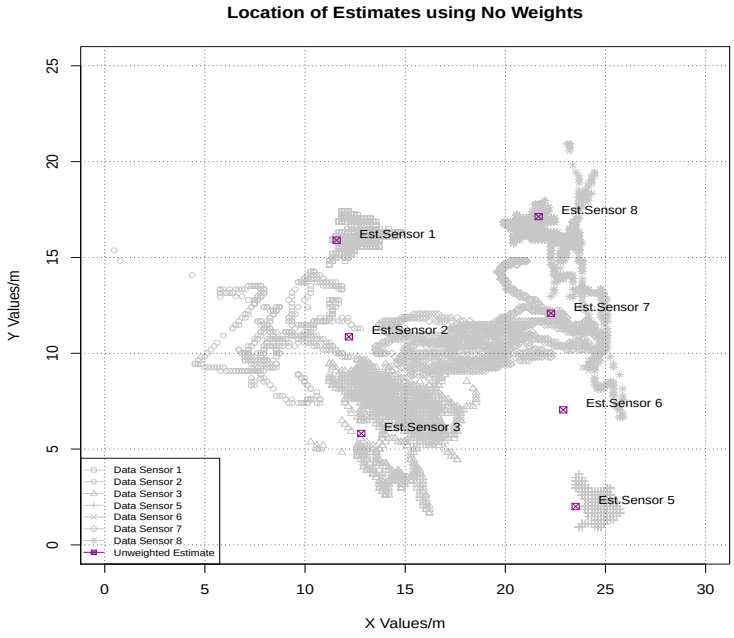


Figure 4.17: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Base Optimization.

4.6 Weighted Quadratic Optimization Estimates of Locations of GPS Sensors for Stationary Platforms

In the results shown in Figure 4.18, the quadratic optimization calculating location estimates use the Huber weighting function to perform outlier analysis using the Equation 3.12, while the results in Figure 4.19 show the location estimates on stationary data using the Huff weighting function described in Equation 3.13. To test the effect of the situation under which only 2 of the sensors report observed data, the Huber weight function was calculated for only Sensor 1 and Sensor 5, with the rest of the observed sensor data not being accounted for in the quadratic optimization, produces the location estimates in Figure 4.20, while a similar experiment with only Sensor 1 and Sensor 8 under consideration produces Figure 4.21 as the location estimates. From both these experiments, we can see the orientation of the mobile platform straightening itself slightly. This leads to the conclusion that sending more observed data to the quadratic optimization methodology may not be useful, if this data is erroneous. Figure 4.22 shows the effect of using a combined weight of Huff weight $\times$ Huber weight shown in Equation 3.14. Using the method described in Section 3.3.2 to calculate deleted residuals and apply quadratic optimization, we obtain the results in Figure 4.22.

Figure 4.24 shows a summary of all the different techniques of outlier analysis on stationary data to generate location estimates for the GPS sensors. From the summary figure, we can conclude that there are slight advantages to using the different weighting techniques, and they seem to produce more accurate estimates of location. We cannot judge objectively the efficacy of these methods on stationary data due to the fact that the true value of the location of the GPS sensors is not known, therefore, we do not have accurate metrics to compare. In the case of location estimates while the platform is in motion, the simulation described in Section 4.1.2 provides us with

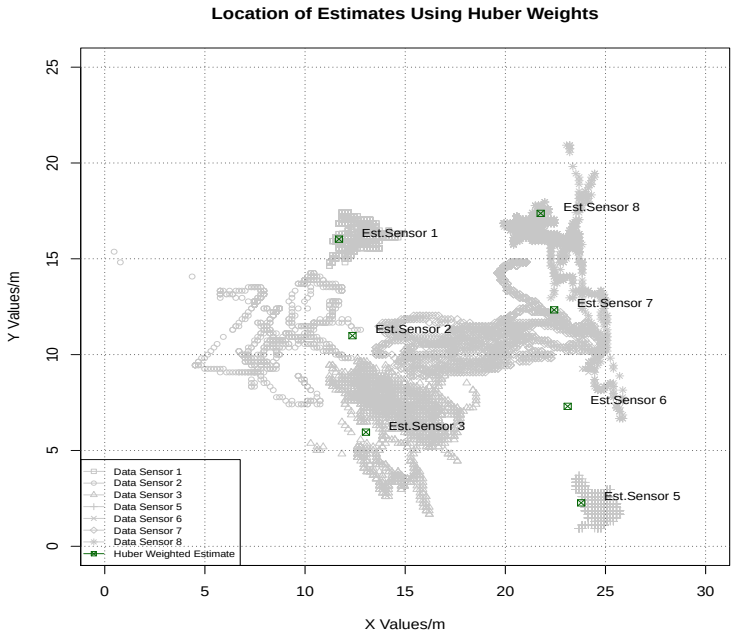


Figure 4.18: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Huber Weighted Optimization.

the true value of the location, thus allowing us to compare objectively the efficacy of the quadratic solutions, sliding windows and outlier analysis schemes.

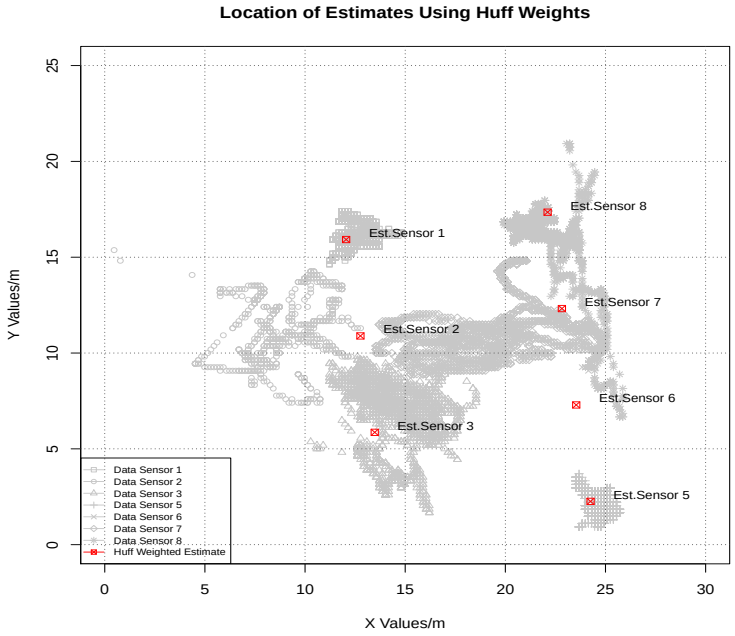


Figure 4.19: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Huff Weighted Optimization.

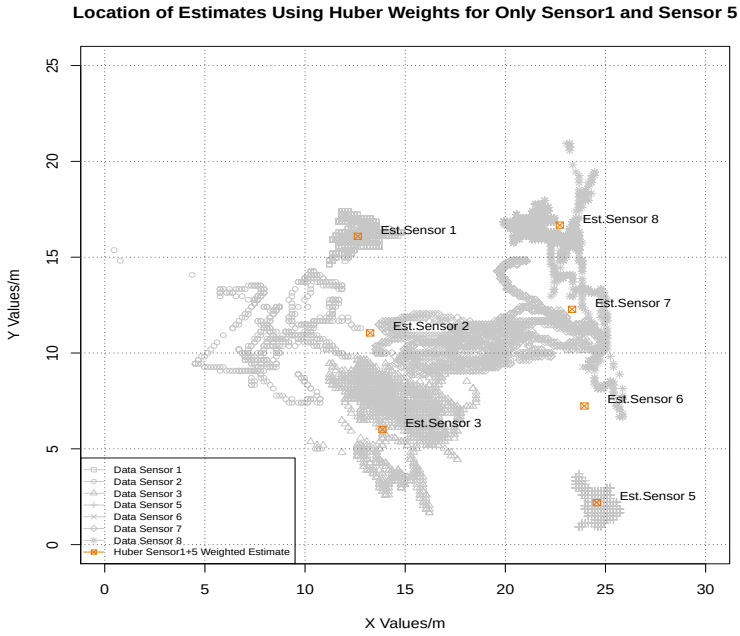


Figure 4.20: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Huber Weighted Optimization with Observed Data from Sensor 1 and Sensor 5 only.

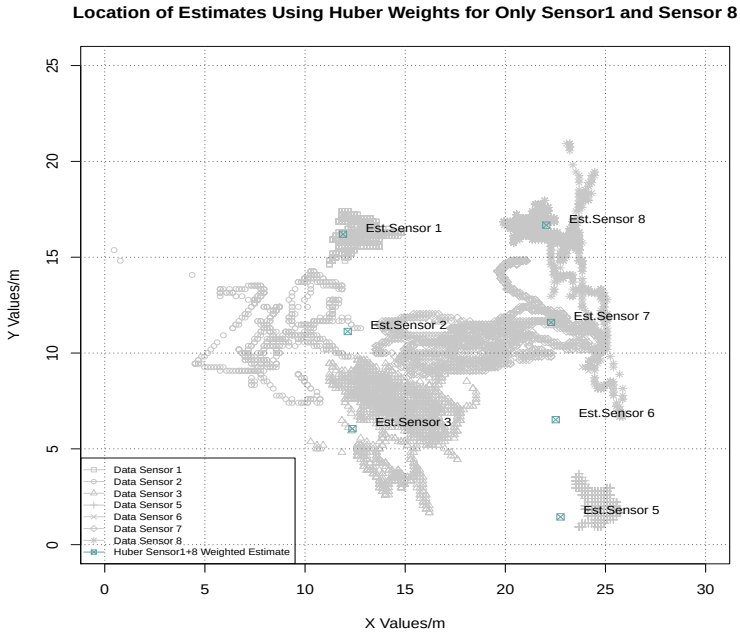


Figure 4.21: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Huber Weighted Optimization with Observed Data from Sensor 1 and Sensor 8 only.

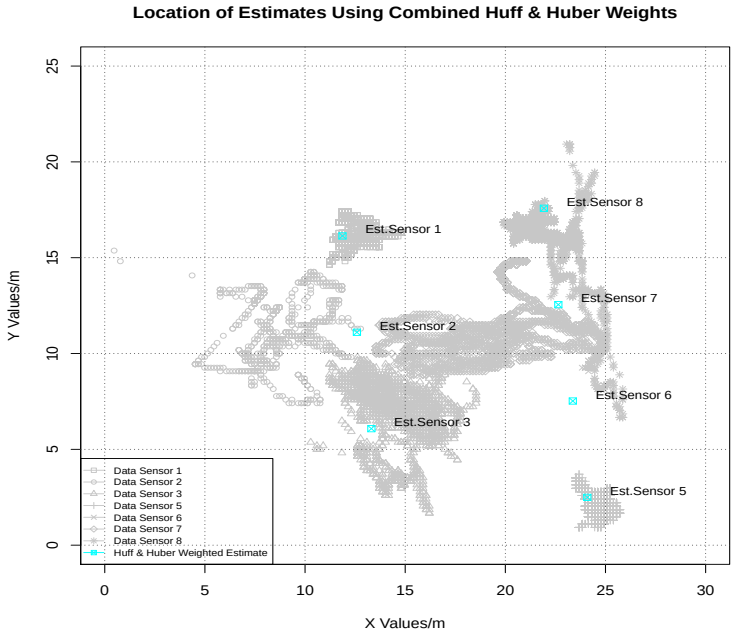


Figure 4.22: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Huff × Huber Weighted Optimization.

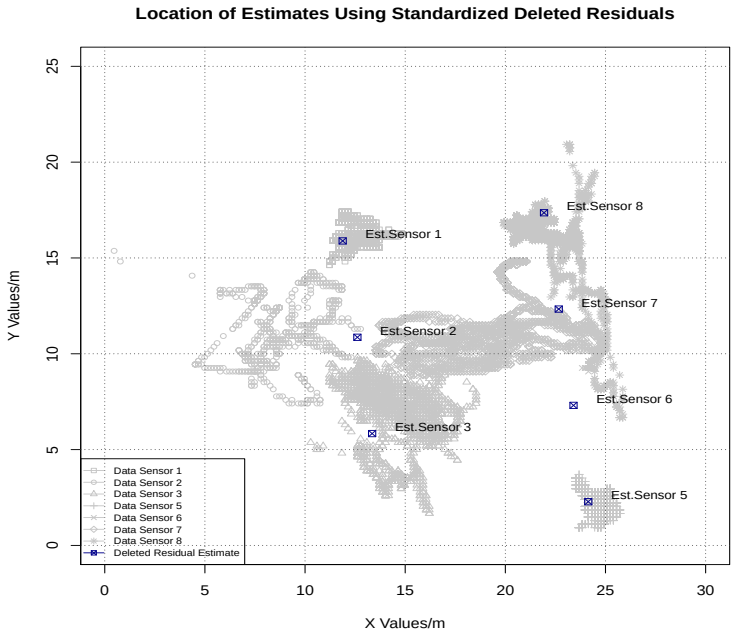


Figure 4.23: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Deleted Residuals Optimization.

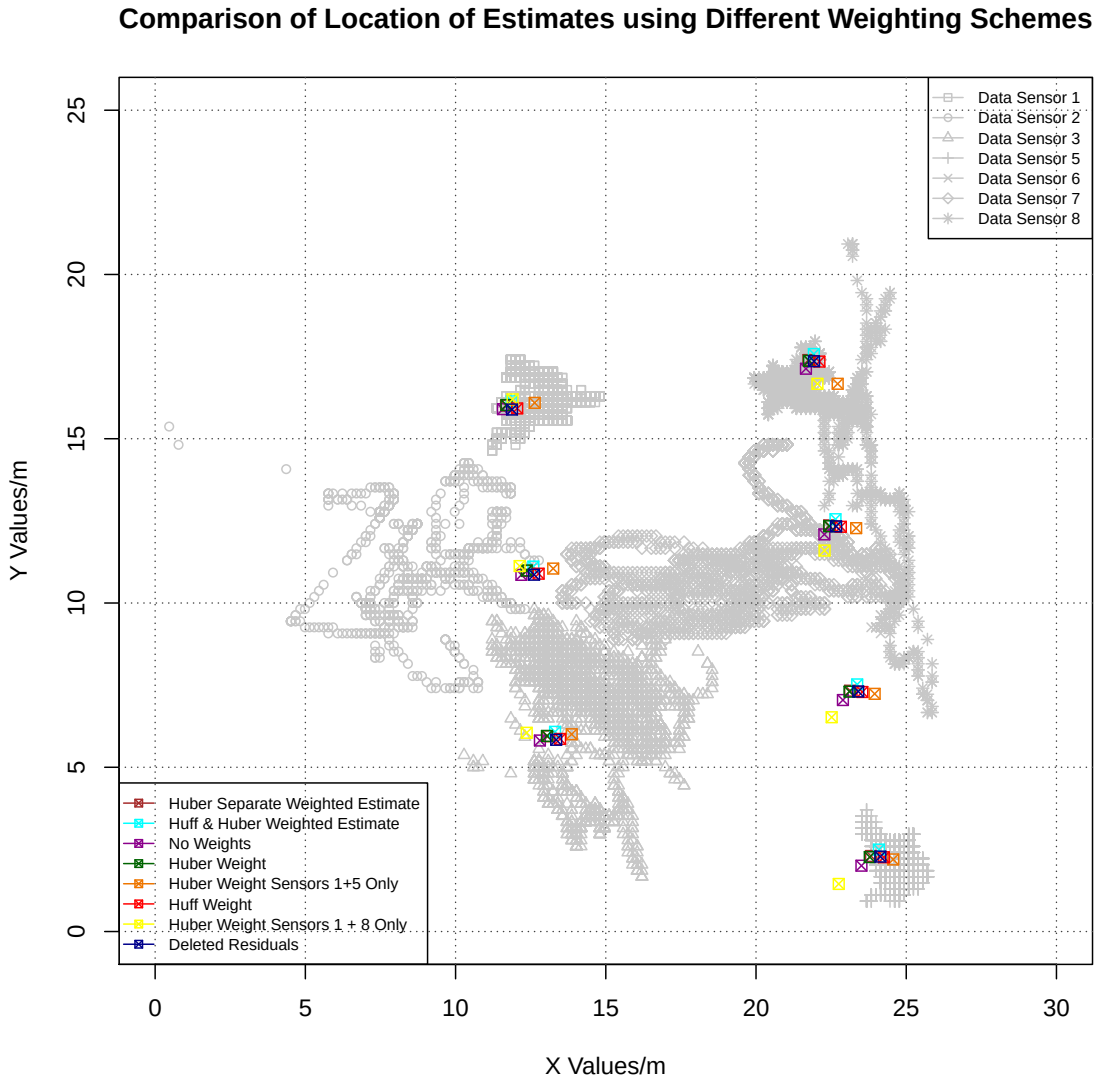
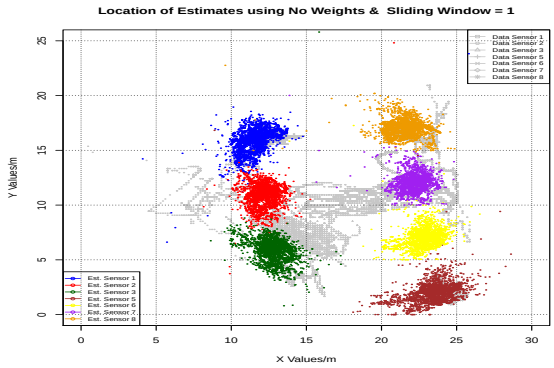


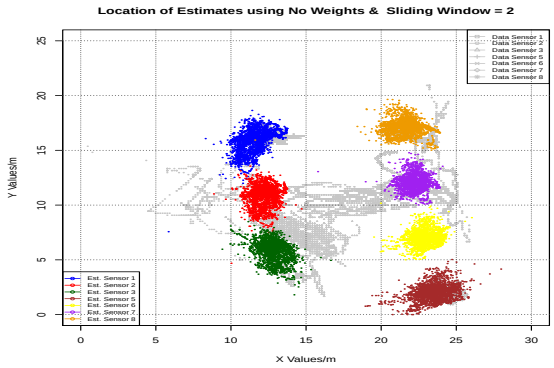
Figure 4.24: GPS Sensors Observed and Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates Summary of Weighted and Unweighted Optimization.

4.7 Unweighted Quadratic Optimization Estimates of Locations of GPS Sensors for Stationary Platform using Sliding Windows

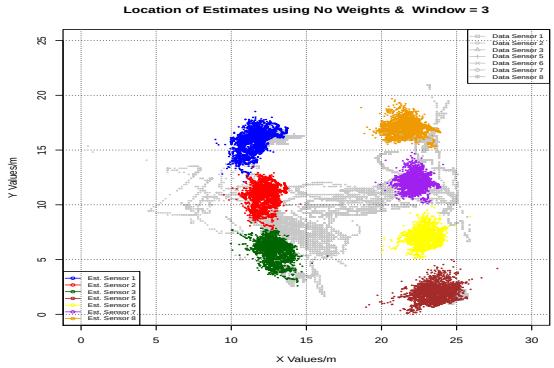
To calculate the estimates of locations of the GPS sensors for the stationary platform, all of the stationary data was divided into iterative sections using sliding windows, over which the estimates are produced. It follows that the number of estimates produced will be the total amount of data less the initial start- up sliding window size. Figures 4.25 and 4.26 summarize the use of sliding windows in the stationary situation. Although the use of sliding windows when the platform is standing still may not be useful at first glance, we may be able to ascertain their utility in handling missing data by observing the spread of the cluster of estimates being produced for each sliding window.From the graph that depicts the comparisons between different sliding windows in Figure 4.27, we can see a definite decrease in the spread of the estimates of locations being produced for each of the sensors as the number of data points in the sliding window increases. This behavior may be attributed to the decrease in effect of missing data.



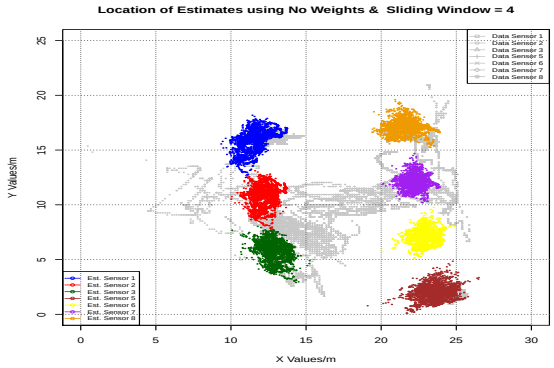
(a) Sliding Window = 1.



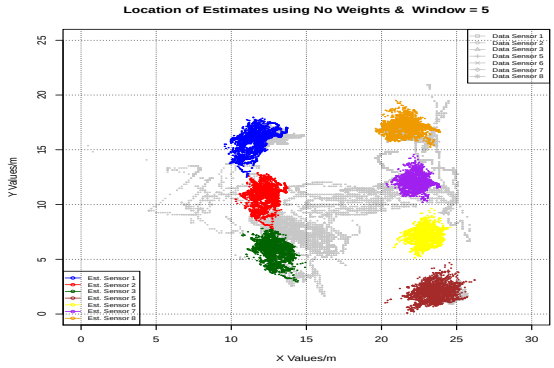
(b) Sliding Window = 2.



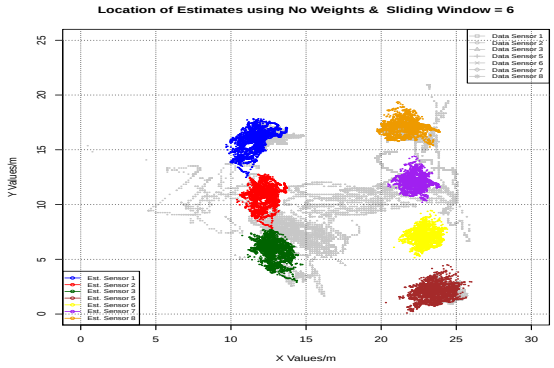
(c) Sliding Window = 3.



(d) Sliding Window = 4.

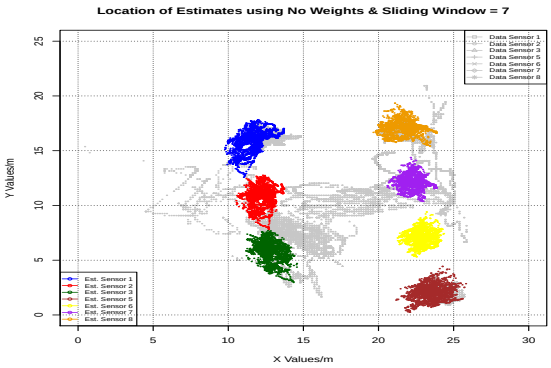


(e) Sliding Window = 5.

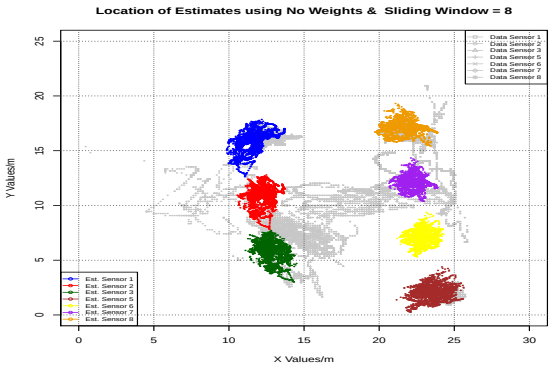


(f) Sliding Window = 6.

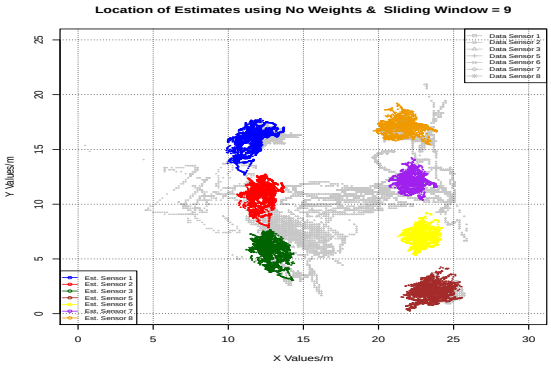
Figure 4.25: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Different Sizes of Sliding Window.



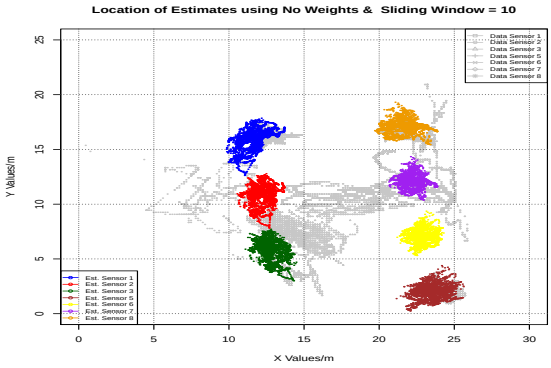
(a) Sliding Window = 7.



(b) Sliding Window = 8.



(c) Sliding Window = 9.



(d) Sliding Window = 10.

Figure 4.26: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Different Sizes of Sliding Window (contd.).

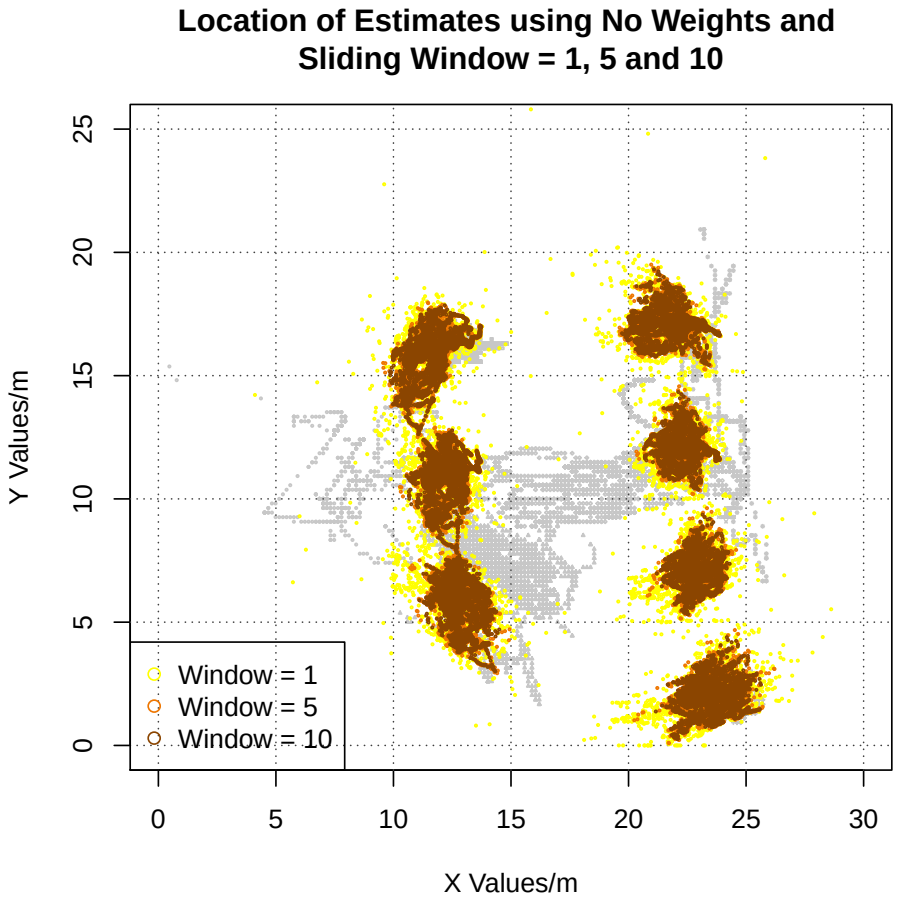


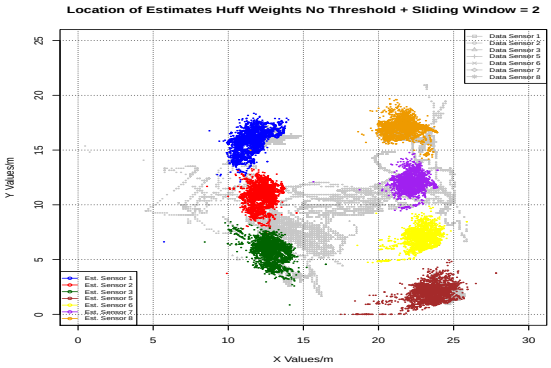
Figure 4.27: GPS Sensors Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates using Sliding Windows of Size = 1,5 and 10 Comparison.

4.8 Huff Weighted Quadratic Optimization Estimates of Locations of GPS Sensors for Stationary Platform using Sliding Windows

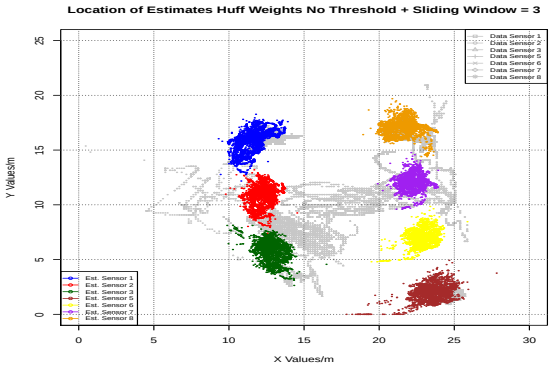
To observe the effect of using a weight function on the estimates for locations of the GPS sensors, we conducted tests using the Huff weighting function combined with different sizes of sliding window. In the description of the Huff weighting algorithm in Section 3.3, there is a threshold over which the error values are considered too high to continue being considered as useful to the calculation of the weights and subsequently estimates. To determine which value of threshold we should use, we calculated estimates of location with respect to the threshold being not present and the threshold being set to 3.5 units, so as to empirically determine the effectiveness of the threshold value. The results are presented in the following sections.

4.8.1 Huff Weight Function With No Threshold

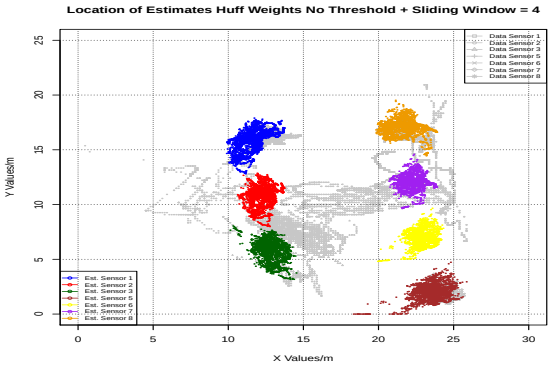
When the threshold is not set in the Huff algorithm, we obtain the results for different sliding window sizes, as shown in Figure 4.28 and 4.29 with a comparison showing the reduction in cluster of the estimates produced using larger sliding window sizes shown in Figure 4.30.



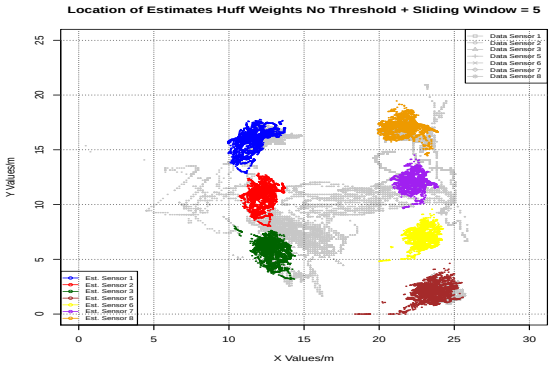
(a) Sliding Window = 2.



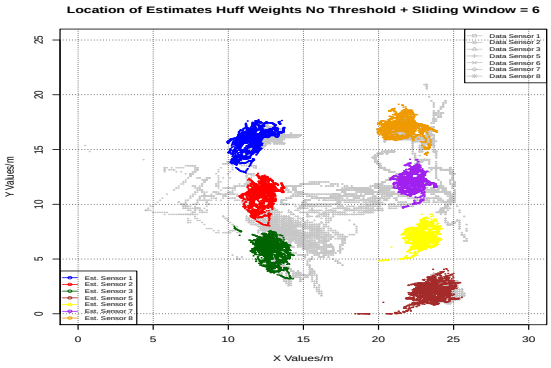
(b) Sliding Window = 3.



(c) Sliding Window = 4.

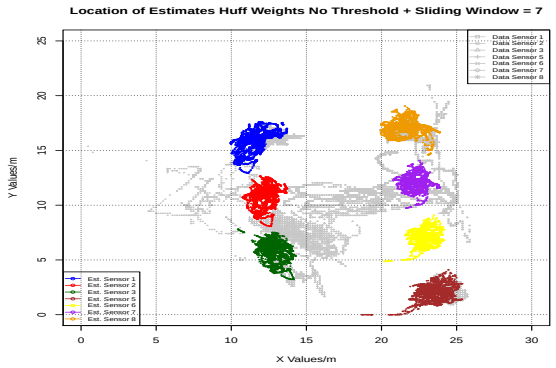


(d) Sliding Window = 5.

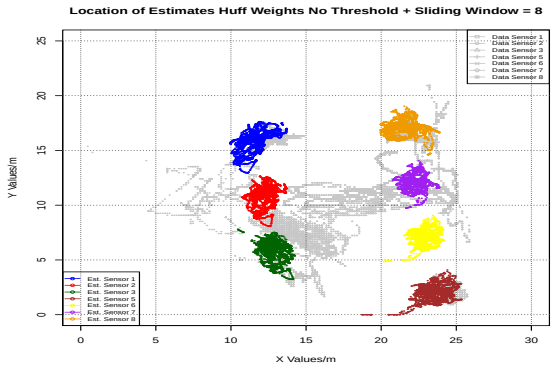


(e) Sliding Window = 6.

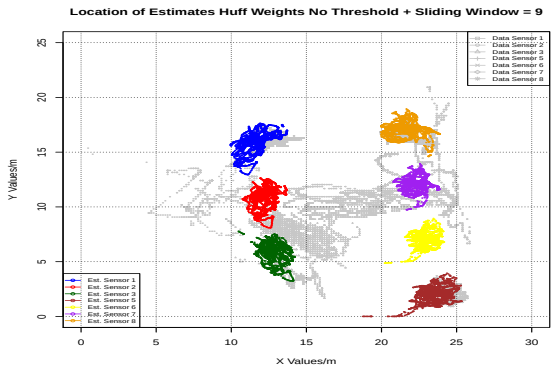
Figure 4.28: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Huff Weighting Function No Threshold and Different Sizes of Sliding Window.



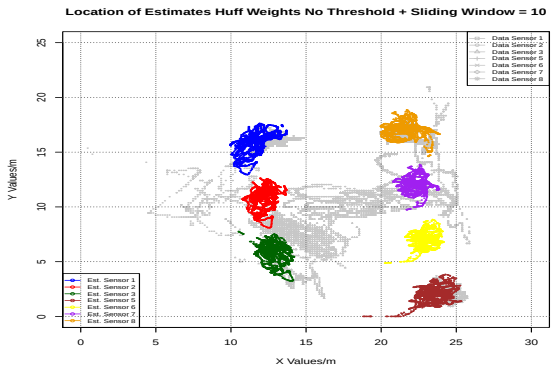
(a) Sliding Window = 7.



(b) Sliding Window = 8.



(c) Sliding Window = 9.



(d) Sliding Window = 10.

Figure 4.29: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Huff Weighting Function No Threshold and Different Sizes of Sliding Window (contd.).

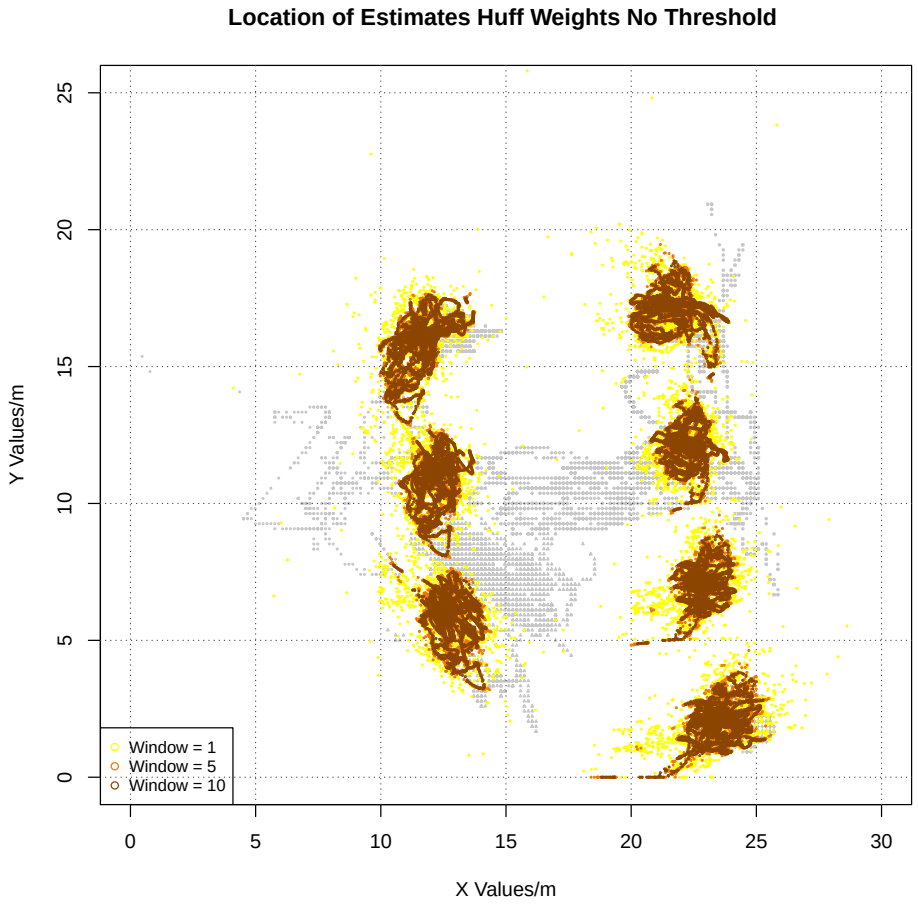
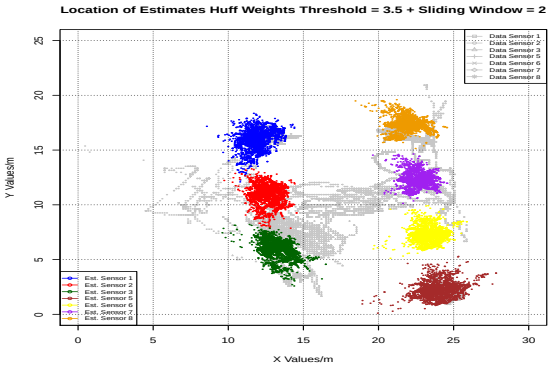


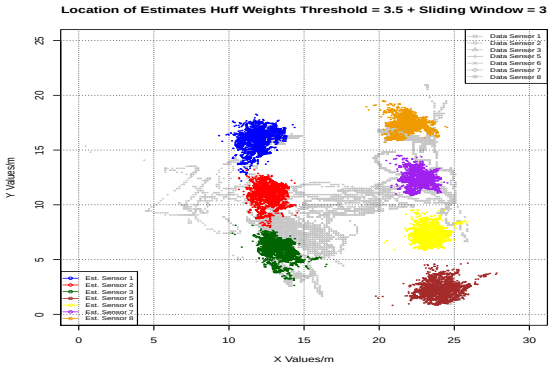
Figure 4.30: GPS Sensors Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates with Huff Weighting Function No Threshold using Sliding Windows of Size = 1,5 and 10 Comparison.

4.8.2 Huff Weight Function With Threshold Value = 3.5

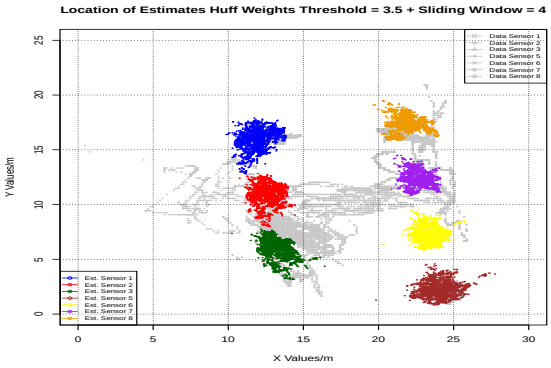
When the threshold is set to 3.5 in the Huff algorithm, we obtain the results for different sliding window sizes, as shown in Figure 4.31 and 4.32 with a comparison showing the reduction in cluster of the estimates produced using larger sliding window sizes shown in Figure 4.33.



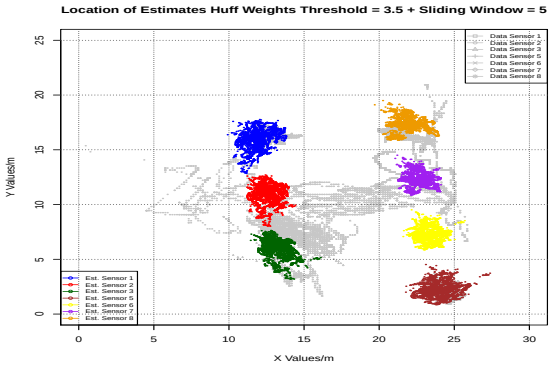
(a) Sliding Window = 2.



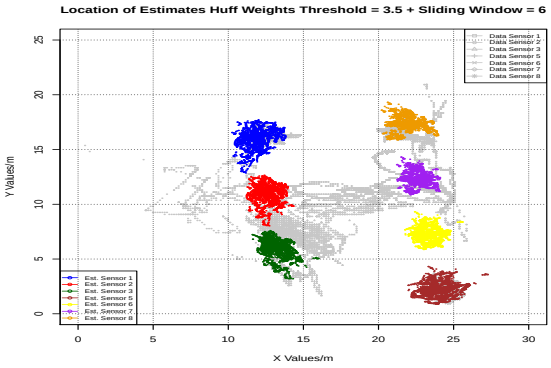
(b) Sliding Window = 3.



(c) Sliding Window = 4.

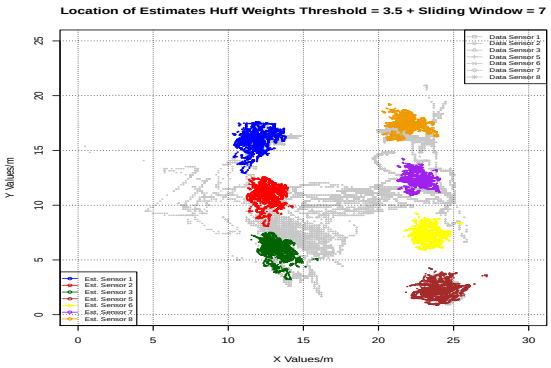


(d) Sliding Window = 5.

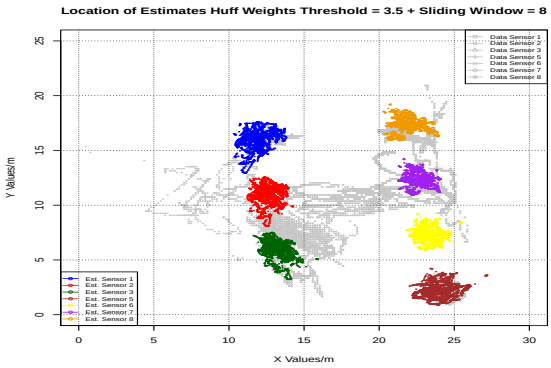


(e) Sliding Window = 6.

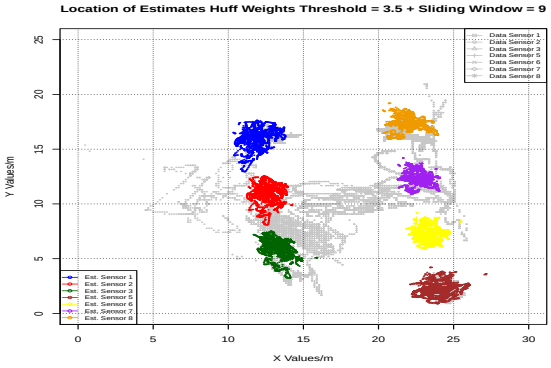
Figure 4.31: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Huff Weighting Function Threshold = 3.5 and Different Sizes of Sliding Window.



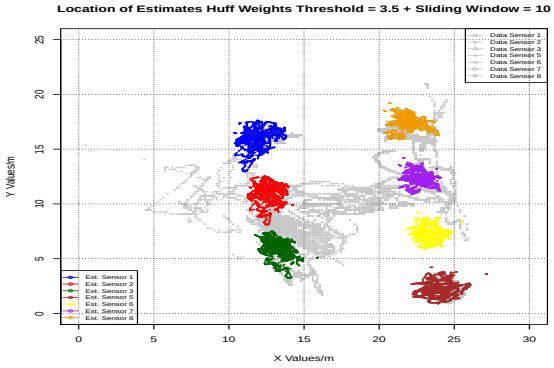
(a) Sliding Window = 7.



(b) Sliding Window = 8.



(c) Sliding Window = 9.



(d) Sliding Window = 10.

Figure 4.32: Estimates for Location of GPS Sensors UTM Y Coordinates vs. X Coordinates for Stationary Platform with Huff Weighting Function Threshold = 3.5 and Different Sizes of Sliding Window (contd.).

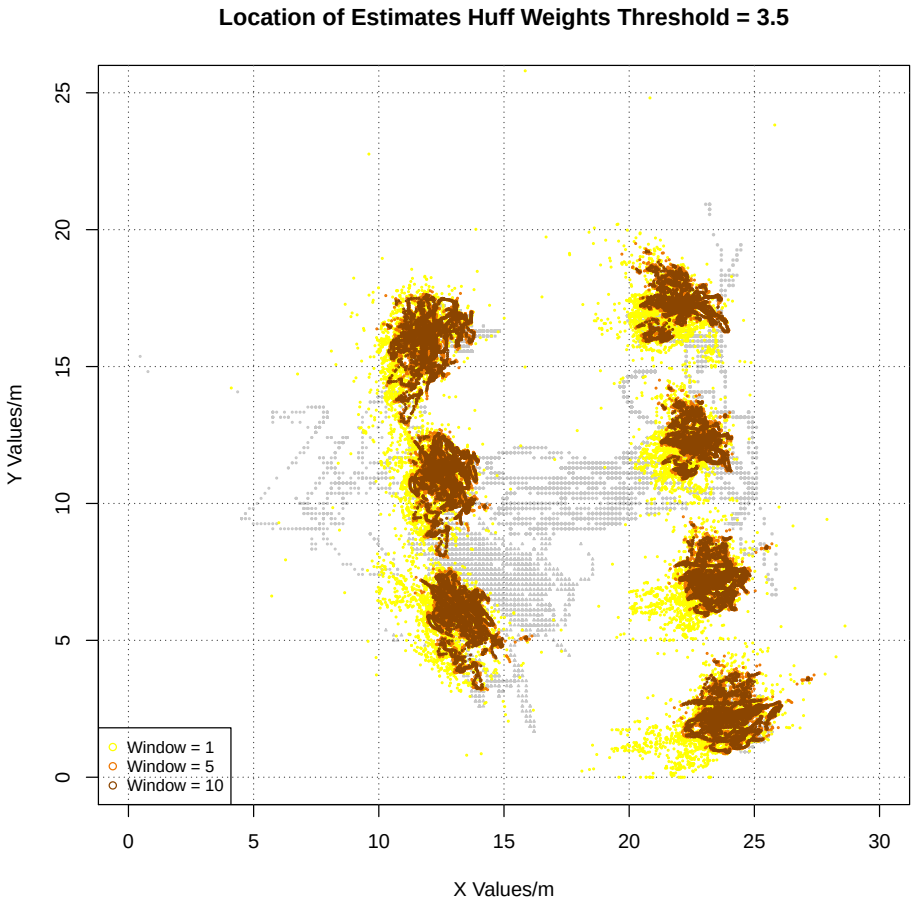


Figure 4.33: GPS Sensors Estimated Stationary Locations UTM Y Coordinates vs. X Coordinates with Huff Weighting Function Threshold = 3.5 using Sliding Windows of Size = 1,5 and 10 Comparison.

4.9 Quadratic Optimization Estimates of Locations of GPS Sensors for Moving Platform

Using the setup described in Section 4.1.2, the mobile platform traveled through given paths that consist of four target points described by the following (x, y) coordinates :

1. Path 1 : Starting at $(0, 0) \rightarrow (100, 0) \rightarrow (100, 100) \rightarrow (0, 100) \rightarrow (0, 0)$.
2. Path 2 : Starting at $(0, 0) \rightarrow (200, 0) \rightarrow (200, 200) \rightarrow (0, 200) \rightarrow (0, 0)$.

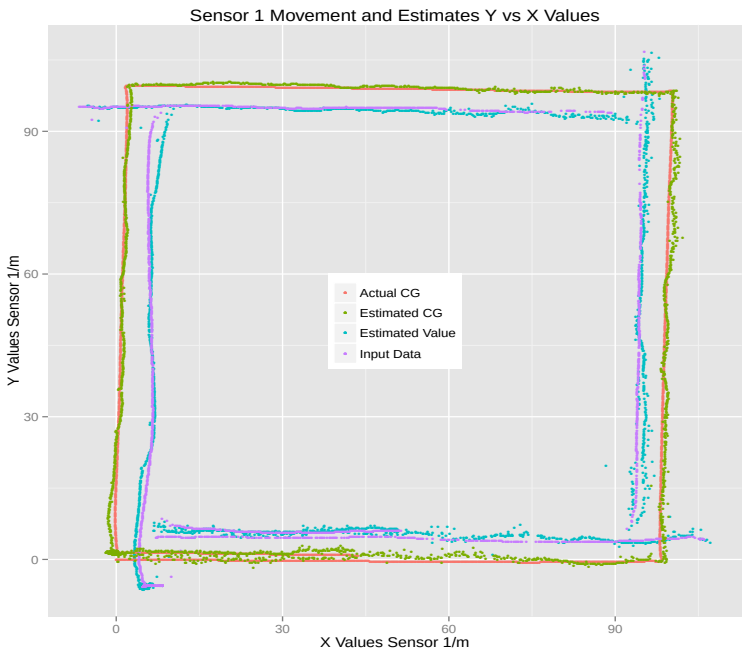
Due to the use of target areas of a certain diameter around the target points defined above, the mobile platform does not necessarily move in a straight line to the target points and may take a curved path as it attempts to reach the target point. Additionally, sliding windows are used to provide real-time estimates of the locations of the sensors on the platform as opposed to the stationary case where since there was no movement, there is no expectation of rapid change in the location of the platform. As mentioned before in Chapter 3, the only method of weighting that is applicable in the situation where the platform is moving is the Huff weighting function, which is compared with random weighting and no weighting for performance.

4.10 Unweighted Quadratic Optimization Estimates of Locations of GPS Sensors for Moving Platform using Sliding Windows

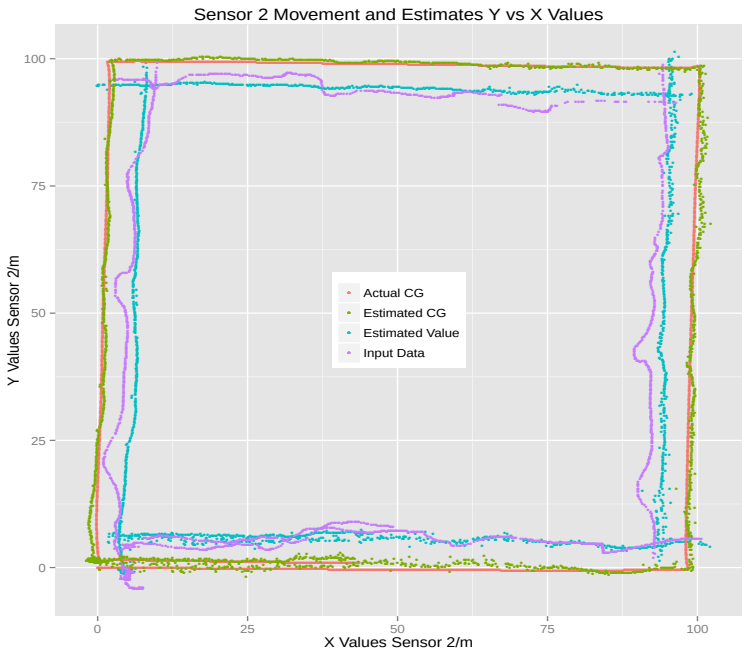
Using the schemes and methodologies established in Section 3.1.2, we can iteratively calculate the location estimates of the platform for different window sizes as the vehicle moves along the pre-determined path. The following sections refer to different paths, as described in Section 4.9.

4.10.1 Location Estimates Using Path 1

For Path 1, Figures 4.34, 4.35, 4.36, 4.37 for sliding window = 1; Figures 4.38, 4.39, 4.40, 4.41 for sliding window = 5; Figures 4.42, 4.43, 4.44, 4.45 for sliding window = 10; Figures 4.46, 4.47, 4.48 and 4.49 for sliding window = 100 show the results of calculating the location estimates for each of the sensors that reported values from Sensor 1,2,3,5,6,7 and 8, along with sliding window values of 1, 5, 10 and 100. In these figures, the red line represents the most accurate path that the platform can take, with the blue line presenting the estimate of the sensor's location, the purple is the observed data generated by the simulation and finally, the green plot representing the estimated center of gravity. On comparison between the red plot and the green plot, we can notice a significant decrease in noise from the observed and estimated data plot. Note that the sensor locations have a constant distance between the center of gravity of the mobile platform and themselves.

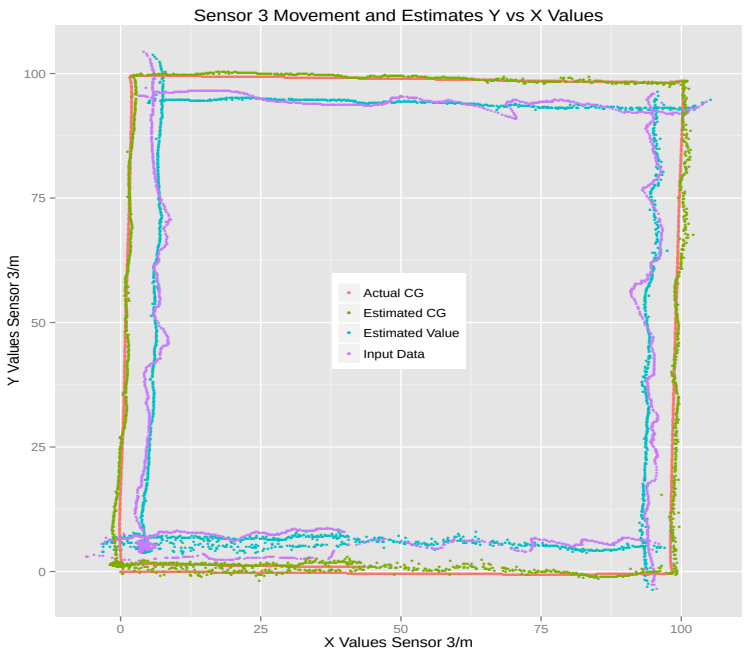


(a) Sensor 1.

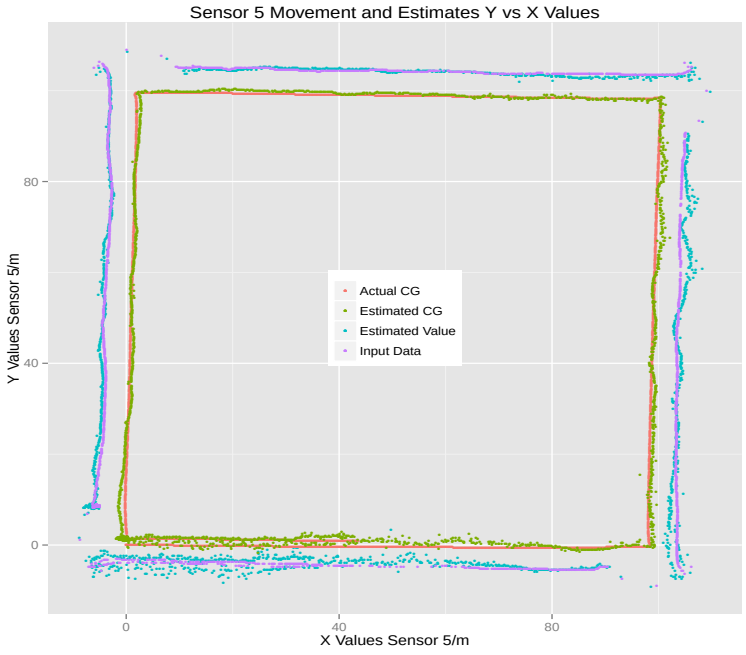


(b) Sensor 2.

Figure 4.34: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8.

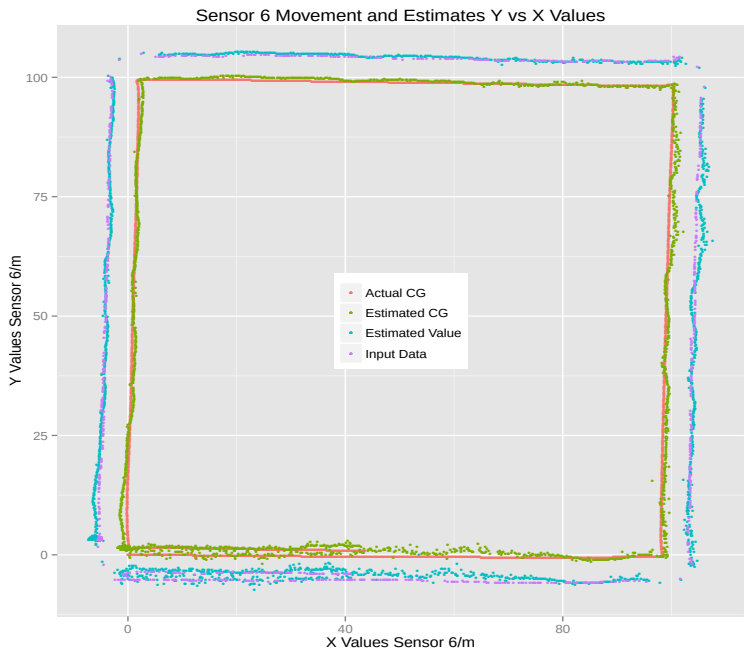


(a) Sensor 3.

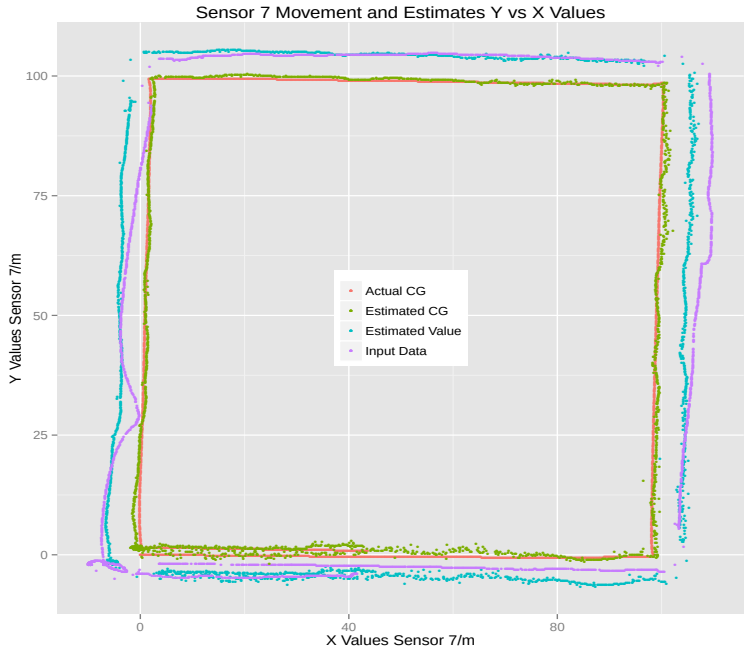


(b) Sensor 5.

Figure 4.35: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

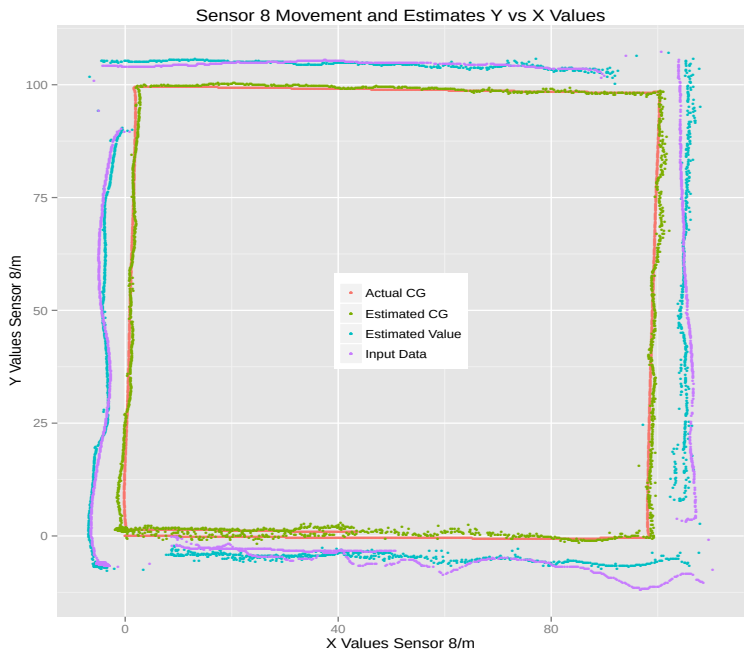


(a) Sensor 6.



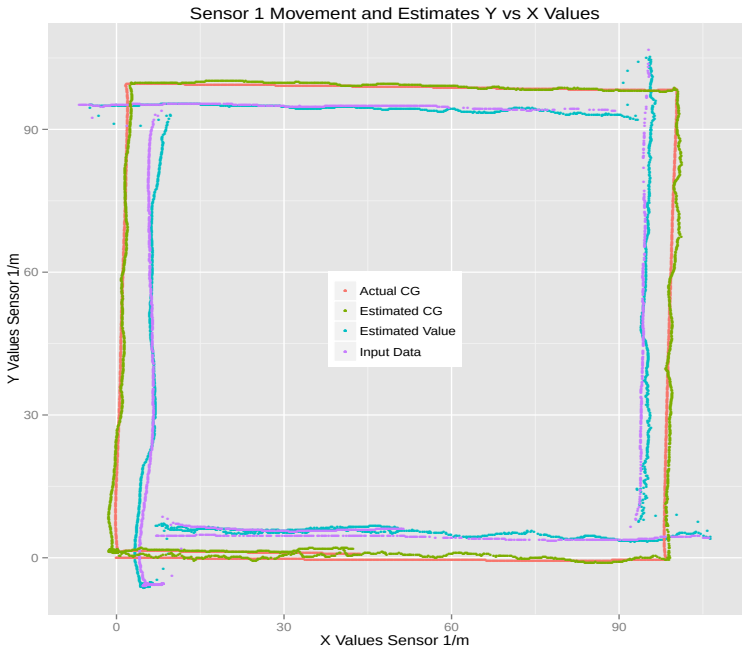
(b) Sensor 7.

Figure 4.36: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

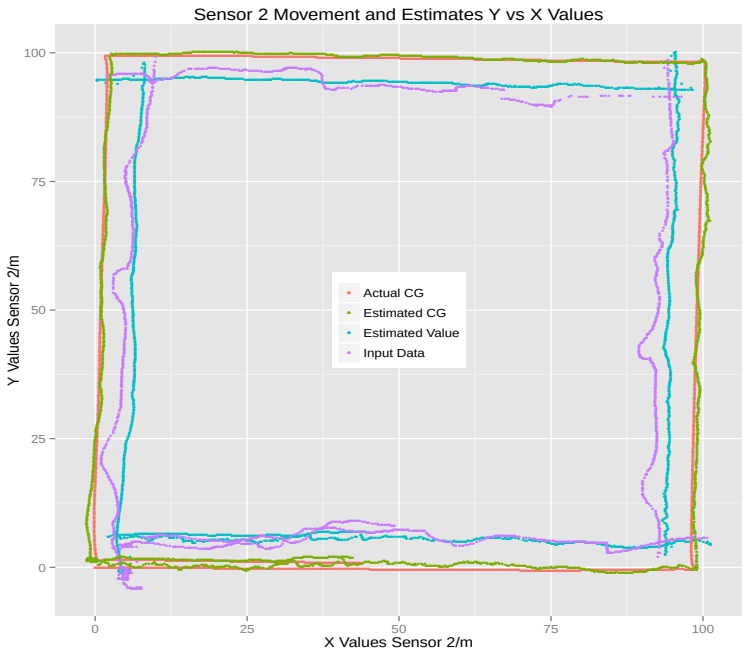


(a) Sensor 8.

Figure 4.37: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

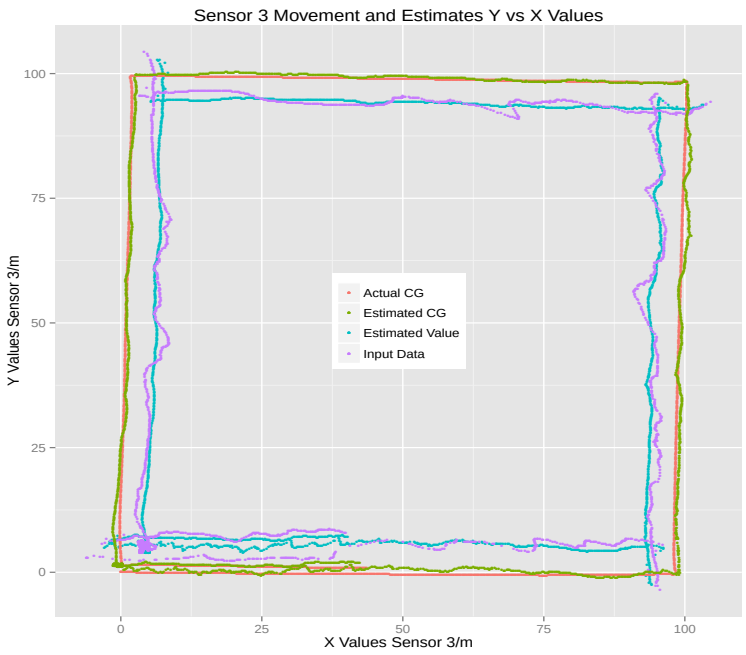


(a) Sensor 1.

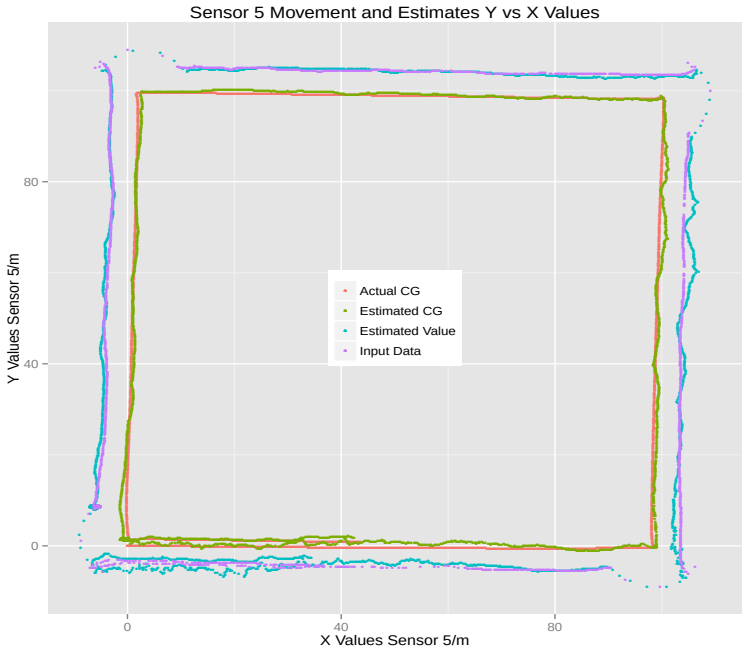


(b) Sensor 2.

Figure 4.38: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8.

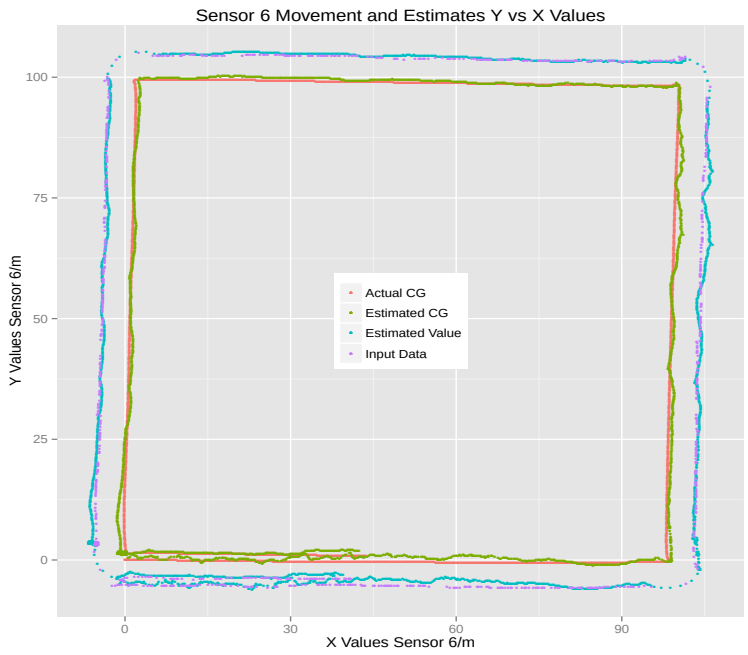


(a) Sensor 3.

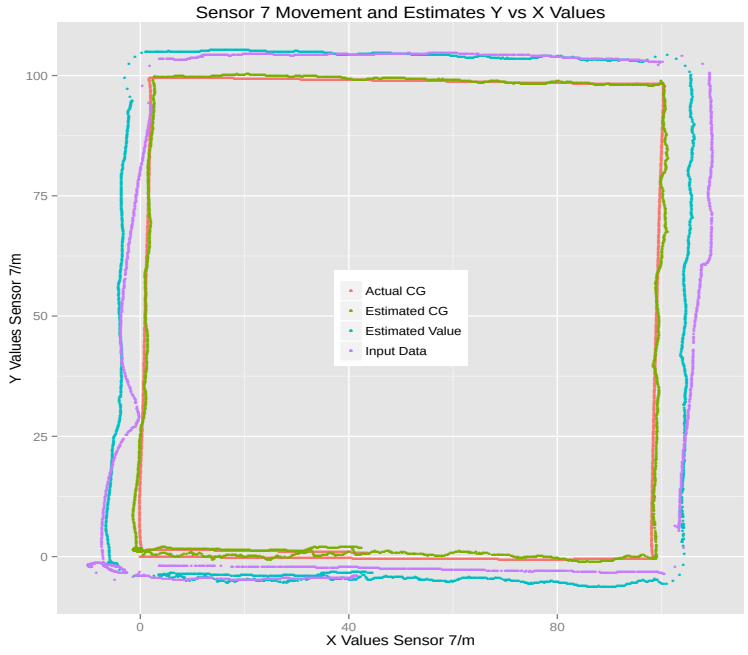


(b) Sensor 5.

Figure 4.39: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

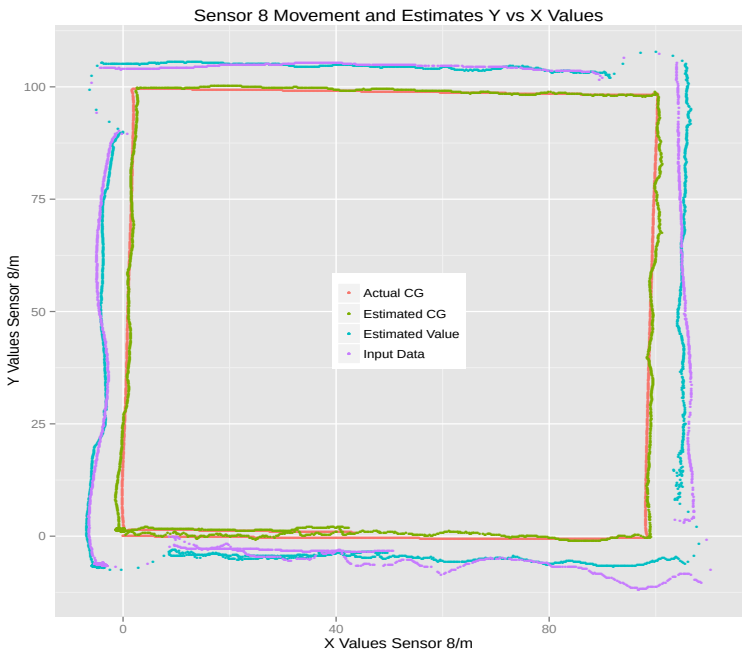


(a) Sensor 6.



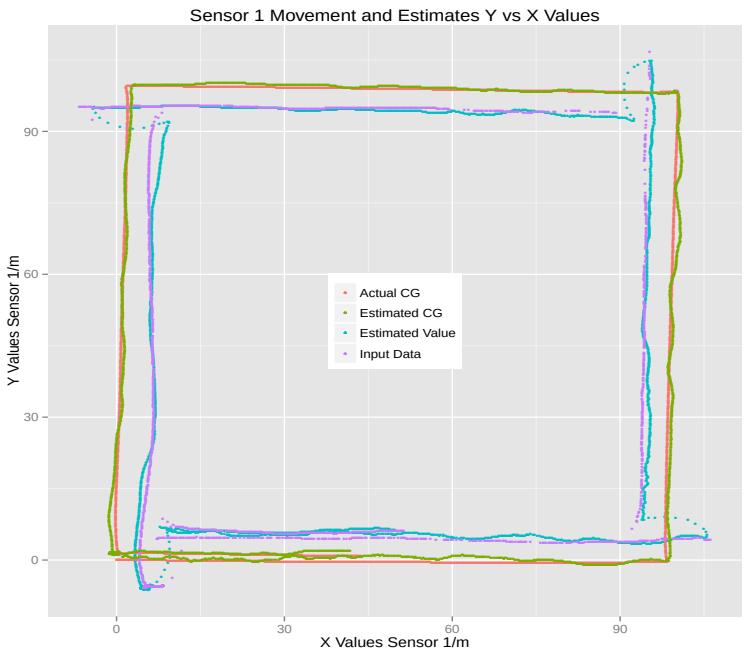
(b) Sensor 7.

Figure 4.40: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

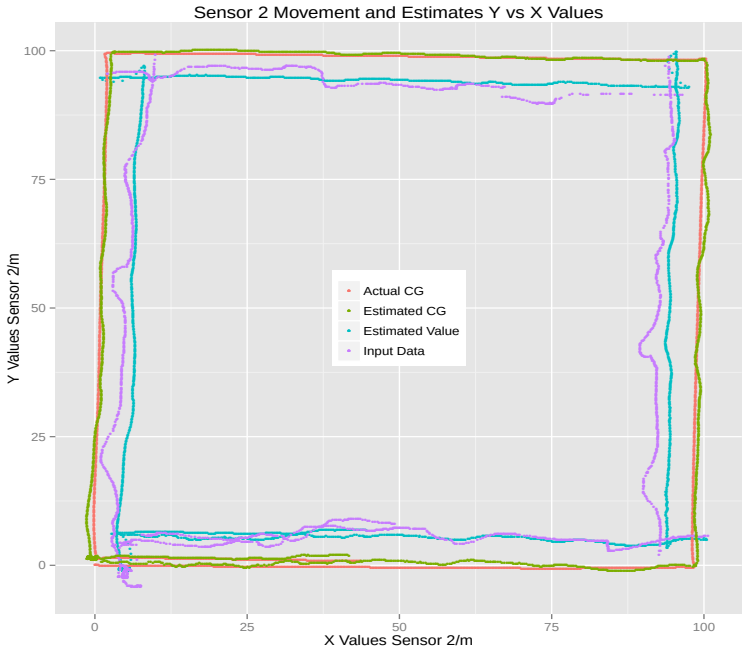


(a) Sensor 8.

Figure 4.41: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

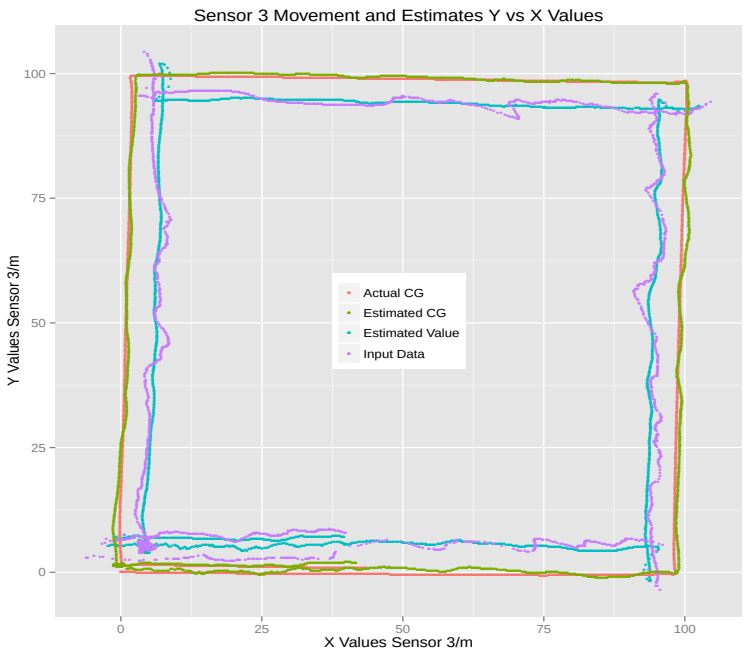


(a) Sensor 1.

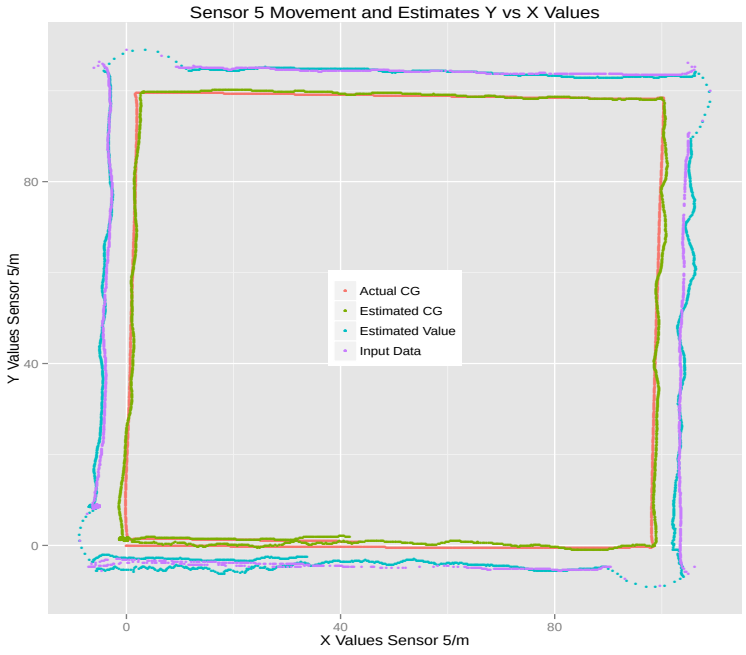


(b) Sensor 2.

Figure 4.42: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8.

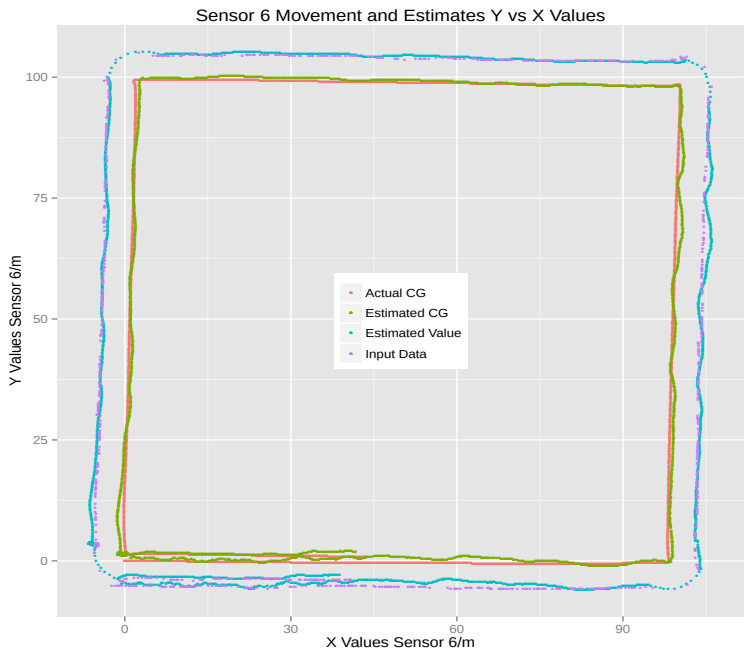


(a) Sensor 3.

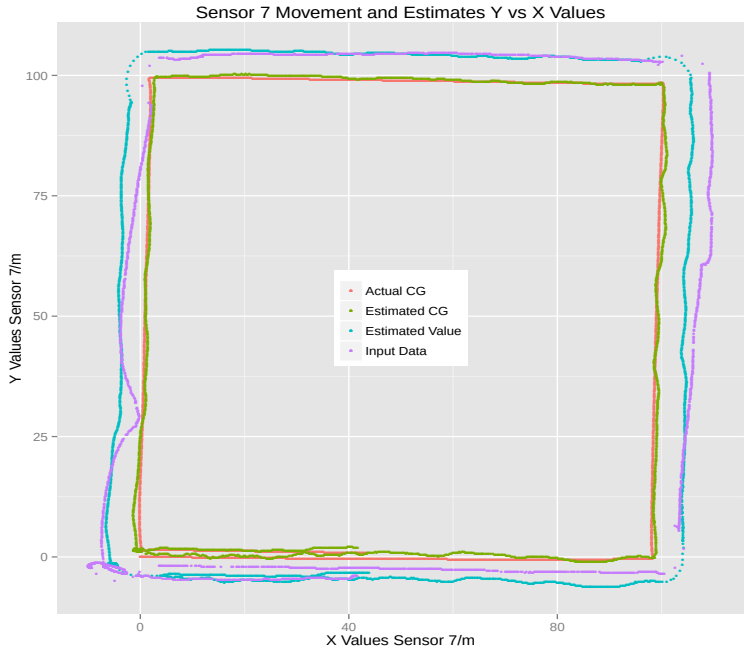


(b) Sensor 5.

Figure 4.43: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

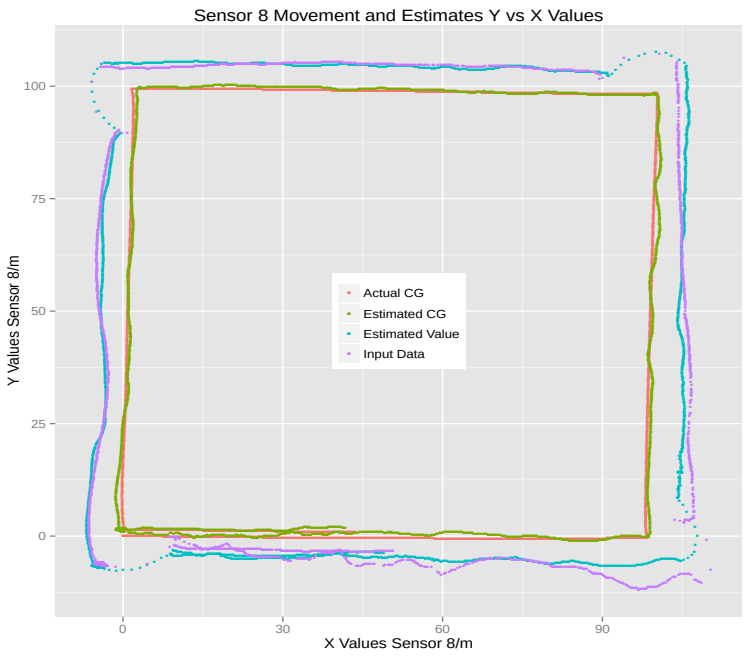


(a) Sensor 6.



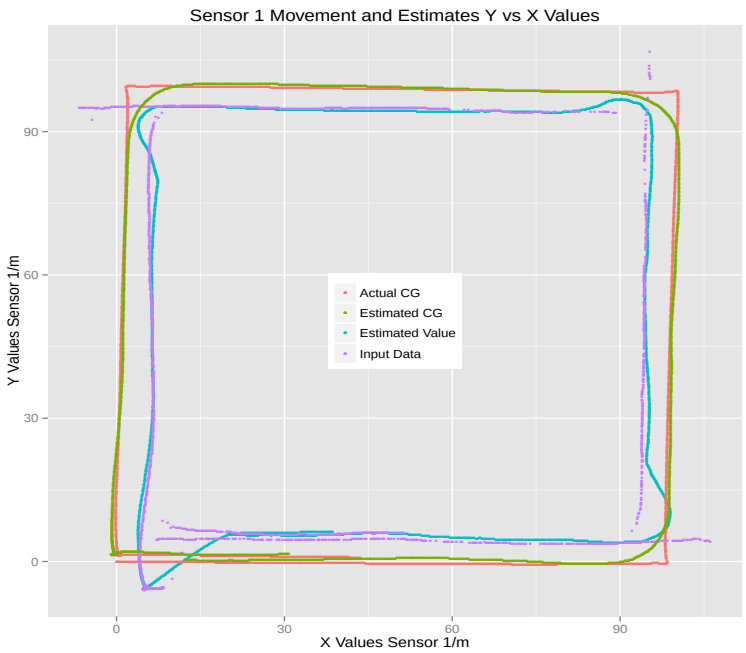
(b) Sensor 7.

Figure 4.44: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

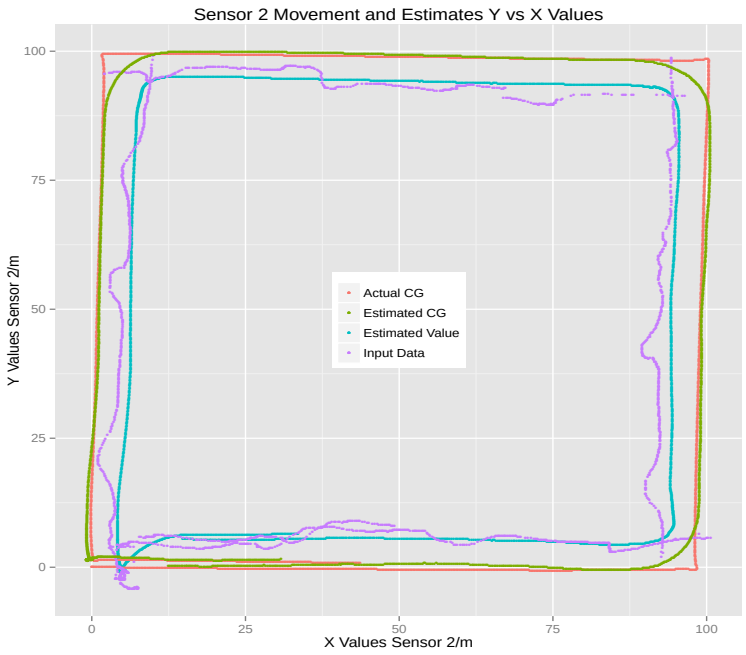


(a) Sensor 8.

Figure 4.45: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

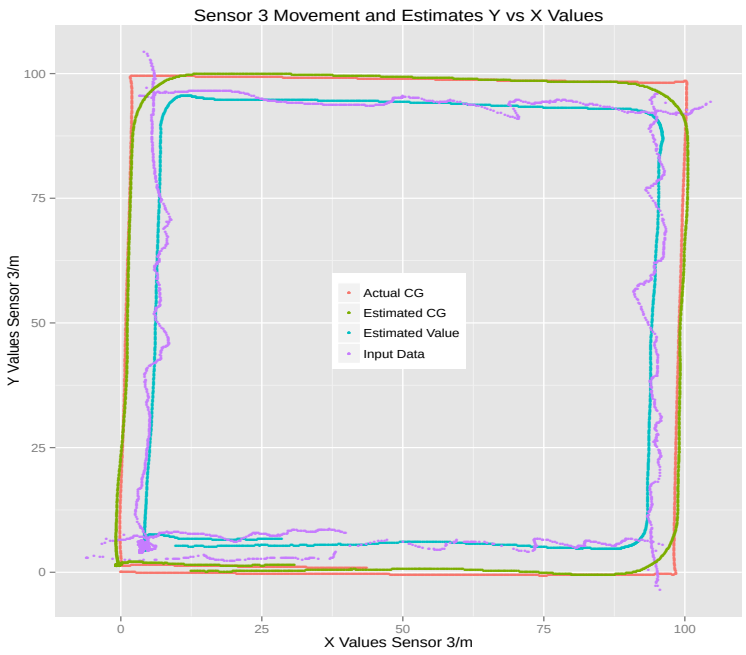


(a) Sensor 1.

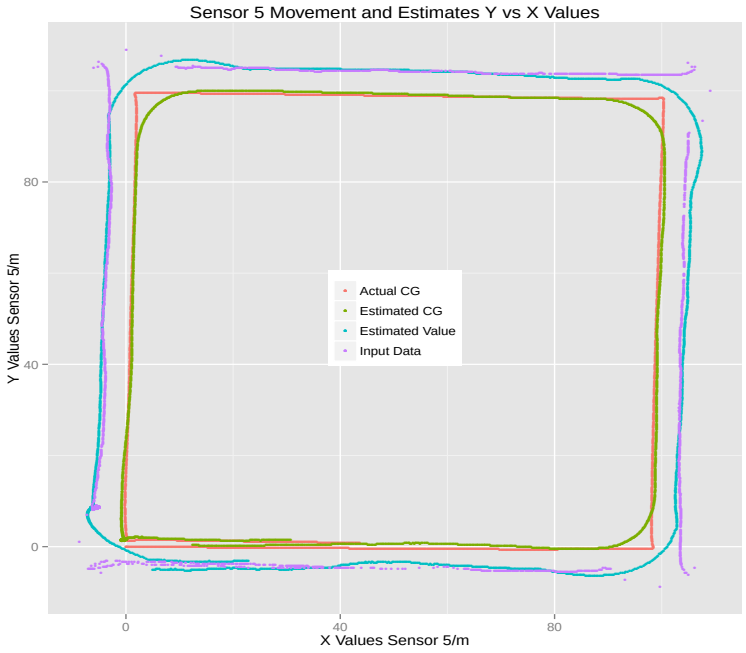


(b) Sensor 2.

Figure 4.46: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8.

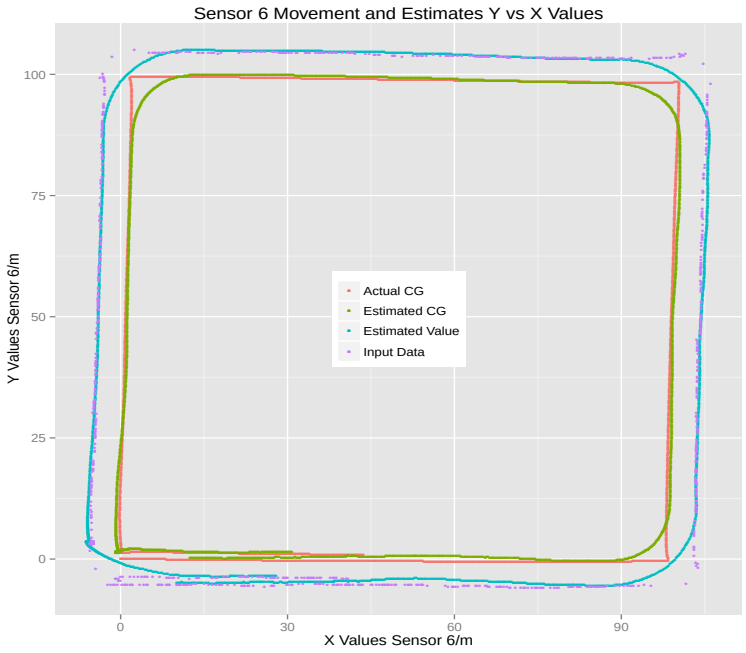


(a) Sensor 3.

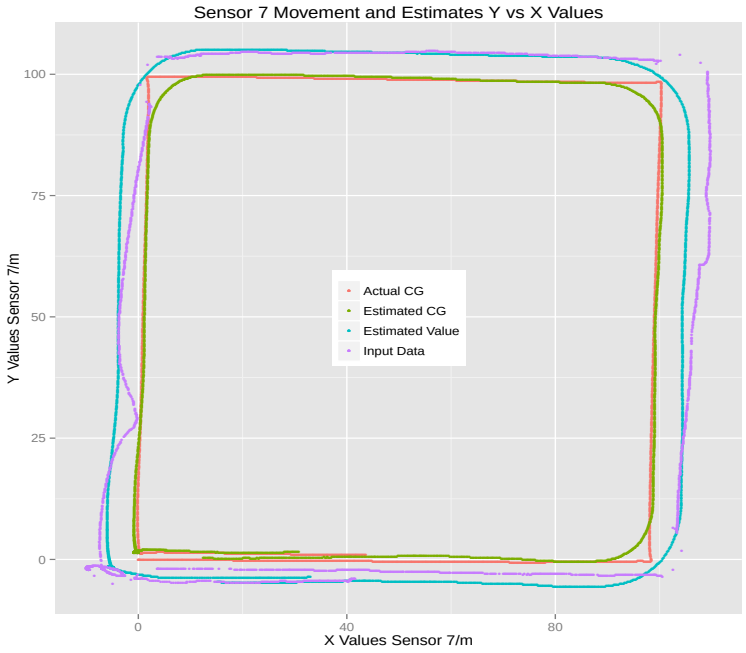


(b) Sensor 5.

Figure 4.47: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

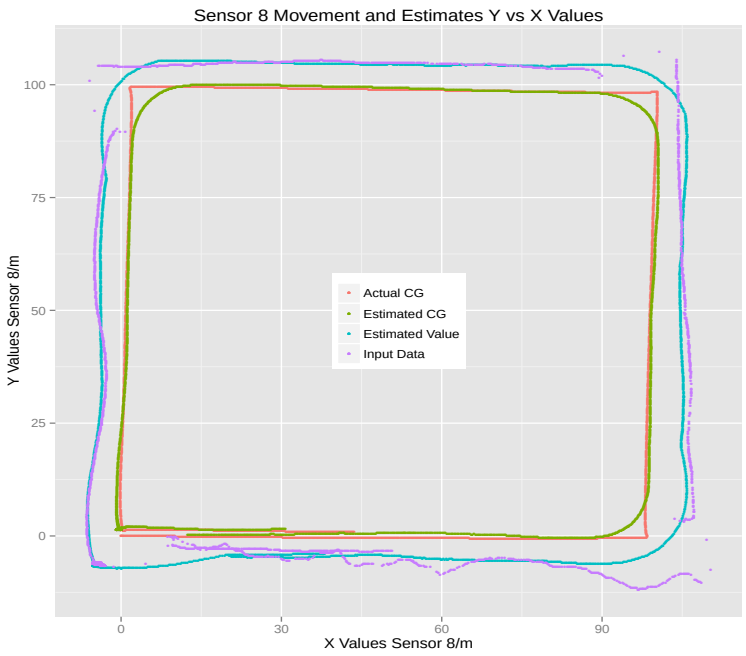


(a) Sensor 6.



(b) Sensor 7.

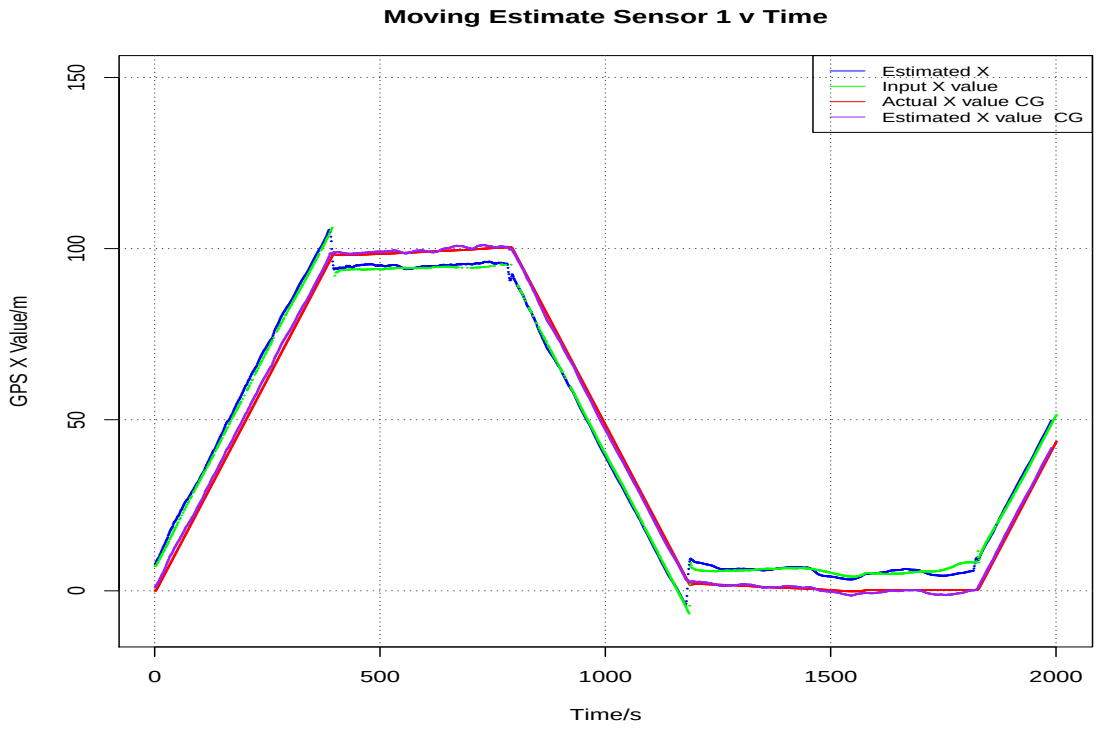
Figure 4.48: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).



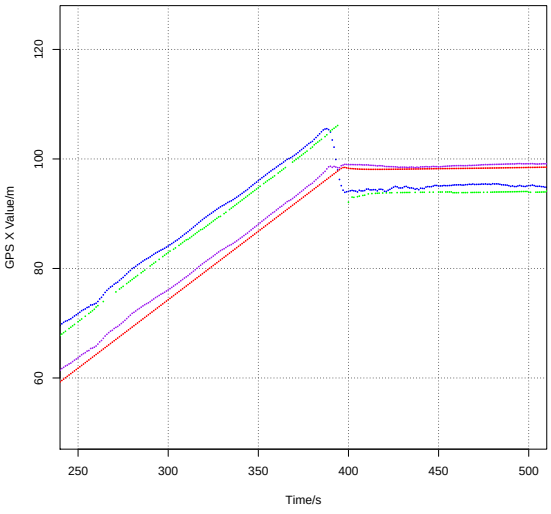
(a) Sensor 8.

Figure 4.49: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

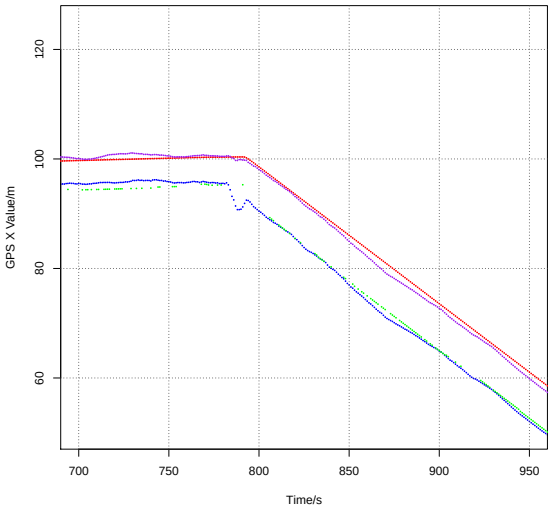
From the plots, we can observe that as the sliding window size increases, the variation from one estimate to the next is greatly reduced. However we cannot indiscriminately increase the sliding window size as the lag effect becomes more pronounced as seen for Sensor 1 in sliding windows 10 in Figures 4.50 4.51 and sliding window = 100 in Figures 4.52 and 4.53.It is clear that the location estimates calculated through quadratic optimization using the configuration of the sensors as constraints for a large sliding window seem to fall behind and attempt to "catch up" to the true location values.



(a) Sensor 1.

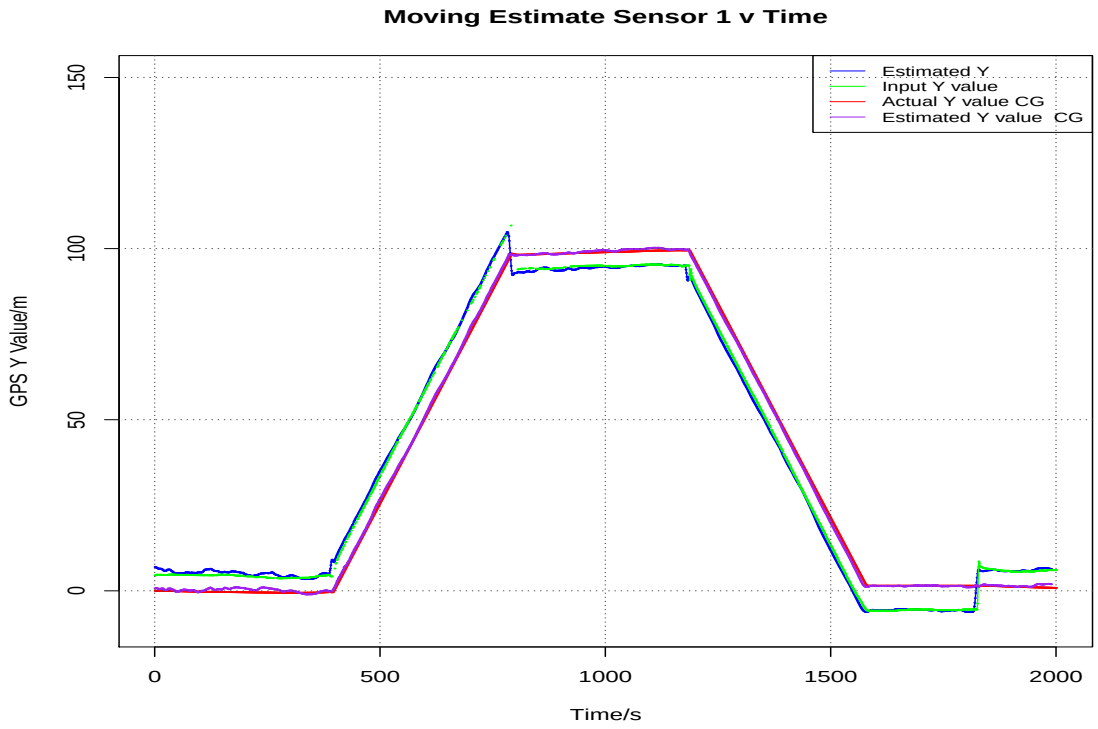


(b) Sensor 1 Positive Slope Magnified.

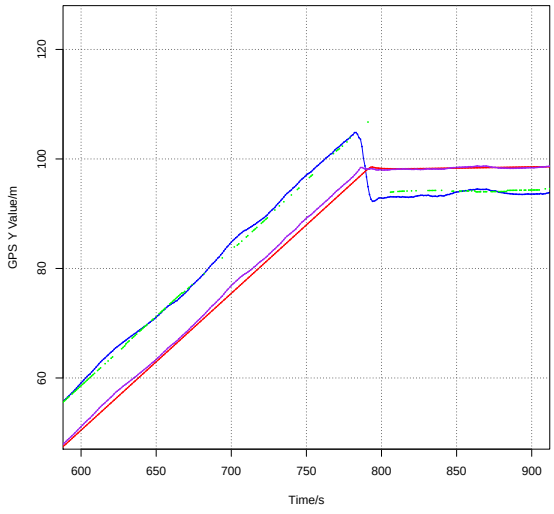


(c) Sensor 1 Negative Slope Magnified.

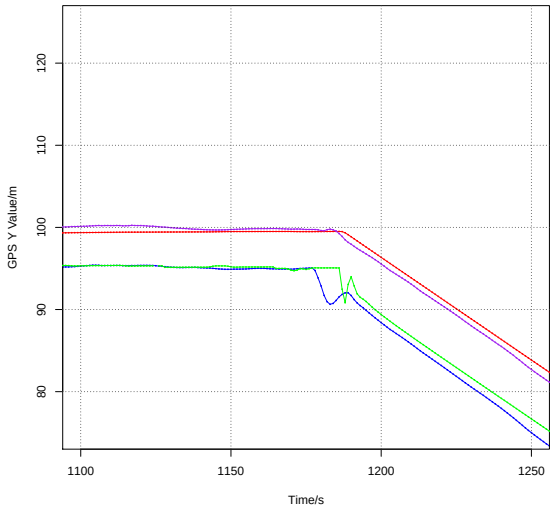
Figure 4.50: Location Estimates of Sensor 1 UTM X Coordinates vs. Time No Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

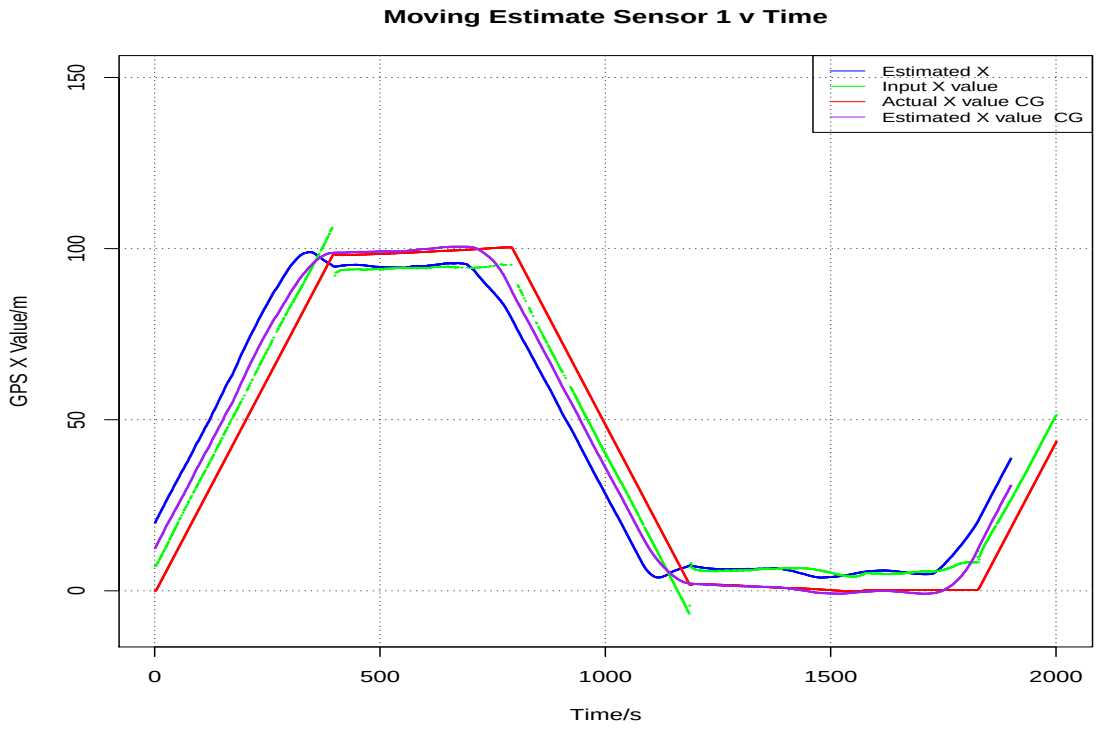


(b) Sensor 1 Positive Slope Magnified.

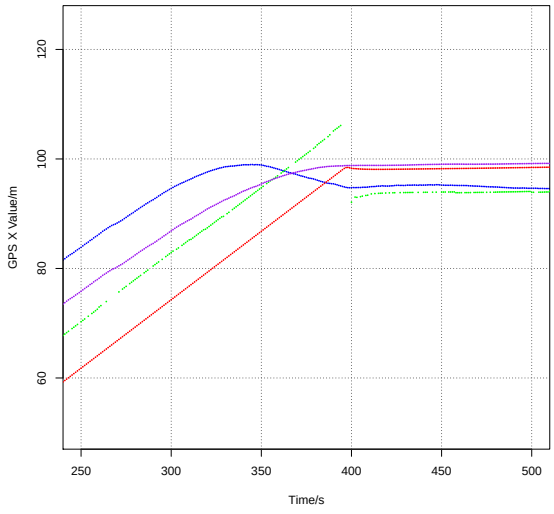


(c) Sensor 1 Negative Slope Magnified.

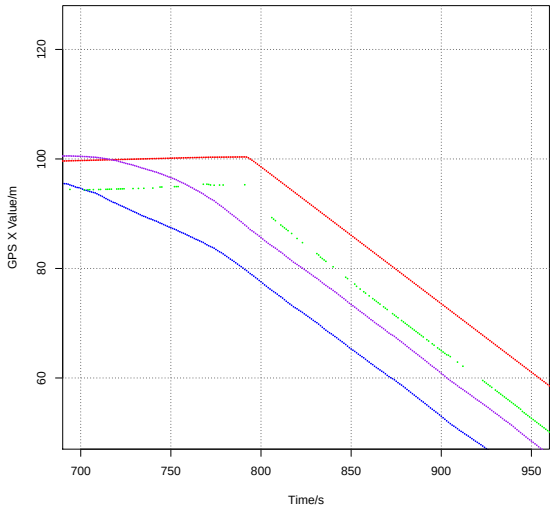
Figure 4.51: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time No Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

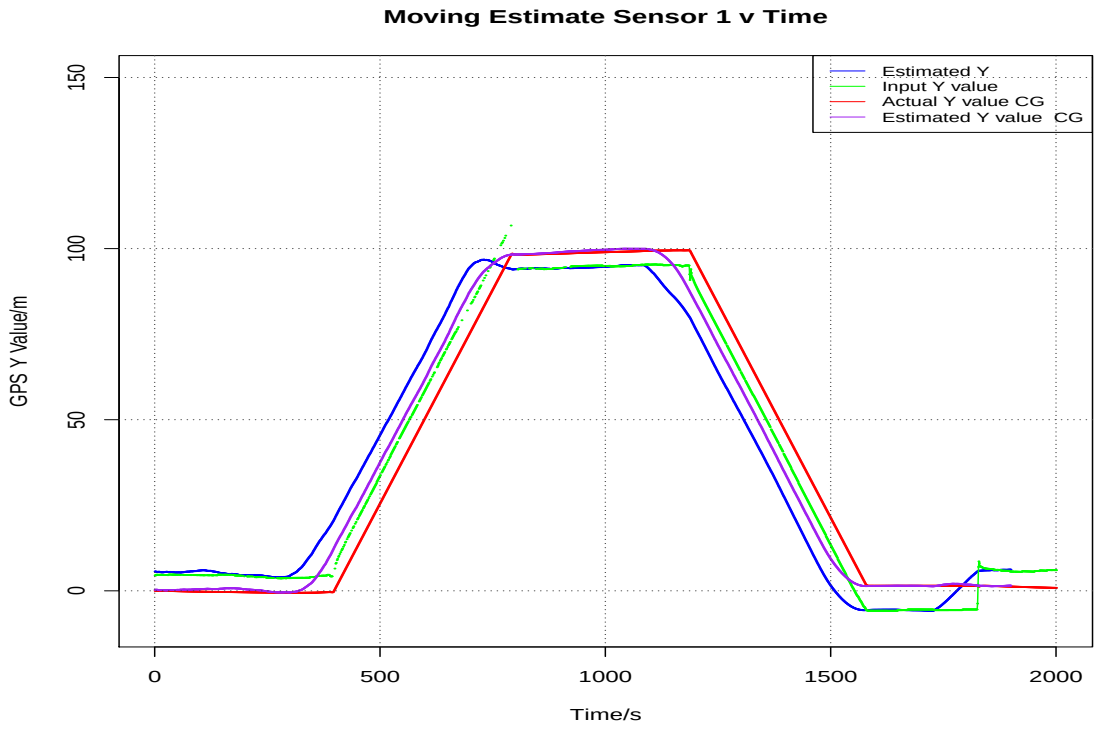


(b) Sensor 1 Positive Slope Magnified.

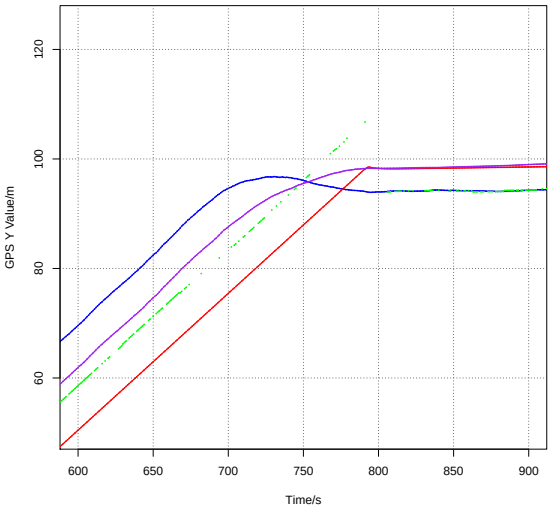


(c) Sensor 1 Negative Slope Magnified.

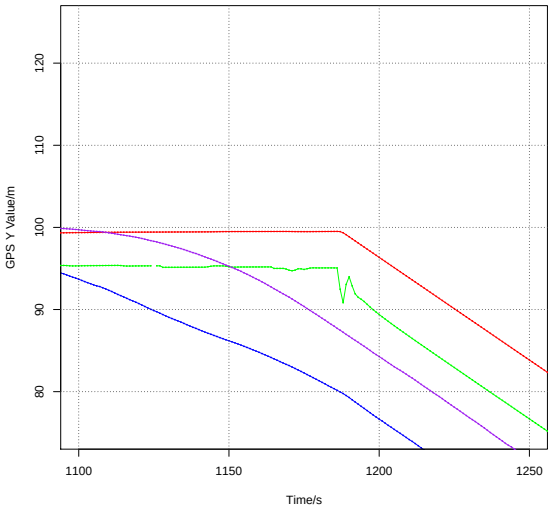
Figure 4.52: Location Estimates of Sensor 1 UTM X Coordinates vs. Time No Weights Sliding Window = 100 $(+veX - North, +veY - East)$.



(a) Sensor 1.



(b) Sensor 1 Positive Slope Magnified.

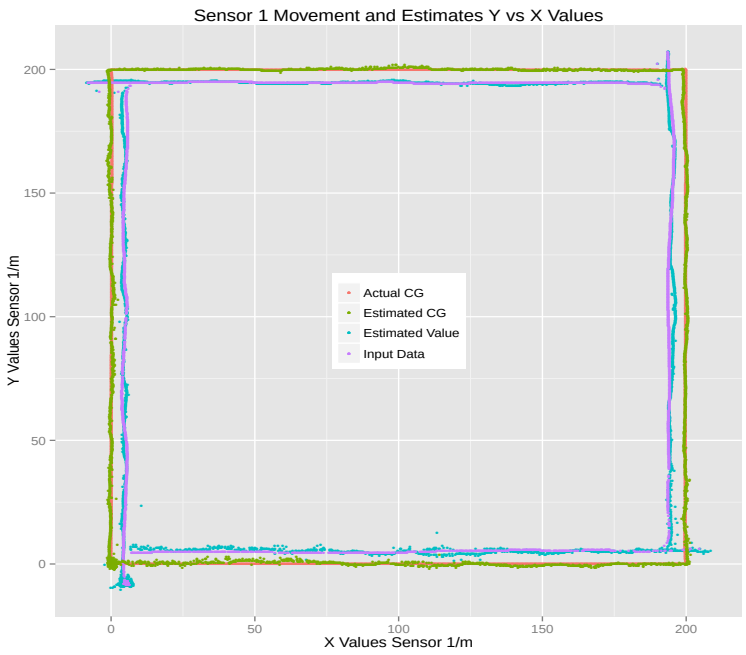


(c) Sensor 1 Negative Slope Magnified.

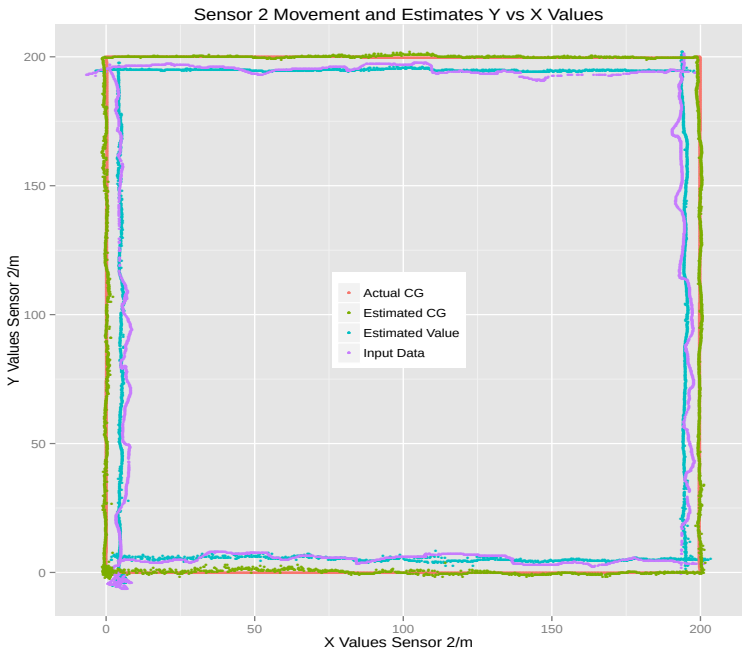
Figure 4.53: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time No Weights Sliding Window = 100 ($+veX - North, +veY - East$).

4.10.2 Location Estimates Using Path 2

For Path 2, Figures 4.54, 4.55, 4.56, 4.57 for sliding window = 1; Figures 4.58, 4.59, 4.60, 4.61 for sliding window = 5; Figures 4.62, 4.63, 4.64, 4.65 for sliding window = 10; Figures 4.66, 4.67, 4.68 and 4.69 for sliding window = 100 show the results of calculating the location estimates for each of the sensors that reported values from Sensor 1,2,3,5,6,7 and 8, along with sliding window values of 1, 5, 10 and 100. In these figures, the red line represents the most accurate path that the platform can take, with the blue line presenting the estimate of the sensor's location, the purple is the observed data generated by the simulation and finally, the green plot representing the estimated center of gravity. On comparison between the red plot and the green plot, as with Path 1, we can notice a significant decrease in noise from the observed data and location estimates plot.

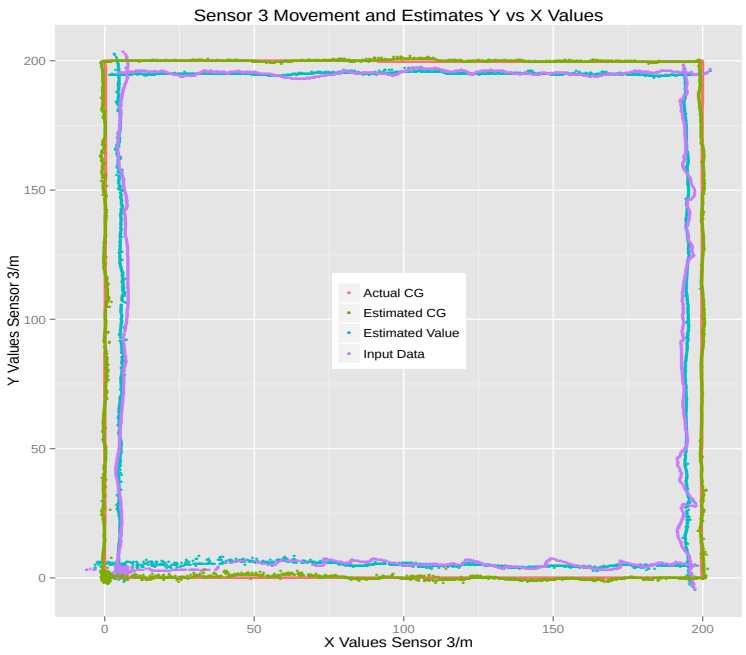


(a) Sensor 1.

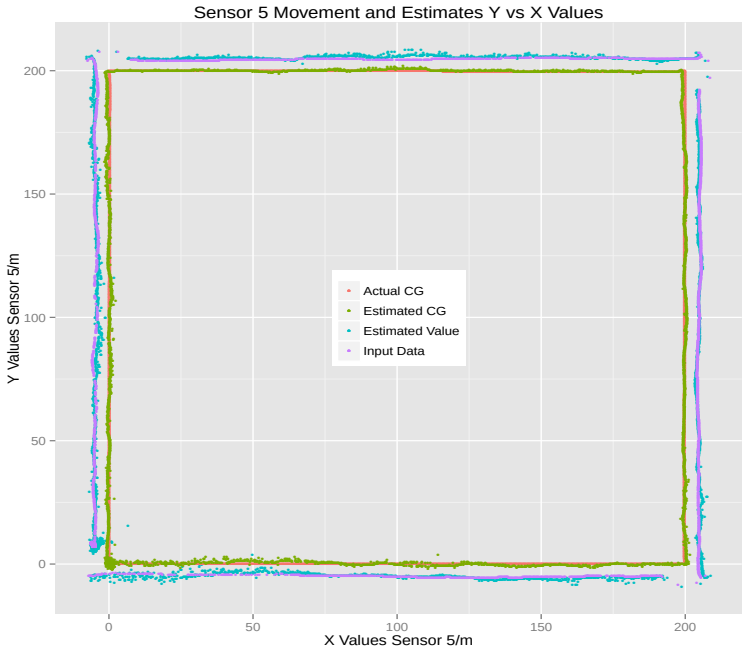


(b) Sensor 2.

Figure 4.54: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8.

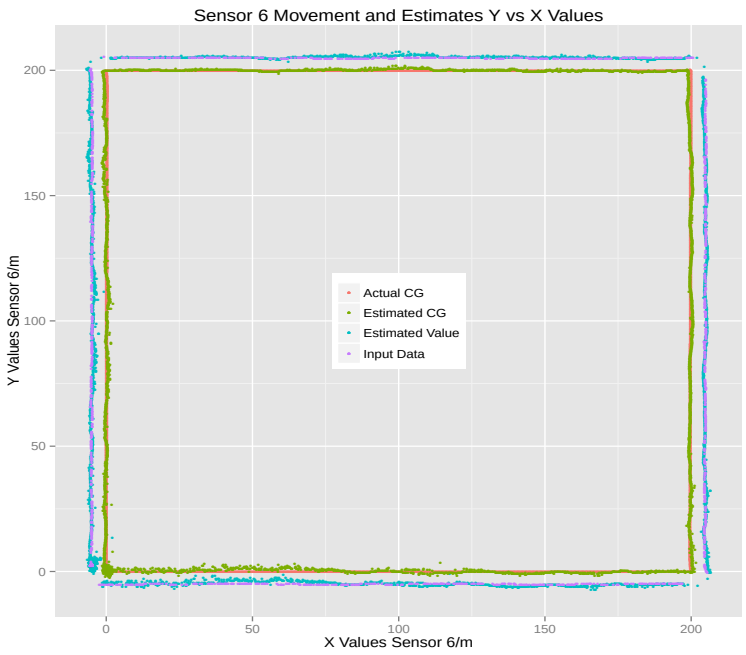


(a) Sensor 3.

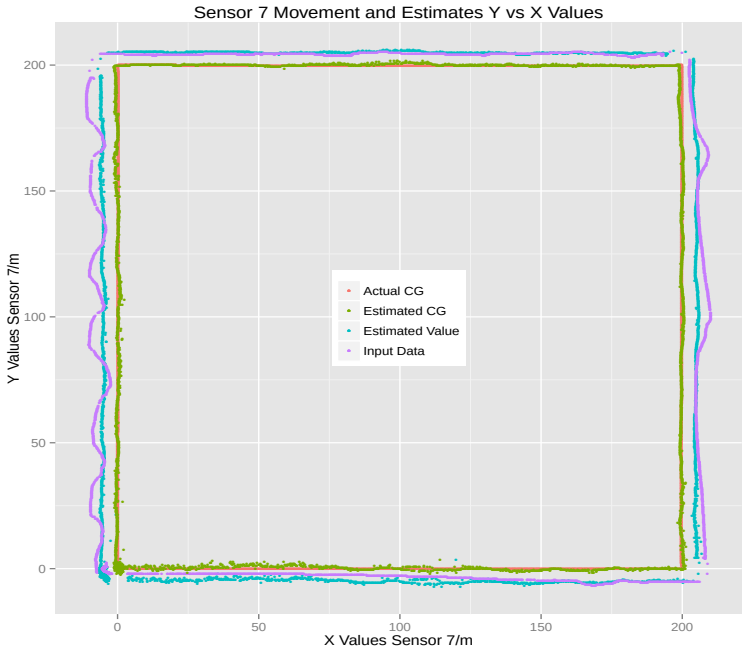


(b) Sensor 5.

Figure 4.55: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

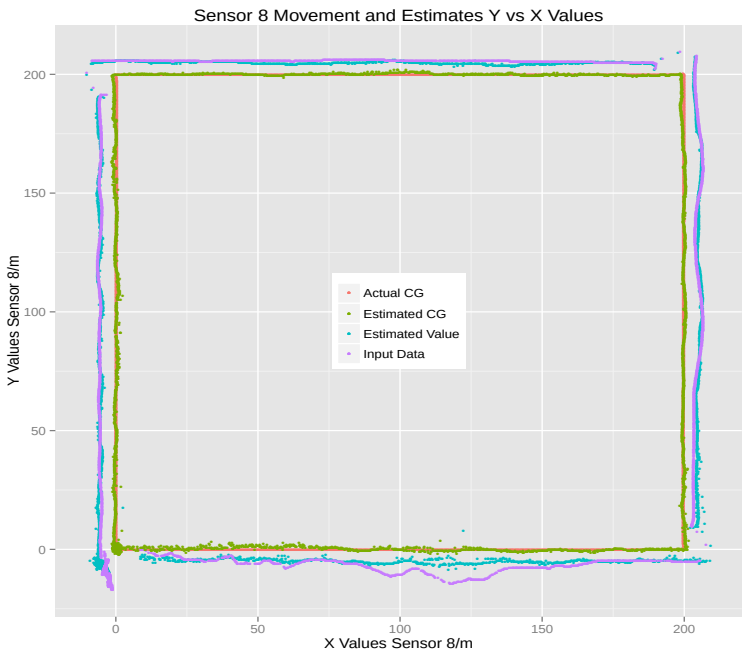


(a) Sensor 6.



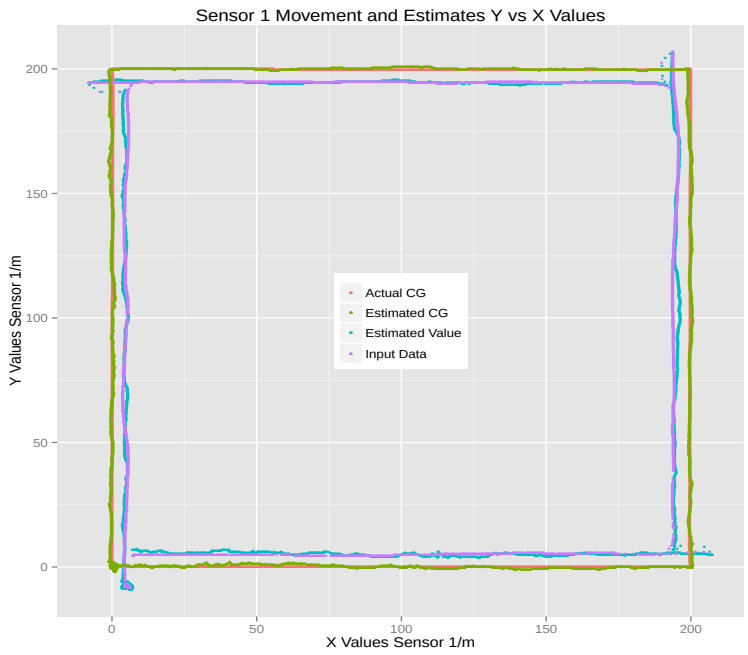
(b) Sensor 7.

Figure 4.56: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

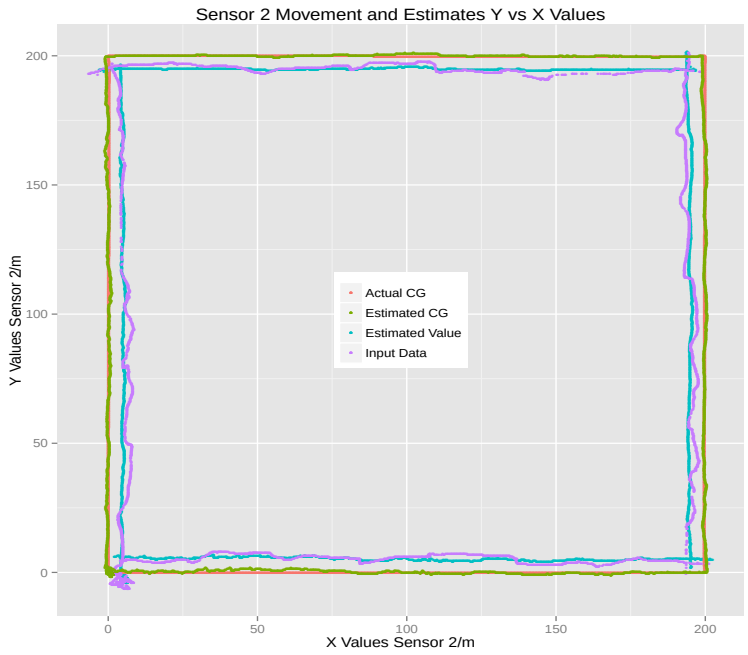


(a) Sensor 8.

Figure 4.57: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

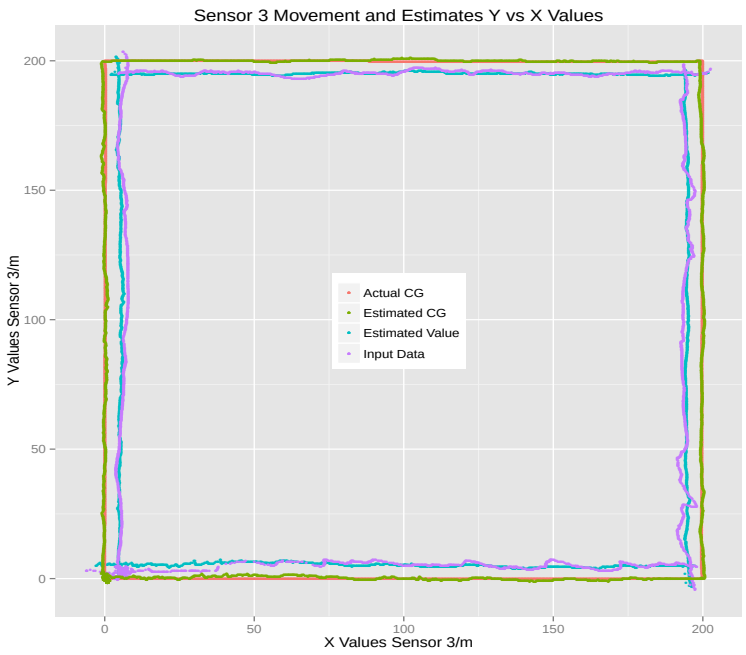


(a) Sensor 1.

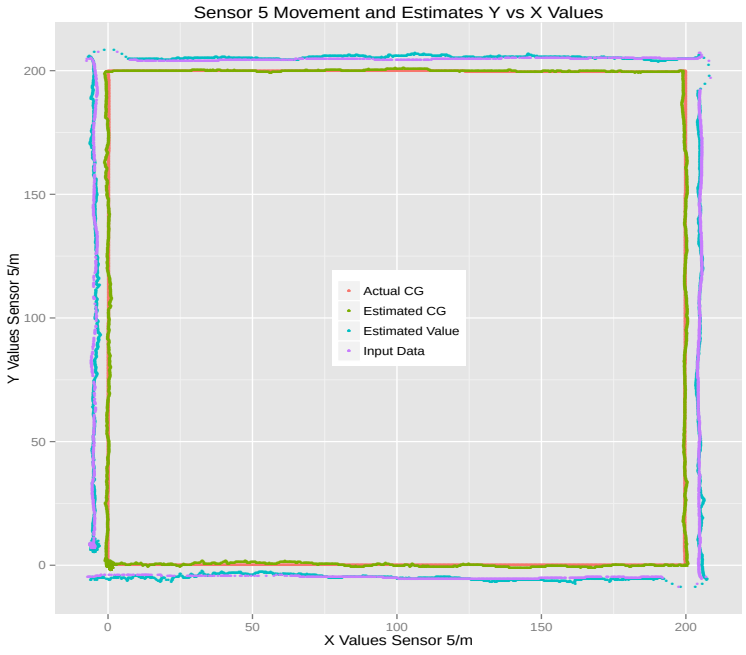


(b) Sensor 2.

Figure 4.58: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8.

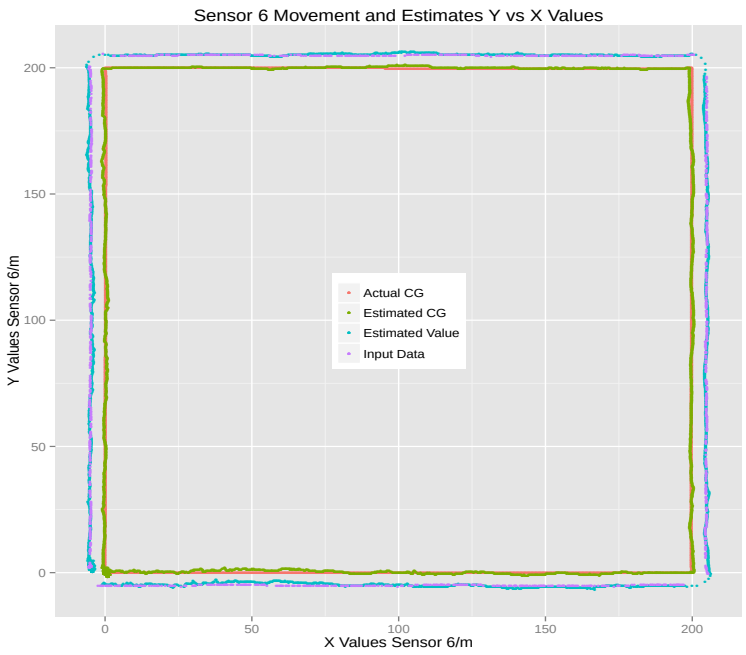


(a) Sensor 3.

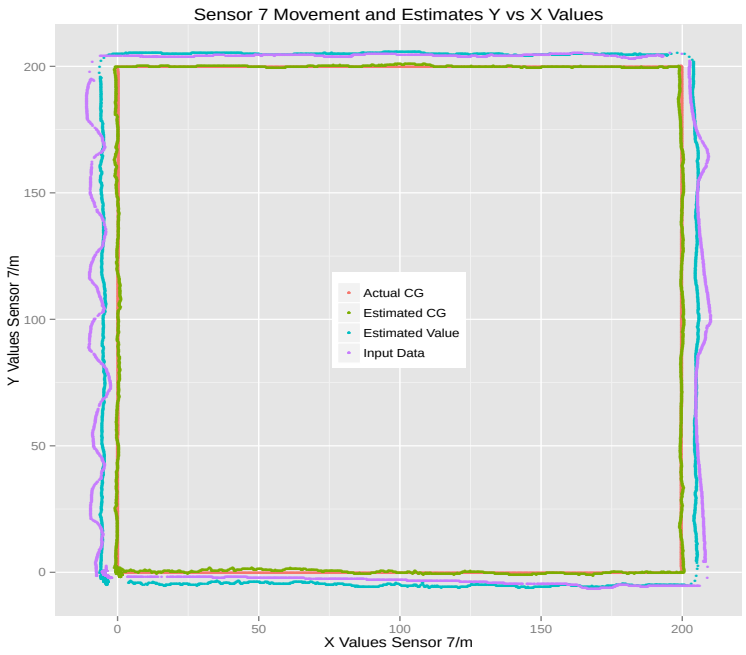


(b) Sensor 5.

Figure 4.59: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

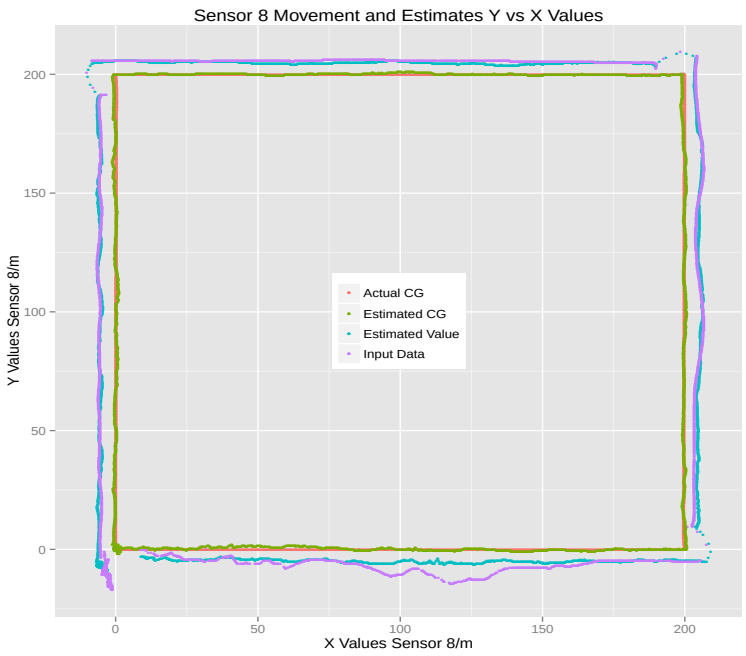


(a) Sensor 6.



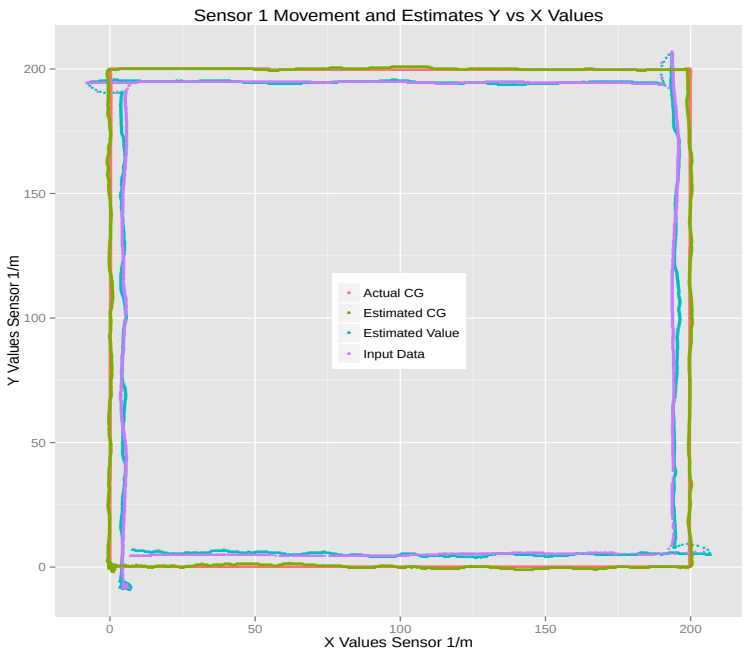
(b) Sensor 7.

Figure 4.60: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

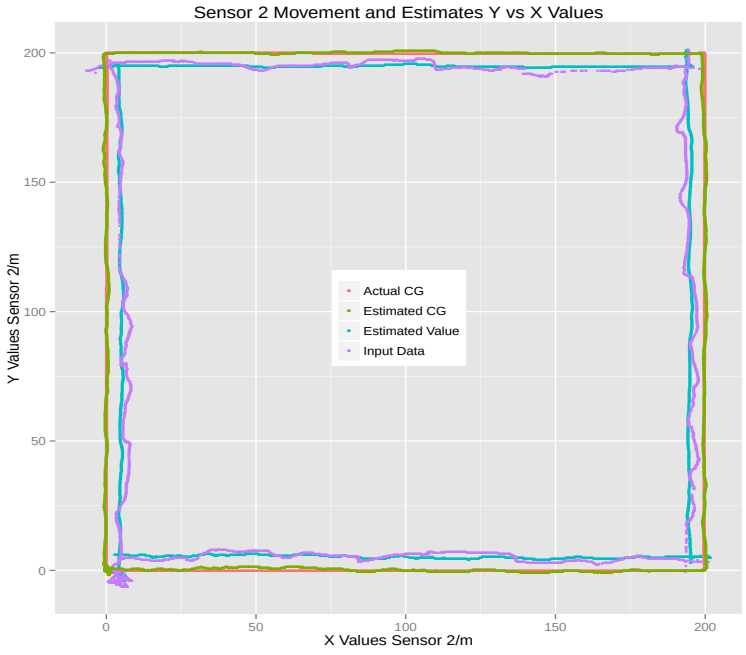


(a) Sensor 8.

Figure 4.61: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

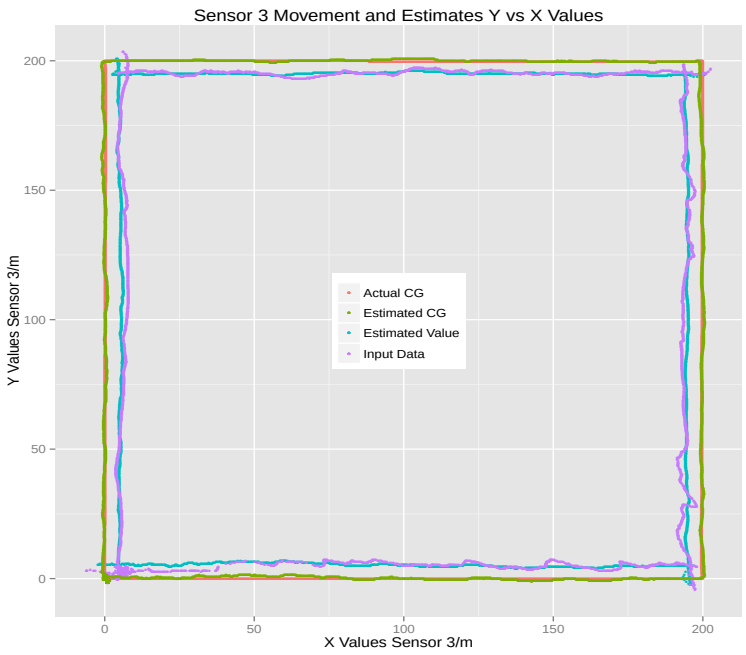


(a) Sensor 1.

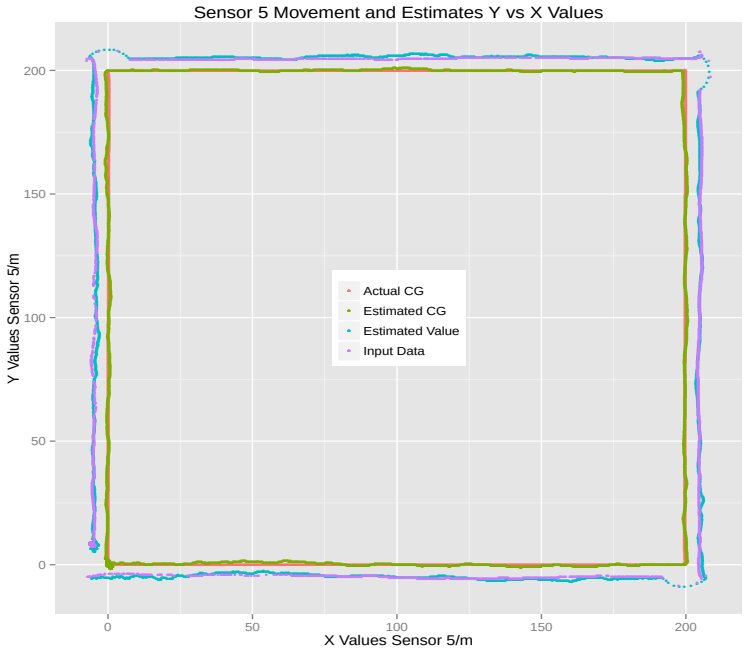


(b) Sensor 2.

Figure 4.62: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8.

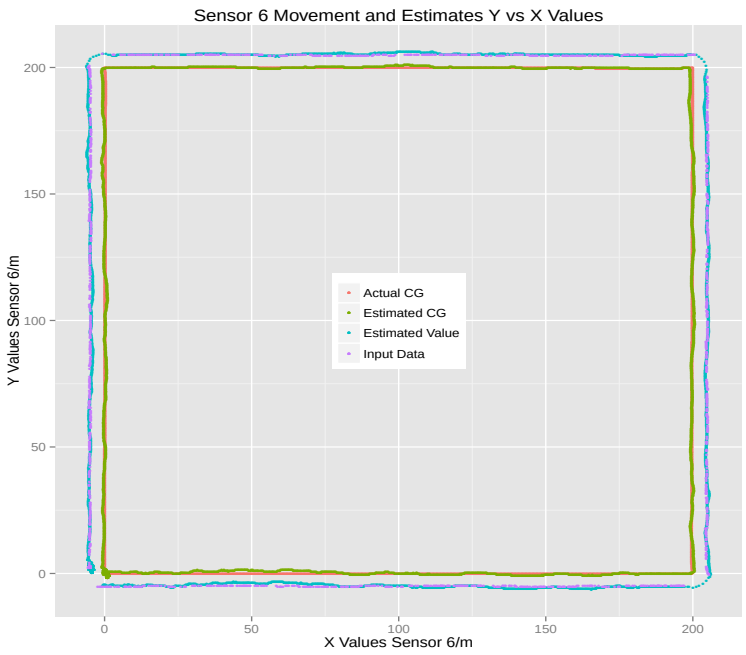


(a) Sensor 3.

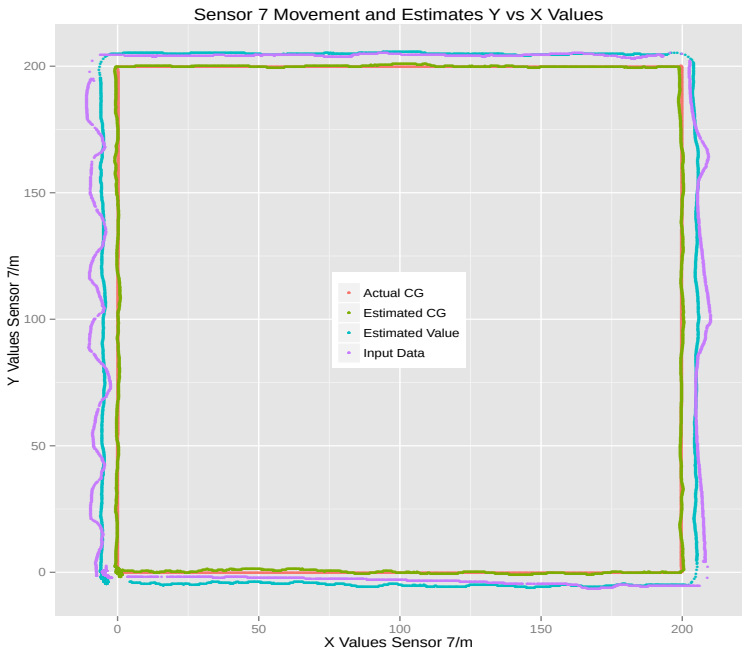


(b) Sensor 5.

Figure 4.63: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

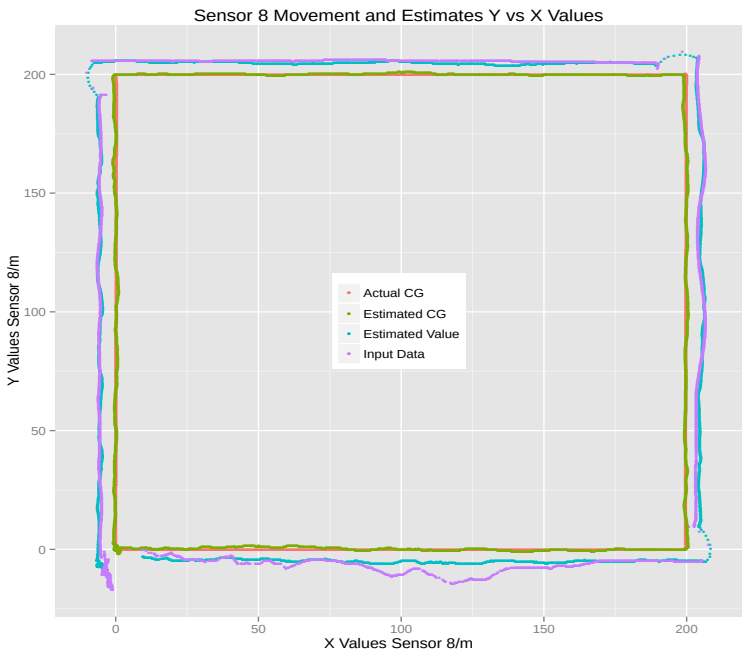


(a) Sensor 6.



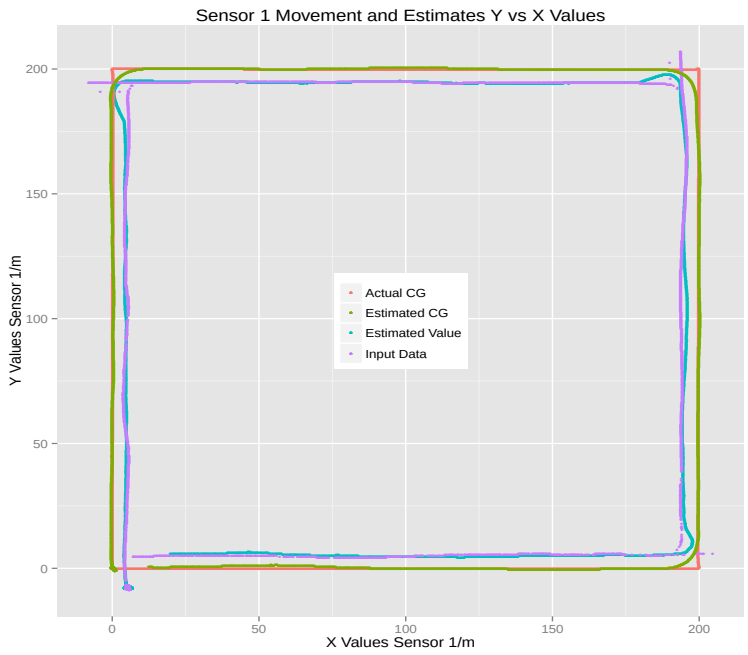
(b) Sensor 7.

Figure 4.64: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

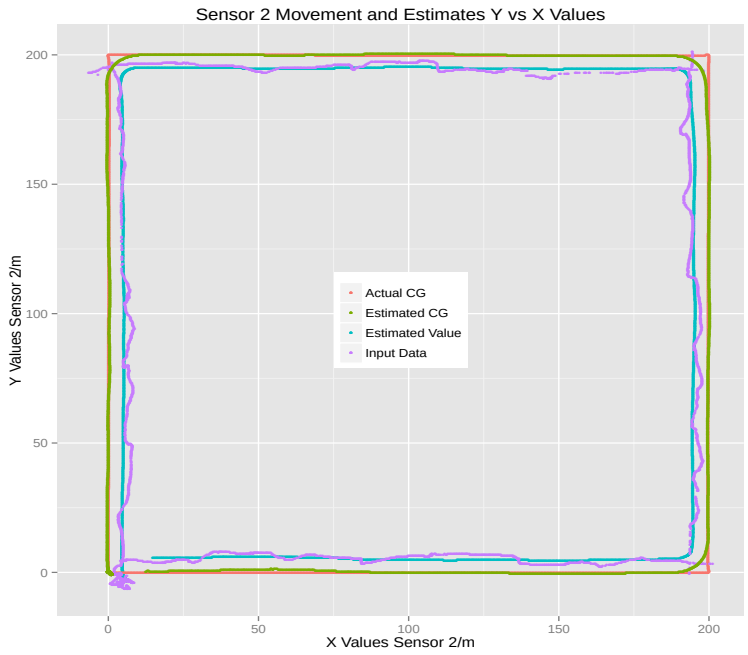


(a) Sensor 8.

Figure 4.65: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

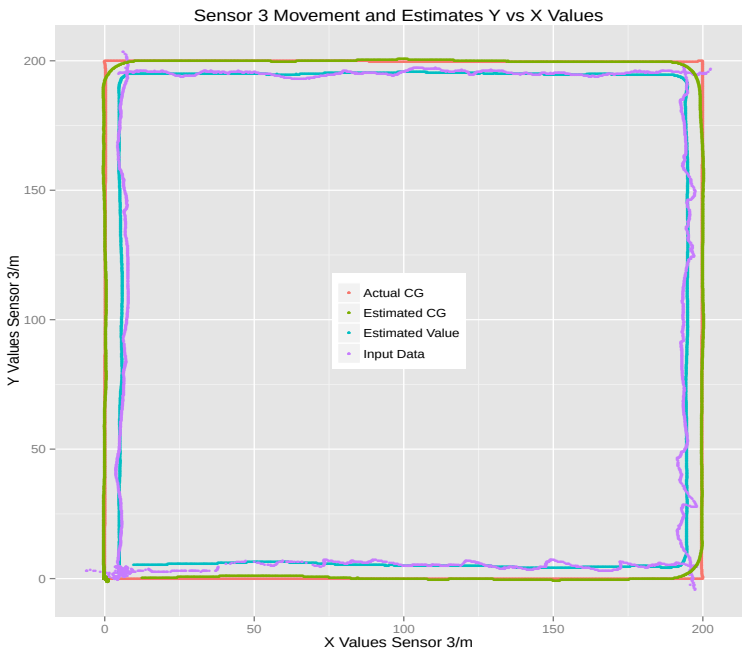


(a) Sensor 1.

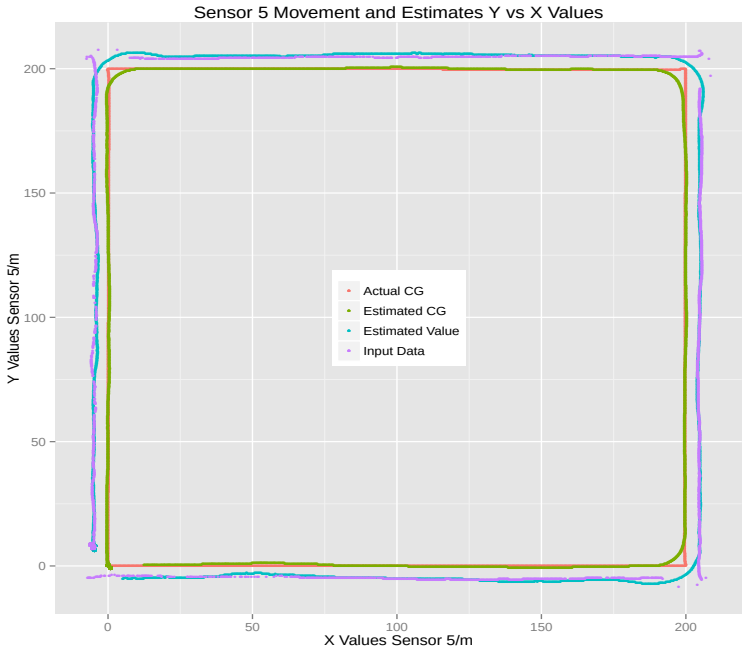


(b) Sensor 2.

Figure 4.66: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8.

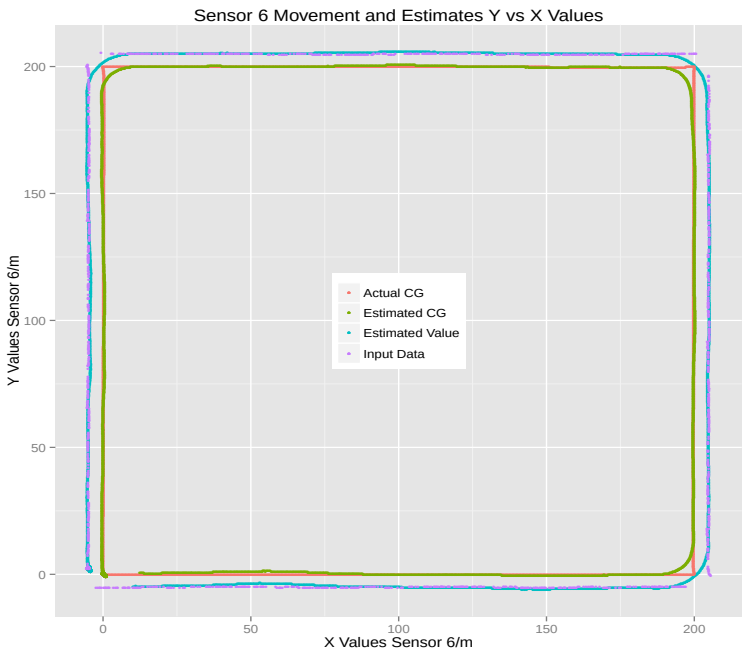


(a) Sensor 3.

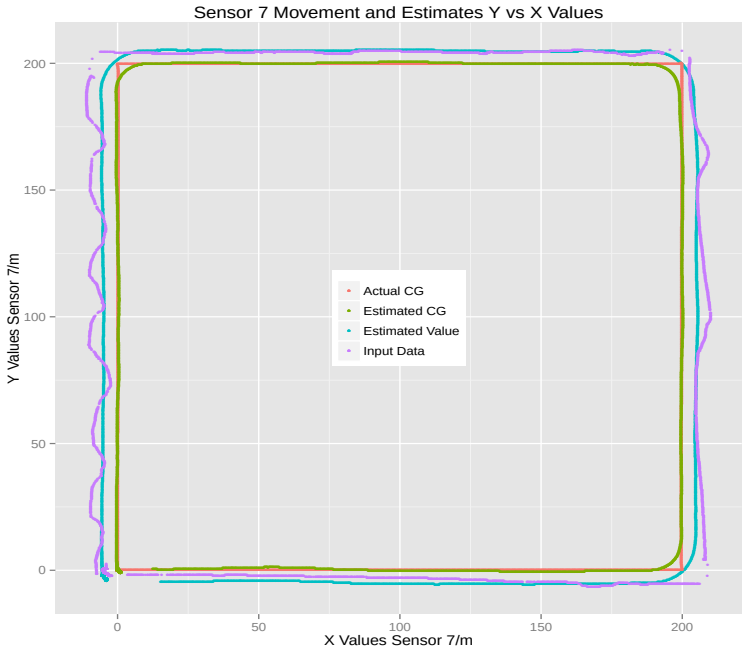


(b) Sensor 5.

Figure 4.67: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

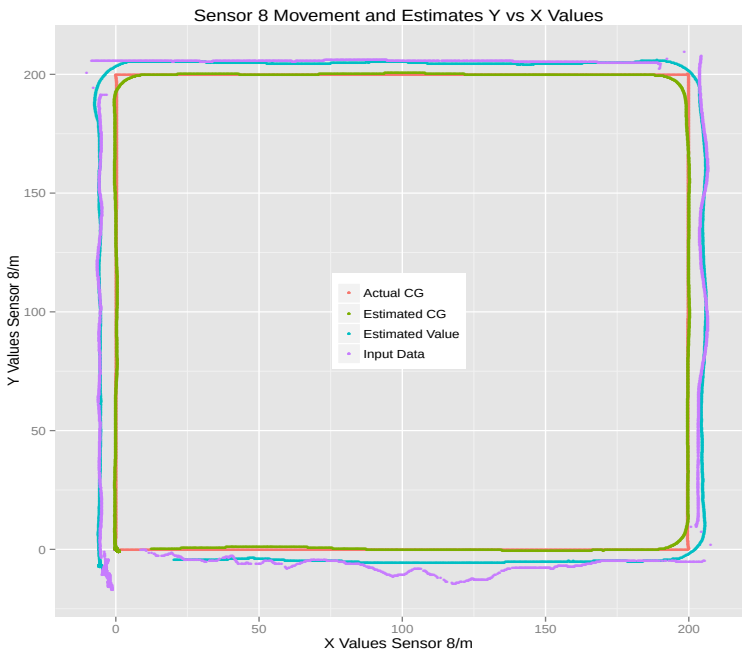


(a) Sensor 6.



(b) Sensor 7.

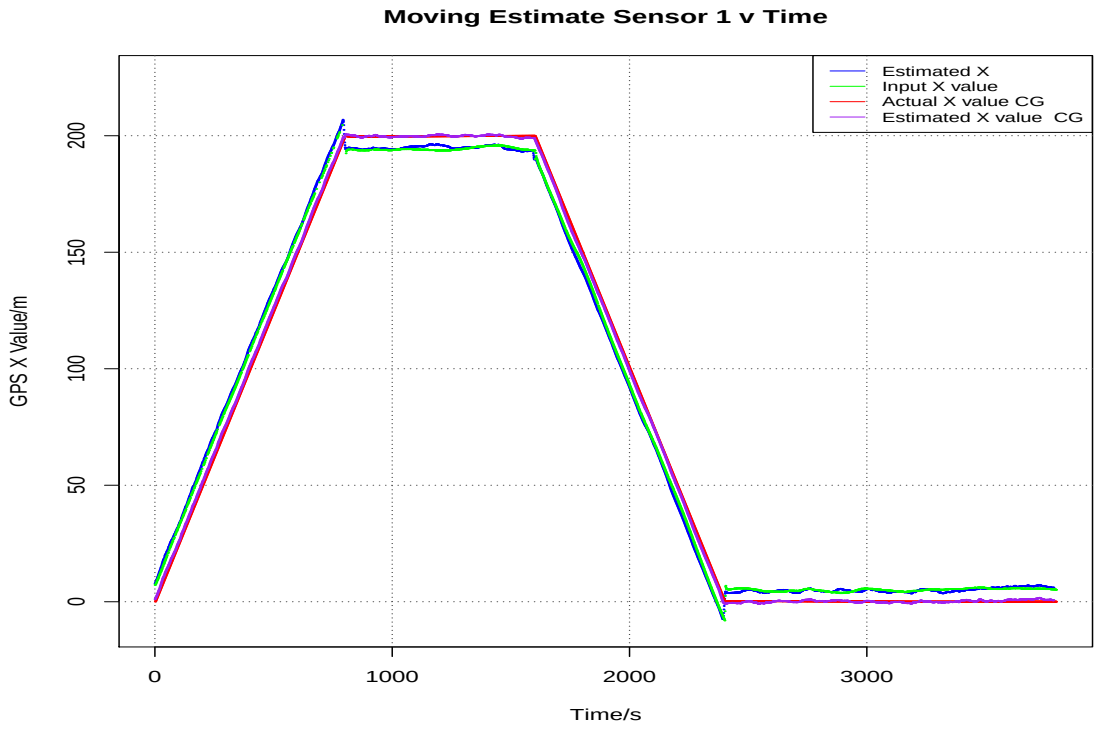
Figure 4.68: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).



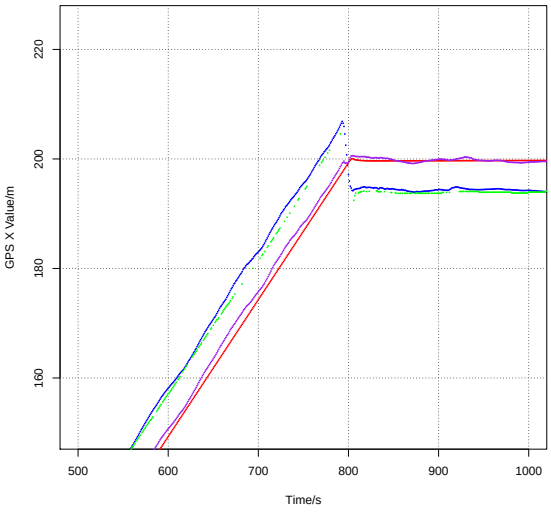
(a) Sensor 8.

Figure 4.69: Location Estimates Using Quadratic Optimization with No Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

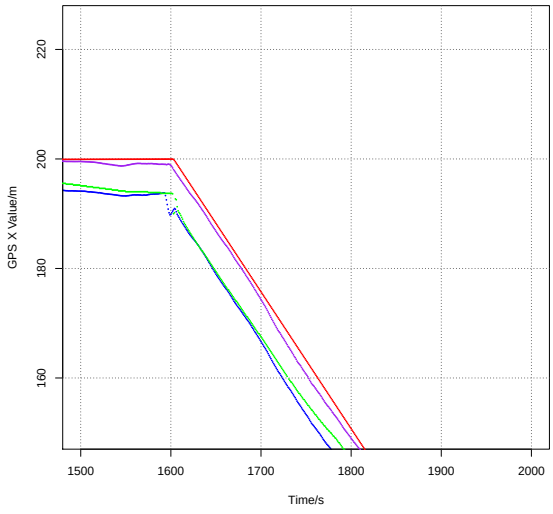
From the plots, we can observe that as the sliding window size increases, the variation from one estimate to the next is greatly reduced. However we cannot indiscriminately increase the sliding window size as the lag effect becomes more pronounced as seen for Sensor 1 in sliding windows 10 in Figures 4.70, 4.71 and sliding window = 100 in Figures 4.72 and 4.73. As in the situation with Path 1, for Path 2, it is clear that the location estimates calculated through quadratic optimization using the configuration of the sensors as constraints for a large sliding window seem to fall behind and attempt to "catch up" to the true location values. These results are similar to those calculated in Path 1 except that the lag effects are more pronounced.



(a) Sensor 1.

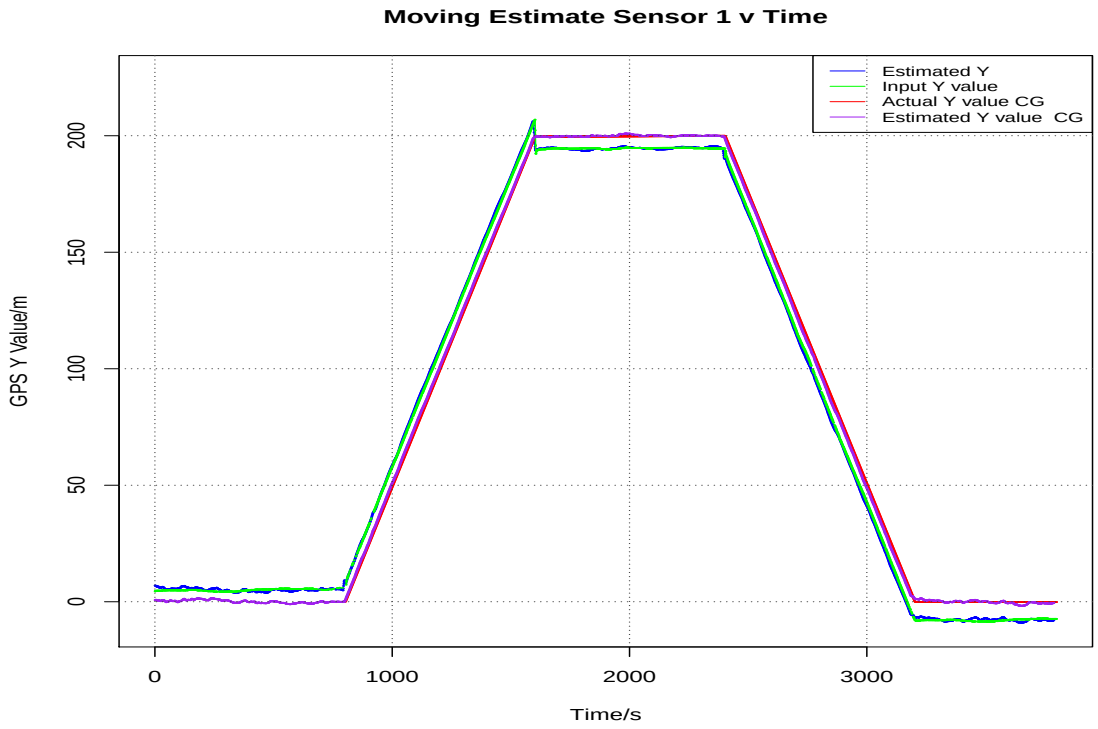


(b) Sensor 1 Positive Slope Magnified.

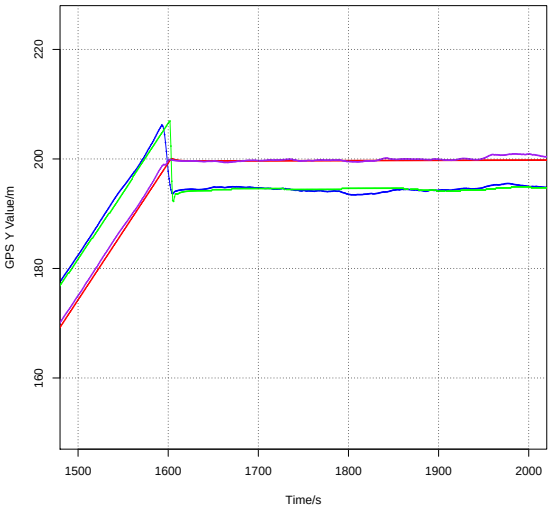


(c) Sensor 1 Negative Slope Magnified.

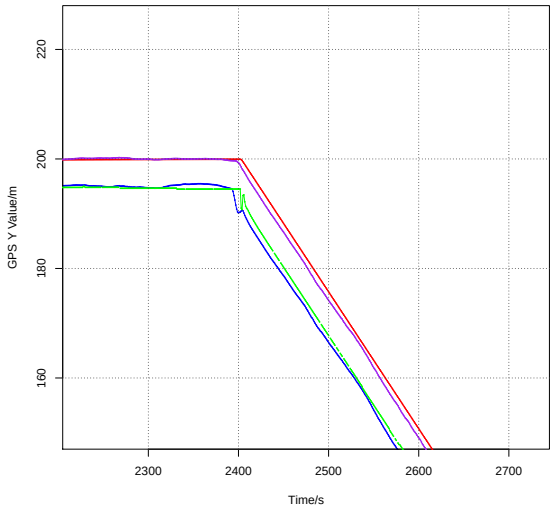
Figure 4.70: Location Estimates of Sensor 1 UTM X Coordinates vs. Time No Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

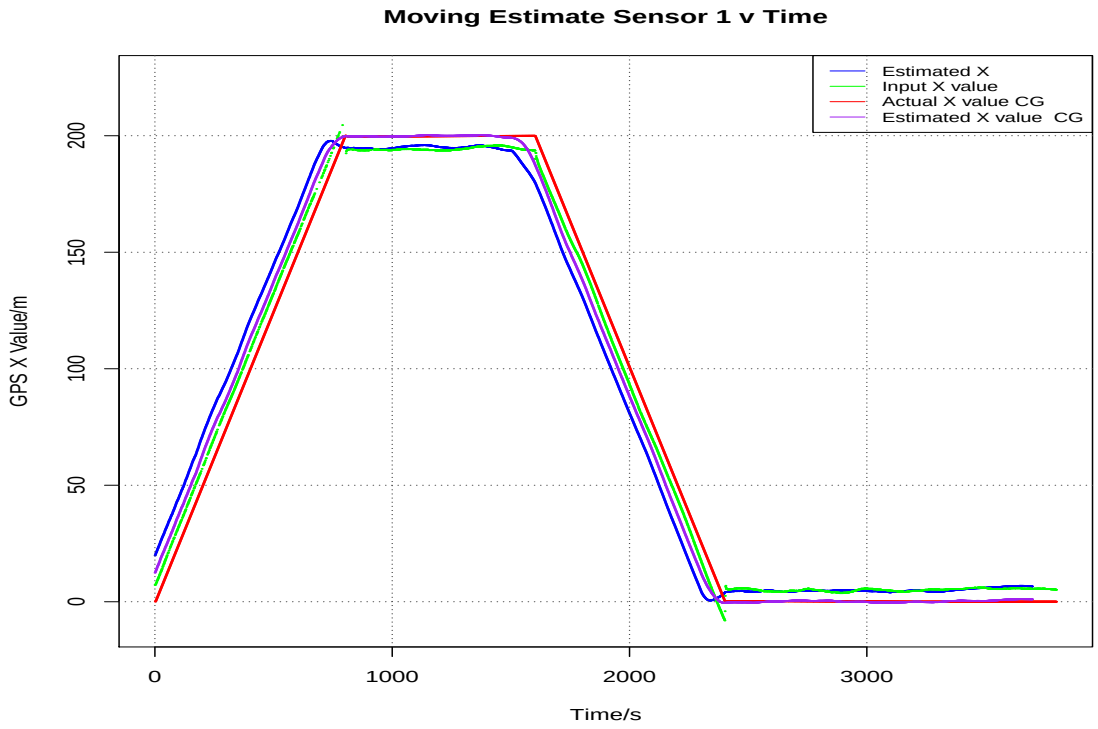


(b) Sensor 1 Positive Slope Magnified.

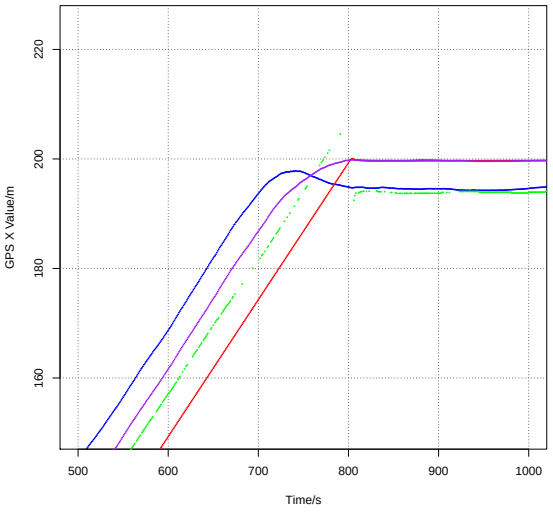


(c) Sensor 1 Negative Slope Magnified.

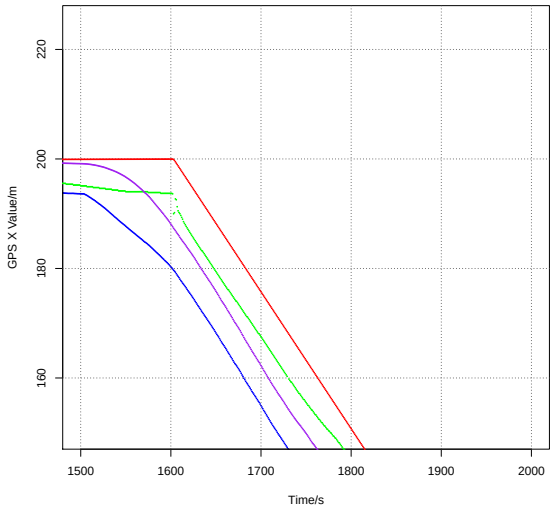
Figure 4.71: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time No Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

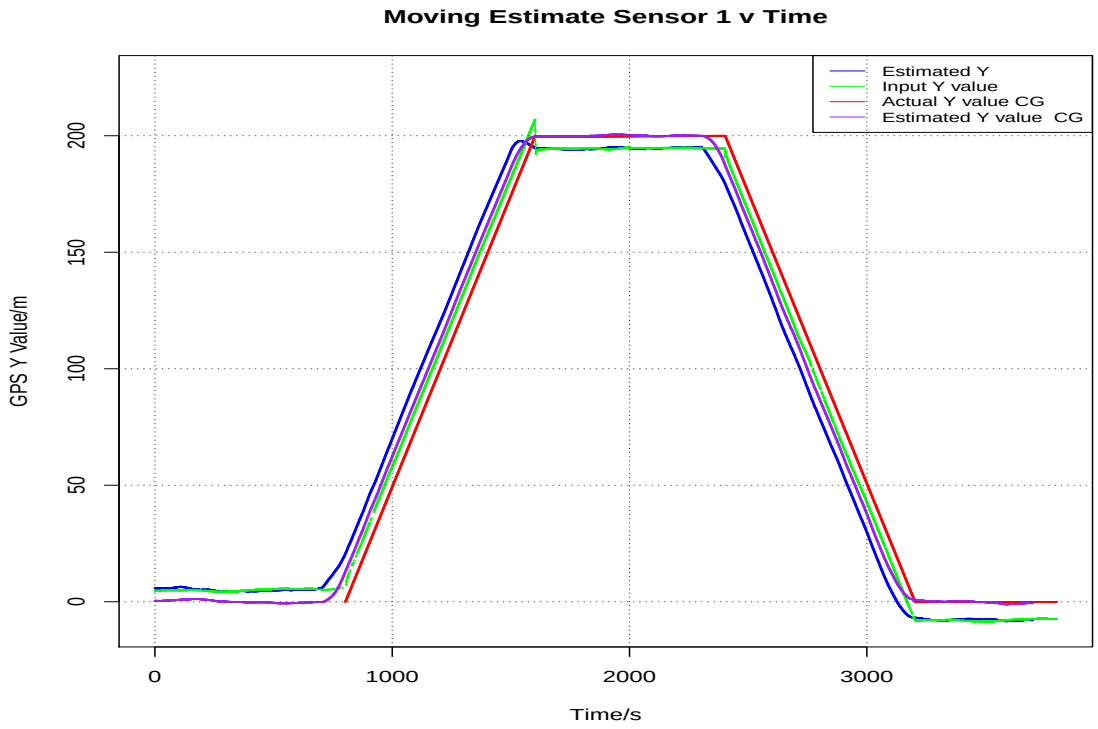


(b) Sensor 1 Positive Slope Magnified.

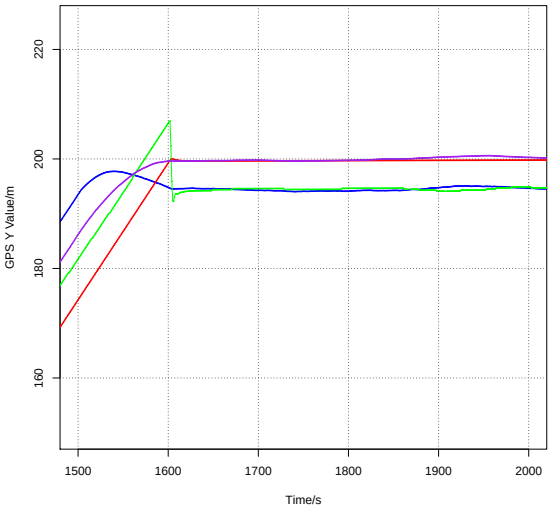


(c) Sensor 1 Negative Slope Magnified.

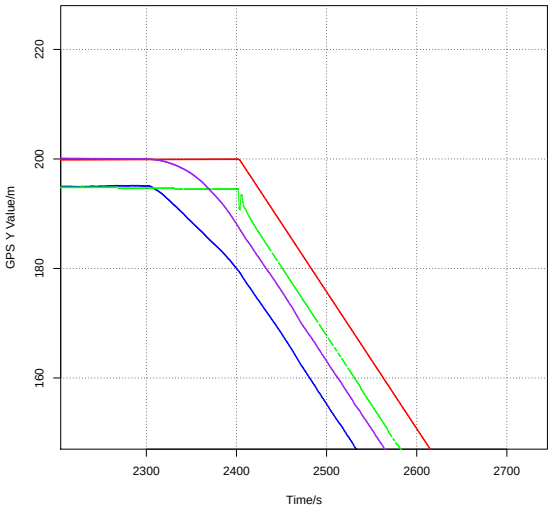
Figure 4.72: Location Estimates of Sensor 1 UTM X Coordinates vs. Time No Weights Sliding Window = 100 $(+veX - North, +veY - East)$.



(a) Sensor 1.



(b) Sensor 1 Positive Slope Magnified.



(c) Sensor 1 Negative Slope Magnified.

Figure 4.73: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time No Weights Sliding Window = 100 $(+veX - North, +veY - East)$.

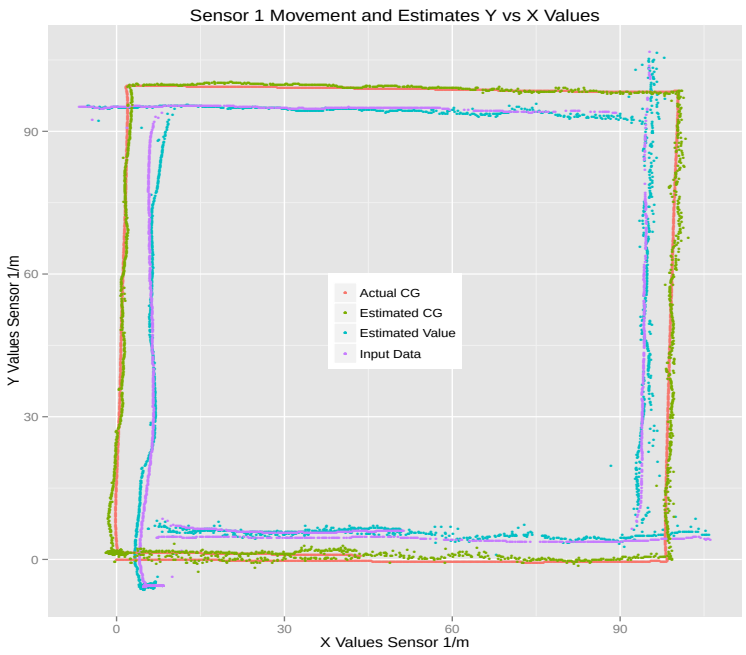
4.11 Weighted Quadratic Optimization Estimates of Locations of GPS Sensors for Moving Platform using Sliding Windows

As discussed in prior sections, the use of weighting functions is for the purpose of reducing the effect of missing and unreliable observed data. To calculate the estimates of the locations of the sensors when the platform is in motion along with outlier analysis and use of a weighting function, we performed tests using the Huff weighting function. The Huff weighting function describes a method of calculating weights depending on the presence of the data point in that particular time step and deviation from the theoretical distances between the sensors in Section 3.3.3. We have investigated the effect of Huff weighting on obtaining estimates of GPS sensor location using Path 1 and Path 2 (in Section 4.9).

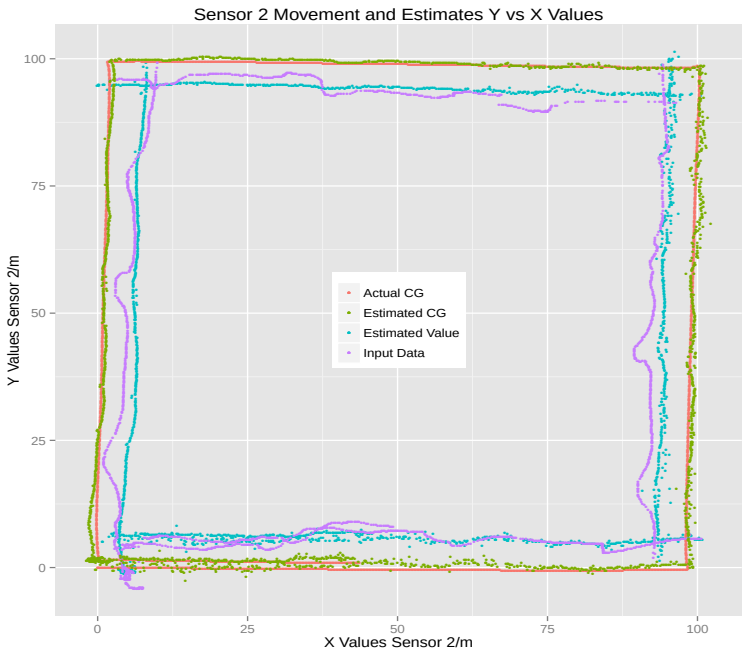
4.11.1 Location Estimates Using Path 1

For Path 1, Figures 4.74, 4.75, 4.76,4.77 for sliding window = 1; Figures 4.78, 4.79, 4.80, 4.81 for sliding window = 5; Figures 4.82, 4.83, 4.84, 4.85 for sliding window = 10; and Figures 4.86, 4.87, 4.88, 4.89 for sliding window = 100 show the results of calculating the location estimates for each of the sensors that reported values from Sensor 1,2,3,5,6,7 and 8 along with the combination of sliding window values of 1,5,10 and 100. As in the unweighted cases in the previous sections, the red line represents the actual center of gravity which is the most accurate path that the platform can take, with the blue line presenting the estimate of the sensor's location, the purple line is observed data generated by the simulation and finally, the green plot, which represents the estimated center of gravity derived from the location estimates through quadratic optimization. As with the unweighted cases of obtaining location estimates, the red and green plots seem to coincide very closely with one another, and as the sliding window size increases, a smoothing effect takes place reducing the magnitude

of noise. Even with the utilization of the Huff weight function, the issue of lag for the higher sliding window sizes still exists and will be accounted for in future sections as seen for sliding window = 10 in Figures 4.90 , 4.91 compared to sliding window = 100 in Figures 4.92,4.93.

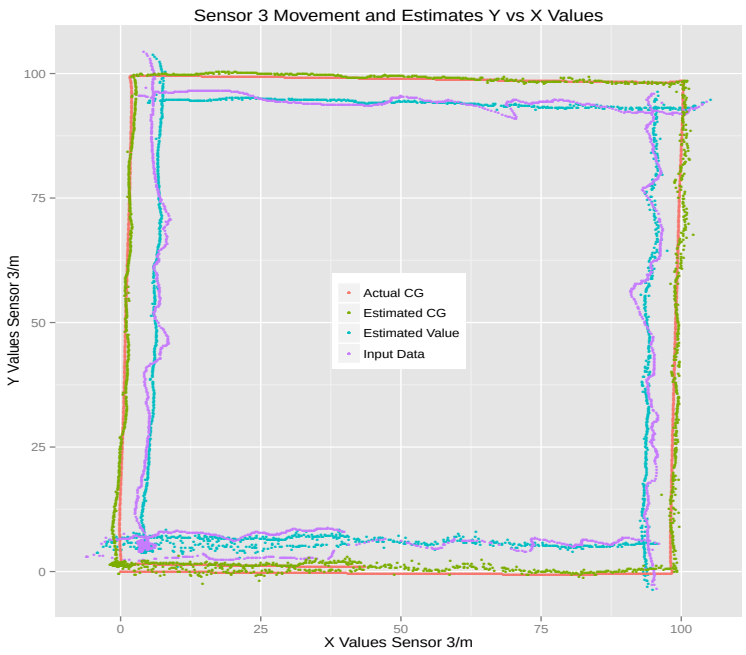


(a) Sensor 1.

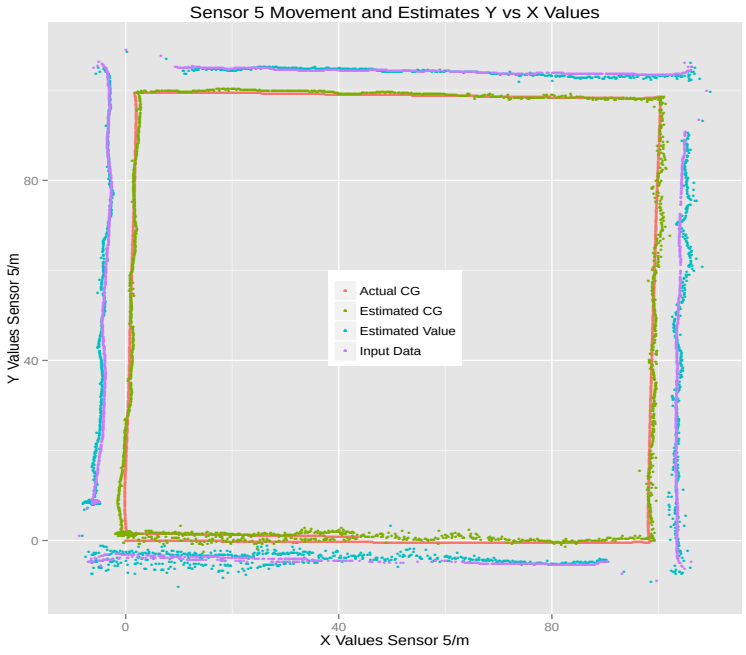


(b) Sensor 2.

Figure 4.74: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8.

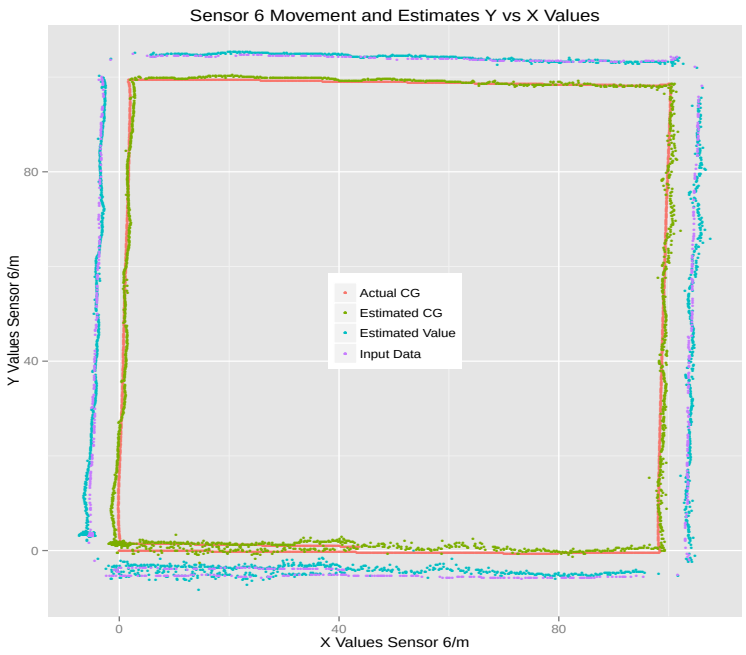


(a) Sensor 3.

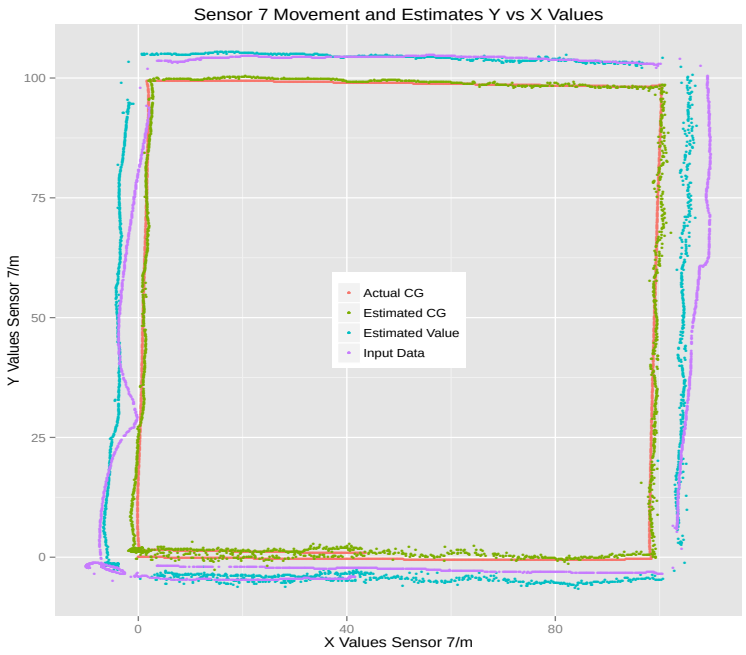


(b) Sensor 5.

Figure 4.75: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

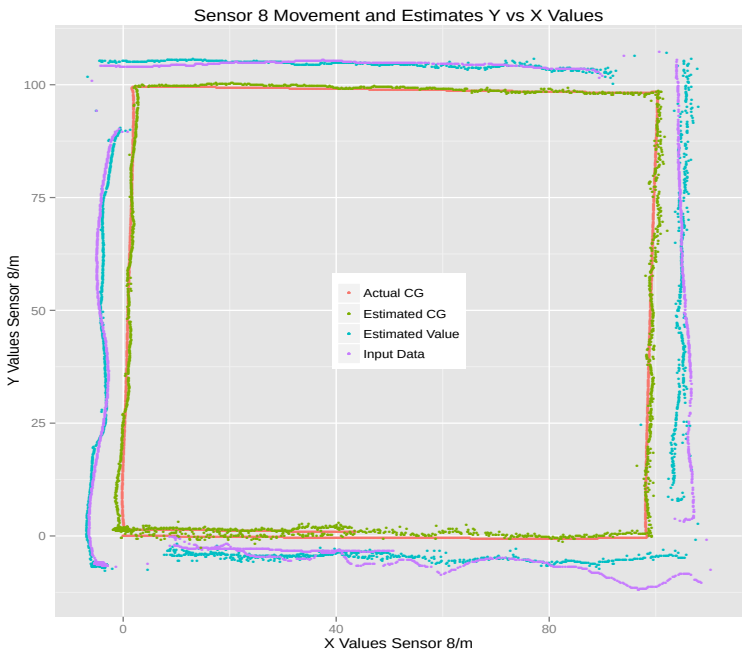


(a) Sensor 6.



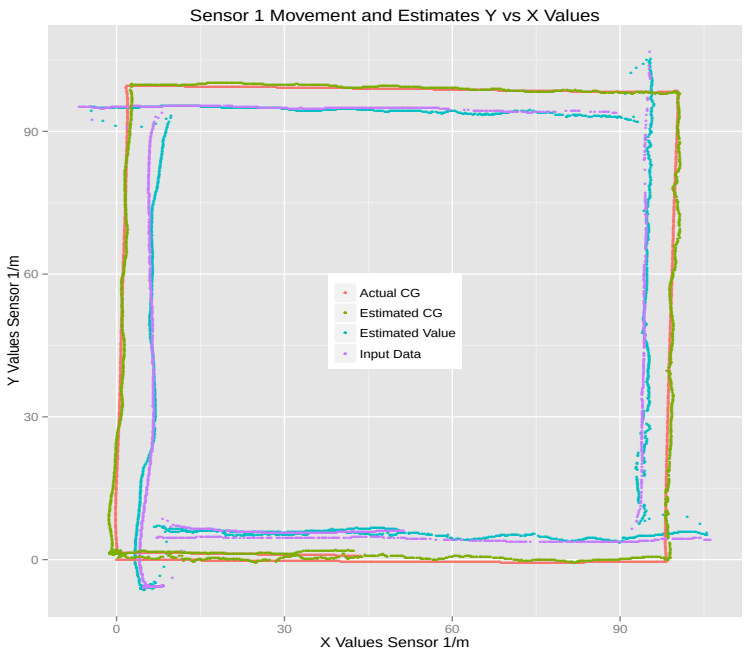
(b) Sensor 7.

Figure 4.76: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

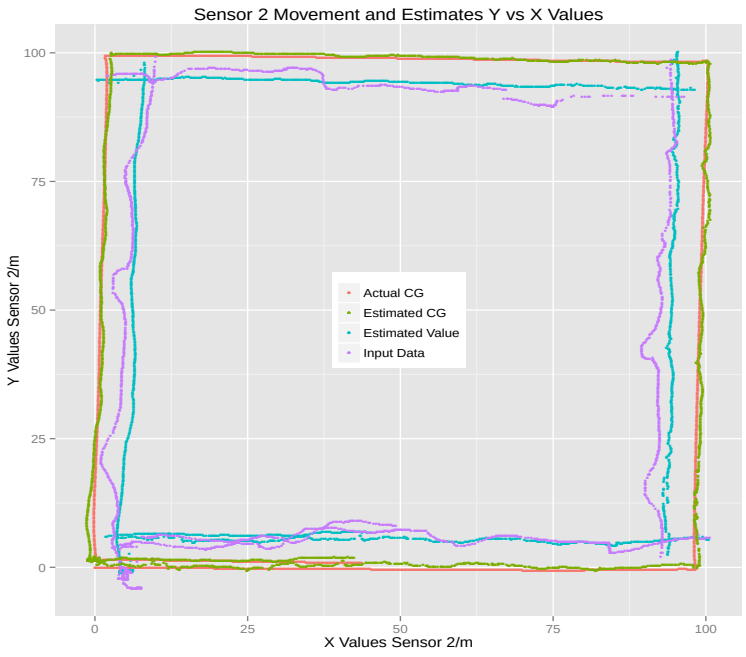


(a) Sensor 8.

Figure 4.77: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

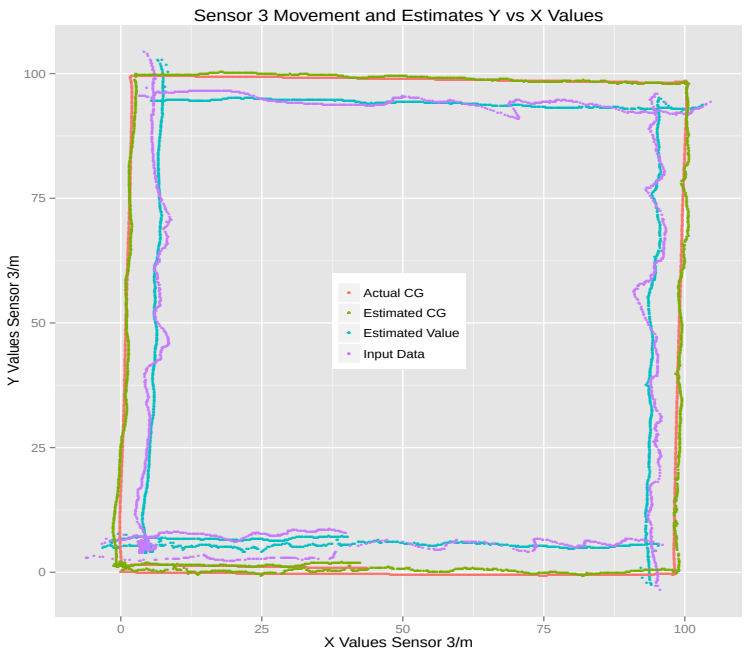


(a) Sensor 1.

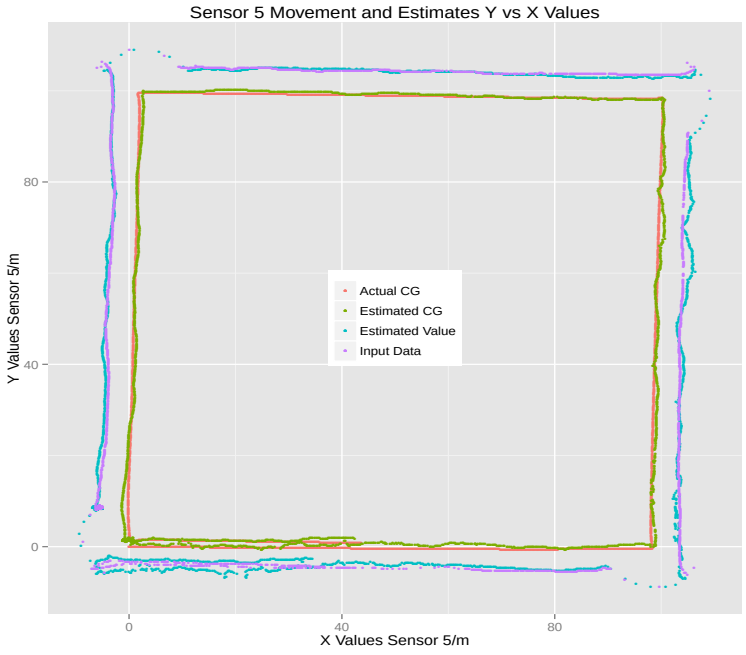


(b) Sensor 2.

Figure 4.78: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8.

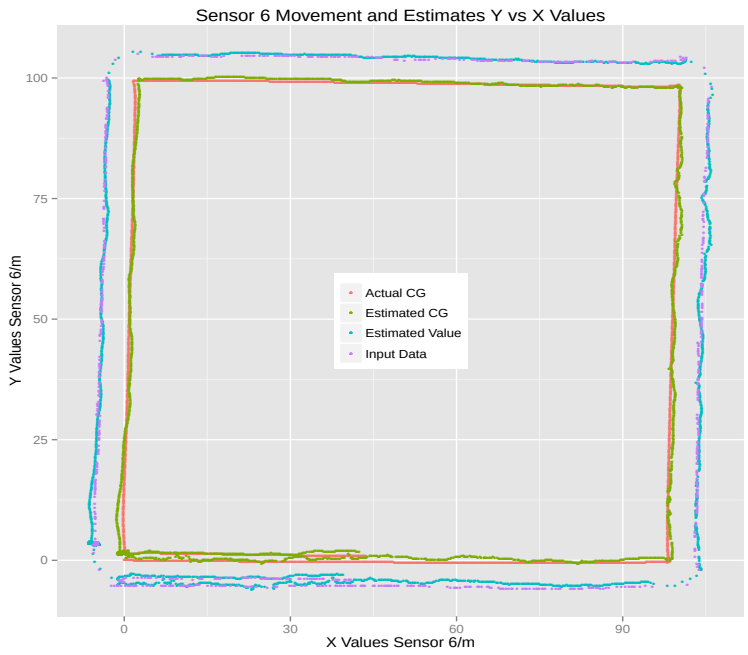


(a) Sensor 3.

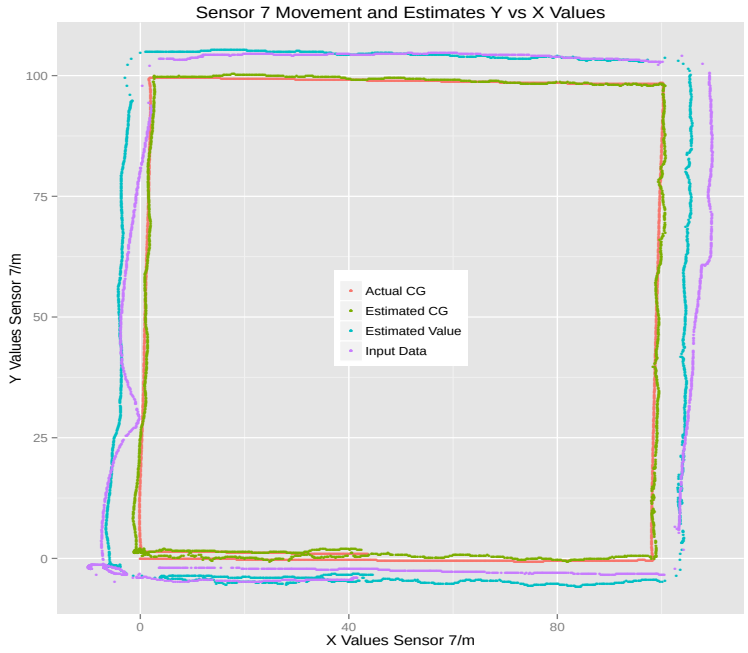


(b) Sensor 5.

Figure 4.79: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

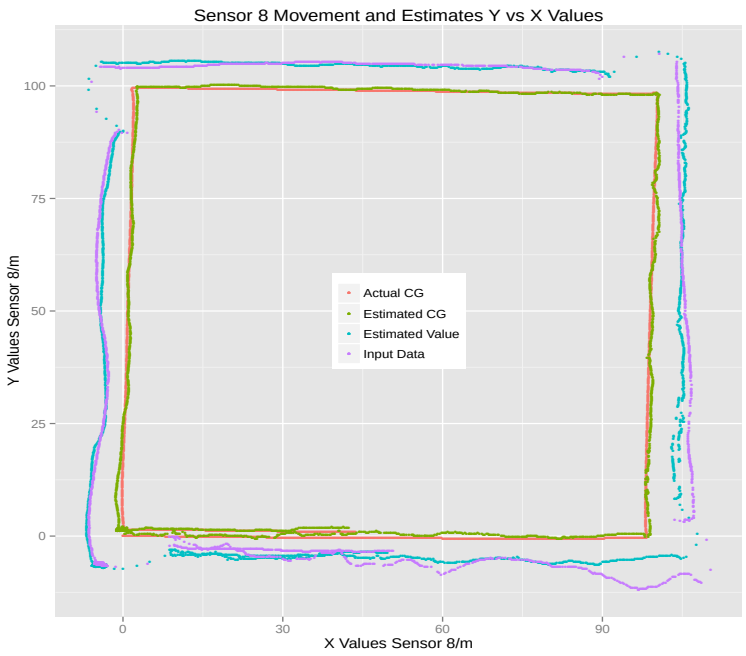


(a) Sensor 6.



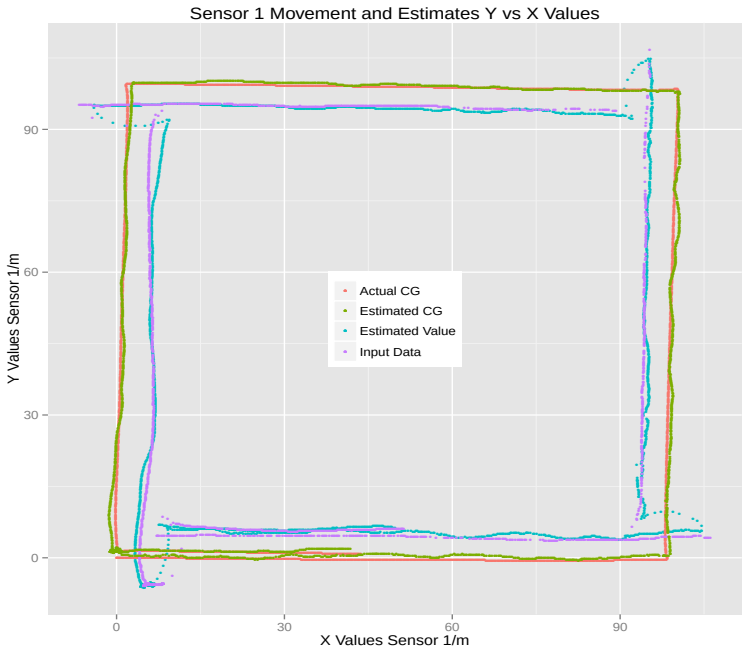
(b) Sensor 7.

Figure 4.80: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

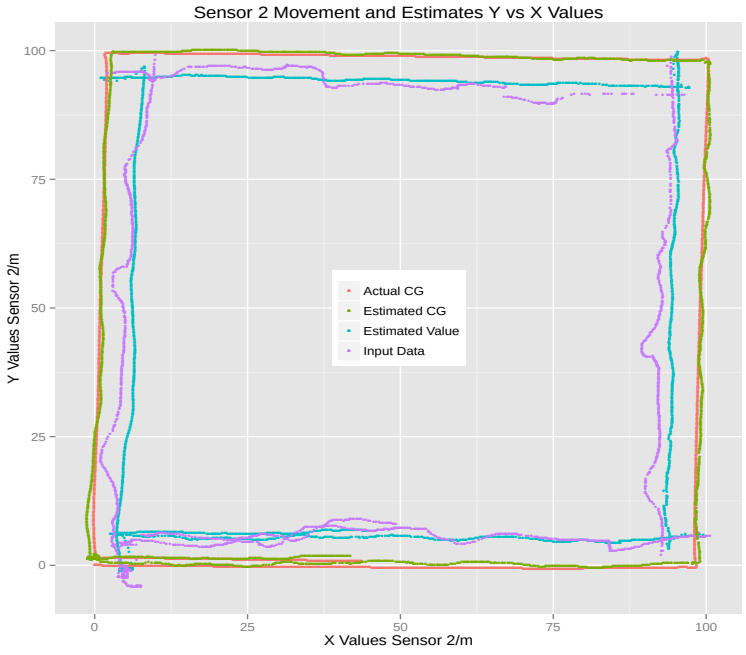


(a) Sensor 8.

Figure 4.81: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

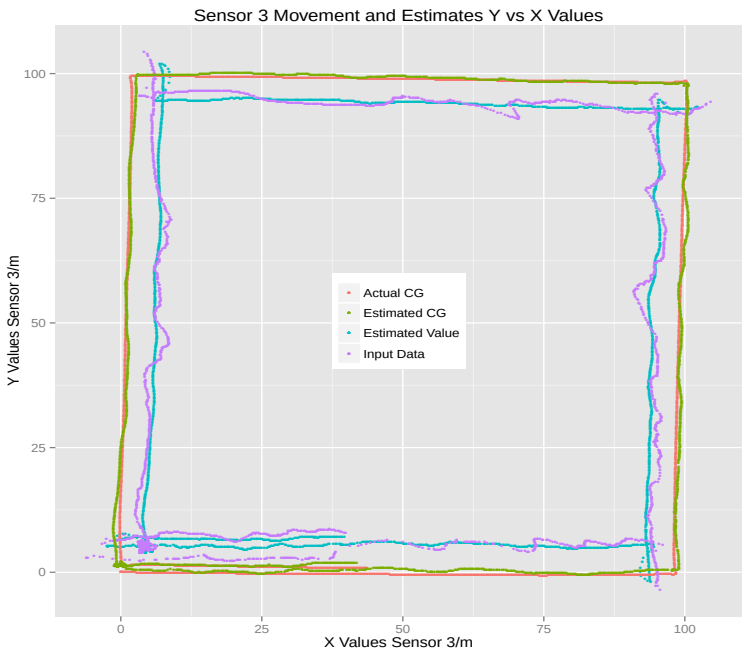


(a) Sensor 1.

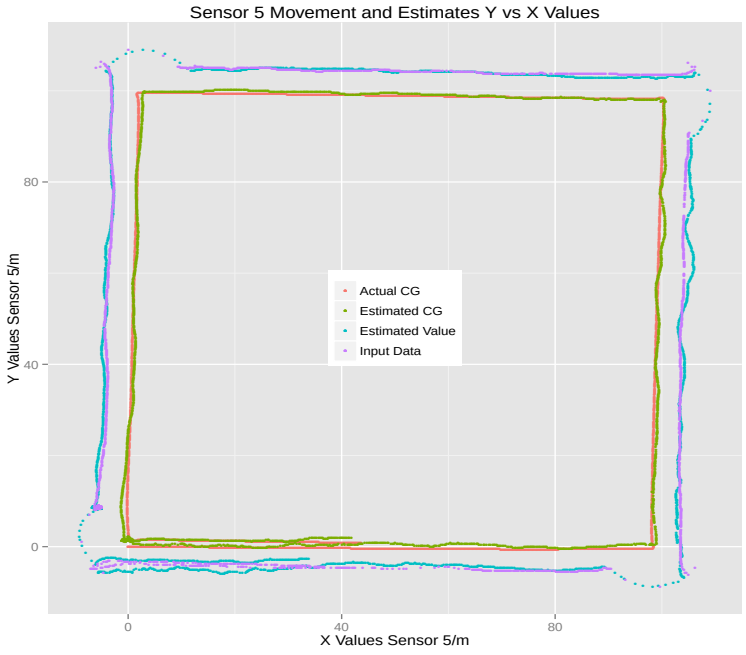


(b) Sensor 2.

Figure 4.82: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8.

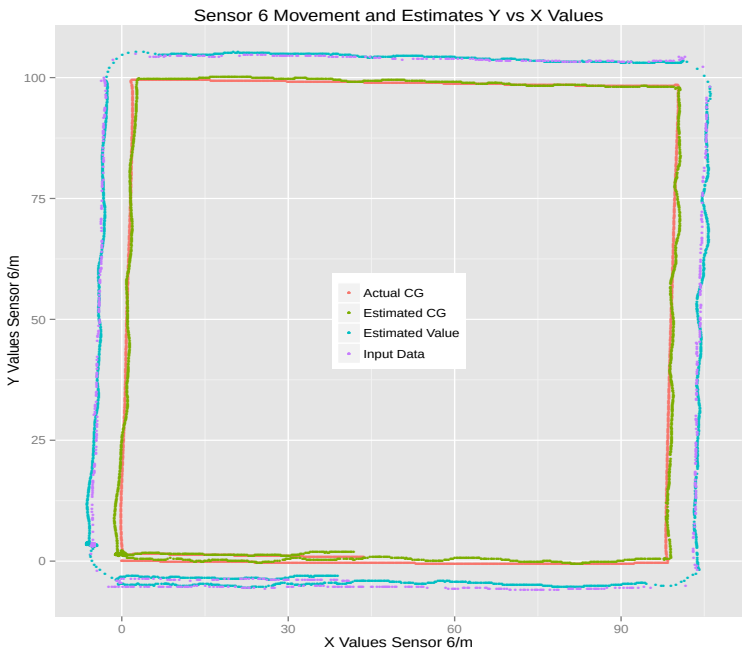


(a) Sensor 3.

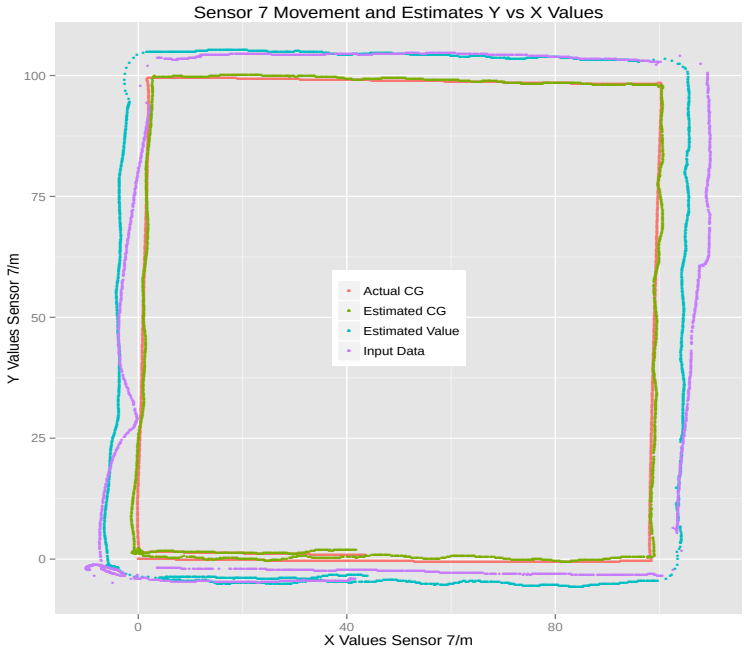


(b) Sensor 5.

Figure 4.83: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

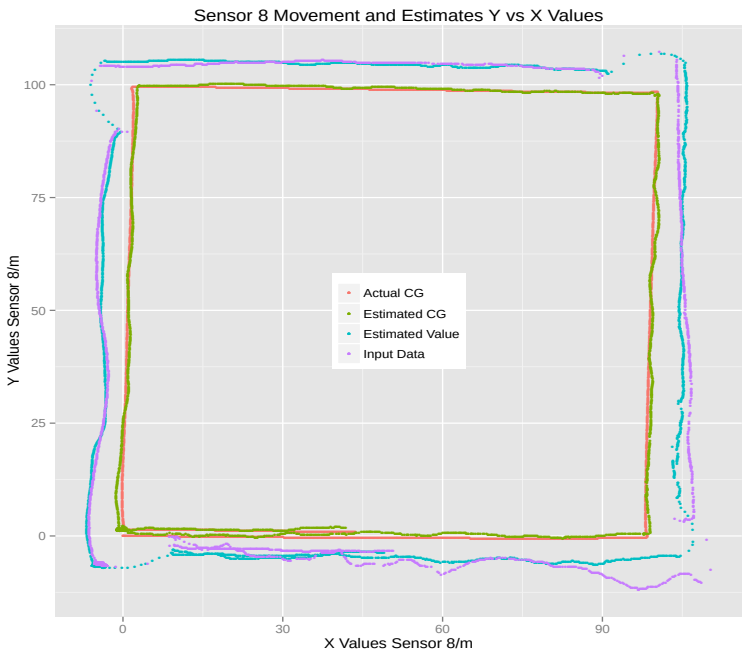


(a) Sensor 6.



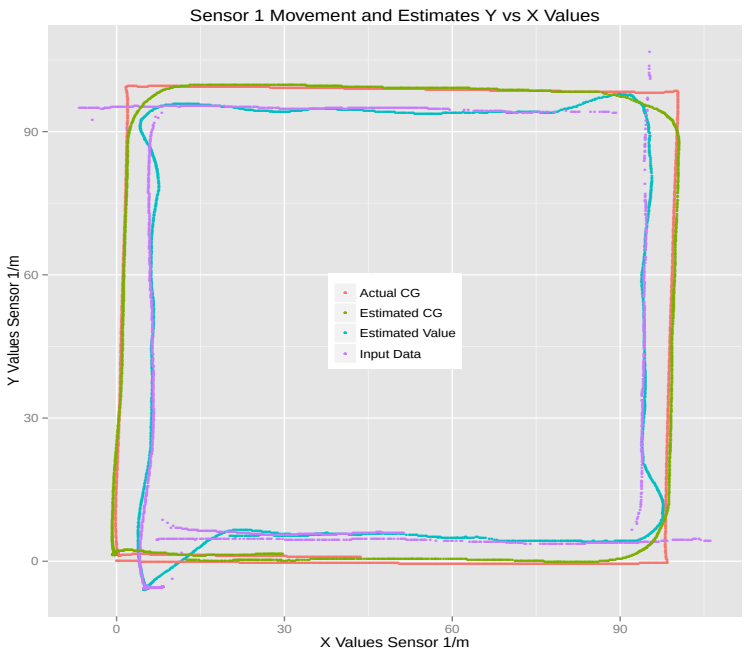
(b) Sensor 7.

Figure 4.84: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

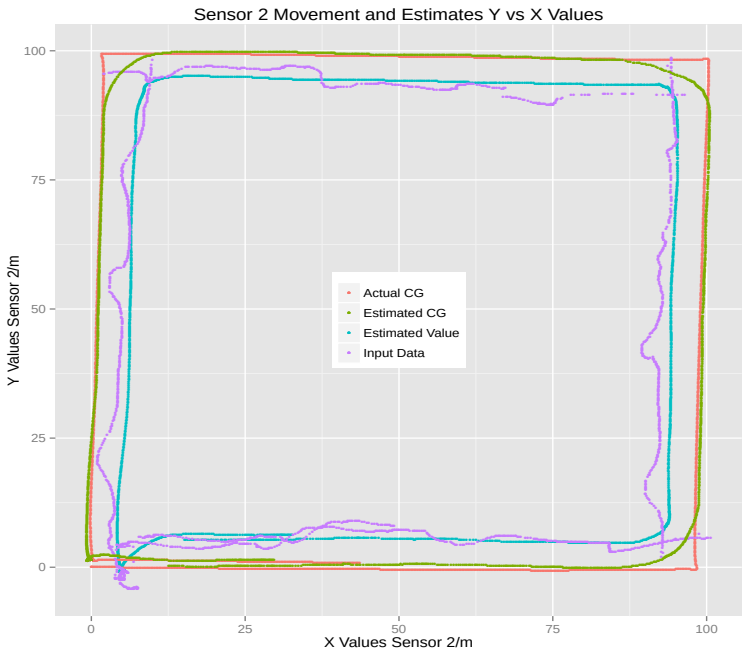


(a) Sensor 8.

Figure 4.85: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.)

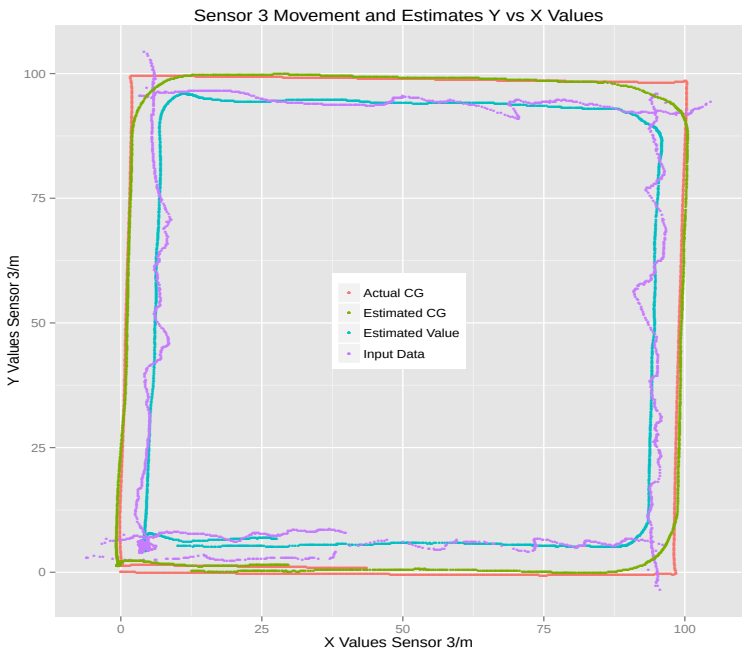


(a) Sensor 1.

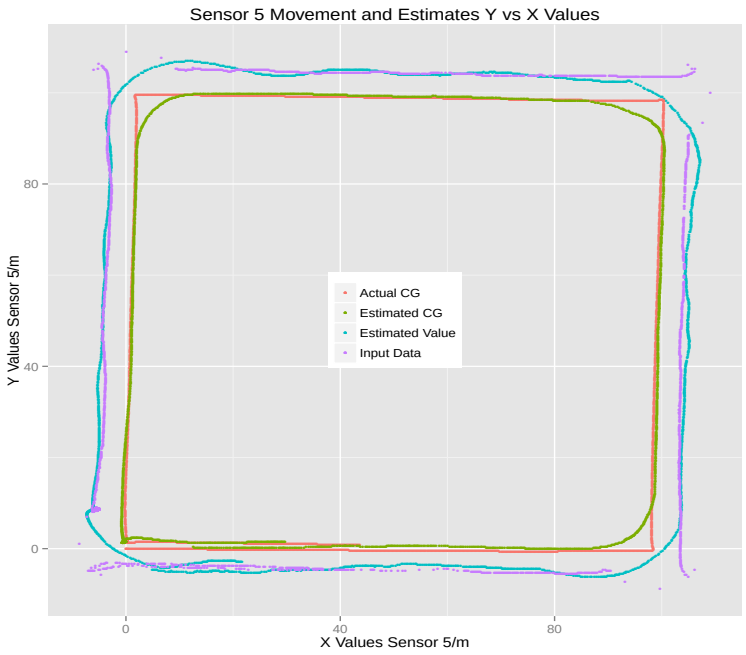


(b) Sensor 2.

Figure 4.86: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8.

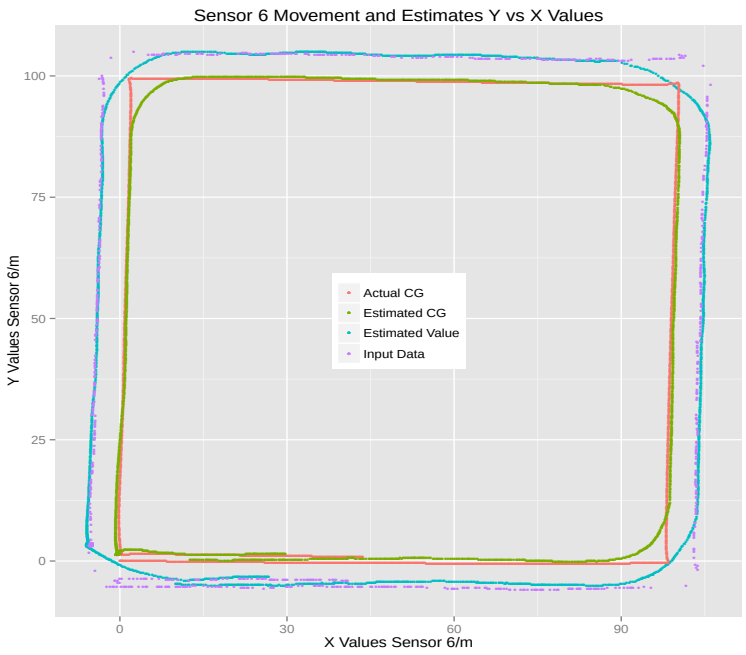


(a) Sensor 3.

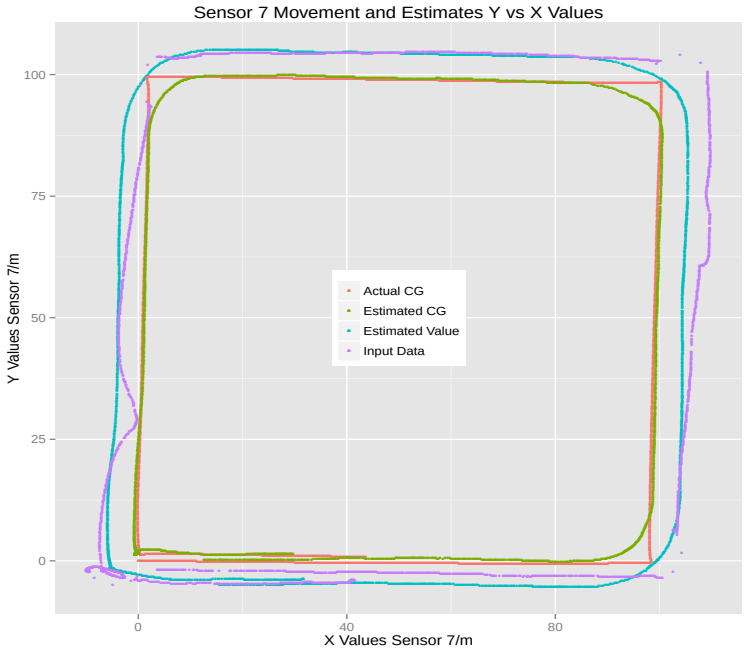


(b) Sensor 5.

Figure 4.87: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

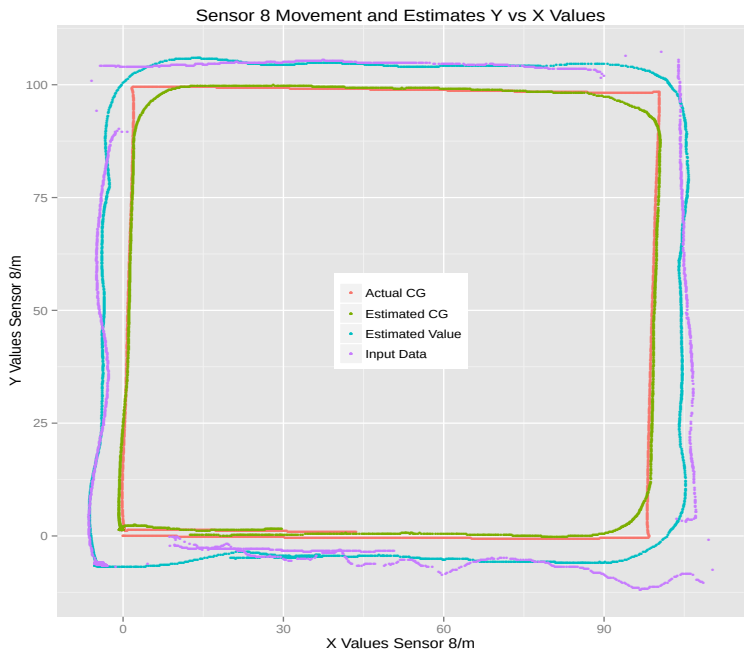


(a) Sensor 6.



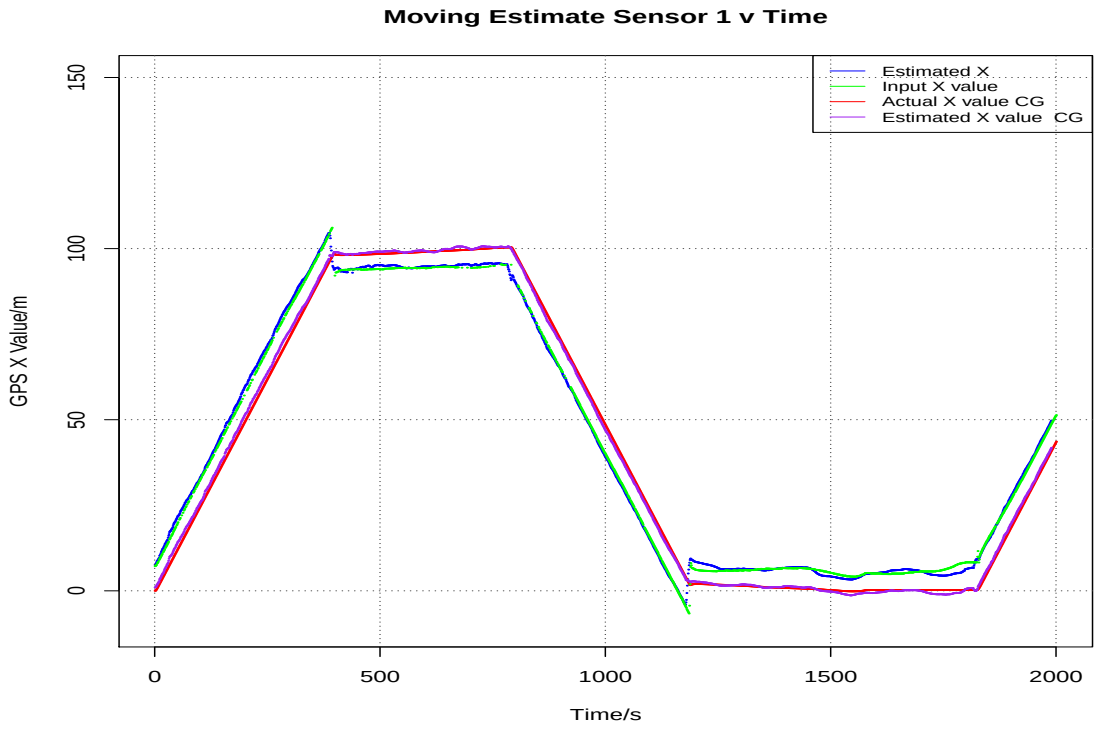
(b) Sensor 7.

Figure 4.88: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

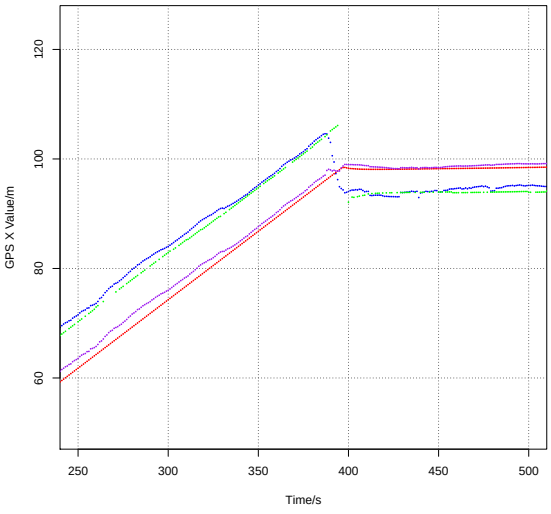


(a) Sensor 8.

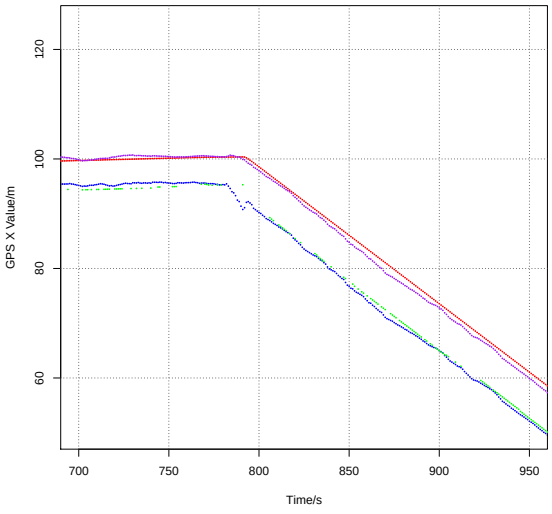
Figure 4.89: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).



(a) Sensor 1.

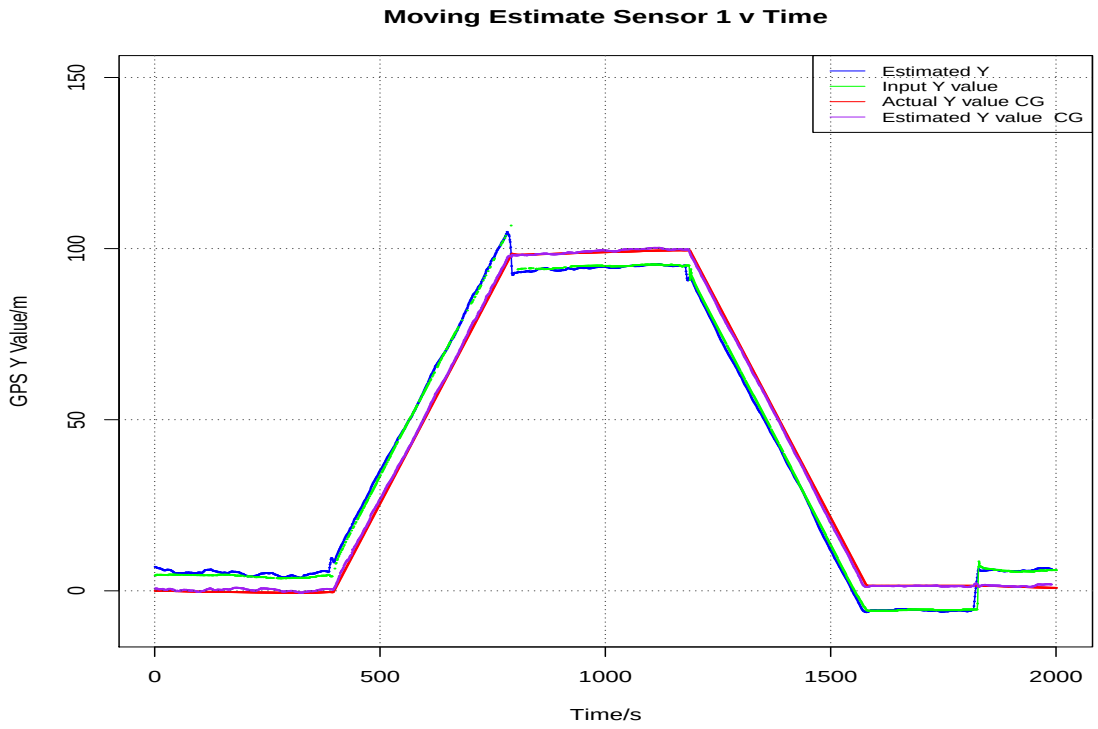


(b) Sensor 1 Positive Slope Magnified.

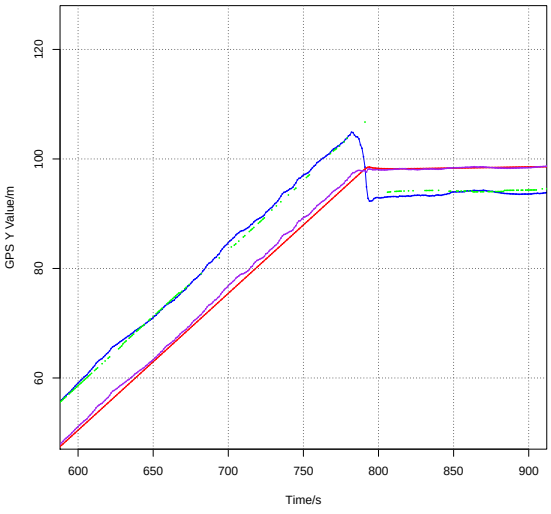


(c) Sensor 1 Negative Slope Magnified.

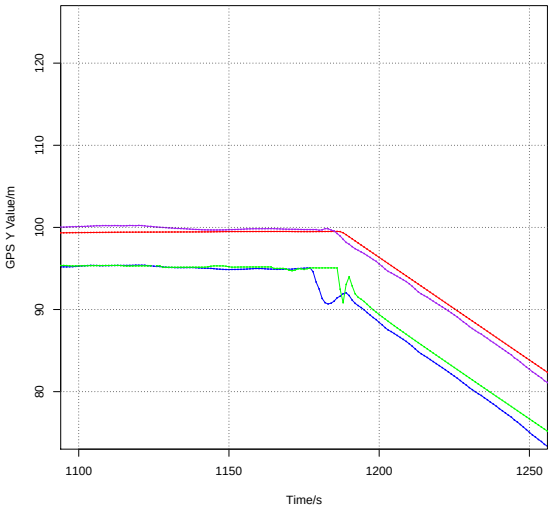
Figure 4.90: Location Estimates of Sensor 1 UTM X Coordinates vs. Time Huff Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

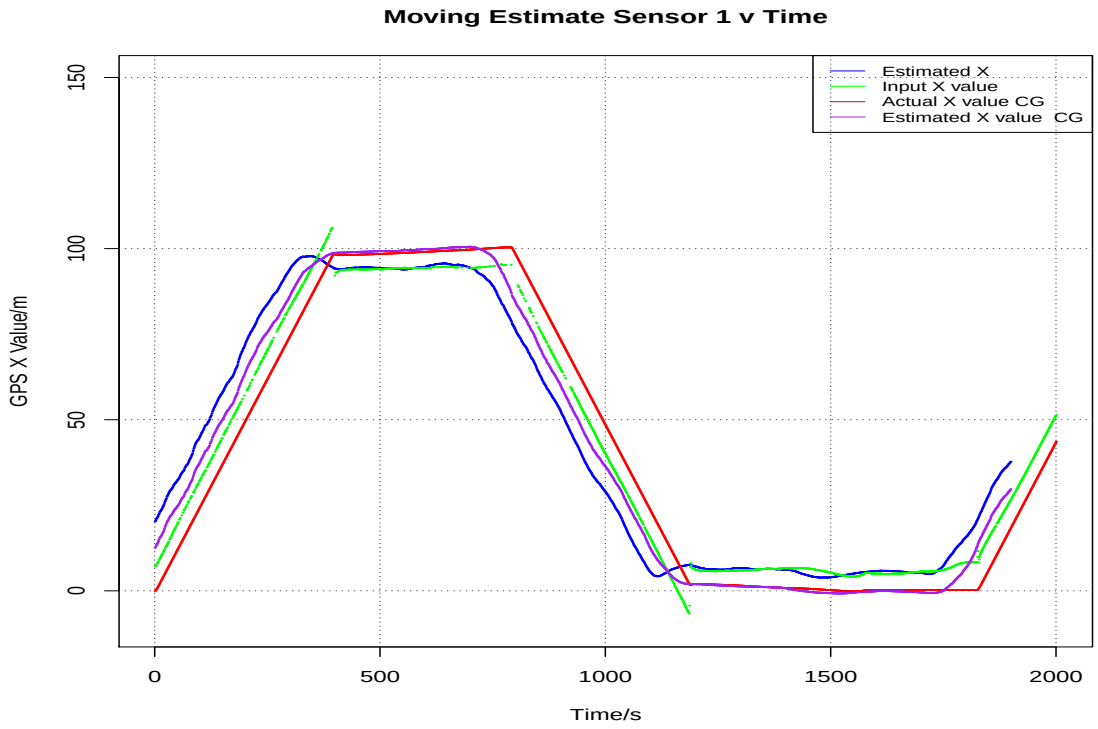


(b) Sensor 1 Positive Slope Magnified.

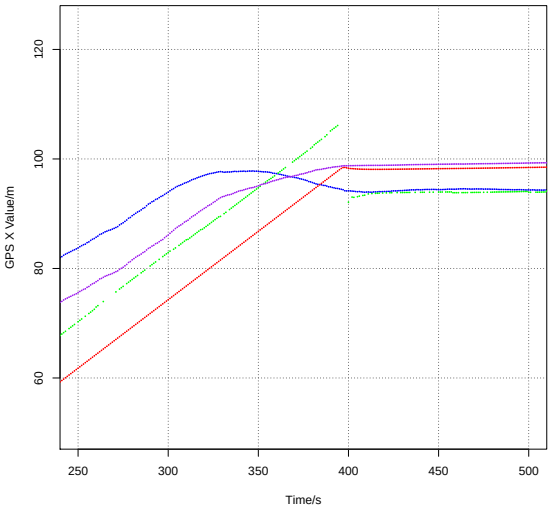


(c) Sensor 1 Negative Slope Magnified.

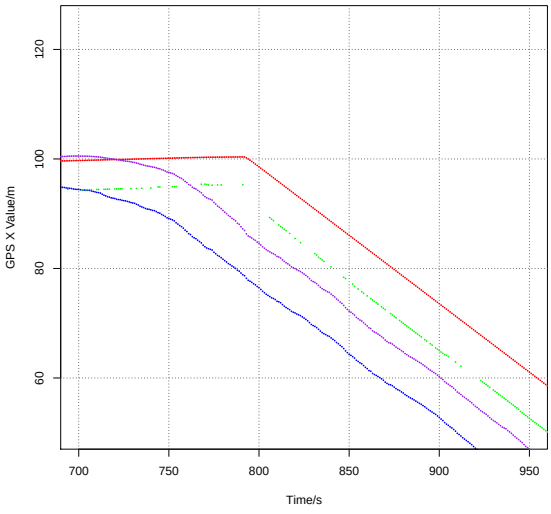
Figure 4.91: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time Huff Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

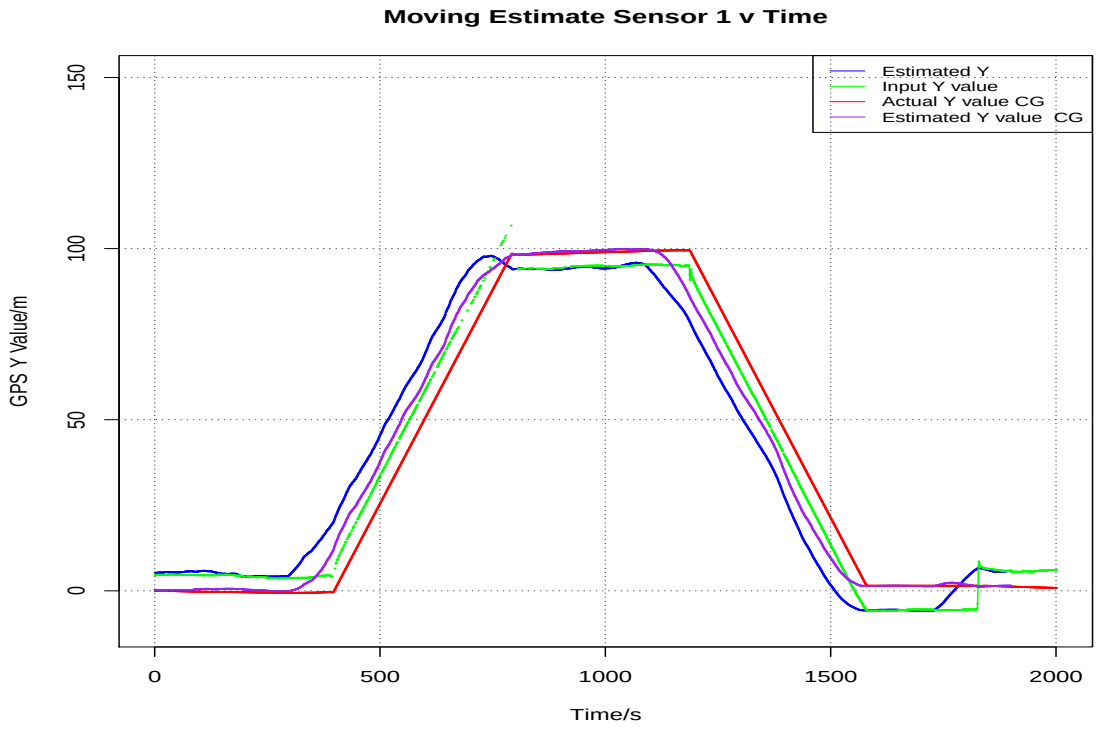


(b) Sensor 1 Positive Slope Magnified.

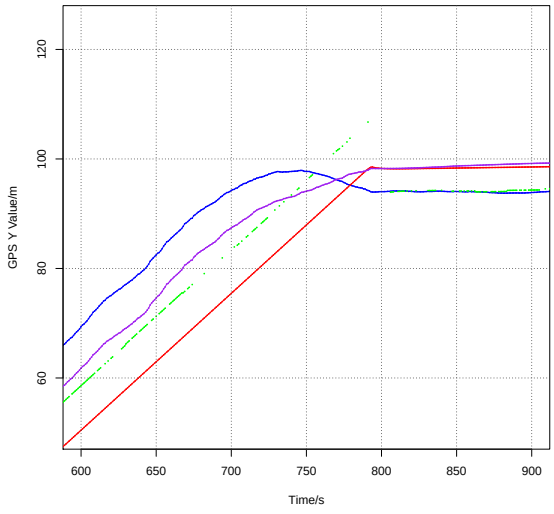


(c) Sensor 1 Negative Slope Magnified.

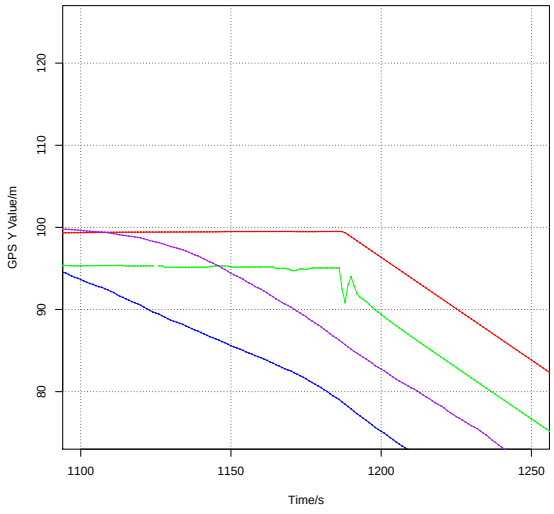
Figure 4.92: Location Estimates of Sensor 1 UTM X Coordinates vs. Time Huff Weights Sliding Window = 100 ($+veX - North, +veY - East$).



(a) Sensor 1.



(b) Sensor 1 Positive Slope Magnified.

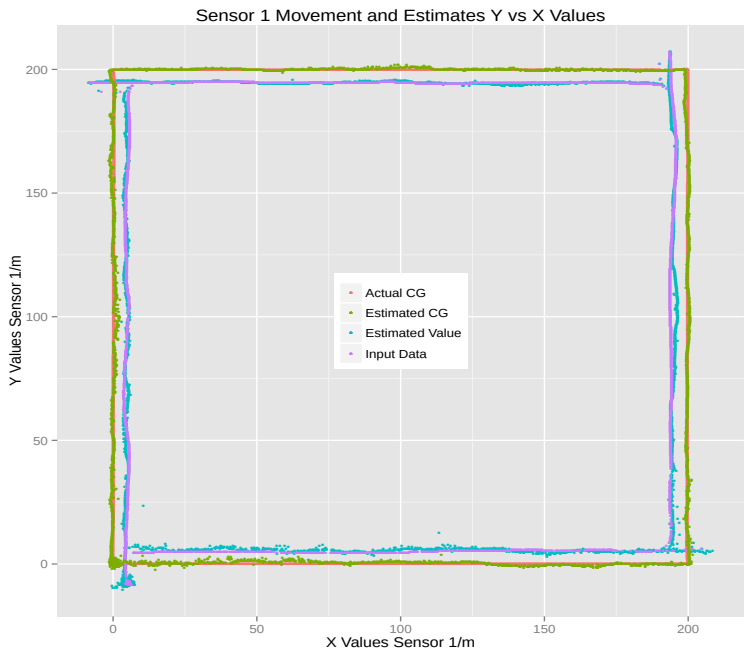


(c) Sensor 1 Negative Slope Magnified.

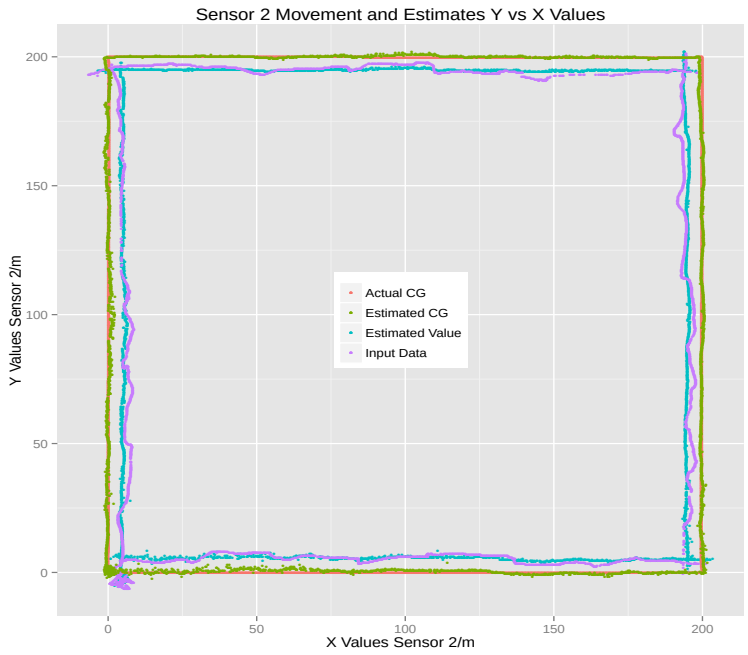
Figure 4.93: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time Huff Weights Sliding Window = 100 $(+veX - North, +veY - East)$.

4.11.2 Location Estimates Using Path 2

As with Path 1, Path 2 shows similar results for the location estimates, with a smoothing possibly due to the larger scale of the plots of paths that the platform follows on. Figures 4.94, 4.95, 4.96, 4.97 for sliding window = 1; Figures 4.98, 4.99, 4.100, 4.101 for sliding window = 5; Figures 4.102, 4.103, 4.104, 4.105 for sliding window = 10; and Figures 4.106, 4.107, 4.108, 4.109 for sliding window = 100 show the results of calculating the location estimates for each of the sensors that reported values from Sensor 1,2,3,5,6,7 and 8 along with the combination of sliding window values of 1,5,10 and 100. As seen with the results for Path 1, the red and green plots seem to coincide very closely with one another, and as the sliding window size increases, a smoothing effect takes place reducing the magnitude of noise. The red plot represents the actual center of gravity of the platform and the green plot represents the estimated center of gravity derived from the location estimates obtained from the quadratic optimization using the sensor configuration.

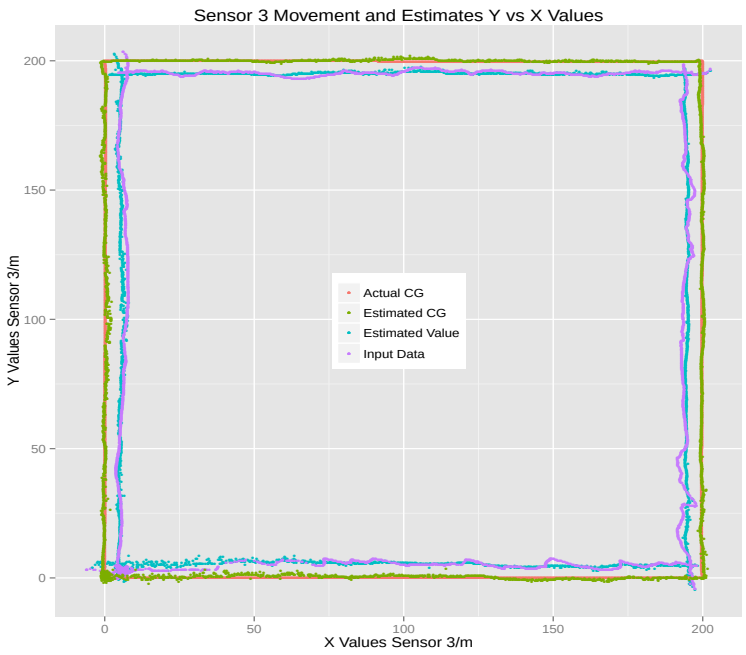


(a) Sensor 1.

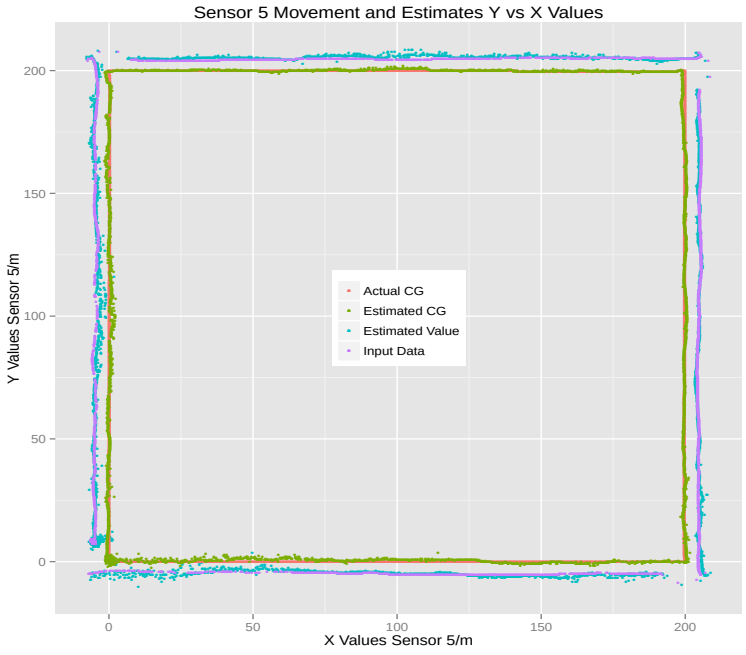


(b) Sensor 2.

Figure 4.94: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8.

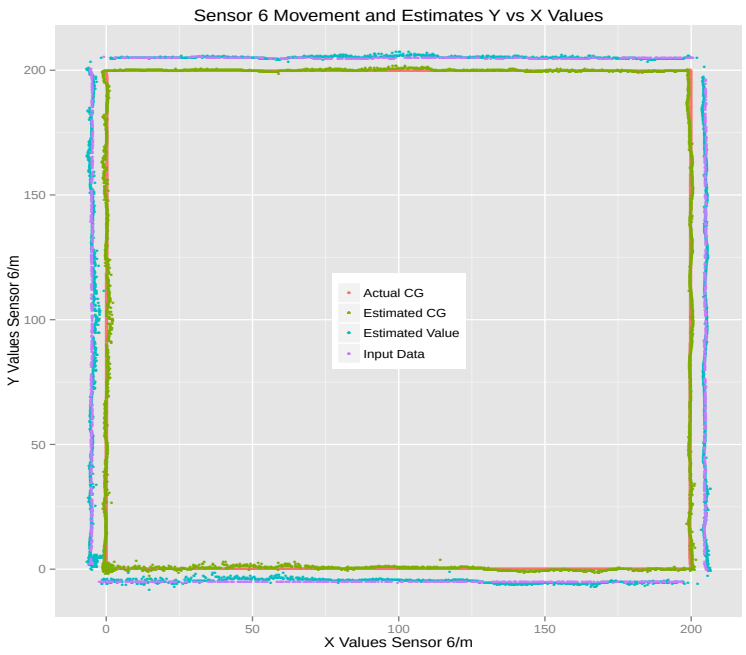


(a) Sensor 3.

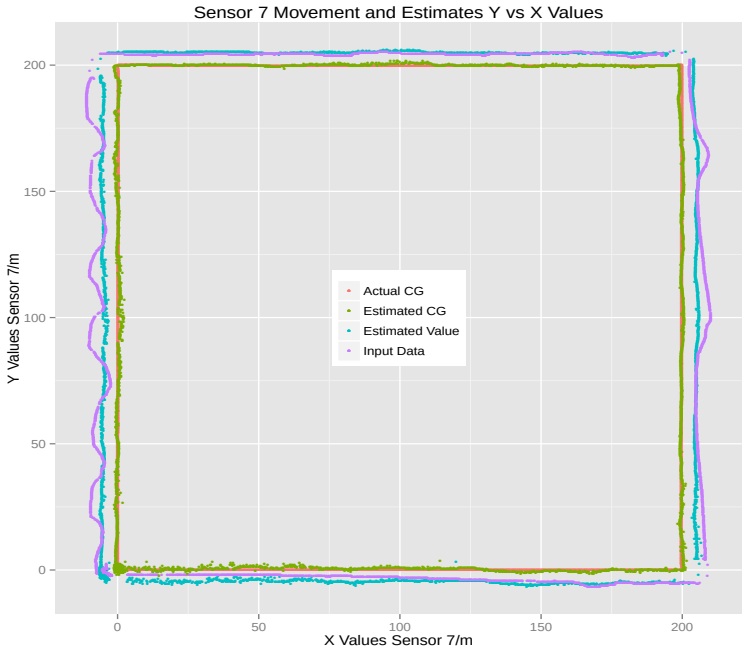


(b) Sensor 5.

Figure 4.95: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

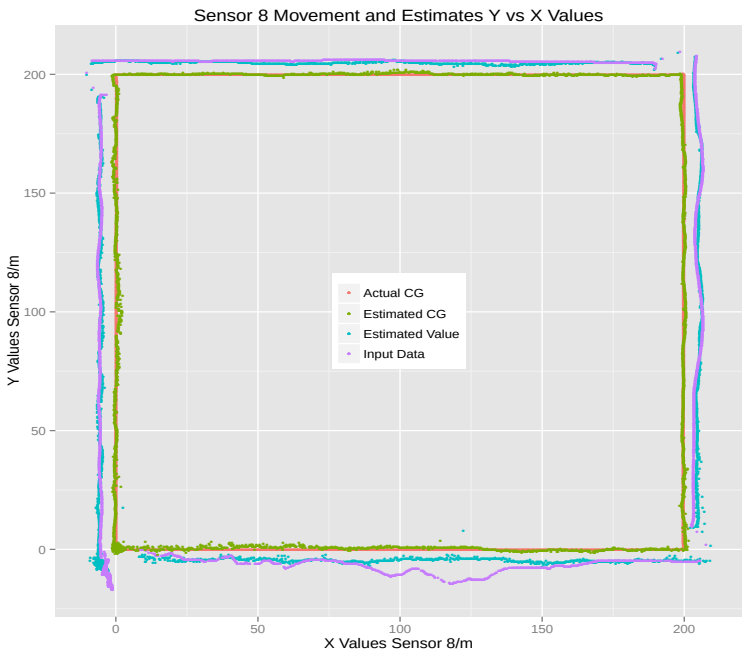


(a) Sensor 6.



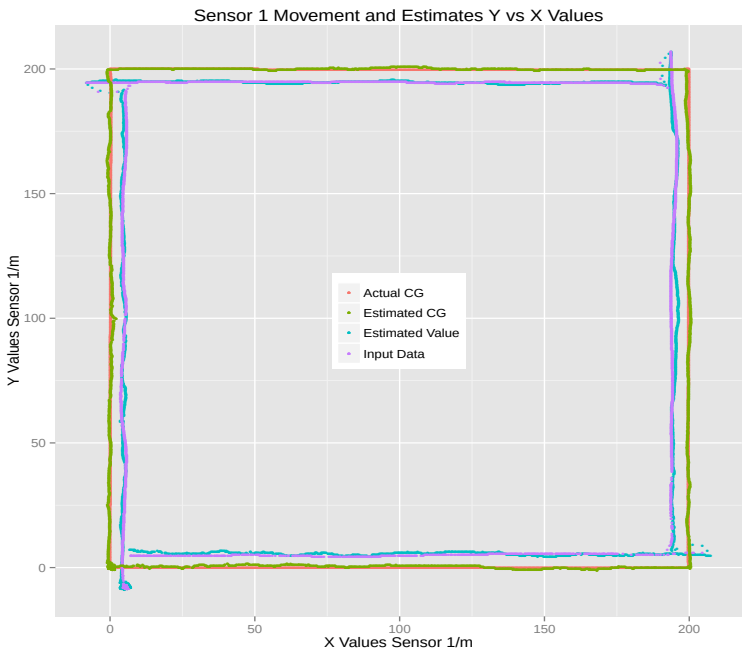
(b) Sensor 7.

Figure 4.96: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

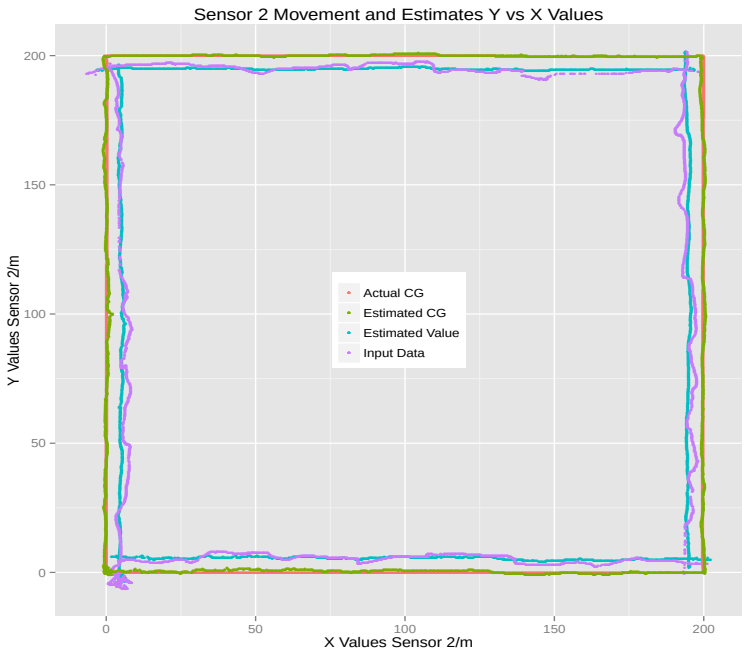


(a) Sensor 8.

Figure 4.97: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 1 For Sensors 1-8 (contd.).

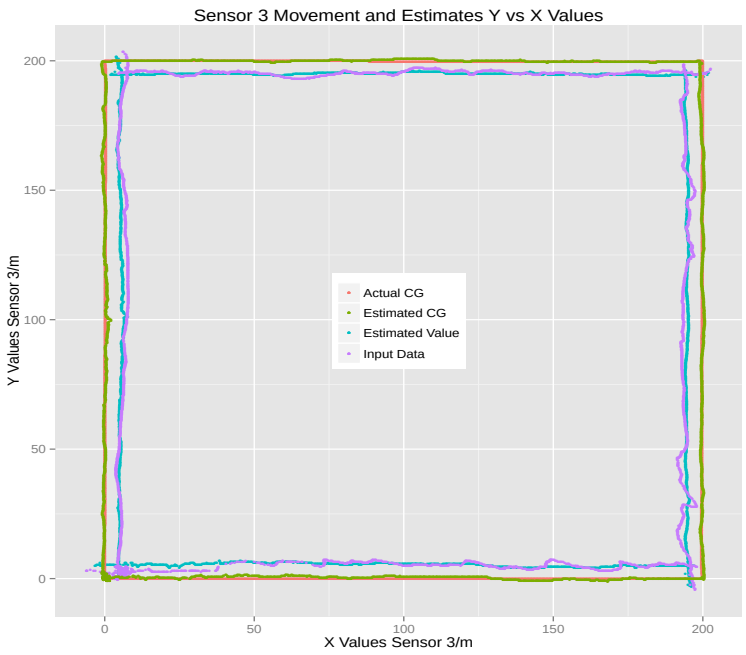


(a) Sensor 1.

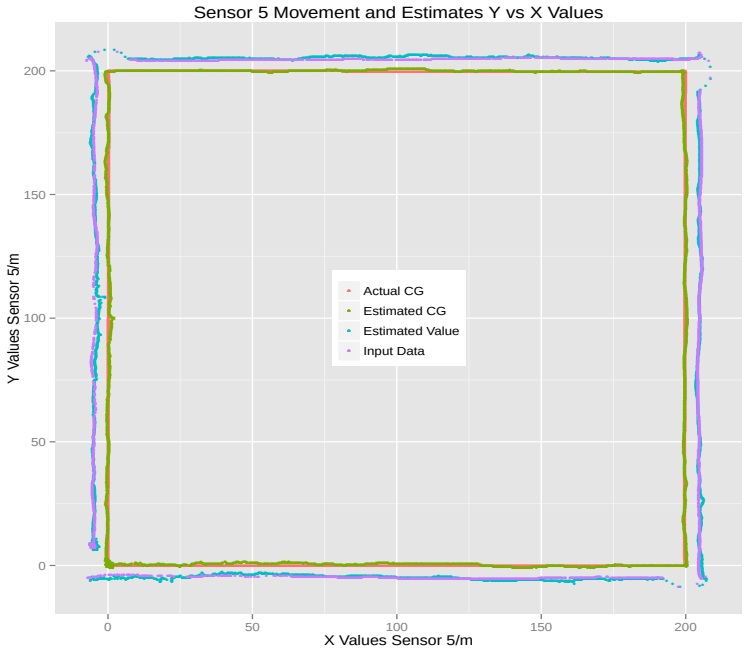


(b) Sensor 2.

Figure 4.98: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8.

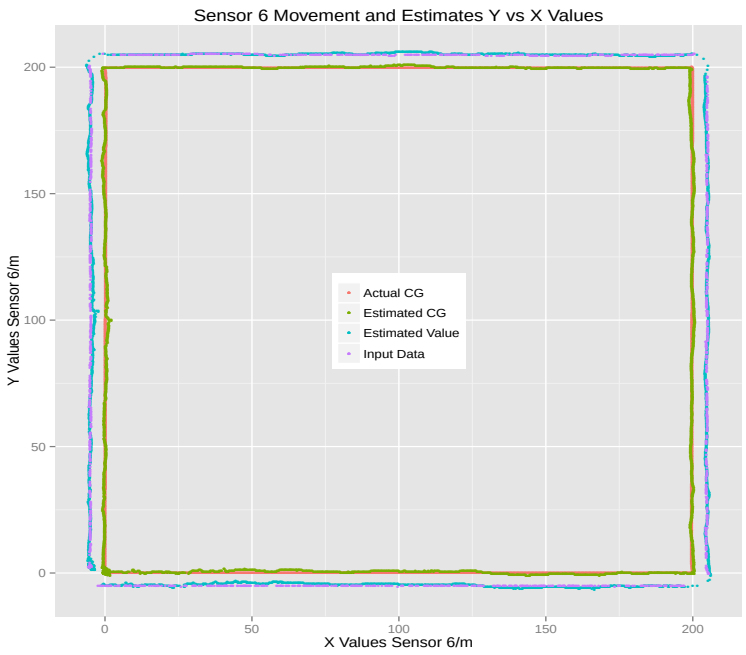


(a) Sensor 3.

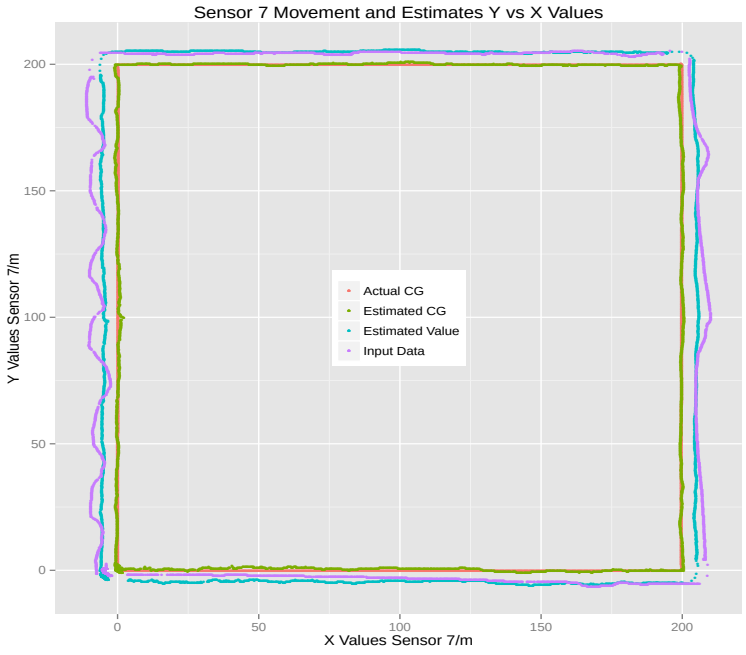


(b) Sensor 5.

Figure 4.99: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

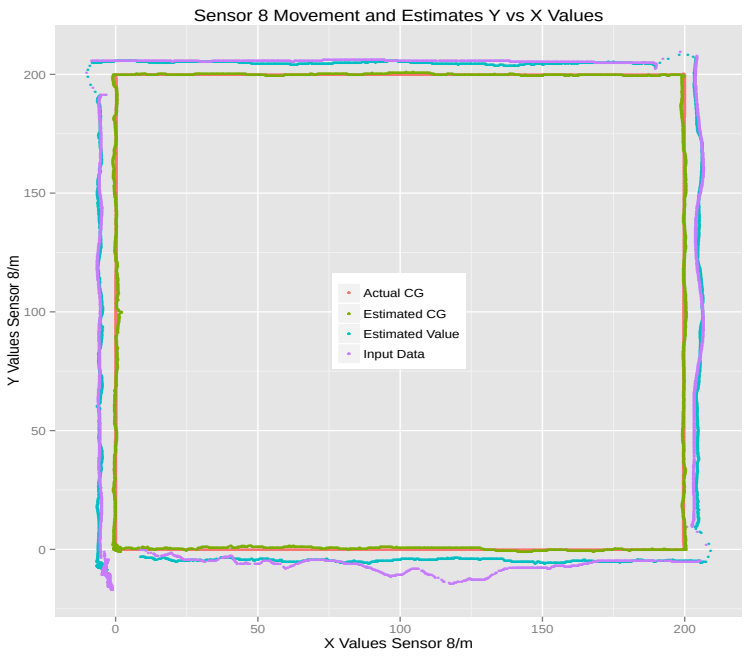


(a) Sensor 6.



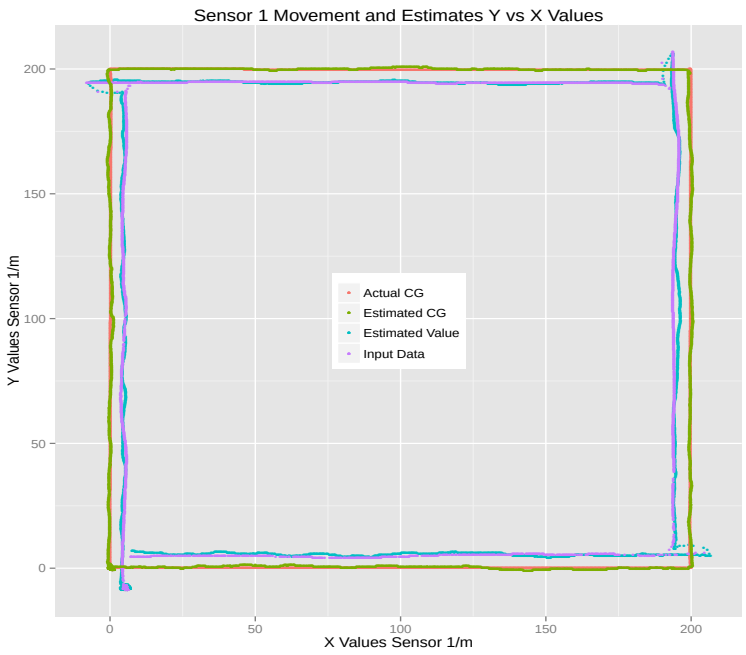
(b) Sensor 7.

Figure 4.100: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

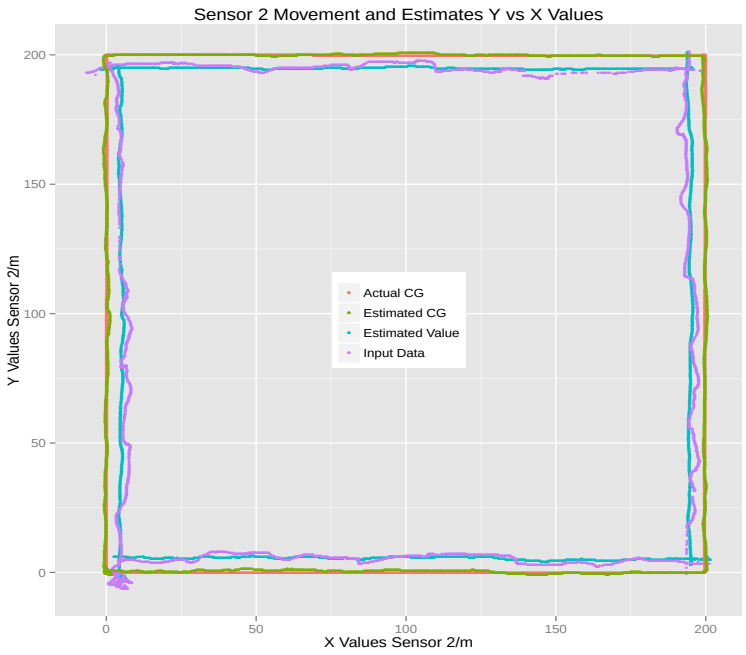


(a) Sensor 8.

Figure 4.101: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 5 For Sensors 1-8 (contd.).

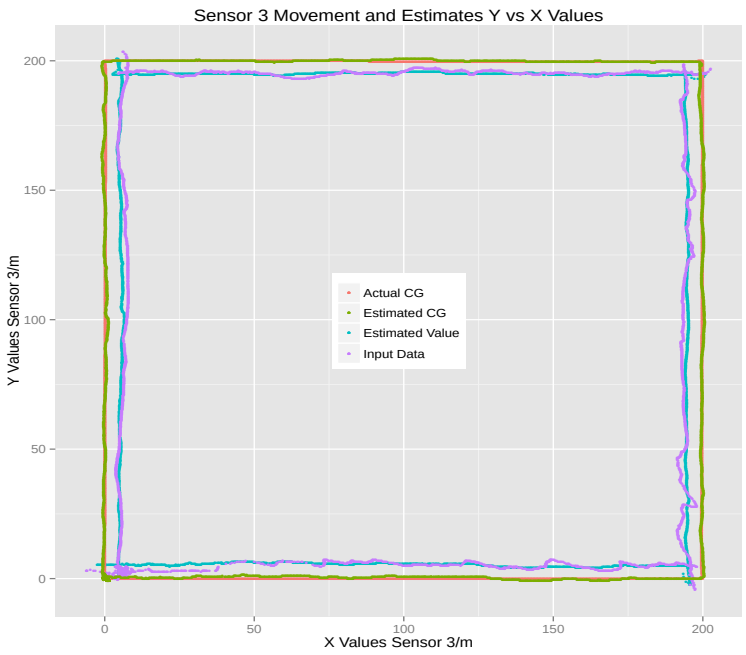


(a) Sensor 1.

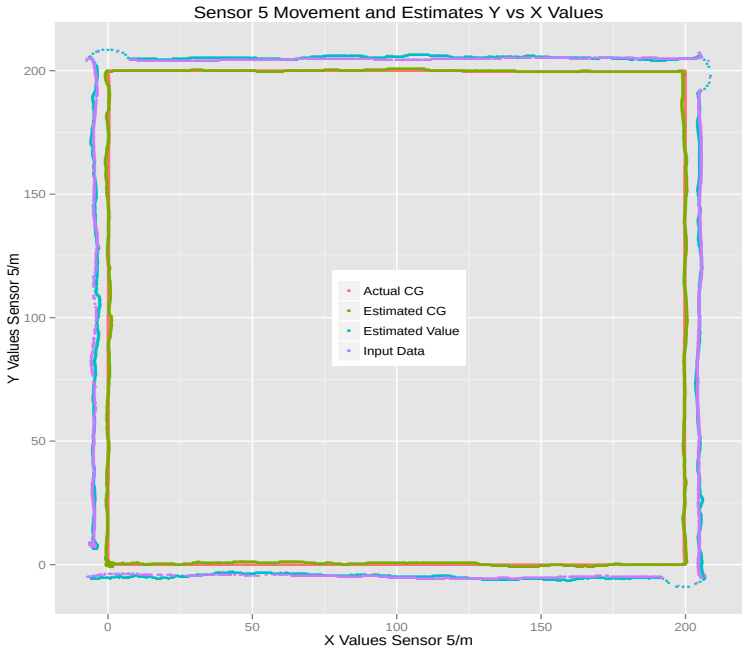


(b) Sensor 2.

Figure 4.102: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8.

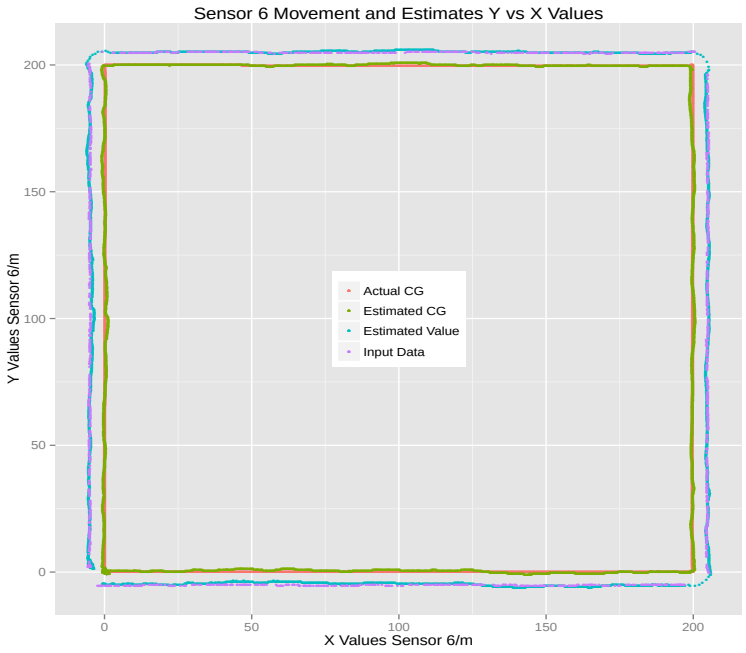


(a) Sensor 3.

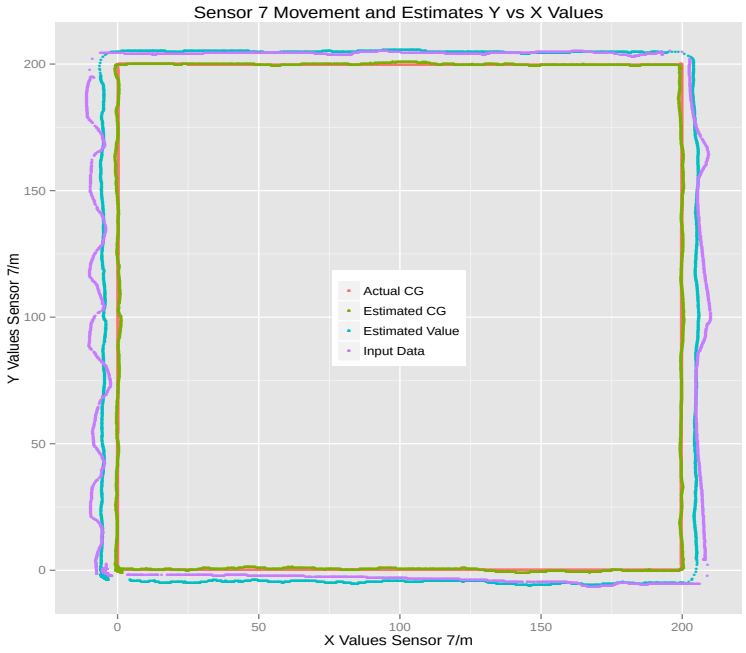


(b) Sensor 5.

Figure 4.103: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

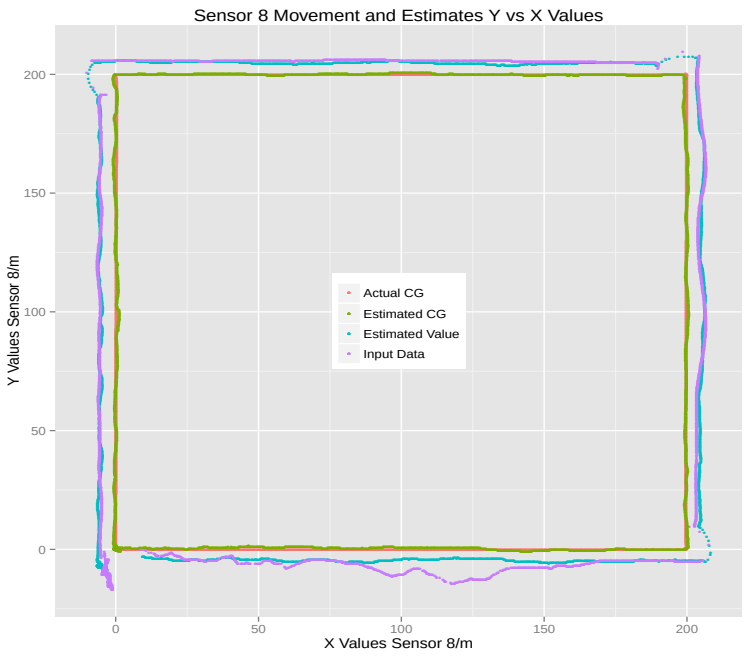


(a) Sensor 6.



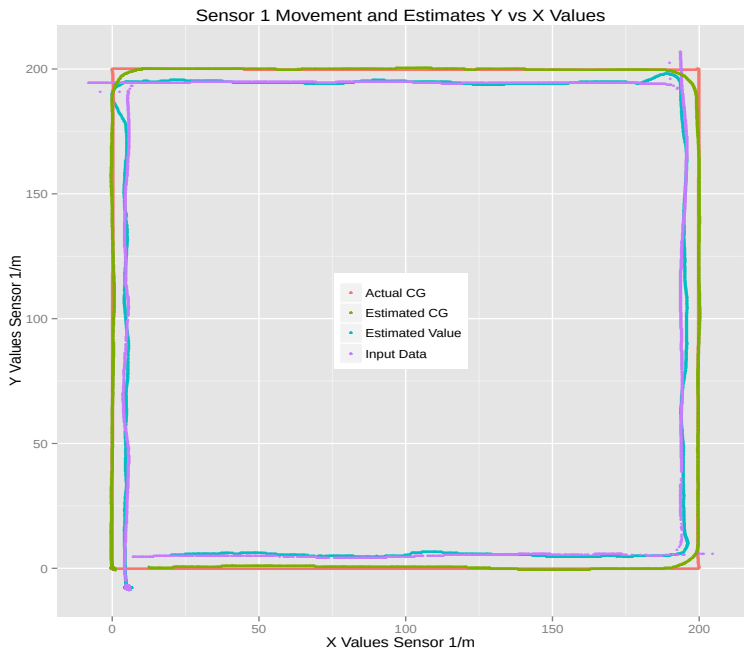
(b) Sensor 7.

Figure 4.104: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

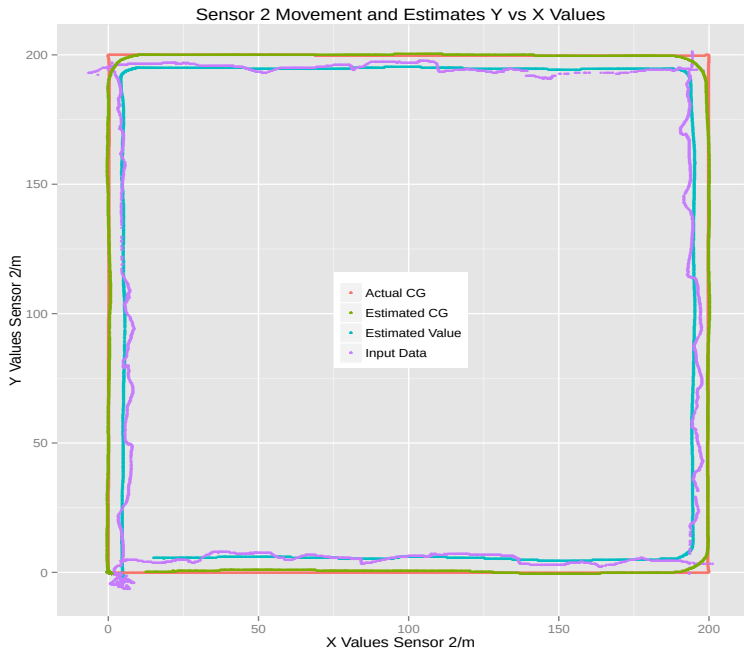


(a) Sensor 8.

Figure 4.105: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 10 For Sensors 1-8 (contd.).

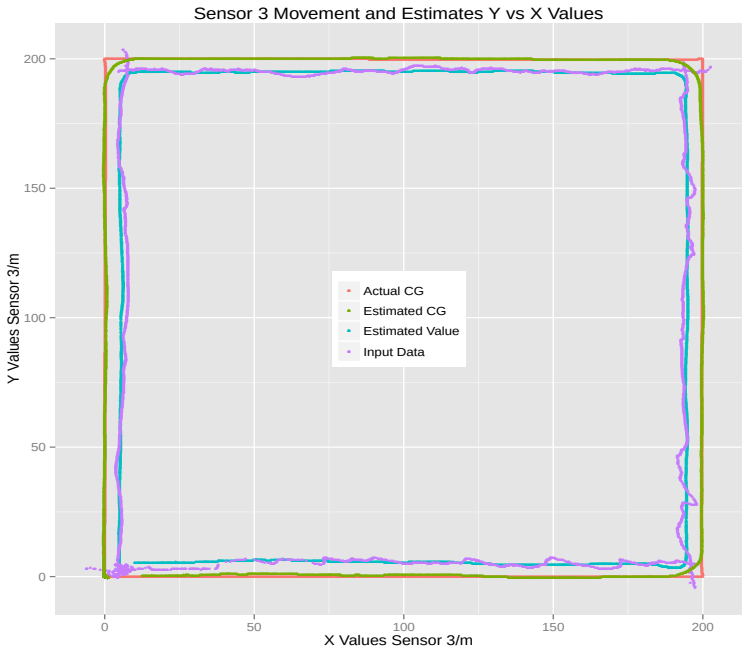


(a) Sensor 1.

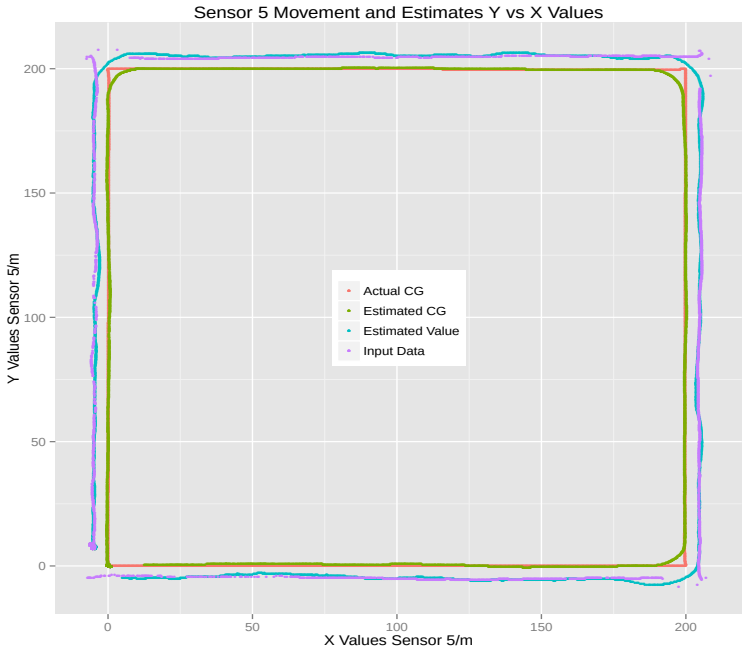


(b) Sensor 2.

Figure 4.106: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8.

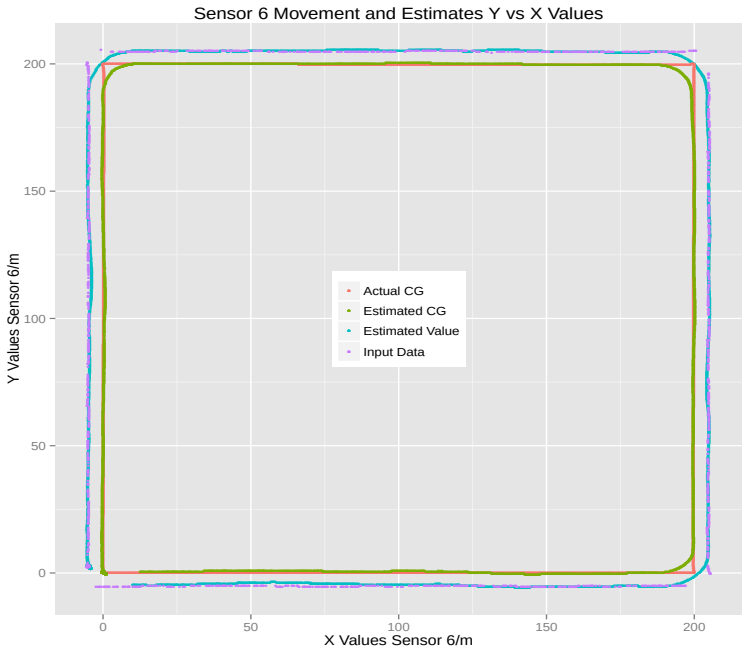


(a) Sensor 3.

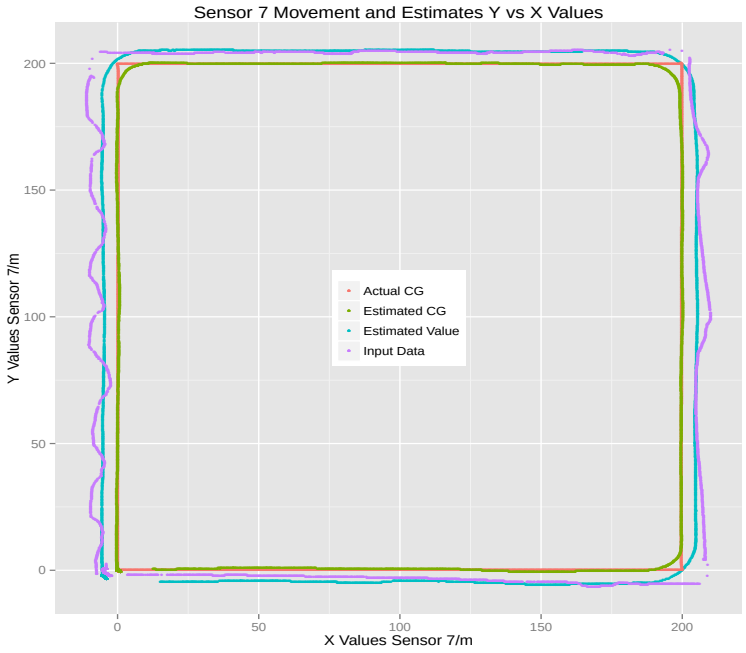


(b) Sensor 5.

Figure 4.107: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

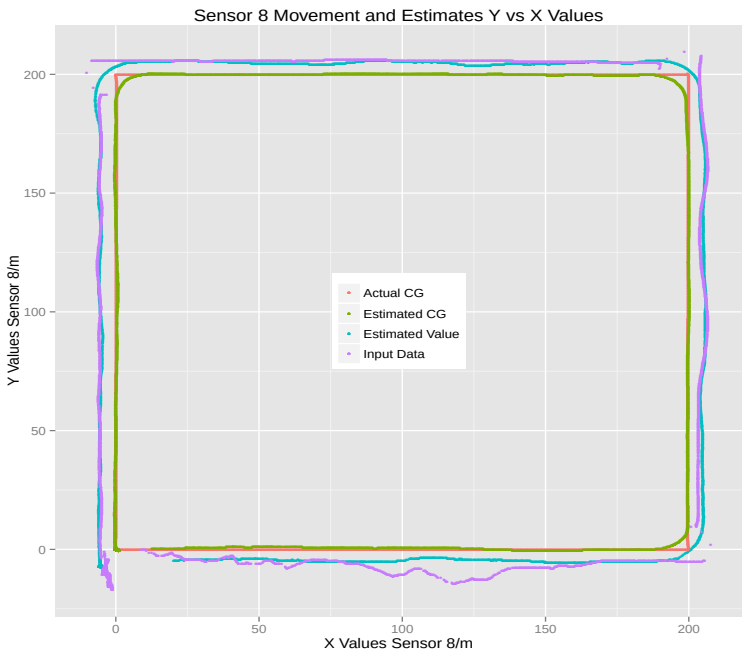


(a) Sensor 6.



(b) Sensor 7.

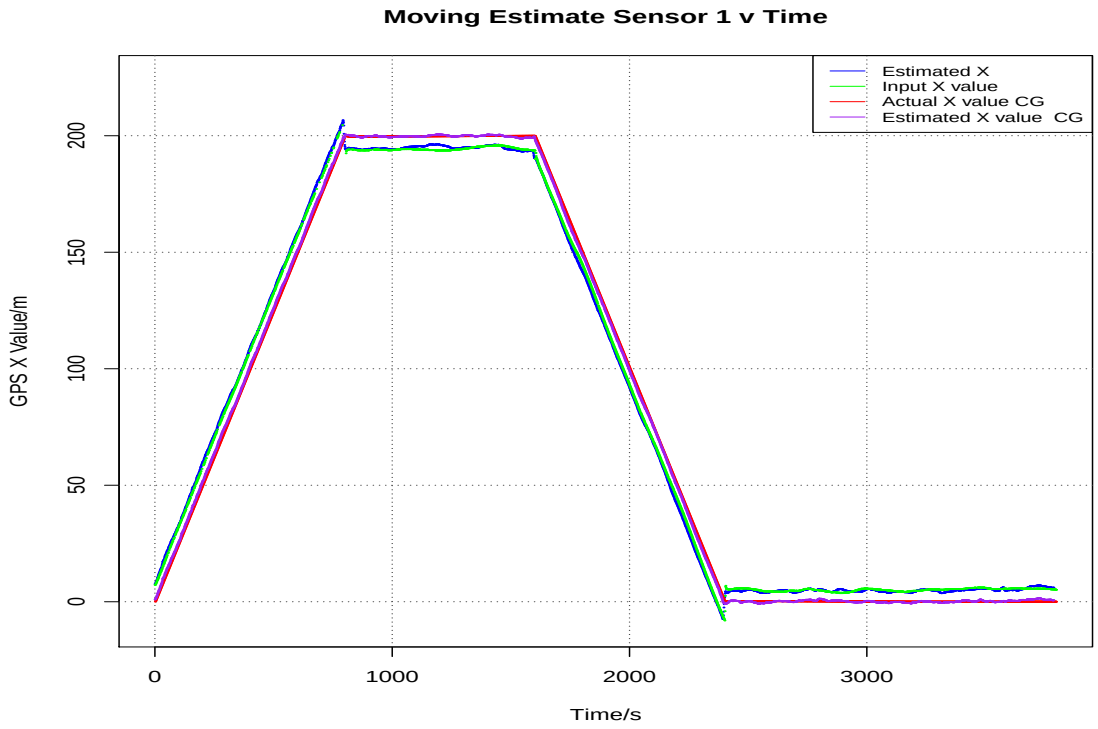
Figure 4.108: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).



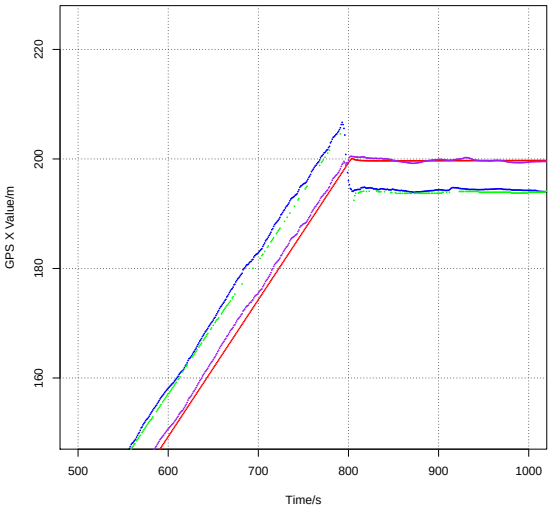
(a) Sensor 8.

Figure 4.109: Location Estimates Using Quadratic Optimization with Huff Weights $(+veX - North, +veY - East)$ Sliding Window = 100 For Sensors 1-8 (contd.).

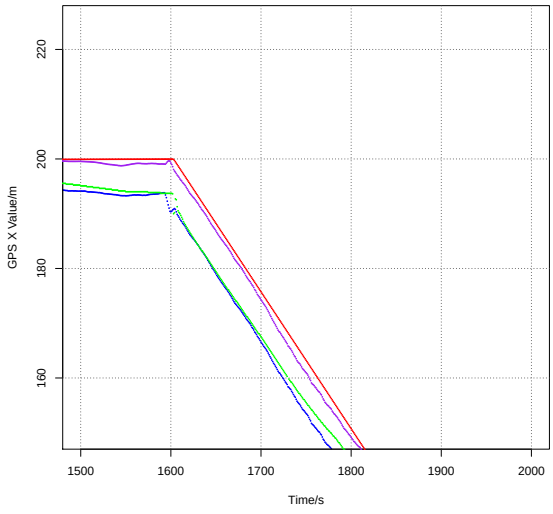
Using a longer path with Huff weighting still does not and will not be able to solve the issue of lag for higher sliding window values shown in Figures 4.110, 4.111 vs. 4.112, 4.113 for sliding window = 10 and sliding window = 100 respectively. As seen in the previous sections, the estimates for center of gravity and actual center of gravity adhere very close to one another for a myriad of sliding window sizes and weighting schemes. The next section introduces the methodology to choose the optimum size value for sliding windows and weighting empirically through the use of performance metrics.



(a) Sensor 1.

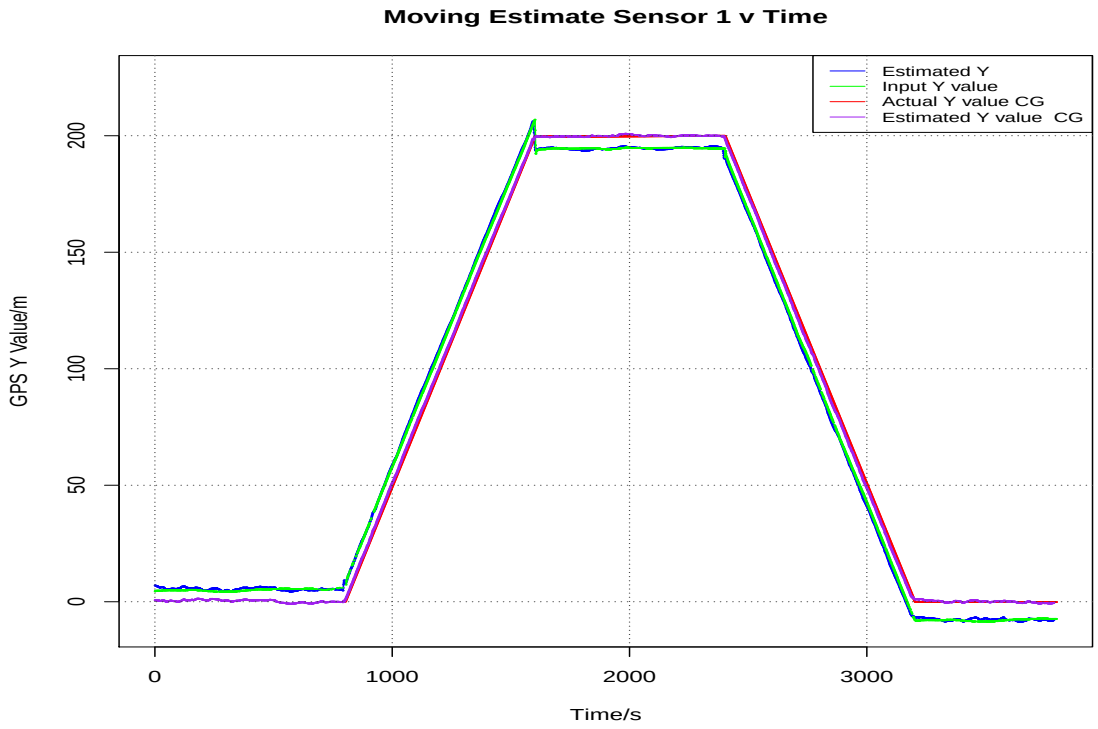


(b) Sensor 1 Positive Slope Magnified.

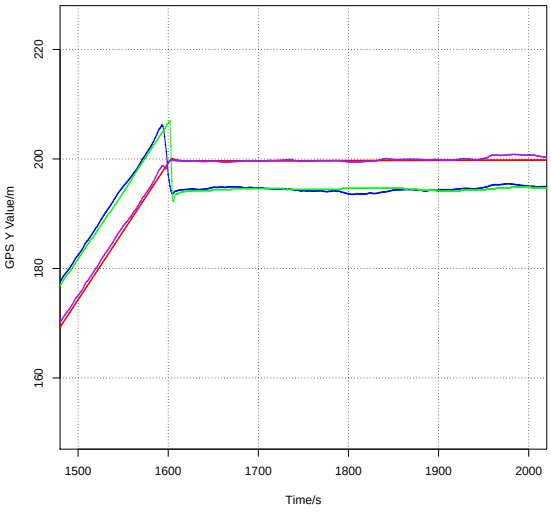


(c) Sensor 1 Negative Slope Magnified.

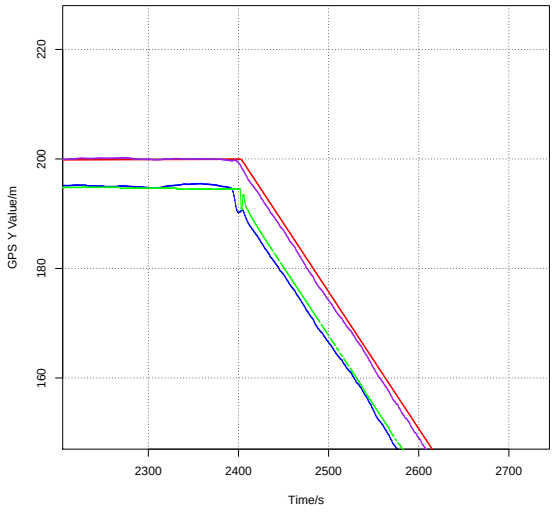
Figure 4.110: Location Estimates of Sensor 1 UTM X Coordinates vs. Time Huff Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

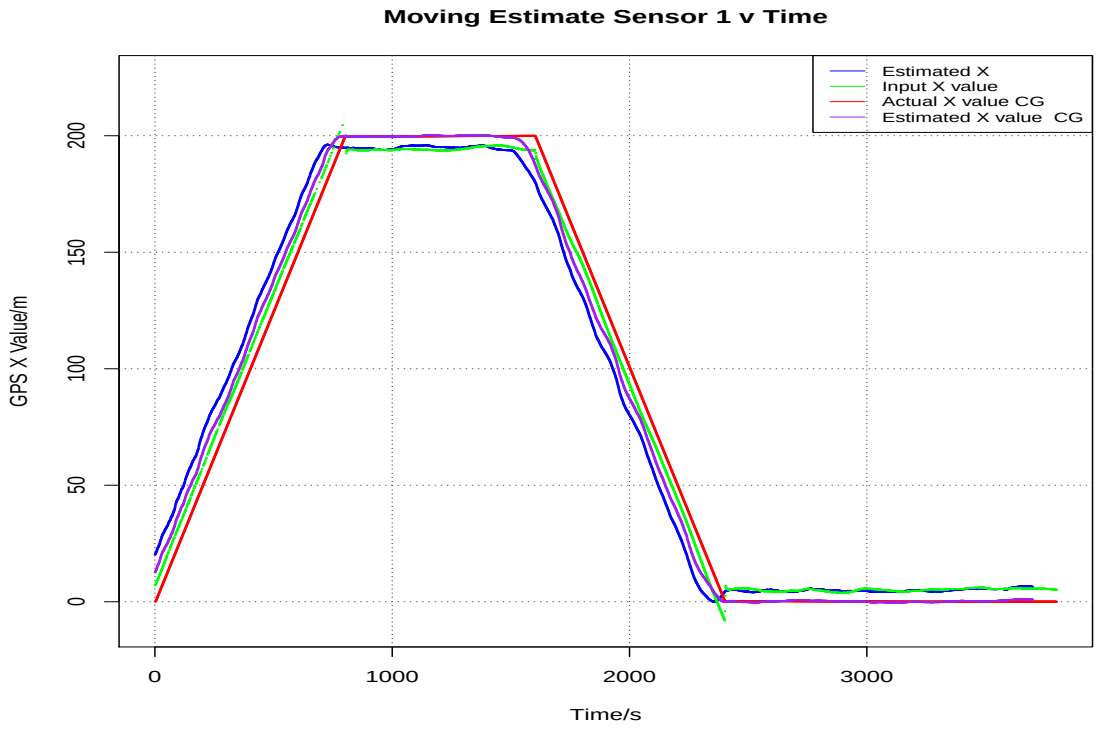


(b) Sensor 1 Positive Slope Magnified.

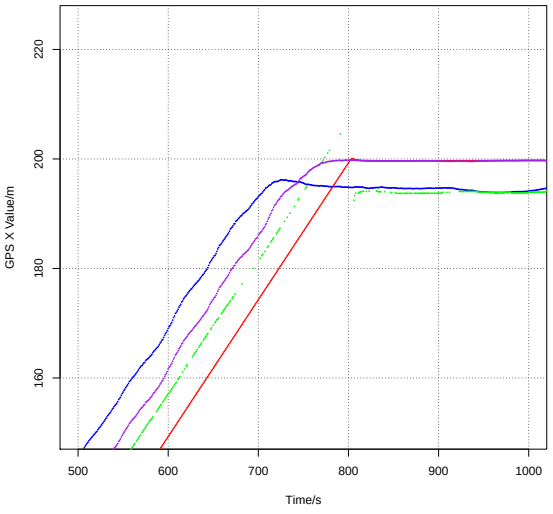


(c) Sensor 1 Negative Slope Magnified.

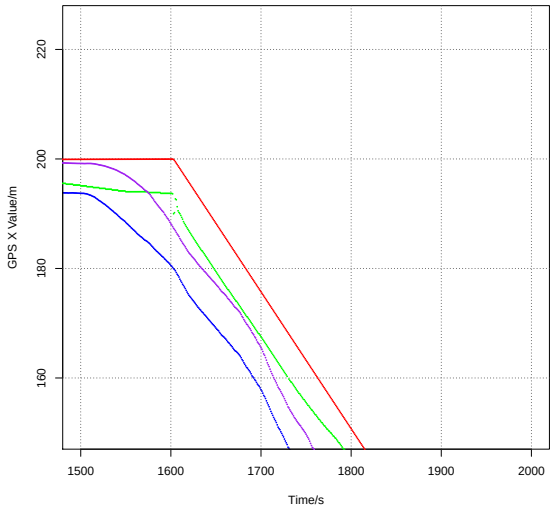
Figure 4.111: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time Huff Weights Sliding Window = 10 $(+veX - North, +veY - East)$.



(a) Sensor 1.

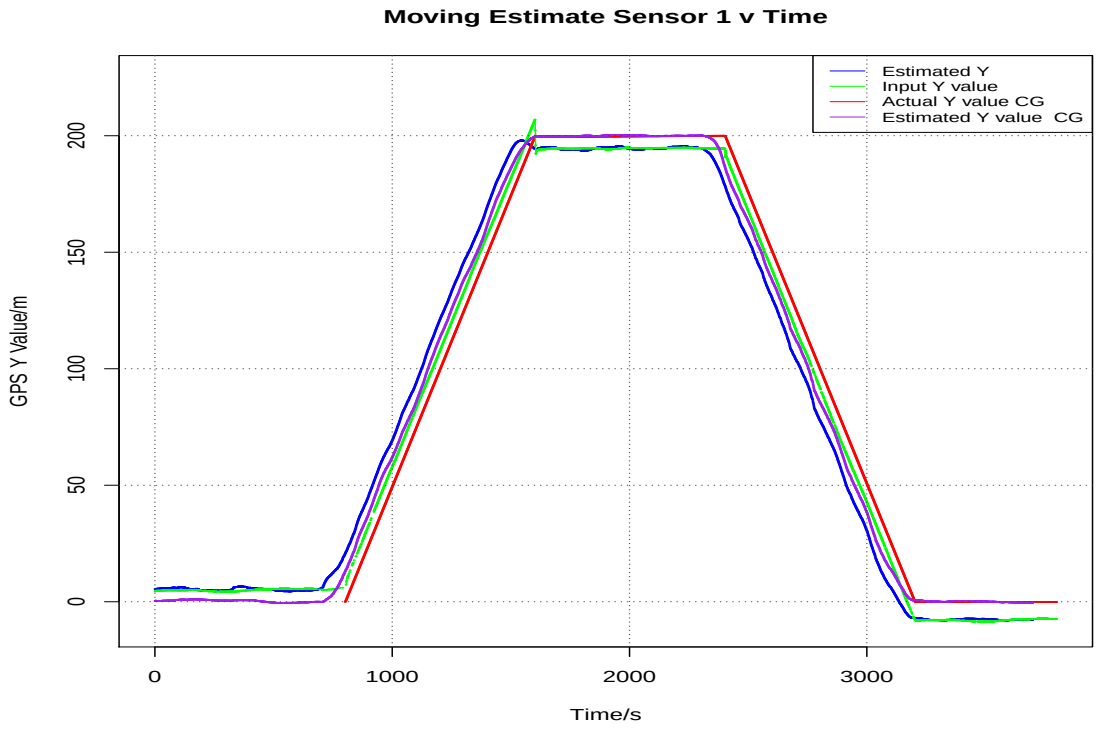


(b) Sensor 1 Positive Slope Magnified.

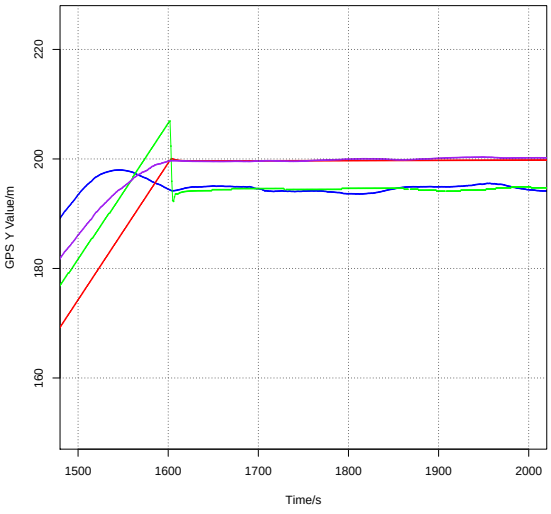


(c) Sensor 1 Negative Slope Magnified.

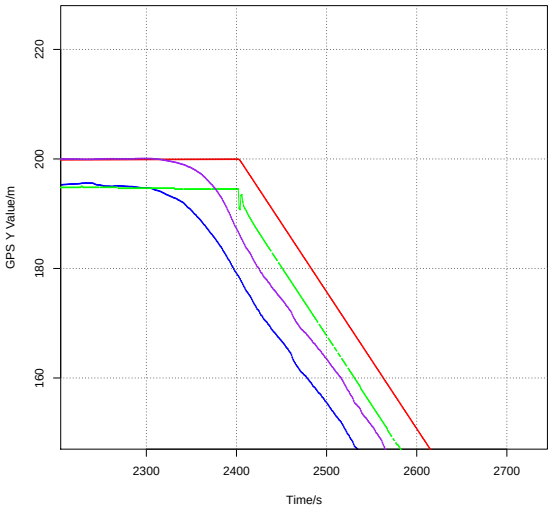
Figure 4.112: Location Estimates of Sensor 1 UTM X Coordinates vs. Time Huff Weights Sliding Window = 100 ($+veX - North, +veY - East$).



(a) Sensor 1.



(b) Sensor 1 Positive Slope Magnified.



(c) Sensor 1 Negative Slope Magnified.

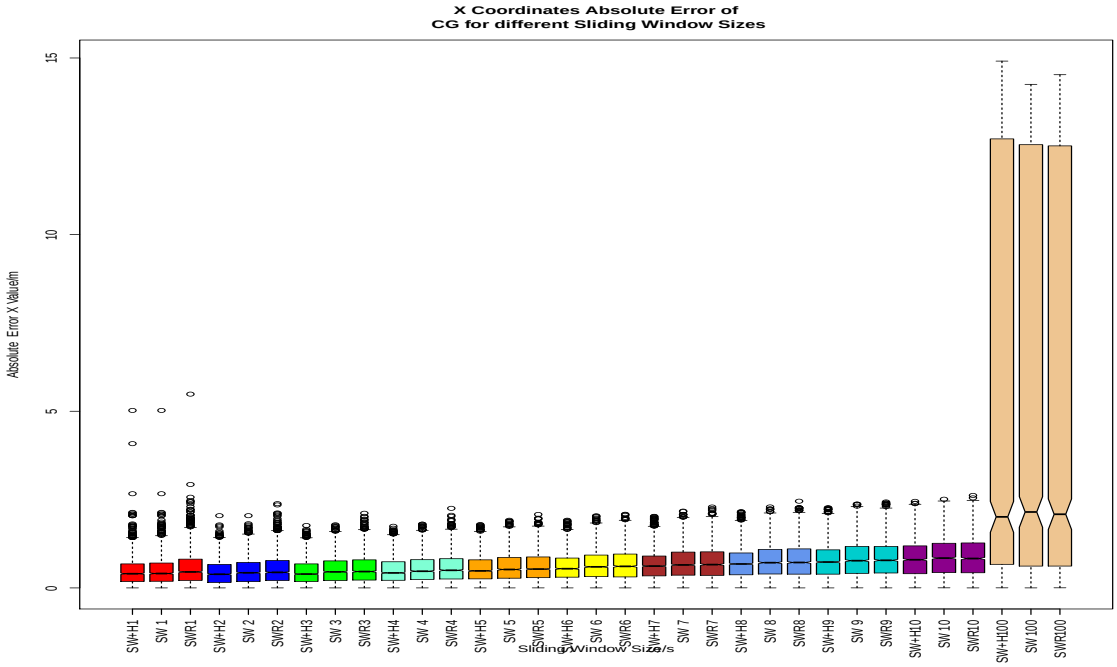
Figure 4.113: Location Estimates of Sensor 1 UTM Y Coordinates vs. Time Huff Weights Sliding Window = 100 ($+veX - North, +veY - East$).

4.12 Performance

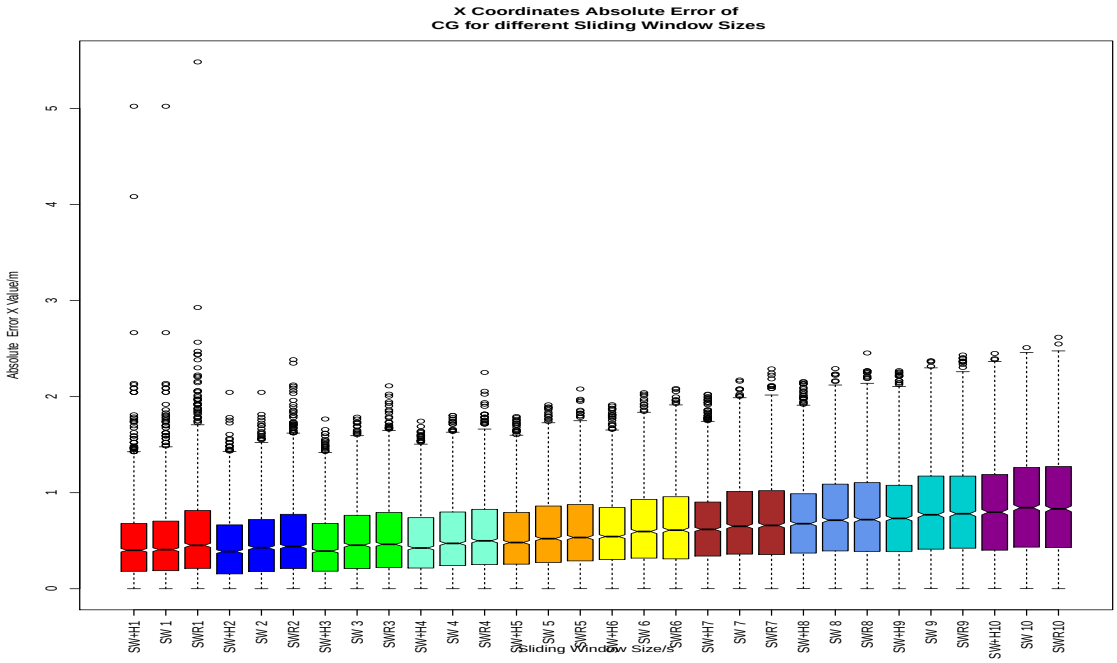
To be able to accurately choose which of the sliding window sizes is the most suitable for the application to a mobile robot platform during motion, an objective measure can be used to differentiate between their performance. Here, performance is defined by how accurate the estimated center of gravity of the platform is when compared to the actual center of gravity. The estimated center of gravity is derived from using the sensor location estimates using Equation 3.8 while the actual center of gravity is provided by the SimuLink® mobile platform simulation described in Section 4.1.2. The performance metric used is the absolute difference between the estimated center of gravity and the actual center of gravity represented as $(x_t^{\mathrm{CG}}, y_t^{\mathrm{CG}})$, which is shown in Equation 4.1, with $error_{xt}$ for measuring the performance on x-coordinates and $error_{yt}$ to measure the performance on y-coordinates of the locations of the sensors.

$$\begin{aligned} error_{xt} &= \left| \widehat{x_t^{\mathrm{CG}}} - x_t^{\mathrm{CG}} \right| \\ error_{yt} &= \left| \widehat{y_t^{\mathrm{CG}}} - y_t^{\mathrm{CG}} \right| \end{aligned} \tag{4.1}$$

For Path 1, the performance metrics in Figure 4.114 and 4.115 show that on using a smaller sliding window size or no sliding window at all, there would be a large number of outliers or the location estimations are highly inaccurate, while as the we scale up to using a larger sliding window, the outliers seem to all but disappear, as seen in the case of using a sliding window of size 10, without a large change in the value of the median error. Moreover, the effect of using a sliding window of a large size may not be very useful due to the previously discussed lag effect.

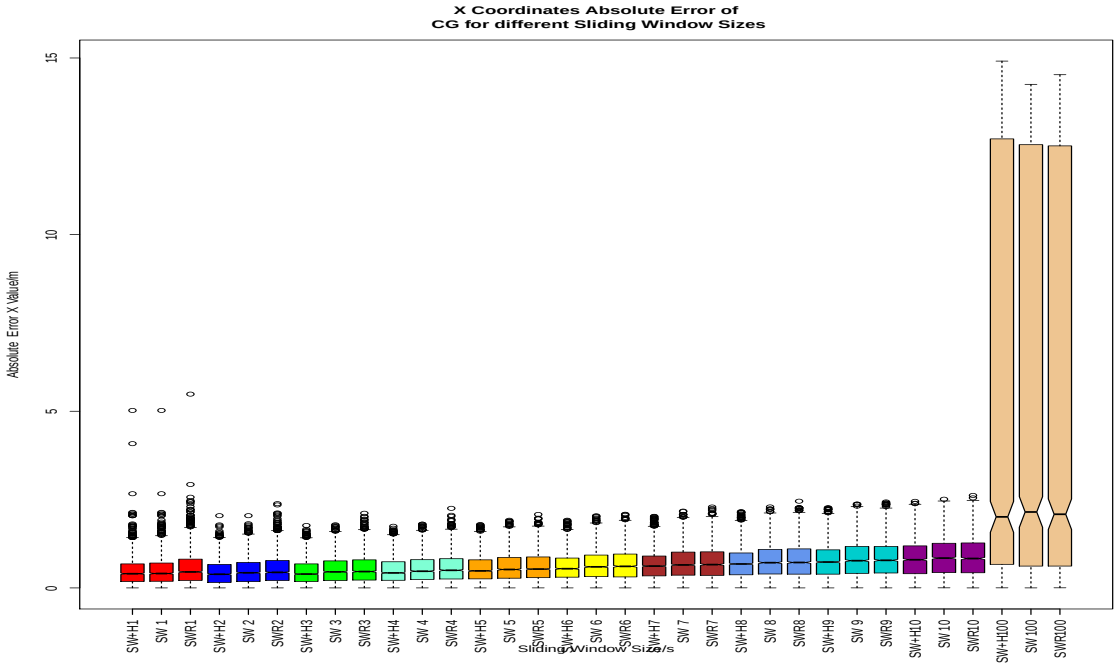


(a) Comparison of performance for UTM X Coordinates for Sliding Window Size 1-100 with No Weights, Huff Weights and Random Weights.

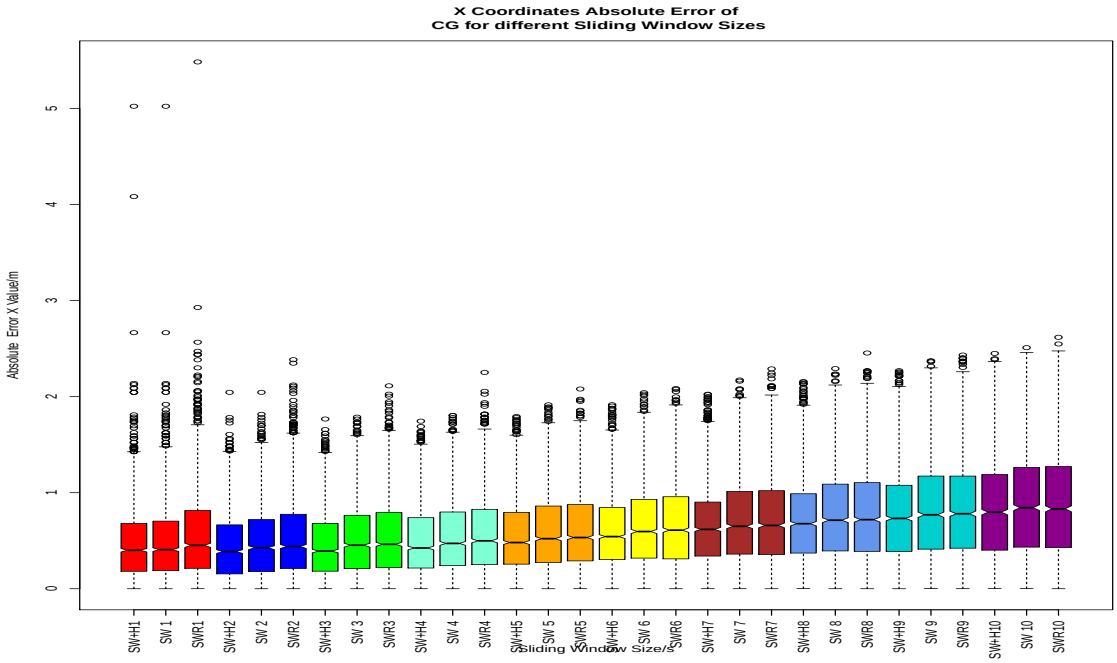


(b) Comparison of performance for UTM X Coordinates for Sliding Window Size 1-10 with No Weights, Huff Weights and Random Weights.

Figure 4.114: Comparison of Performance for Path 1 w.r.t Sliding Windows and Weights for UTM X Coordinates.



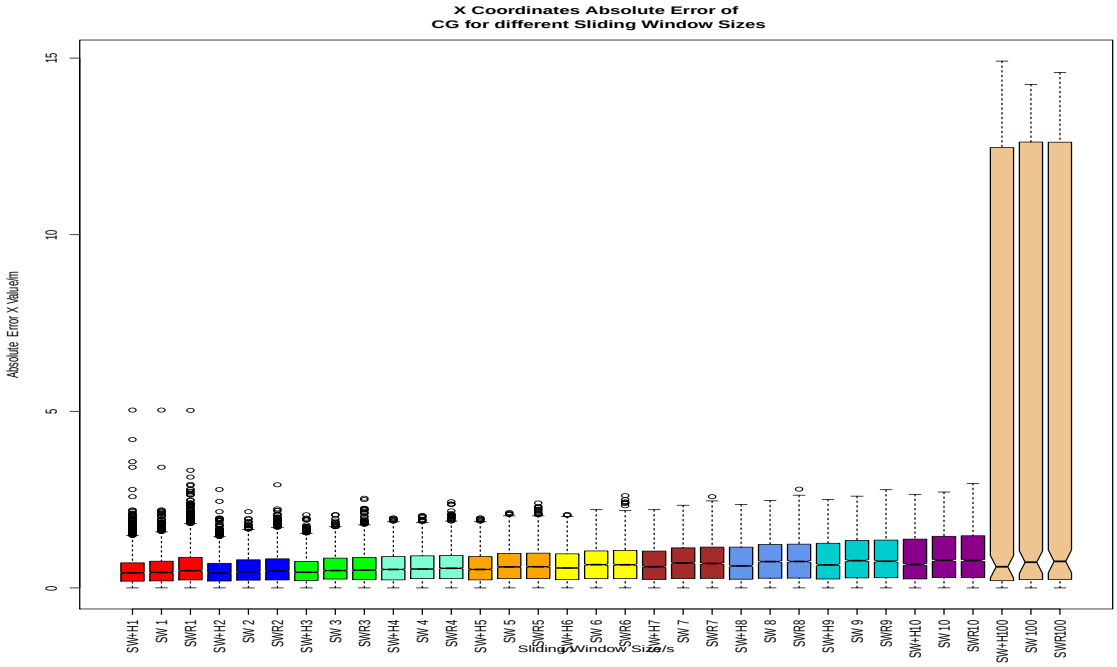
(a) Comparison of performance for UTM Y Coordinates for Sliding Window Size 1-100 with No Weights, Huff Weights and Random Weights.



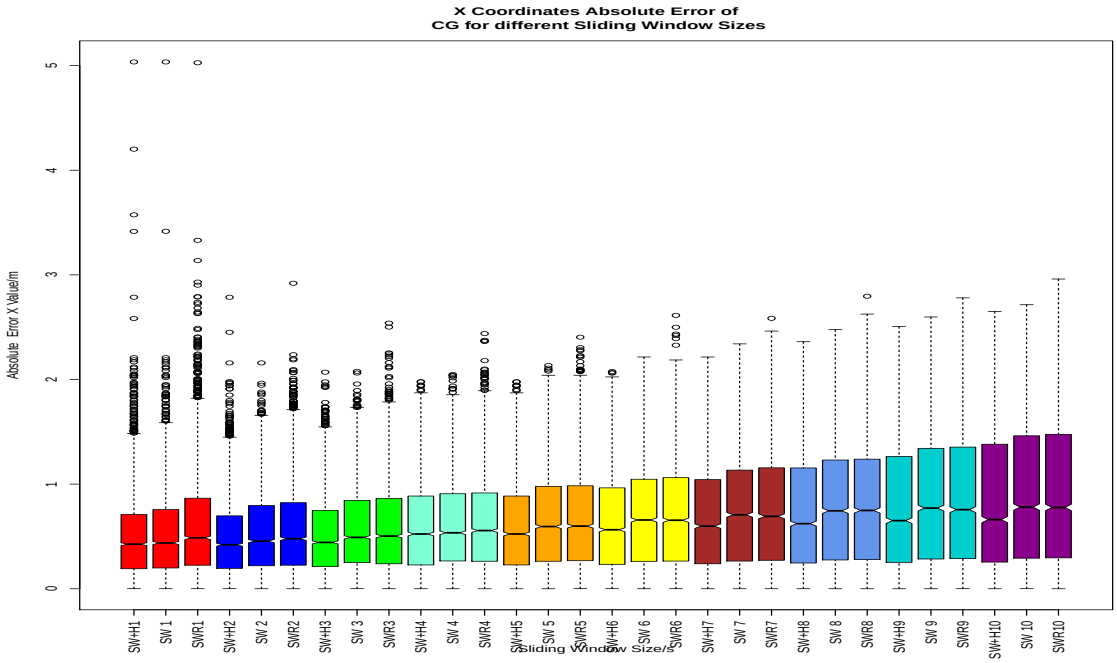
(b) Comparison of performance for UTM Y Coordinates for Sliding Window Size 1-10 with No Weights, Huff Weights and Random Weights.

Figure 4.115: Comparison of Performance for Path 1 w.r.t Sliding Windows and Weights for UTM Y Coordinates.

Additionally, we see that in all cases, the use of Huff weights provides an equal or better performance than using no weights at all or using random weights. These results are reflected similarly for Path 2, as seen in Figures 4.116 and 4.117.

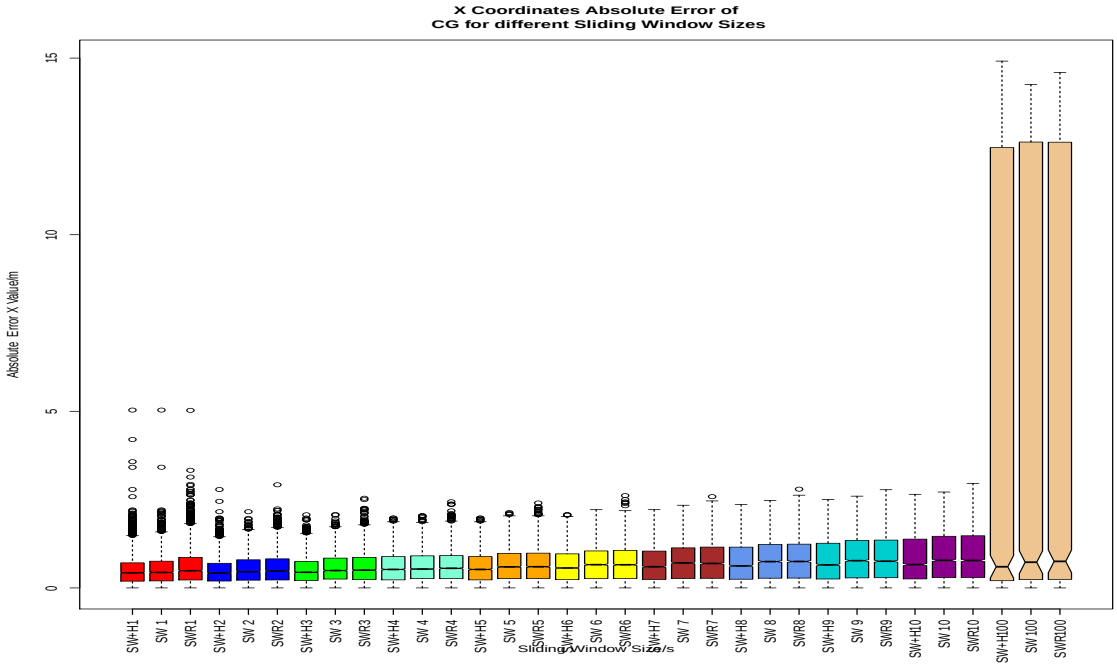


(a) Comparison of performance for UTM X Coordinates for Sliding Window Size 1-100 with No Weights, Huff Weights and Random Weights.

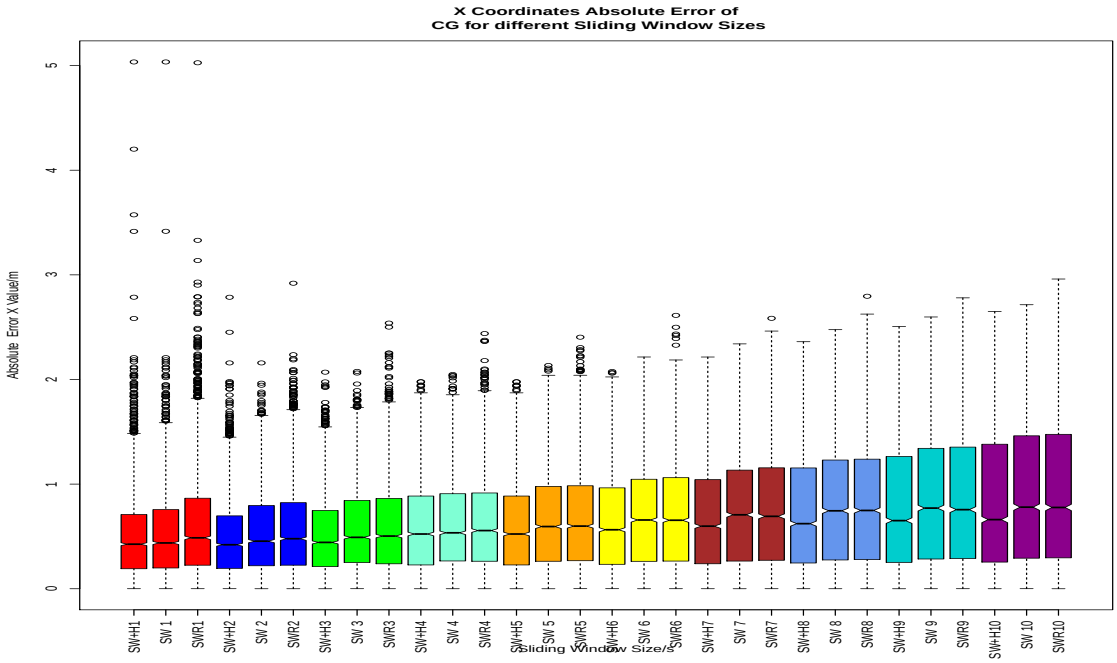


(b) Comparison of performance for UTM X Coordinates for Sliding Window Size 1-10 with No Weights, Huff Weights and Random Weights.

Figure 4.116: Comparison of Performance for Path 2 w.r.t Sliding Windows and Weights for UTM X Coordinates.



(a) Comparison of performance for UTM Y Coordinates for Sliding Window Size 1-100 with No Weights, Huff Weights and Random Weights.



(b) Comparison of performance for UTM Y Coordinates for Sliding Window Size 1-10 with No Weights, Huff Weights and Random Weights.

Figure 4.117: Comparison of Performance for Path 2 w.r.t Sliding Windows and Weights for UTM Y Coordinates.

4.12.1 Deriving Orientation from Quadratic Optimization Estimates of Locations of GPS Sensors For A Moving Platform

Using the methodology described in Section 3.2, the estimates for orientation of the platform were obtained for Path 1, as shown in Figure 4.118. From this graph, we can see that there are some negative values of orientation. This is due to the fact that in polar coordinates, negative angles of orientation are the same as subtracting the magnitude of the angle in radians from 2π. Overall, the orientation matches the actual heading of the vehicle calculated by the simulation. To compare performance, we calculate the absolute error of orientation per time-step $error_t^{\theta}$, as shown in Equation 4.2 as the absolute difference between the heading denoted by θ_t and the estimated orientation calculated from the location estimates obtained from quadratic optimization denoted as $\widehat{\theta}_t$.

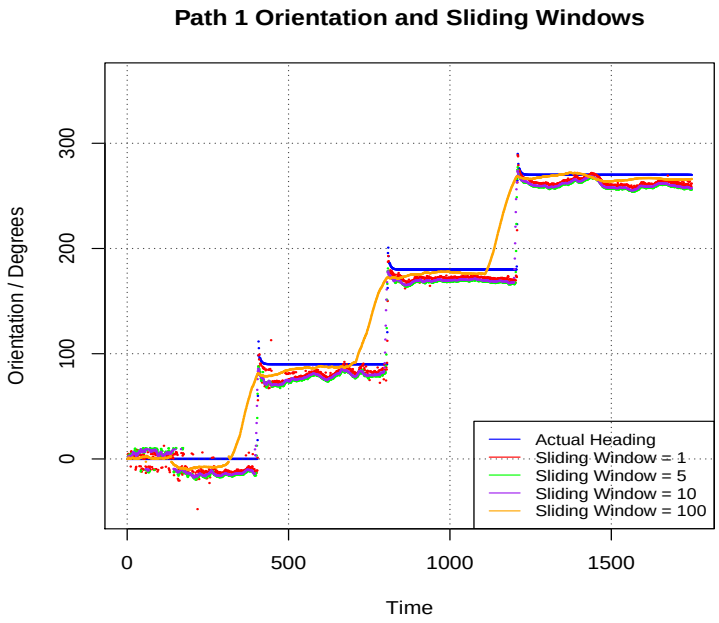


Figure 4.118: Path 1 Orientation of Platform vs. Time for Various Sliding Windows = 1,5,10,100.

$$error_t^{\theta} = \left|\theta_t - \widehat{\theta}_t\right| \tag{4.2}$$

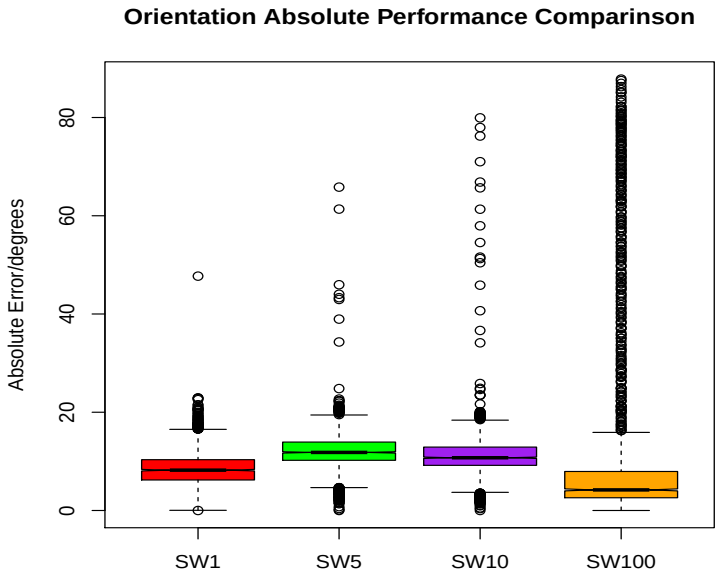


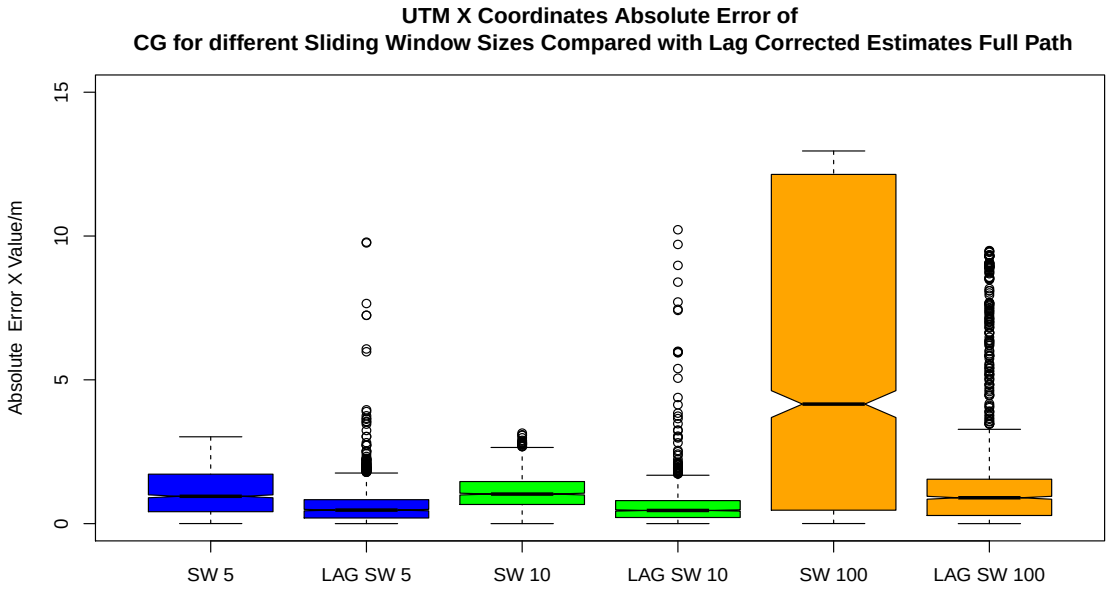
Figure 4.119: Box Plot Comparison Path 1 Orientation Error w.r.t Sliding Windows = 1,5,10,100.

The comparison on the basis of sliding window size to orientation is shown in Figure 4.119. From the box plot of absolute errors, we can note the large number of outliers when the sliding window = 100 although the median error is even lower than that of sliding window = 1. Therefore, if we can use separate sliding window sizes for estimating orientation and for estimating location, there is a possibility of being able to harness the best performance from both worlds. This is in part, the motivation for correction of lag faced in sliding window = 100, which will further reduce the number and magnitude of the outliers.

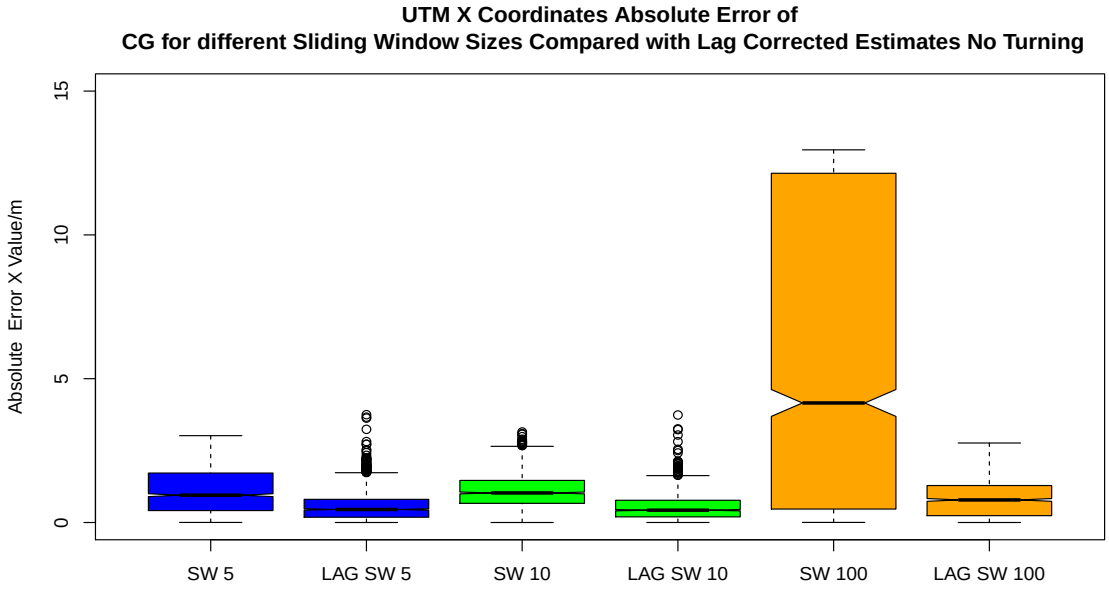
4.12.2 Performance Using Correction For Lag

Using the algorithm outlined in Section 3.4.2 to correct for the lagging behavior for estimates produced using large window sizes, we obtained the performance metrics compared in Figures 4.120 for x-coordinates and 4.121 for y coordinates Box Plot. These plots were calculated using the performance metrics defined in Equation 4.1. Figure 4.120(a) shows the performance when the entire path of the platform is considered while Figure 4.120(b) considers the path where the platform is not turning. Since it is expected in a fully-developed environment that the control information will be provided regarding whether the platform is turning or not, we simulate that by manually selecting sections of time-steps around which turning occurs. More precise results may be obtained through the collection of data from a more independent mobile platform application environment.

The performance metric the platform using lag correction is compared with performance using sliding windows and no lag correction at all. Additionally, the comparisons are not made with respect to sliding window = 1 since prediction and lag correction in that case is unnecessary. We can observe a significant decrease in the median error values for all large window sizes. The number of outliers for the case where the full path is considered is much higher than that for the no-turns path due to the fact that the correction algorithm is still using the large window size estimates that contain outdated observed values. To alleviate those errors, the adaptive/dynamic sliding window size technique has been used in Section 4.12.3.

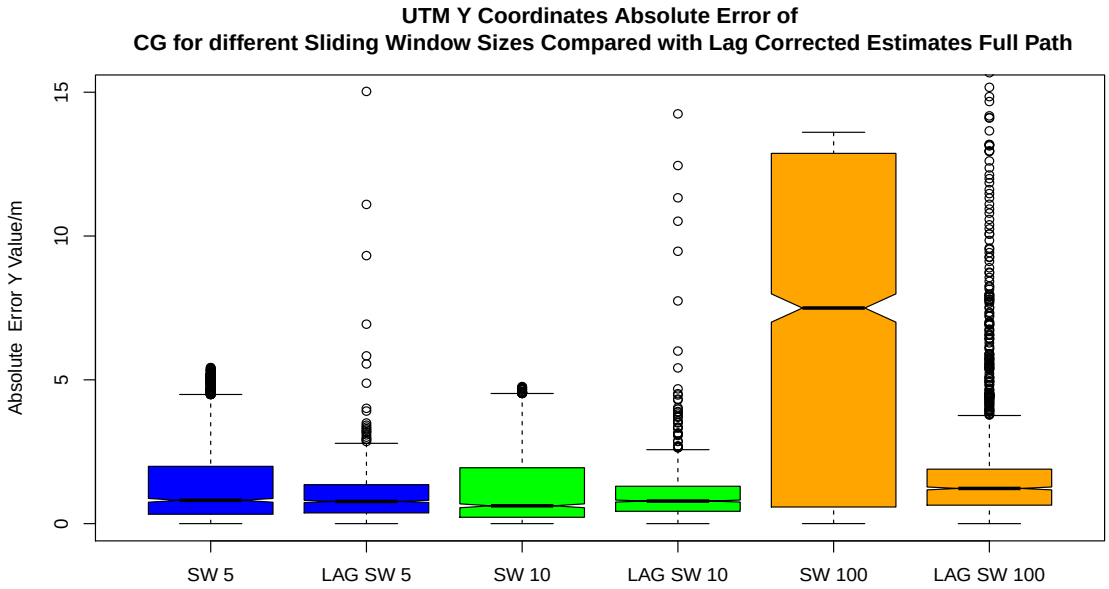


(a) Comparison of performance for UTM X Coordinates for Sliding Window Size 5-100 with Full Path Using Lag Correction.

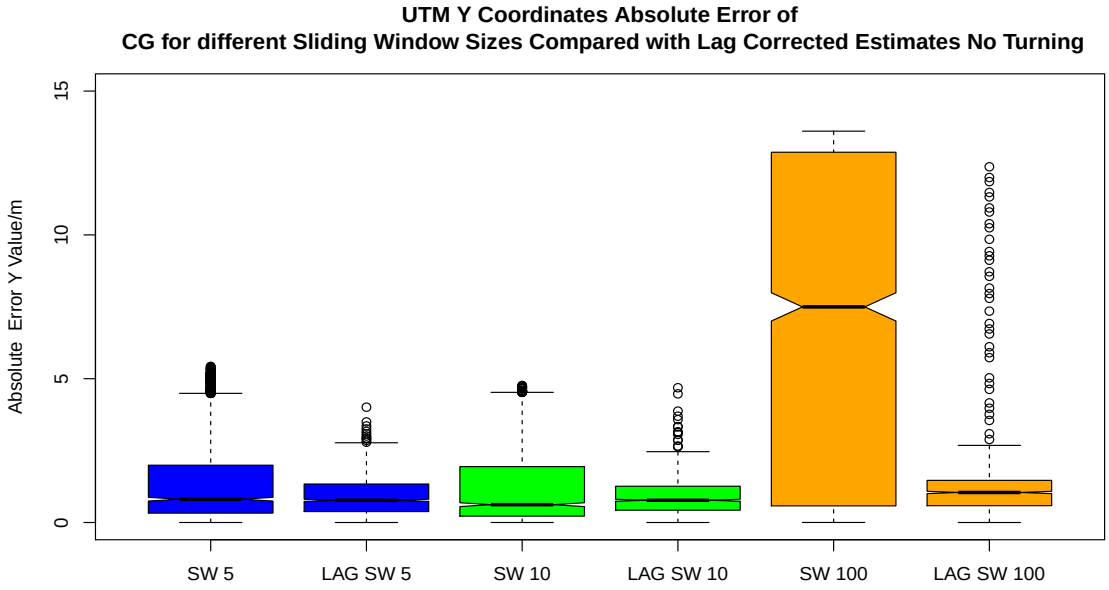


(b) Comparison of performance for UTM X Coordinates for Sliding Window Size 5-100 with No Turns On Path.

Figure 4.120: Comparison of Performance for Path 1 w.r.t Lag Correction for Full Path and No Turning Path Using Lag Correction for X-Coordinates.



(a) Comparison of performance for UTM Y Coordinates for Sliding Window Size 5-100 with Full Path Using Lag Correction.



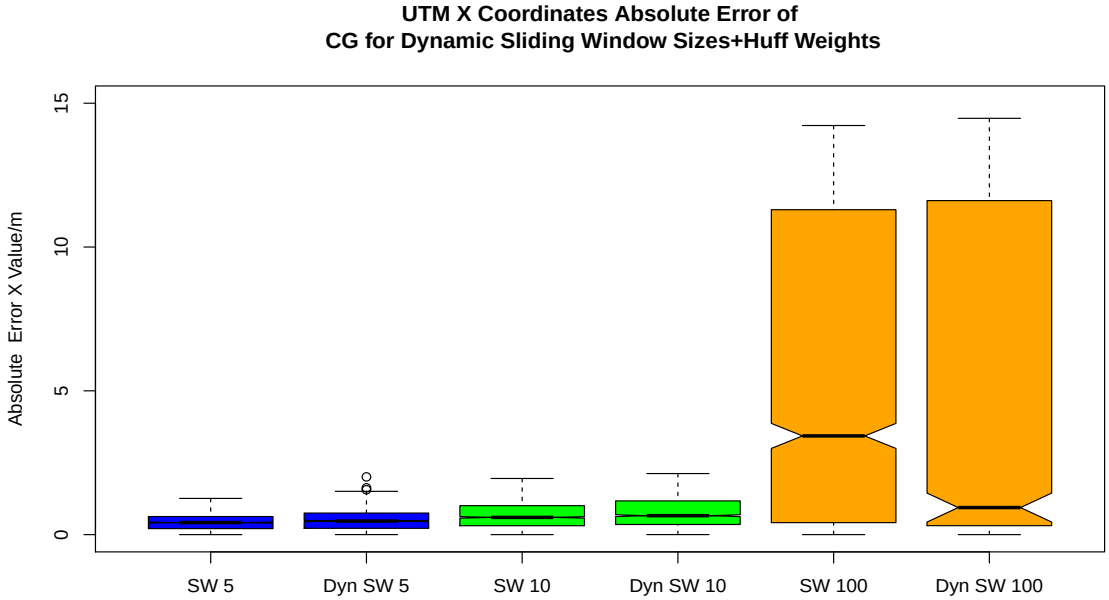
(b) Comparison of performance for UTM Y Coordinates for Sliding Window Size 5-100 with No Turns On Path.

Figure 4.121: Comparison of Performance for Path 1 w.r.t Lag Correction for Full Path and No Turning Path Using Lag Correction for Y-Coordinates.

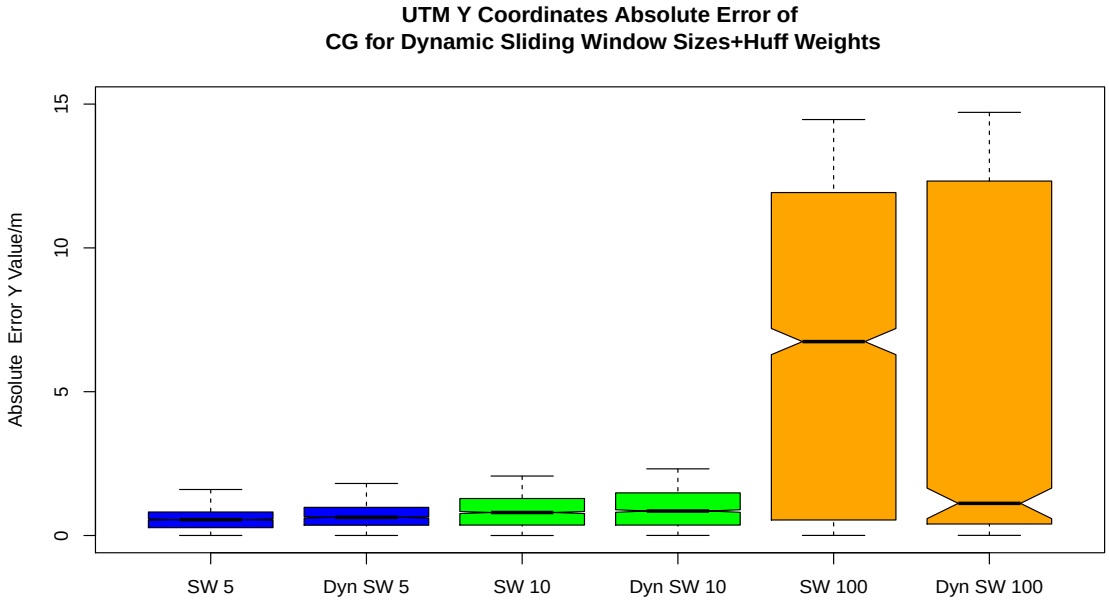
4.12.3 Performance Using Dynamic Sliding Window Size

Figure 4.122 compares the use of dynamic sliding window size with no correction for lag and where the platform uses large sliding window sizes when the path is straight and a sliding window of size 1 when the platform is turning. As described in Section 3.4.2.1, the use of the smaller sliding window size when the platform is turning or changing orientation is because the observed data needs to updated quickly and maintaining historical information is not needed.

The box-plots of the errors compare the performance of the estimates using dynamic sliding windows size vs. static sliding windows sizes for Huff weighted estimates, and there is not much difference in the median values of the sliding window sizes 5 and 10 but there is a significant drop in the median when using sliding window = 100. This implies that there may be a significant decrease in error based performance metric if both dynamic sliding window sizes are combined with correction for lag, which is discussed in Section 4.12.4



(a) Comparison of performance for UTM X Coordinates for Sliding Window Size 5-100 using Huff Weights and Dynamic Sliding Window Size.

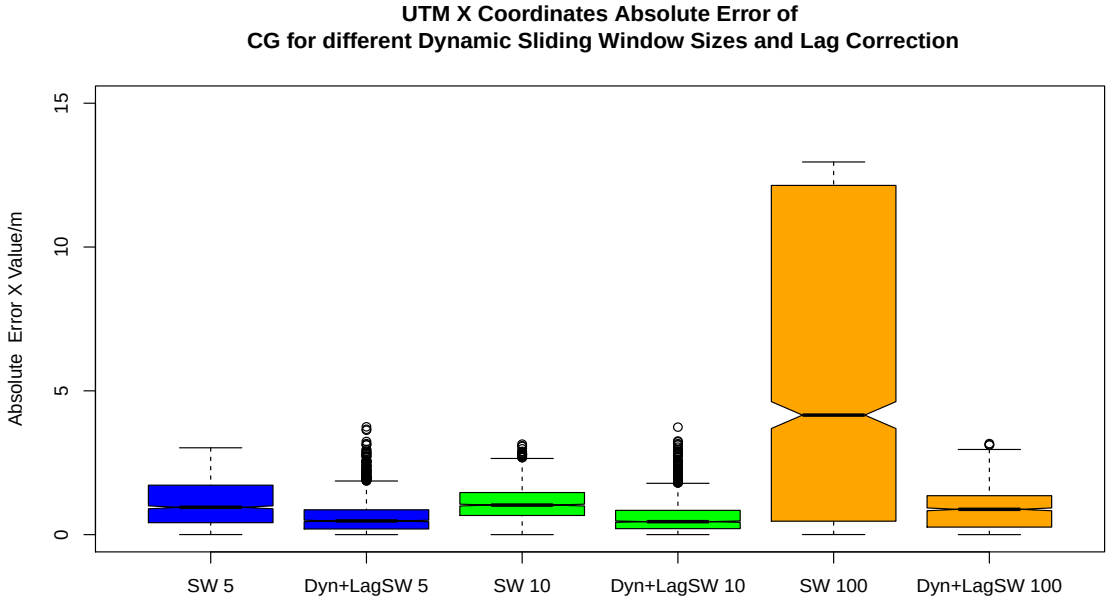


(b) Comparison of performance for UTM Y Coordinates for Sliding Window Size 5-100 using Huff Weights and Dynamic Sliding Window Size.

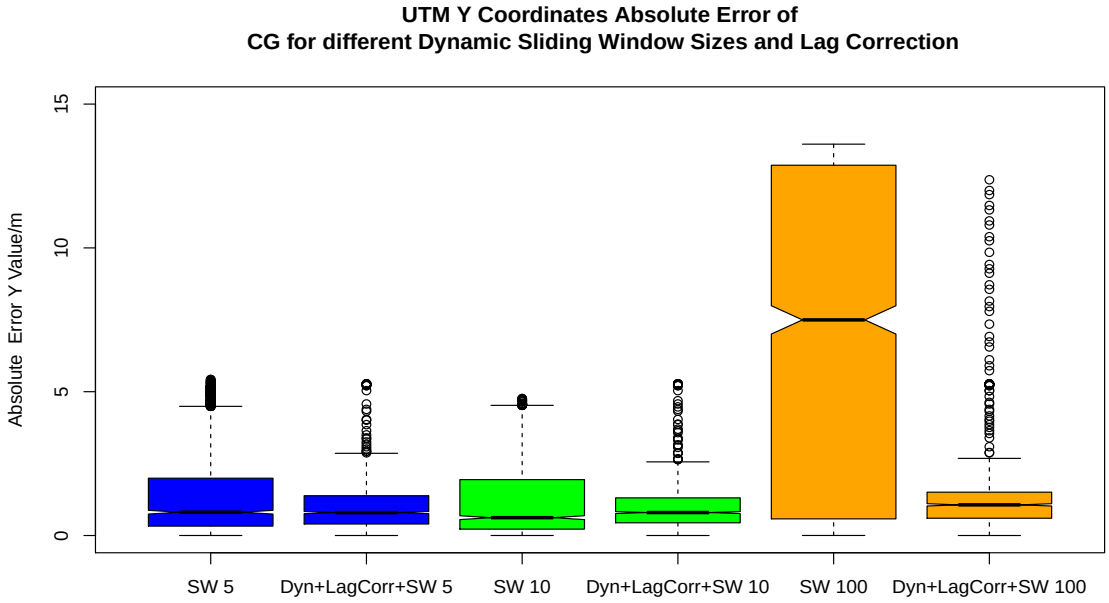
Figure 4.122: Comparison of Performance for Path 1 w.r.t Huff Weights and Dynamic Sliding Window Size.

4.12.4 Performance Using Correction For Lag Combined with Dynamic Sliding Window Size

The performance metrics for estimating the location of the center of gravity of the platform by combining the correction for lag and dynamic sliding window size for the full path is demonstrated in Figure 4.123(a) for the x-coordinates and Figure 4.123(b) for the y-coordinates. The box-plots show an increase in the number of outliers from the previous box plots. This is due to the fact that the location estimates over the entire path are being taken into consideration, and the outliers can be traced back to the use of sliding window = 1 during the turning sections. However, on comparison with lag correction without dynamic sliding window sizes, there is a significant decrease in the number of outliers. Therefore, we can conclude that both the correction for lag and the dynamic sliding window size techniques will be instrumental in producing more accurate estimates for the locations of the sensors and platforms.



(a) Comparison of performance for UTM X Coordinates for Sliding Window Size 5-100 using Lag Correction and Dynamic Sliding Window Size.



(b) Comparison of performance for UTM Y Coordinates for Sliding Window Size 5-100 using Lag Correction and Dynamic Sliding Window Size.

Figure 4.123: Comparison of Performance for Path 1 w.r.t Huff Weighted Dynamic Sliding Window Size.

4.12.5 Summary of Performance Metrics for Lag Correction and Dynamic Sliding Window Size

The following set of tables summarize the improvements in performance described in the previous sections using correction for lag, dynamic sliding window sizes and their combinations in comparison to not using any enhancements, as described in Sections 4.12.2, 4.12.3 and 4.12.4. Table 4.1 shows a summary of the mean, median and the standard deviation of absolute error computed using Equation 4.1 for the UTM x-coordinates, while Table 4.2 shows the same for y-coordinates. In Tables 4.1 and 4.2, "LC" stands for correction for lag, "Dyn SW" stands for using dynamic sliding window sizes only, and "LC + Dyn SW" stands for using both techniques together.

Window Size	**Technique**	**Mean**	**Median**	**Standard Deviation**
5	None	1.095	0.950	0.795
5	LC Full Path	0.629	0.471	0.723
5	LC No Turns	0.568	0.453	0.505
5	Dyn SW	0.510	0.472	0.337
5	LC + Dyn SW	0.630	0.476	0.588
10	None	1.064	1.028	0.562
10	LC Full Path	0.62	0.458	0.781
10	LC No Turns	0.541	0.428	0.447
10	Dyn SW	0.768	0.659	0.503
10	LC + Dyn SW	0.606	0.451	0.546
100	None	5.564	4.16	4.691
100	LC Full Path	1.454	0.9	2.052
100	LC No Turns	0.879	0.785	0.971
100	Dyn SW	5.691	0.941	5.818
100	LC+ Dyn SW	0.923	0.879	0.731

Table 4.1: Summary of Performance Metrics for UTM X-Coordinates Comparing Mean, Median And Standard Deviation of the Absolute Error of Location Estimates for Center of Gravity for Mobile Platform.

Window Size	Technique	Mean	Median	Standard Deviation
5	None	1.639	0.81	1.768
5	LC Full Path	0.939	0.773	0.85
5	LC No Turns	0.895	0.772	0.63
5	Dyn SW	0.674	0.631	0.395
5	LC + Dyn SW	0.982	0.797	0.811
10	None	1.419	0.620	1.562
10	LC Full Path	0.969	0.785	1.07
10	LC No Turns	0.874	0.773	0.596
10	Dyn SW	0.926	0.852	0.626
10	LC + Dyn SW	0.963	0.798	0.791
100	None	5.126	5.889	4.355
100	LC Full Path	1.811	1.224	2.287
100	LC No Turns	1.232	1.044	1.342
100	Dyn SW	5.939	1.117	5.981
100	LC+ Dyn SW	1.325	1.066	1.410

Table 4.2: Summary of Performance Metrics for UTM Y-coordinates Comparing Mean, Median And Standard Deviation of the Absolute Error of Location Estimates for Center of Gravity for Mobile Platform.

From Tables 4.1 and 4.2, we can surmise any method of correction for lag or dynamic sliding window sizes has a lower mean error, median error and standard deviation of error than not using any method at all. All of these values also agree with the findings discussed in the previous sections. For sliding windows of size 10 and 100, using correction for lag on the entire path has a higher error mean, median and standard deviation than using correction for lag with dynamic sliding window over the entire path. Furthermore , the mean, median and standard deviation of absolute error show that use of dynamic sliding window sizes solely does not scale well as the size of window increases. Overall, the use of both correction for lag and dynamic sliding window size shows the most promise for reducing the absolute error in the location of the center of gravity of the platform.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Concluding Remarks

To summarize the results from the previous chapters, we formulated a quadratic optimization based solution to estimate the location of multiple sensors arranged in a particular configuration and the center of gravity of the mobile platform. The quadratic optimization was performed by minimizing the cost of the squared errors between the observed data and the estimated locations and constraining the results to the configuration or the distances and orientations between the sensors (also our prior knowledge). Through the results shown in Chapter 4, we have shown the success of this formulation when the platform is stationary or when the platform is in motion, thus adding to the generality of our solution. The results in Chapter 4 demonstrate the low error of the quadratic optimization results with respect to the calculations of center of gravity of the platform when the platform is in motion.

We recognized the need for outlier analysis by noting that the observed values contain a high degree of noise (high bias and high variance), which cannot fully be counteracted by the use of quadratic optimization. In that regard, two statistical methods of weight functions were introduced, the Huber Weight Function and the Standardized Deleted Residuals for outlier analysis that were tested for stationary cases while the Huff Weight Function was developed to handle both stationary and non-stationary cases. In the case of the Huber weight function and the standardized deleted residuals, the metrics used to assign differing weights, consider the x-coordinates and y-coordinates per sensor separately. The residuals in both these

methods and therefore, their weights are calculated separately for x-coordinates and y-coordinates. This runs counter to the intuition of the application environment where distance is the paramount metric. The Huff weight function metrics utilize a distance based residual, combining the x-coordinates and the y-coordinates to one value through calculating the Euclidean distance for the purposes of assigning weight. Additionally, the use of Huff weights over sliding windows showed a marked improvement in performance over using no weights or random weights. We can conclude that the use of Huff weights can be beneficial to producing more accurate estimates.

We also introduced the concept of sliding windows to be able to obtain real-time results when the platform is in motion. Through our performance analysis, sliding windows were deemed as a worthy option in dealing with missing data and a sound tool in producing better location estimates that have real-time constraints, which given that the application space contains possibly cheap and unreliable sensors. As shown in Chapter 4, the balance between retaining a large amount of history and maintaining the value of the most recently observed data point can be done through changing the size of the sliding window. With large sliding window sizes, we observed the lag effect and successfully calculated the change in orientation of the platform from a reference position, thus allowing us to correct for the lag. We also investigated correction for the lag and the use of dynamic sliding windows sizes together and can conclude through implementation of both concepts on the application environment, that there is a significant decrease in the median error, either when using large or small window sizes, compared to using no correction or dynamic sliding window.

All of the previously discussed methodologies are tools essential to formulating the best estimates of locations of the sensors but there remains other options that require further investigation, discussed in Section 5.2.

5.2 Future Work

As discussed in Chapter 3 and 4, we implemented a prototype system to estimate the location of a mobile platform using quadratic optimization and by improving the estimates through the application of weights. Also, the simulation is capable of handling the production of estimates when the platform is in motion. We observed the lag effect for large sizes of sliding window which can be corrected to obtain more current location estimates. To test the true effectiveness of the system, the next phase of implementation will involve applying the principles of the prototype in a real-world environment.

The purpose of these methodologies was also to formulate a general solution for calculating estimates of different types of sensors. Although in this dissertation, the focus has been on position and orientation information, we are keen on future applications of this framework to sensors that measure velocity or altitude etc. This will possibly involve the introduction of multiple features besides position to the framework and also increasing the number of constraints to describe the configuration of sensors. We are also interested in looking at different sensor configurations and would like to investigate the effect of more complex shapes on the complexity of the quadratic optimization problem. The issue of altitude will expand the current 2-D framework to a 3-D space. Moreover, the additional issues of rotation on multiple axis will append more dimensions to the current problem space.

This framework can be further expanded to account for tracking of locations of large vehicles such as trains, airplanes or boats and other forms of transportation, and may be successfully tested in the presence of tracking data being collected from these application spaces. Moreover with trains, the constraints could contain the configuration of the network of train tracks, thus adding to the dimensionality of our basic optimization framework solution.

During the process of investigating solutions for the problem statement, we encountered the use of the extended Kalman Filter algorithm for the process of filtering noisy data from the observed data from GPS sensors. We intend to implement a system that takes advantage of extended Kalman Filter and the quadratic optimization framework using sensor configuration knowledge to result in a system that can define approaches on both sides of the problem statement : the filtering of the data AND the sensor configuration knowledge.

REFERENCES

[1] R. Freund. (2004) Nonlinear programming. [Online]. Available: http://ocw.mit.edu/courses/sloan-school-of-management/15-084j-nonlinear-programming-spring-2004/lecture-notes/

[2] MATLAB. (2012) Constrained nonlinear optimization. [Online]. Available: http://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[3] F.-T. Y. Katta G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*. University of Michigan, Ann Arbor, 1997.

[4] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Application with R Examples*, 2nd ed. Springer-Verlag, 2006.

[5] T. P. S. University. (2012) Stat 510 - applied time series analysis. [Online]. Available: https://onlinecourses.science.psu.edu/stat510/?q=node/33

[6] G. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis : Forecasting and Control*, 4th ed. Wiley Series in Probability and Statistics, 2008.

[7] A. C. Acock, "Working with missing values," *Journal of Marriage and Family*, vol. 67, pp. 1012–1028, Nov. 2005.

[8] A. Donders, G. van der Heijden, T. Stijnen, and K. G. Moons, "Review : A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 65, pp. 1087–1091, Jan. 2006.

[9] D. C. Montgomery, *Statistical Quality Control : A Modern Introduction*, 6th ed. John Wiley and Sons, 2011.

[10] NIST and SEMATECH. (2012) Univariate and multivariate control charts. [Online]. Available: http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc3.htm

[11] R. L. Mason, N. D. Tracy, and J. C. Young, "A practical approach for interpreting multivariate t^2 control chart signals," *Journal of Quality Technology*, vol. 29, pp. 369–501, Oct. 1997.

[12] M. G. de la Parra and P. Rodriguez-Loaiza, "Application of the multivariate t^2 control chart and mason-tracy-young decomposition procedure to the study of the consistency of impurity profiles of drug substances," *Quality Engineering*, pp. 127–142, Aug. 2003.

[13] D. Koks and S. Challa, "An introduction to bayesian and dempster-shafer data fusion," Australian Government Department of Defence : Defence Science and Technology Organisation, Australia, Tech. Rep. AR-012-775, Nov. 2005.

[14] Z. Yi, H. Y. Khing, C. C. Seng, and Z. X. Wei, "Multi-ultrasonic sensor fusion for mobile robots," *IEEE Intelligent Vehicles Symposium*, Oct. 2000.

[15] M.-A. Simard, E. Lefebvre, and C. Helleur, "Multisource information fusion applied to ship identification for the recognised maritime picture," *Sensor Fusion : Architectures, Algorithms, and Application IV*, vol. 4051, Nov. 2000.

[16] F. Cremer, J. Schavemaker, E. den Breejen, and K. Schutte, "Detection of anti-personnel land-mines using sensor fusion techniques," *EuroFusion*, vol. 4051, Oct. 1999.

[17] G. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive- integrated moving average time series models," *Journal of the American Statistical Association*, vol. 65, pp. 1509–1526, Dec. 1970.

[18] ROS. (2012) Ros documentation. [Online]. Available: http://www.ros.org/wiki/

[19] ——. (2012) Ros documentation for gps common. [Online]. Available: http://www.ros.org/doc/api/gps_common/html/msg/GPSFix.html

[20] G. Baddeley. (2001) Gps - nmea sentence information. [Online]. Available: http://aprs.gids.nl/nmea/

BIOGRAPHICAL STATEMENT

Roochi Mishra was born in Algiers, Algeria and raised in India, Malaysia, Uzbekistan and Switzerland. She received her B.S degree in Computer Science and Engineering from University of Texas at Arlington, Texas, in 2004, her M.S degree in Computer Science from University of Pittsburgh, Pennsylvania , in 2006, and is currently working on obtaining her Ph.D. degree in Industrial Engineering from University of Texas at Arlington, Texas in 2013. In 2003, she interned at Sprint Inc. as a system administrative intern and in 2010, she worked as Research and Development Software Testing Engineer for Firmware and Software in Boston Scientific Neuromodulation. Her current research interests are in Predictive Analytics in Big Data, Data Science and Linguistics.