Radio based Cooperative Positioning for Vehicle-to-Vehicle Systems in Urban Scenarios

Hao Wu
wir15hwu@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor:
Fredrik Tufvesson
Meifang Zhu

Examiner: Fredrik Rusek

June 15, 2018

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This master thesis aims at improving vehicle positioning in high-speed movement scenarios where global navigation satellite system (GNSS) does not work well. The standard 802.11p, as one of the WiFi family members, it can support to share vehicle information between vehicle and vehicle or between vehicle and infrastructure in a high-speed movement environment. Without the help of GNSS, the vehicles can estimate their position by sharing position information with other vehicles. In order to reach highly accurate positioning in urban scenarios, non-linear filters, such as extended Kalman filter (EKF), square root of cubature Kalman filter (SCKF) and particle filters (PF) are investigated in this thesis. These filter algorithms are simulated in MATLAB to evaluate the positioning performance. Useful information from vehicles are used in the algorithms, such as velocity, acceleration of vehicles. To obtain realistic scenarios, vehicles are simulated in different road networks in SUMO and obtain the vehicle information from one another in GEMV^2 and NS3. SUMO, GEMV^2 and NS3 are the tools to help the simulation. In the simulations, the positioning accuracy is greatly improved when sharing vehicle information and utilizing the filter algorithm. This thesis compares the advantages and disadvantages of two filter algorithms. One is square root of cubature Kalman filter, the other is particle filter. As a conclusion from the simulation result, the SCKF works better than the particle filter and it improves the accuracy of positioning when the GNSS is not well received.

Popular Science Summary

Global navigation satellite system makes vehicles to locate themselves with the help of satellites. Each vehicles needs at least three satellites sending time information from distance between it and the satellites. With geometry tool, the vehicles can estimate their positions on the earth approximately. However, the satellite signals travel a long distance to the vehicle so that the strength of the signal is weak to be detected. These signals are easily blocked by objects, which introduces an error in the positioning estimation. With the help of signal from base station, it improves the positioning. However, the signal from base station is also reflected and degenerated by the objects in the way of its propagation. Inspired by the idea that selecting a close signal access point to navigation, the nearest access points are picked up instead, what these points have the same function as satellite and base station is to measure the distance from themselves to vehicles in this thesis. These nearest access points are vehicles, since there are a lot of them running on the road. The thesis focuses on how the vehicles can obtain good accurate positions with the help of other vehicles. Vehicles supported by distance measurement among each other is the main idea in this thesis. The WiFi standard 802.11p supports vehicle-to-vehicle communication and vehicle-to-infrastructure communication, making vehicles share their information with others easily. Each vehicle collects all information from the surround vehicles to improve its own positions.

802.11p, as a member of IEEE 802.11 standard, supports sharing vehicle information in a high-speed dynamic environment. The vehicles share their own information in two ways. The first way is to send it directly to the destination vehicles. The second way is to send it to the infrastructure road side unit or other wireless access points before sending to the target vehicles. The 802.11p standard provides broadcast protocol so that the distance measurement is estimated through the arrival time of the signal. The shared information include position estimated by GNSS, velocity, acceleration and distance measurement of each vehicle. The thesis also compares the simulation results from the Kalman filter and the particle filter as different fusing tools. Both these two filters are good at fusing shared information to obtain good positioning.

The vehicular network simulation software simulates a real road network as not all the tests can be applied in real road networks with the hardware limitation. The software such as SUMO, GEMV^2 and NS3 make vehicles run on the

simulation road network and show all the vehicle information for further analysis. These software provide us the exact vehicle locations, velocities and corresponding distances among vehicles. The vehicle information is implemented into MATLAB to evaluate the filter algorithm performance on positioning. In order to make the simulation more general and see how accurate the positioning, several scenarios are created to examine the robustness of the algorithm. Several scenarios like distance, distance-velocity, GPS-distance, GPS-distance-velocity are discussed in each scenario.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

At presents, vehicle positioning does not always work satisfactory in the urban scenario. Vehicle mostly use satellite signals to positioning, but the signals are easily influenced or blocked completely by objects in the environment. This causes an error in position estimation. Some vehicles can use the signal from nearest base station to assist and improve their position. Nevertheless, the signal from base stations face the same problem as signals from satellites, which vehicles are surrounded by tall buildings, pedestrians, running vehicles and vegetation in the urban scenario. To improve vehicle positioning performance, the 802.11p WiFi standard can be used with the cooperative positioning, which each vehicle contribute their information to others to help them improve positioning. The standard provides the vehicle-to-vehicle communication and vehicle-to-infrastructure communication to help vehicles sharing information with each other. Vehicles estimate their positions by fusing information from surrounding vehicles, which different kind of vehicle information can restrict the vehicle movement in their path. The cooperative positioning method will improve the vehicle positioning even without GNSS.

1.2 Project Aims and Approach

The master thesis aims at investigating positioning in an urban scenario without GNSS. The way is to analyze different methods and find out if the performance is satisfactory for safety related application. To achieve the goal, the thesis simulates the use of the 802.11p standard to share position information among vehicles to positioning in the software. The software helps us to simulate vehicle positioning performance with cooperative positioning, by comparing them with the ground true position intuitively. The data fusion tools for cooperative positioning used are the non-linear Kalman filter and particle filter. In all, there are three approaches needed to be done in order to make a good simulation. Firstly, the simulation road network are done with SUMO and the sharing of information from the vehicles is preformed in the vehicular simulation software, such as GEMV^2 and NS3. Secondly, the non-linear Kalman filter and particle fiter are

implemented and simulated in MATLAB. Finally, the information from vehicle is implemented as observation values in the filter to have a good position in different scenarios. Several scenarios are investigated to analyze the robustness of the filter algorithm.

1.3 Background of Techniques for Positioning

At present, vehicle use the global navigation satellite system (GNSS) to assist their positioning. The GNSS is an abbreviation for global navigation satellite system. It is a big family including Global positioning system (GPS) operated by United State, GLONASS operated by Russia, Galileo operated by European Union and BeiDou navigation system operated by China. The deviation of GNSS, taking Global Positioning System (GPS) as an example, is roughly in meters, sometimes higher. In the thesis, the GNSS positioning is assumed to be obtained from the GPS sensor, while GPS still has the largest number of users. The error is caused by inaccurate signal collection by the electronics and obstructed signal by objects being in the way of the propagating wave. These effects contribute to signal multi-path arrival and wrong arrival time. The power of signal sometimes decreases rapidly then becoming undetectable in the long distance travel. All these influences make positioning more inaccurate. Therefore, devices use cooperative methods to improve their position estimates, which makes devices positionable when GPS is not well received. The common cooperative position methods are summarized into five typical positioning estimation schemes: triangulation, scene analysis, proximity [2] self-measurements and data fusion method. These methods are used both in indoor and outdoor environments. The rest of the section describes five different kinds of cooperative methods.

1.3.1 Triangulation

Triangulation is based on the geometric properties. It estimates the measurement point P (measuring unit) with help of the known reference units A, B, C and the distances from point P to each reference unit. The configuration is shown in Figure 1.1. The reference units can be the nearest base stations, WiFi access points and other wireless access points. Instead of measuring distance to estimate position directly, there are several ways to estimate distance based on different radio properties.

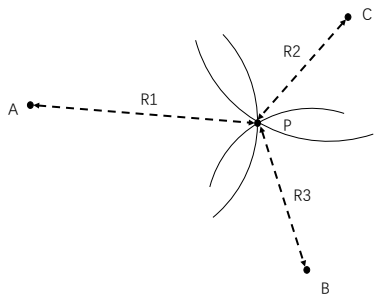

Figure 1.1: Triangulation Positioning

TOA

TOA is an abbreviation for the time of arrival measuring the time of flight (TOF) of signal in the air. What the TOF measure is the flight time from signal leaving reference to signal arriving at measurement point. Once the time of flight t is known, the distance s is calculated by relation $s = c \times t$, where c is the speed of light. Due to an unavoidable error in measuring time t, the distance estimate is not a deterministic value. The range of distance measurement is between two curve lines shown in Figure 1.2, where the position estimate of point P is in the intersection area, shown in Figure 1.2.

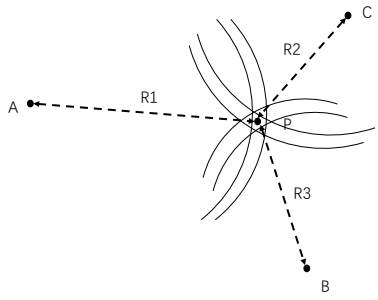

Figure 1.2: Error measurement in distance in position estimation

The final estimate of the position can be anywhere in the intersection area. In order to minimize the error, the cost function with least squares algorithm is applied to optimize the position. It is given by [2] :

$$F(P) = \sum_{i=1}^{N} \alpha_i^2 f_i^2(x), \tag{1.1}$$

where α is the weight based on the reliability of the received signal, subscript i means the information are from the i^{th} reference unit, $P = (x, y, z)^T$ is the position of point P represented by coordinate, $f(x)$ is formed as [2] :

$$f_i(x) = v \times t - \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}. \tag{1.2}$$

The location of (x, y, z) is calculated by minimizing the function $F(P)$ in (1.1).

TDOA

TDOA is an abbreviation for time difference of arrival. Instead of measuring the absolute distance shown in section TOA, TDOA [3] method determines difference distance between PA and PB, where PA and PB are the absolute distance from measuring point P to reference unit A and from measuring point P to reference unit B respectively. In each TDOA measurement, the position of P relies on the hyperboloid. It is shown by the following equation [2] :

$$D_{i,j} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2}, \tag{1.3}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) are the reference units, (x, y, z) is measuring point. The position of the target device is solved at least by measuring two arrival time in two reference points respectively. Due to the error in time measurement and time synchronization, point P is no longer a deterministic point but vary in an area as the TOA. The optimization solution is also to minimize the cost function (1.1), like TOA does. The time t in equation (1.2) is obtained by maximizing the correlation function, represented as [2] :

$$R_{x_i,x_j}(t) = \frac{1}{T} \int_0^T x_i(\tau) x_j(\tau - t) d\tau, \tag{1.4}$$

The TDOA measurement method is widely used in LTE release 9 [3]. Another method called OTDOA is used in LTE release 11 [3]. The difference between these two methods is that TDOA method estimates the position by the base station after receiving mobile phone's broadcast reference signal. OTDOA [3] method estimates the relative location by the mobile phone itself after receiving the reference signal from the base station.

RSS

RSS is an abbreviation for received signal strength. The basic principle of RSS is to estimate the distance by the strength of received signal. A statistical model for

RSS is given by [4]:

$$P(d) = P_0 - 10\gamma log_{10}d + S, \tag{1.5}$$

where $P(d)$ is the strength of received signal based on distance d, P_0 is the strength at one-meter reference point from target device, γ is the path-loss exponent and d is the distance between the receiving device and the reference unit. S is large-scale fading. The more precise model in [5] introduces standard derivation of S and γ. Both them depend on the distance. However, this method is not accurate either. The strength of the signal is affected by many aspects including multi-path signal arrival and non-line of sight (NLOS). Due to the error in distance measurement, the solution is optimized by minimizing the cost function just like equation (1.2), where f_i is represented as [2] :

$$f_i(x) = d - \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}, \tag{1.6}$$

RTOF

RTOF is an abbreviation for round-trip time of flight. It uses the same principle as TOA but calculates the back-and-forth time t_{total} instead of one-way time t. The round-trip time of flight t_{total} contains back and forth time that the signal travels in the air from measuring unit to reference unit. Assuming these two units are static, the signal spends the same time t in the air. Therefore, the relation is shown in the following equation [2] :

$$t_{total} = 2 \times t + \Delta t, \tag{1.7}$$

where Δt is the processing time in the repeating unit from it is receiving the reference signal until it is sending the reference signal back to the measuring unit. After estimating the time t, the measuring unit position is estimated by the least square method.
Synchronization is the hardest part in the communication system. This method avoids those issues by using a relative synchronization clock embedded inside reference unit itself. However, the processing time in measuring unit is uncertainty depending on different kinds of hardware. The hardware also causes some unavoidable error in the time measurement.

PDOA

PDOA is an abbreviation for phase-difference-of-arrival. It assumes that the reference unit transmits two pure cosine waves at the same time with different frequency f_1, f_2 and the same initial phase. After propagating in the medium, measuring unit extracts the phases difference between these two phases Δp shown in following relation equation [2] :

$$\Delta p = 2\pi(f_1 - f_2)t, \tag{1.8}$$

Since Δp and two reference frequencies are known, the propagation time is solved by the above equation. Hence the distance can be calculated. In this method, these two cosine waves are assumed that suffering from the same environment influence. The strength of signal does not affect the phase difference Δp.

DOA

DOA [2] is an abbreviation for direction of arrival. Assuming that measuring unit P sends a signal to two reference units A and B. Reference units A and B are able to measure the direction of the incoming signal. Then they communicate each other to decide where the signal comes from as Figure 1.3 is showing. After calculating the the position in the reference point A and B, they send the position information back to the measuring unit.

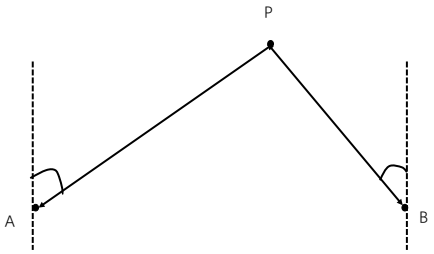

Figure 1.3: Positioning Concept of DOA

In order to measure the angle of arrival of signal, the 2D antenna array implemented in reference units A and B are the basic requirement to be capable of detecting the angle of arriving signal.

1.3.2 Proximity

The proximity algorithm [2] estimates position relating to reference unit position. The position of measuring unit is based on the dense grid of reference units continuously receiving all kinds of signals. Each reference unit has its own position. The measuring unit considers the position of reference unit who receives the strongest strength of signal, as its position. RFID [6], CELL-ID are the representative method based on proximity. Proximity is widely used with low cost, no dedicated hardware and synchronization requirement.

1.3.3 Scene Analysis

Scene Analysis is widely used in indoor positioning. It also called "Fingerprints [7]". It collects some properties of often signal in every position. Then, build a database to store these properties. The properties could be strength of signal, electromagnetic field of signal in X, Y, Z directions. The measuring unit collects

the signals and compares them with the references stored in the database. The estimate is given as position which has the most properties of signal, as the device position.

1.3.4 Self Measurement

Some electronic devices have a self-measurement [4] system. They comprise a small IMUs unit based on microelectromechanical systems (MEMS) for measuring angular velocity and specific force. The specific force is acceleration combining the gravitational and the inertial linear acceleration. Each IMU contains three orthogonal accelerometers and three orthogonal gyroscopes. Knowing these parameters, standalone inertial navigation (INS) is possible with error free angular velocity and acceleration measurement. If acceleration measurement or angular velocity measurement has a small error, the error will increase accumulatively in a long time tracking. In practice, inertial navigation is supplementary to GPS to get an accurate positioning. This is an alternative tool when GPS do not work well.

1.3.5 Data Fusion

Due to the different accuracy requirements in positioning, combining different methods to optimize the solution might be necessary. Let's take one example, using visual information to calibrate the positioning [8] and using inertial navigation based on IMUs and Kalman filter to correct the positioning [9] are better than only one of these methods is used. Kalman filter is adapted to many areas, such as the virtual reality, GamePad, indoor and outdoor positioning. The next chapter focuses on the principle and derivation of Kalman filter, including its modified Kalman filter square root of cubature Kalman filter (SCKF).

1.4 Thesis Structure

In the thesis, the content is divided into three main parts. The first part is to introduce the construction of the Kalman filter algorithm and particle filter algorithm. These two algorithms, as the main filter algorithms, will be compared in the following chapter. The second part is to simulate the real road network through widely-used vehicular simulation software. All the tests are based on simulation data instead of real data. The third part is about implementing the simulation information into MATLAB for further analysis. This part is to understand how accurate the positioning could be by the cooperative algorithm.

Chapter 2
Cooperative Positioning

As higher and higher accuracy of positioning requirement in outdoor, the positioning algorithm can also be used to assist with satellite positioning. One of the typical positioning algorithms is the Kalman filter. The principle of the Kalman filter is based on the process model and measurement model. In the statistics point of view, most noise can be represented as Gaussian model with one mean value m_e and one variance value σ^2, written as $N(m_e, \sigma^2)$. Many extension are based on the original Kalman filter. They include cubature Kalman filter (CKF) [11], unscented Kalman filter (UKF) [12], square root Kalman filter (SCKF) [11], extended Kalman filter (EKF) [10] and so on. In the thesis, original principle of Kalman filter is described in detail.

2.1 Kalman Filter Algorithm

The original Kalman filter algorithm [13] consists of two steps. The first step is to estimate the state vector in prediction step at time k and the second step is to estimate the measurement values in update step up to time k, denoted by $z_{1:k}$. This filter problem is described to compute the probability density function $p(x_k|z_{1:k})$. This probability density function $p(x_k|z_{1:k})$ calculates prediction distribution $p(x_k|z_{1:k-1})$ and then updates to the distribution $p(x_k|z_{1:k})$. In the prediction step, $p(x_k|z_{1:k-1})$ is computed by the previous states which is updated by $p(x_{k-1}|z_{1:k-1})$ at time $(k-1)$ [10] :

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}, \tag{2.1}$$

where $p(x_{k-1}|z_{1:k-1})$ is known due to recursion in every time step and $p(x_k|x_{k-1})$ is computed by equation (2.3). The distribution over x_k is considered as a prior estimate. After receiving the most recent measurement z_k, the distribution over x_k is updated to [10] :

$$p(x_k|z_{1:k}) \propto p(z_k|x_k)p(x_k|z_{1:k-1}). \tag{2.2}$$

In general, the original Kalman filter is optimal choice but it is hard to adapt to highly dynamic environment due to the long time consuming computation and

large memory needed. It can not be carried out analytically, but can be implemented by Monte Carlo sampling. This simplifies the process that the Kalman filter only use the previous states.

Structure of Algorithm

The Kalman filter is the optimal choice due to the fact that the noise is assumed to be Gaussian distribution. Even through there is an error-estimated in measurement caused by noise, the Kalman filter still have capability of solving the problem. The structure of Kalman filter are defined as the a following serial of equations [10] :
Process Equation

$$X_{k|k-1} = FX_{k-1|k-1} + B\mu_k + w_{k-1}, \tag{2.3}$$

Measurement Equation

$$Z_{k|k-1} = HX_{k|k-1} + v_k. \tag{2.4}$$

$X_{..|..}$ is a column vector containing all states. H is a transformation matrix determining the measurement transformation from the states. The method to solve matrix F is called the observer design problem state. The content in the matrix F is not necessary to be known. What the algorithm really want is the outputs of matrix F ($m \times m$). Matrix B ($m \times l$) is to make an optional control on extra values μ_k ($l \times 1$). When B is set to be 0, it means there is no extra values to adjust the input signal. Z is transformation matrix showing the predicted measurements, which is compared with the real measurements collected from sensors. The predicted measurement values are obtained by equation (2.4), which transforms the internal predicted state $X_{k|k-1}$ into measurement (observation) value $Z_{k|k-1}$. In the original Kalman filter, the descriptions about the transformation from $X_{k|k-1}$ to $Z_{k|k-1}$ are linear combinations. It means that the system F and H are all linear. As for nonlinear solutions, they are solved in the following section with extension of Kalman filter or particle filer. The subscript in each element is to distinguish notation in the different time steps. For example, subscript $(k|k-1)$ means that the states is in k step, but the states are calculated based on the previous states in $(k-1)$ time step. The random value w and v are the process noise and measurement noise respectively. Assuming all the noise are modeled as the Gaussian distribution, they are independent of each other. The noise are expressed by [10] :

$$p(w) \sim N(0, Q), \tag{2.5}$$

$$p(v) \sim N(0, R). \tag{2.6}$$

In practice, the covariance of process noise Q and the covariance of measurements noise R could be different when the algorithm is used in different scenarios. But in the thesis, these two noises variance are assumed to be constant. The consistent noise covariance increase the error in estimating current state vector $X_{k|k}$. Affected by different Gaussian noise w and v, predicted states $X_{k|k-1}$ and observation values $Z_{k|k-1}$ have different mean values and covariances respectively like $X_{k|k-1} \sim N(FX_{k-1|k-1}, Q)$ and $Z_{k|k-1} \sim N(HX_{k|k-1}, R)$.

The main idea of the Kalman filter is to use the process equation (2.3) to predict the internal predicted states $X_{k|k-1}$, which is based on the old a posterior state $X_{k-1|k-1}$. Implement predicted (a priori) state to estimate $X_{k|k-1}$ and the corresponding predicted measurement value $Z_{k|k-1}$ in equation (2.4) to update the current state vector (a posterior state vector) $X_{k|k}$. The relationship among these three different states is presented by the following equations [10] :

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - Z_{k|k-1}), \tag{2.7}$$

where Z_k is an observation value from sensors in electronic device and K_k represents Kalman gain in time step k. It determines how much difference is added to the priori states to obtain more accurate states $X_{k|k}$. The principle of obtaining Kalman gain K_k is to minimize a posterior error covariance $P_{k|k}$. It is represented as [10] :

$$P_{k|k} = E\{(X_{k|k} - \hat{X}_{k|k})(X_{k|k} - \hat{X}_{k|k})^T\}, \tag{2.8}$$

where $X_{k|k}$ is an updated states obtained from equation (2.7) and $\hat{X}_{k|k}$ is the average value among $X_{k|k}$. One of solutions to get Kalman gain is to minimize a posteriori error covariance. It is provided by the Brown and Hwang mentioned in [14]. Hence, the expression of Kalman gain is shown as [10] :

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}, \tag{2.9}$$

where $P_{k|k-1}$ is a priori error covariance having the same description as a posterior error covariance $P_{k|k}$. A priori error covariance is shown as [10] :

$$P_{k|k-1} = E\{(X_{k|k-1} - \hat{X}_{k|k-1})(X_{k|k-1} - \hat{X}_{k|k-1})^T\}, \tag{2.10}$$

when K_k infinitely approaches to 0, it means that the measurement from sensors Z_k are much more reliable. Otherwise, the predicted measurement values $Z_{k|k-1}$ are better. The difference $(Z_k - Z_{k|k-1})$ is referred to measurement innovation or residual. If the measurement innovation is 0, it means that measurements from sensor Z_k and predicted measurement $Z_{k|k-1}$ are totally matched. Otherwise, both the measurement innovation and Kalman gain would affect the state update simultaneously. In conclusion, the whole structure of Kalman filter can be derived as [10] :

The Structure of the Kalman Filter

Time Update:

1) Assume old posterior density function is known at time step k:

$$X_{k|k-1} = F\hat{X}_{k-1|k-1} + Bu_k$$

2) Error covariance

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_{k-1}$$

Measurement Update:

1) Kalman Gain

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$$

2) Updated State

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - HX_{k|k-1})$$

3) New Posterior covariance

$$P_{k|k} = (I - K_kH)P_{k|k-1}$$

Constant Model and Dynamic Model

The Kalman filter is modeled as a constant model with the fixed process and fixed measurement transformations (fixed F and H). It also can be modeled as a dynamic model combining many physical models. If a dynamic model is implemented, the algorithm converge only if one of the models converge. So that both states and system will tend to be steady once again. Although the dynamic model is a good way to reflect complex phenomenon, the Kalman filter needs to re-initialize the parameters when one of model fails to express [10]. The constant model is mainly used in the thesis.

Discussion about Error Covariance P and Noise Covariance Q, R

In practice, the measurement noise covariance R is obtained from the sensor measurements. Import the off-line pilot sequence to the information signal and send it to the device along with useful signal. The destination device estimates the noise covariance with off-line pilot. The measurement noise is generated when the signal propagate in the medium such as copper wire, optical fiber and air. It is also generated in such way that the signal is collected by electronic devices and they are affected by the temperature and the inherent error estimate. Fortunately, the most sources of noise are represented with a Gaussian model with one mean value and one variance value in the statistics point of view. In practice, the measurement noise covariance is always set a little larger. The critical parameters in the Kalman filter is the ratio between error covariance $P_{k|k-1}$ and measurement noise R. Both of them determine the Kalman gain K_k in equation (2.7). If $P_{k|k-1}/R$ [17] is underestimated, the Kalman filter converges very slowly. This is not good for estimation in a highly dynamic environment. If $P_{k|k-1}/R$ is overestimated, the Kalman filter has a quick convergence but unstable state estimates due to large measurement noise R. The measurement noise influences greatly on the estimates. However, the process noise covariance Q is generally much harder to estimate. Process state vector $X_{k|k-1}$ is unobserved and unknown at every time step. One possible way is to fix the Q, P and vary R in each iteration. Q and R can be estimated by analyzing off-line sampling in different Kalman filter and

then updates the optimal values to the current Kalman filter. The measurement noise covariance also updates in every updated circle like adaptive variational bayesian cubature Kalman filter (VBCKF) [15] does.

2.2 Extended Kalman Filter

The Kalman filter algorithm only suits linear systems, where both transformations F and H are linear. When either one of matrices F or H is not a linear combination, the original Kalman filter causes a lot of error due to the model mismatch. One solution is to linearize matrix F or H with Jacobian matrix of spatial derivation. The method is referred to the extended Kalman filter [10]. The core idea in extended Kalman filter algorithm is to use the first order derivation of Tayler theorm to approximately represent a nonlinear combination as a linear system. The process and measurement equations are shown in a similar way [10] :

$$X_{k|k-1} \approx \tilde{X}_k + \mathring{F}(X_{k-1} - \hat{X}_{k-1|k-1}) + \mathring{W} \cdot w_{k-1}, \tag{2.11}$$

$$Z_k \approx \tilde{Z}_k + \mathring{H}(X_{k|k-1} - \tilde{X}_k) + \mathring{V} \cdot v_k, \tag{2.12}$$

where $\tilde{X}_k$ and $\tilde{Z}_k$ are approximate states and corresponding approximate predicted measurements respectively. Both them are derived from the equation (2.3) and (2.4) without noise [10] :

$$\tilde{X}_k = F\hat{X}_{k-1|k-1}, \tag{2.13}$$

$$\tilde{Z}_k = H\tilde{X}_k. \tag{2.14}$$

$X_{k|k-1}$ and Z_k are the actual states and measurement vectors. The subscript represents time step k or $(k-1)$. $\mathring{F}$, $\mathring{W}$, $\mathring{H}$ and $\mathring{V}$ are Jacobian Matrix of partial derivatives of F, w, H, v respectively. Jacobian matrix of partial derivation does not work well in high degrees of matrix F and H, especially in the high-dimensional states estimates. High-dimensional state is the large size vector in state X. The algorithm would seriously be punished by the divergence [11] because of inaccuracy underlying physical model, incorrect data collection and high nonlinearity. Estimated by the partial derivations, the non-linear algorithm is transformed into linearity finally. They are easily be implemented in linear combination steps by steps, which is the same as the original filter algorithm.

2.3 Square Root of Cubature Kalman Filter

Instead of linearizing the system as the EKF does, SCKF (square root of cubature Kalman filter), which is the extension of CKF (cubature Kalman filter), uses the Bayesian filter theory in the time update and the measurement update. SCKF assumes both process noise and measurement noise to Gaussian distribution. It

employs deterministic sampling to get an approximately mean value and variance of the state vector. We rewrite the process equation (2.3) and measurement equation (2.4) into [11] :

$$X_{k|k-1} = f(X_{k-1|k-1}, u_{k-1}) + w_{k-1}, \tag{2.15}$$

$$Z_{k|k-1} = h(X_{k|k-1}, u_k) + v_k, \tag{2.16}$$

where the transformation function f and h are no longer linear. As the Bayesian theory is applied, the probability density should be known in each time step when doing the process and measurement update. The most important thing is a posterior density function, which helps the algorithm to find out the optimization state [11] :

$$\hat{X}_{k|k} = \int X_{k|k} \cdot p(X_{k|k}|D_k) dX_{k|k}, \tag{2.17}$$

where $p(X_{k|k}|D_k)$ is called as a posterior density and D_k is measurement value from time step 1 until k, shown as $D_k = \{Z_i\}_{i=1}^{k}$. So that the equation is rewritten as [11] :

$$p(X_{k|k}|D_k) = p(X_{k|k}|D_{k-1}, Z_k) = \frac{1}{C_k} p(X_{k|k-1}|D_{k-1}) p(Z_k|X_{k|k-1}), \tag{2.18}$$

where C_k is used to normalize the posterior density into unit value, $p(X_{k|k-1}|D_{k-1})$ is the predictive density function obtained by equation 2.15, $p(Z_k|X_{k|k-1})$ is measurement likelihood function and Z_k are the values from real measurement. The models are all assumed to be Gaussian, whose density function is represented by the mean value and the error covariance value. Thus, the predictive density function and the likelihood density function are shown as below separately [11] :

$$p(X_{k|k-1}|D_{k-1}) = N(X_{k|k-1}; \hat{X}_{k|k-1}, P_{k|k-1}), \tag{2.19}$$

$$p(Z_k|X_{k|k-1}) = N(Z_k; \hat{Z}_{k|k-1}, P_{zz,k|k-1}), \tag{2.20}$$

where N(a;b,c) means the probability to get the value "a" in the Gaussian model with mean "b" and its noise covariance "c". The mean value of predicted state $X_{k|k-1}$ and predicted measurements $Z_{k|k-1}$ are calculated as [11] :

$$\hat{X}_{k|k-1} = \int f(X_{k-1|k-1}) \cdot N(X_{k-1|k-1}; \hat{X}_{k-1|k-1}, P_{k-1|k-1}) dX_{k-1|k-1}, \tag{2.21}$$

$$\hat{Z}_{k|k-1} = \int h(X_{k|k-1}) \cdot N(X_{k|k-1}; \hat{X}_{k|k-1}, P_{k|k-1}) dX_{k|k-1}, \tag{2.22}$$

where $X_{k-1|k-1}$ is the old posterior state from old posterior density $p(X_{k-1|k-1}|D_{k-1})$. These equations make recursion automatically to get a more correct states. The predicted error covariance $P_{k|k-1}$ and innovation error covariance $P_{zz,k|k-1}$ are shown below [11] :

$$\begin{aligned} P_{k|k-1} &= E[(X_{k|k-1} - \hat{X}_{k|k-1})\,(X_{k|k-1} - \hat{X}_{k|k-1})^T | Z_{1:k-1}] \\ &= \int_{R^{n_x}} f(X_{k-1|k-1})\, f^T(X_{k-1|k-1}) \cdot N(X_{k-1|k-1}; \hat{X}_{k-1|k-1}, P_{k-1|k-1})\, dX_{k-1|k-1} \\ &- \hat{X}_{k|k-1}\hat{X}_{k|k-1}^T + Q_{k-1}, \end{aligned} \tag{2.23}$$

$$\begin{aligned} P_{zz,k|k-1} &= E[(Z_{k|k-1} - \hat{Z}_{k|k-1})\,(Z_{k|k-1} - \hat{Z}_{k|k-1})^T | Z_{1:k-1}] \\ &= \int_{R^{nx}} h(X_{k|k-1})\, h^T(X_{k|k-1}) \cdot N(X_{k|k-1}; \hat{X}_{k|k-1}, P_{k|k-1})\, dX_{k|k-1} \\ &- \hat{Z}_{k|k-1}\, \hat{Z}_{k|k-1}^T + R_k, \end{aligned} \tag{2.24}$$

After receiving new measurement values Z_k, the Bayesian filter computes the a posterior density $p(X_{k|k}|Z_{1:k})$ [11] :

$$p(X_{k|k}|Z_{1:k}) = N(X_{k|k}; \hat{X}_{k|k}, P_{k|k}), \tag{2.25}$$

where the posterior estimate state is [11] :

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k(Z_k - \hat{Z}_{k|k-1}), \tag{2.26}$$

the posterior error covariance is [11] :

$$P_{k|k} = P_{k|k-1} - K_k P_{zz,k|k-1} K_k^T, \tag{2.27}$$

and the Kalman gain is [11] :

$$K_k = P_{xz,k|k-1} P_{zz,k|k-1}^{-1}, \tag{2.28}$$

where the $P_{xz,k|k-1}$ is the cross covariance [11] :

$$\begin{aligned} P_{xz,k|k-1} &= E[(X_{k|k-1} - \hat{X}_{k|k-1})(Z_{k|k-1} - \hat{Z}_{k|k-1})^T | Z_{1:k-1}] \\ &= \int_{R^{n_x}} X_{k|k-1} h^T(X_{k|k-1}) \cdot N(X_{k|k-1}; \hat{X}_{k|k-1}, P_{k|k-1}) dX_{k|k-1} \\ &\qquad - \hat{X}_{k|k-1}\hat{Z}_{k|k-1}^T. \end{aligned} \tag{2.29}$$

So far, we derive the cubature Kalman filter from the linear Kalman filter. From the construction of the equation, the main core to solve the *CKF* algorithm is to solve the function $non-linear\ transformaion \times Gaussian\ distribution$ [11] :

$$I(f) = \int f(x) \cdot N(x; \mu, \Sigma) dx, \tag{2.30}$$

where Gaussian distribution $N(x; \mu, \Sigma)$ is:

$$N(x; \mu, \Sigma) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\Sigma}}, \tag{2.31}$$

Letting $x = \sqrt{2\Sigma}y + \mu$, makes $y = \frac{x-\mu}{\sqrt{2\Sigma}}$ and $dx = \sqrt{2\Sigma}dy$. So that the function is [11] :

$$\begin{aligned} &\int f(x) \cdot N(x;\mu,\Sigma)dx \\ &= \int f(x)\frac{1}{\sqrt{2\pi\Sigma}}e^{-\frac{1}{2}\frac{(x-\mu)^2}{\Sigma}}dx \\ &= \frac{1}{\sqrt{\pi}^n}\int f(\sqrt{2\Sigma}y+\mu)e^{-y^Ty}dy \\ &= \frac{1}{\sqrt{\pi}^n}\int f(\sqrt{2\Sigma}x)e^{-x^Tx}dx \end{aligned} \tag{2.32}$$

Therefore, the question is shrunk to solve the function $\int f(x)e^{-x^Tx}dx$. The article written by Simon Haykin [11] solves this with third-degree spherical-radial method. The result shows like [11] :

$$\int f(x)w(x)dx \approx \sum_{i=1}^{m} w_i f(x_i), \tag{2.33}$$

where weight function is $w(x) = e^{-x^Tx}$. To avoid complicated computation from numerous points as inputs in the algorithm, invariant theory [16] employs the symmetric points into $f(x)$ and corresponding equally weight function $w(x)$. Constructing symmetric cubature points simplifies the approximation function (2.33) from infinity points down to $2n$ points. According to invariant theory, n represents the total dimension of states vector.
First, the function above at the left side are transformed into combination of radius r and direction vector $\vec{y}$ as follows: $x = r\vec{y}$ with $\vec{y}^T\vec{y} = 1$ leading to $x^Tx = r^2$ with r from 0 to ∞. Thus, the function above is rewritten as [11] :

$$\int f(x)e^{-x^Tx}dx = \int_0^\infty \int_{U_n} f(r\vec{y})r^{n-1}e^{-r^2}d\sigma(\vec{y})dr. \tag{2.34}$$

The equation (2.34) could be divided into two parts:

$$I(f) = \int_0^\infty S(r)r^{n-1}e^{-r^2}dr, \tag{2.35}$$

$$S(r) = \int_{U_n} f(r\vec{y})d\sigma(\vec{y}). \tag{2.36}$$

Then employ Generalized Gaussian Quadrature rule to solve 2.35 and spherical cubature rule to solve (2.36). The spherical rule leads to a result that the cubature points locate at the intersection of the unit sphere and their axes. The cubature points are expressed in a mathematical point of view [11] :

$$[1] = \begin{pmatrix} 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \vdots & 0 & -1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 0 & \cdots & \cdots & -1 \end{pmatrix}. \tag{2.37}$$

There are two symmetric cubature points around the original points for each dimension of state vector. According to the generalized Gaussian Quadrature with three degrees rule, there is only one cubature point locating at $\sqrt{2n/2}$. Therefore, the whole algorithm is based on total full symmetric cubature points. The weight for each point is $1/(2n)$ [11]. Integrating these two solutions of equation (2.35) and (2.36), third-degree spherical radial rule is extended to solve function in (2.30). Take Gaussian distribution $N(x;0,I)$ for example, it is shown as [11] :

$$I_N(f) = \int_{R^n} f(x)N(x;0,I)dx \approx \sum_{i=1}^{m} w_i f(y_i), \tag{2.38}$$

where

$$\begin{aligned} y_i &= \sqrt{\frac{2n}{2}}[1]_i \\ w_i &= \frac{1}{2n}, \end{aligned} \tag{2.39}$$

where i=1,2,$\cdots$,2n, and $[1]_i$ is the i_{th} column in matrix $[1]$. When extending the result to a normal case, Gaussian distribution $N(x;\hat{x},P)$ is shown like [11] :

$$I_N(f) = \int_{R^n} f(x)N(x;\hat{x},P)dx \approx \sum_{i=1}^{m} w_i f(y_i), \tag{2.40}$$

where

$$\begin{aligned} y_i &= \sqrt{\frac{2n}{2}}\sqrt{P}[1]_i + \hat{x} \\ w_i &= \frac{1}{2n}. \end{aligned} \tag{2.41}$$

The error covariance matrix P satisfies $P = \sqrt{P}\sqrt{P}^T$. At the end, implement the final expression (2.40) into every Kalman filter algorithm with $non-linear\ function \times Gaussian\ Distribution$. The complete *CKF* algorithm is shown in the Appendix.

Briefly, the idea of CKF is to create symmetric cubature points by adding a permutating and changing the sign to the original point to build the generator $[1]$. Then, insert every cubature point into the process and the measuring algorithm respectively. Average all of them to get a more accurate state. In this thesis, SCKF algorithm is the main tool to increase accuracy of positioning, which uses the least-squares method to compute the Kalman gain and uses triangularizations for covariance updates. The CKF algorithm is superior to EKF algorithm. The reason is that the EKF is only suitable for the low dimension state vector and low-linearity system. SCKF algorithm is superior to CKF due to these phases. Firstly, SCKF can maintain the error covariance matrix $P_{k|k}$ symmetry and positive definiteness in every update cycle. If not, the algorithm will be crash immediately and then wastes time to re-do the algorithm until it succeeds. In the view, SCKF is much stable and much more anti-inference than CKF. Secondly, due to

the finite word-length digital computation, information is often lost. It influences on the covariance properties and the lost information ruins the whole algorithm. These computations include matrix square-root, matrix inversion and division between two matrices in the covariance update. SCKF is designed to avoid these kinds of computations [11], increasing stability by using the least-squares method to against matrix inversion, using matrix triangular factorizations to against the square root of the matrix and using forward/back substitution to against the division between two metrics. The structure SCKF is showing in the following:

The SCKF Algorithm [11]

Process update:

1) Cubature points

$$X_{i,k-1|k-1} = \sqrt{\frac{2n}{2}} S_{k-1|k-1}[1]_i + \hat{X}_{k-1|k-1},$$

, where i=1,2,$\cdots$,2n.
2) Propagated cubautre points

$$X^*_{i,k|k-1} = f(X_{i,k-1|k-1})$$

3) Predicted state

$$\hat{X}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} X^*_{i,k|k-1}$$

4) Square-root factor of Predicted error covariance

$$S_{k|k-1} = Tria([X^*_{k|k-1}\, S_{Q,k-1}]),$$

where

$$Q_{k-1} = S_{Q,k-1} S^T_{Q,k-1}$$

and weighted, centered matrix:

$$X^*_{k|k-1} = \frac{1}{\sqrt{2n}} [X^*_{1,k|k-1} - \hat{X}_{k|k-1}\, X^*_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X^*_{m,k|k-1} - \hat{X}_{k|k-1}],$$

and *Tria* is general triangularization algorithm (such as QR decomposition), S=Tria[A]. In order to get the lower triangular matrix using QR decomposition. Let the upper triangular matrix R be the QR decomposition of A^T, so that the lower triangular matrix would be $S = R^T$.

Measurement Update:
1) Cubature points

$$X_{i,k|k-1} = \sqrt{\frac{2n}{2}} S_{k|k-1}[1]_i + \hat{X}_{k|k-1} \tag{2.42}$$

2) Propagated Cubature points

$$Z_{i,k|k-1} = h(X_{i,k|k-1}) \tag{2.43}$$

3) Predicted Measurement

$$\hat{Z}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} Z_{i,k|k-1} \tag{2.44}$$

4) Square-root of the innovation covariance

$$S_{zz,k|k-1} = Tria([Z_{k|k-1}\, S_{R,k}]), \tag{2.45}$$

where $S_{R,k}$ is the square-root of R_k that $R_k = S_{R,k} S_{R,k}^T$. And weighted, centered matrix is:

$$Z_{k|k-1} = \frac{1}{\sqrt{2n}} [X_{1,k|k-1} - \hat{X}_{k|k-1}\, X_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X_{m,k|k-1} - \hat{X}_{k|k-1}] \tag{2.46}$$

5) The cross-covariance matrix

$$P_{xz,k|k-1} = X_{k|k-1} Z_{k|k-1}^T, \tag{2.47}$$

where the weighted, centered matrix

$$X_{k|k-1} = \frac{1}{\sqrt{2n}} [X_{1,k|k-1} - \hat{X}_{k|k-1}\, X_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X_{m,k|k-1} - \hat{X}_{k|k-1}] \tag{2.48}$$

6) Kalman gain

$$K_k = (P_{xz,k|k-1} / S_{zz,k|k-1}^T) / S_{zz,k|k-1} \tag{2.49}$$

7) Updated State

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k (Z_k - \hat{Z}_{k|k-1}) \tag{2.50}$$

8) Square-root factor of the new posteriori error covariance

$$S_{k|k} = Tria([X_{k|k-1} - K_k Z_{k|k-1}\, K_k S_{R,k}]) \tag{2.51}$$

2.4 Particle Filter

As discussed in the original Kalman filter section, the original Kalman filtering is implemented with the method of Monte Carlo sampling. Sequential importance sampling (SIS) is the principle method. The Monte Carlo method is also used in particle filter [18]. Instead of considering the marginal full posterior distribution $p(x_{k|z_{1:k}})$ in equation (2.2), SIS needs to consider of all the previous states up to time k, $p(x_{0:k}|z_{1:k})$. They are associated weighted set of samples $(x^i_{0:k}, w^i_k)^N_{i=1}$, where N is the number of total particles at time step k. The states $x^i_{0:k}$ are a set of support points up to time k with its weight w^i_k. In each time step, weight is normalized to be one that $\sum_i w^i_k = 1$. Therefore, the posterior density at k is approximated as [18] :

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} w^i_k \delta(x_{0:k} - x^i_{0:k}). \tag{2.52}$$

The definition of weight w^i_k uses importance sampling method. Assuming that the particles state $x^i_{0:k}$ is drawn from an importance density function $q(x_{0:k}|z_{1:k})$, the weight in equation (2.52) is expressed as [18] :

$$w^i_k \propto \frac{p(x^i_{0:k}|z_{1:k})}{q(x^i_{0:k}|z_{1:k})}. \tag{2.53}$$

In the sequential situation, the existing particle distribution $p(x_{0:k-1}|z_{1:k-1})$ adds other new sets of particles to form new probability density distribution $p(x_{0:k}|z_{1:k})$ in every recursion. Therefore, the importance density $q(.)$ is factorized as [18] :

$$q(x_{0:k}|z_{1:k}) = q(x_k|x_{0:k-1}, z_{1:k}) q(x_{0:k-1}|z_{1:k-1}). \tag{2.54}$$

The new set of particles $x^i_{0:k} \sim q(x_{0:k-1}|z_{1:k})$ are derived from existing samples $x^i_{0:k-1} \sim q(x_k|x_{0:k-1}|z_{1:k-1})$ by adding new state $x^i_k \sim q(x_k|x_{0:k-1}, z_{1:k})$. For the target distribution $p(x_{0:k}|z_{1:k})$, it use Bayes' rule to factorize them as [18] :

$$p(x_{0:k}|z_{1:k}) = \frac{p(z_k|x_{0:k}, z_{1:k-1}) p(x_{0:k}|z_{1:k-1})}{p(z_k|z_{1:k-1})}, \tag{2.55}$$

where $p(x_{0:k}|z_{1:k-1})$ is written as:

$$p(x_{0:k}|z_{1:k-1}) = p(x_k|x_{0:k-1}, z_{1:k-1}) p(x_{0:k-1}|z_{1:k-1}). \tag{2.56}$$

Due to the recursion up to time step k, there are two equations: $p(x_k|x_{0:k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$ and $p(z_k|x_{0:k}, z_{1:k-1}) = p(z_k|x_k)$. The equation (2.55) is factorized into [18] :

$$p(x_{0:k}|z_{1:k}) \propto p(z_k|x_k) p(x_k|x_{k-1}) p(x_{0:k-1}|z_{1:k-1}). \tag{2.57}$$

Then, substituting equation (2.54) and (2.57) into (2.53), the weight updating equation is represented as [18] :

$$w^i_k \propto w^i_{k-1} \frac{p(z_k|x^i_k) p(x^i_k|x^i_{k-1})}{q(x^i_k|x^i_{0:k-1}, z_{1:k})}. \tag{2.58}$$

Normally, only filtered $p(x_k|z_k)$ is required at each step so that one can ignore the path $x^i_{0:k-1}$ and history observation $z_{1:k-1}$. Then the weight is updated to be [18] :

$$w^i_k \propto w^i_{k-1} \frac{p(z_k|x^i_k)p(x^i_k|x^i_{k-1})}{q(x^i_k|x^i_{k-1}, z_k)}. \tag{2.59}$$

2.4.1 Generic Particle Filter

There is one problem in the particle filter algorithm after a few iterations. It is called degeneracy problem, which one but all particles are going to lose their own weights. The degeneracy problem is typically measured by the effective sample size N_{eff} [18] :

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w^i_k)^2}, \tag{2.60}$$

where w^i_k is the normalized weight in equation (2.53). Small N_{eff} indicates serious degeneracy. When the degeneracy happens, there are two methods to solve it in some degree. The first one is to pick up a good importance density, such as local linearization techniques [19] and unscented transforms [20]. It is good to choose importance density as following equation to simplify the implementation [18] :

$$q(x_k|x^i_{k-1}, z_k) = p(x_k|x^i_{k-1}), \tag{2.61}$$

so that the weight in equation (2.53) is simplified as [18] :

$$w^i_k \propto w^i_{k-1} p(z_k|x^i_k). \tag{2.62}$$

The other way is to resample the particles when the effective particle size N_{eff} drops below a certain threshold. What the resampling step do is to eliminate the particles with small weights and keep the particles with large weights. Generate the new set of particles to keep the total particles size constant. The weight of the new sets of particles are reset to $w^i_k = 1/N$ due to new particles in independently and identically distribution. Thus, resampling effectively manages the degeneration problem by getting rid of particles with small weight. At the same time, some particles with high weight are chosen several times and particles with small weight are totally discarded. It will cause some new practical problems, such as sample impoverishment problem. The problem decreases diversity of the particles. In this thesis, systematic resampling [21] is applied due to its easy implementation and uncomplicated operation [18] .

$$A = \sum_{i=0}^{\infty} \beta\alpha^i = \frac{\beta}{1-\alpha} \quad |\alpha| < 1$$

Chapter 3
Simulation Software

3.1 Introduction to V2V Communication

Vehicle-to-Vehicle communication is introduced in the WiFi 802.11p standard, which uses Dedicated Short-Range Communications (DSRC) protocol to assist vehicle communication operating in the licensed ITS high frequency band. Vehicles share information with others for convenient driving, intelligent navigation and avoidance of accident. The advantage of DSRC protocol is that there is no prepare any phases such as building link between each other before starting sharing information. The sharing begins when the vehicle is at each other's range. However, the V2V communication is affected by shadow fading and hidden terminal issue if there is crowed vehicles on the road network. The signal could not arrive at the destination vehicle. One possible solution is Vehicle-Roadside-Vehicle Relay Network [22]. The information would arrive at the destination vehicle before it passing through roadside access points. Another possible way is to use detectable vehicles as a jump point then send the information to the destination vehicle. Otherwise, use the broadcast protocol to collect the as much information as possible from different detectable vehicles [23] and then use these limiting information to improve the position. In the thesis, the broadcast protocol is preferred with its convenient distance measurement and easy operation on their structures. With these external information, each vehicle collects all information of other vehicles to calibrate its own position time by time. The great attraction is that the vehicle can estimate time of flight (TOF) of signal using 802.11p broadcast protocol to obtain distances among vehicles. Use them subsequently to restrict the vehicle position in a correct way.

3.2 Tools

The vehicular simulation tools help us establish a representative true environment. These simulation tools include *SUMO* [24], GEMV^2 and *NS*3 [26]. *SUMO* is used to extract the open-street-map and subsequently make vehicles run on their own predefined path in the open-street-map. The vehicles are assumed running in a typical environment in the urban scenario, where there are many trees, lakes and pedestrians around vehicles. GEMV^2 determines the paths of signal

transmitting and decides how much signal power can be reached at the destination vehicles. *NS3* [26] simulates the real network package communication that the packages are suffered by degeneration, multi-path components and so on.

3.2.1 SUMO

SUMO is an abbreviation for simulation of urban mobility. SUMO can define the paths of virtual vehicles in the simulated road network manually. It can also set vehicles to wait when they meet each other at the traffic light in the crossing. The Figure 3.1 shows the an urban scenario how vehicles meet at crossing.

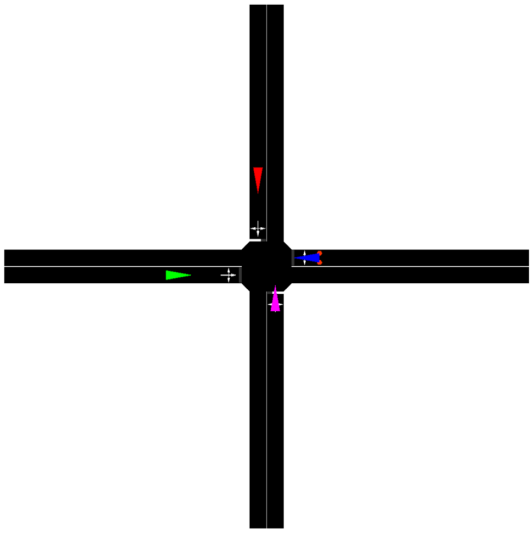

Figure 3.1: Vehicles meeting in a Crossroad

The colorful triangles represent various vehicles. SUMO performs very well in simulating how the vehicles run in a given road network. The road network can also be simulated by the SUMO, which can establish a complicated surroundings including pedestrians, vehicles and so on. The structure of road network is used in the GEMV^2 to calculate the signal power reaching at destination vehicles. In order to simulate running vehicles in the predefined path, powerful tool SUMO can set the types of vehicles, speed of vehicles, lanes in multiple-lanes streets, right-of-way rules and traffic lights in road network. SUMO also provides a fast OpenGL graphical user interface for users to review the open-street-map [25]. The open-street-map is saved into ".osm" file containing all road networks and environmental information.

3.2.2 GEMV^2

GEMV^2 is an abbreviation for geometry-based, efficient propagation model for vehicle-to-vehicle and vehicle-to-infrastructure communication. GEMV^2 uses out-

lines of vehicles, buildings and other objects in the scenarios to calculate how much signal can be reached at the vehicles. Most propagation factors are taken into account in GEMV^2, such as line-of-sight (LOS), non-line-of-sight (NLOS), path-loss, small-scale fading, large-scale fading and so on. Small-scale fading of the signal is calculated based on the number and size of the surrounding objects around the vehicle. The large-scale fading is affected by the object that standing in the way of wave propagation. GEMV^2 can detect the power of signal at destination vehicles in the urban, sub-urban, highway and the open space scenarios. If the power of signal is under the threshold, the signal is considered as fading away. Otherwise they are considered that the vehicles have succeeded in establishing the link between each other. One of the drawbacks of the GEVM^2 is that it does not support the 3D construction environment. GEMV^2 still works in the thesis, since our algorithm works in 2D as well.

3.2.3 NS3

NS3 is a discrete-event network simulator to provide a solution to simulate the V2V communication. It combines different device modules flexibly in complete networks, such as LTE, WIFI and GPS. *NS3* is also used in the non-Internet network analysis. *NS3* sets up the link among vehicles and shares the NS3-generated packets with each other. Through the link, the vehicles estimate the distance among the target vehicles.

3.2.4 Structure in Simulation

The structure of the simulation with these software is shown in Figure 3.2.

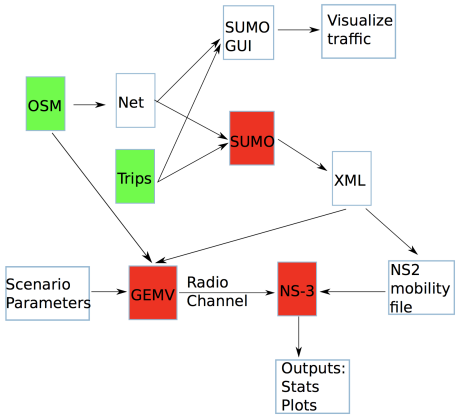

Figure 3.2: The whole Simulation Process

Firstly, extract one scenario in open-street-map and save them in the file. Secondly, set the route of each vehicle in the road network in the predefined open-street-map. Thirdly, set the sumo configuration including road network and trips of vehicle. They are all saved in the XML file. The XML will be both provided

to GEMV^2 and *NS*2. Fourthly, with scenario parameters and the sumo configuration XML file, GEMV^2 does the simulation on the radio channel. Finally, *NS*3 will simulate the entire network to collect the distance measurement, GPS, velocity and acceleration of each vehicle.

3.3 Target Tracking Model

In the thesis, each vehicle collects all the information from neighboring vehicles. These information include positioning from GPS estimate, velocity and acceleration from each vehicle. Here, we are going to introduce the global system model in this thesis. The global system is that each vehicles improve its positioning not only by using the information from its own data sensors, but also collecting and using the information from surrounding vehicles. Thanks to IEEE 802.11p standard introduces the V2X communication protocol in broadcast protocol, the vehicles can share the information in a high-speed dynamic environment and measure the distance among around vehicles from estimating the arrival time of signal. Distance measurement restricts the estimation of position helping vehicle with better positioning. As the global system is used, all vehicles is assumed to have the same state vector represented as:

$$X_i = \begin{bmatrix} x_i & v_{xi} & y_i & v_{y_i} & a_{xi} & a_{y_i} \end{bmatrix}, \tag{3.1}$$

where x_i and y_i are real positions, v_{xi} and v_{y_i} are velocity in x and y direction respectively, a_{xi} and a_{y_i} represent acceleration, and the subscription i represents the i_{th} vehicle. In the global system, each vehicle should take other vehicle states into consideration so that the states vector looks like:

$$X = \begin{bmatrix} X_1 & X_i & \cdots & X_N \end{bmatrix}. \tag{3.2}$$

The principle of global system is to use all states from different vehicles instead of only their own state. Each vehicle predicts not only its own position but also other vehicles' positions using the same transform function shown in equation (3.3). The prediction is based on the previous posterior states and the process noise. Therefore, the predicted process equation in equation (2.3) is written as:

$$\begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix}_k = \begin{pmatrix} F_1 & Z & \cdots & Z \\ Z & F_i & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ Z & \cdots & \cdots & F_N \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix}_{k-1} + w_{k-1}, \tag{3.3}$$

where subscript i represents different vehicle, N represents the number of vehicles in all scenarios, k is the time step, Z is zero matrix with size 6×6, F_i is

transition function which is shown in equation (3.4).

$$F_i = \begin{pmatrix} 1 & T & 0 & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & T & 0 & T^2/2 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.4}$$

where T is the time step according to the 802.11p standard. It is set to be 100 ms. w_{k-1} is the process noise with a singular covariance Q :

$$Q = diag\left([\phi_1 \quad \phi_i \quad \cdots \quad \phi_N]\right). \tag{3.5}$$

For each ϕ_i, they are assumed to be the same:

$$\phi_i = diag\left([\sigma_{x_i}^2 \quad \sigma_{v_{xi}}^2 \quad \sigma_{y_i}^2 \quad \sigma_{v_{y_i}}^2 \quad \sigma_{a_{xi}}^2 \quad \sigma_{a_{y_i}}^2]\right), \tag{3.6}$$

where $\sigma_{x_i}^2, \sigma_{v_{xi}}^2, \sigma_{y_i}^2, \sigma_{v_{y_i}}^2, \sigma_{a_{xi}}^2, \sigma_{a_{y_i}}^2$ represent noise variance of x position, x velocity, y position, y velocity, x acceleration and y acceleration respectively in i_{th} vehicle.

The measurement values mostly depend on what kind of sensors are used. Some of them are non-linear measurements such as distance, the angle of arrival. Others are linear measurements such as GPS, velocity and acceleration. Therefore, equation (2.4) is written in a most common way:

$$Z_k = h(X_k) + v_k, \tag{3.7}$$

where $h(.)$ represents any kind of transition function. It is described in details in the following chapter based on different sensors implementation.

The initial states in equation (3.2) are based on the first time measurement from the sensors like GPS sensor, velocity sensor and the acceleration sensor. The associated covariance is given by:

$$P_{0/0} = diag\left([K_1 \quad K_i \quad \cdots \quad K_N]\right), \tag{3.8}$$

where K_i looks like:

$$K_i = \left(\sigma_{0x_i}^2 \quad \sigma_{0v_{xi}}^2 \quad \sigma_{0y_i}^2 \quad \sigma_{0v_{y_i}}^2 \quad \sigma_{0a_{xi}}^2 \quad \sigma_{0a_{y_i}}^2\right), \tag{3.9}$$

where $\sigma_{0x_i}^2, \sigma_{0v_{xi}}^2, \sigma_{0y_i}^2, \sigma_{0v_{y_i}}^2, \sigma_{0a_{xi}}^2, \sigma_{0a_{y_i}}^2$ are the error variance of x position, x velocity, y position, y velocity, x acceleration and y acceleration respectively in i_{th} vehicle at very beginning time step.

Chapter 4
Data Analysis

4.1 Parameter Settings

In the filter algorithm, there are four parameters that need to be initialized. They are the process noise, the measurement noise, the initial error covariance and the initial state vector. The process noise Q tries to stabilize the filter against filter becoming overconfident in estimation. Low Q in equation (3.5) means that the modeling error can be neglected. High Q means the opposite. The process noise setting is the most difficult part for designing the Kalman filter algorithm, which is unknown for all system considering it as a black box. In the different scenarios, the process noise settings are different. For example, if the vehicle has a very bad initial positions, high Q helps the vehicle to track itself back very quickly. It has a large variation around its real path after it tracking back, while low Q takes much more time to track the vehicle back to its own path but it keeps fairly smooth with small variation after tracking back. One possible way to estimate the process noise is to collect measurement information from sensors. Then, implement these measurement values into the algorithm. While keeping the measurement noise fixed, try to input many possible process noises to find out the better performance of positions. A more-likely process noise is found in specific scenario after many trials. The algorithm could be overconfident when the vehicle runs in the highway scenario due to its smooth driving, while the algorithm maybe lack confidence when the vehicle runs in the urban scenario because the vehicle's movement is much more unpredictable. In the thesis, a high process noise Q is applied since the urban scenario is discussed. The vehicles meet at the cross and do some reaction such as running straightly or making turns correspondingly. The parameters in ϕ_i in (3.6) are shown in the Table 4.1, assuming all vehicles suffering the same predicted noise covariance.

Standard Derivation of Q	Value (unit)
σ_{x_i}	$0.2\ m$
$\sigma_{v_{xi}}$	$1.4\ m/(0.1s)$
σ_{y_i}	$0.2\ m$
$\sigma_{v_{y_i}}$	$1.4\ m/(0.1s)$
$\sigma_{a_{xi}}$	$1\ m/(0.1s)^2$
$\sigma_{a_{y_i}}$	$1\ m/(0.1s)^2$

Table 4.1: Process Noise Parameter

The cooperative algorithm SCKF is influenced by the initial position and its corresponding covariance. If the initial position has a small error but with large covariance or if the initial position has a large error but with small error variance, both of them push the vehicles in the erroneous and bad prediction on their initial positions. A wrong prediction of initial positions makes algorithm take much more time to converge back to real path. The values of error covariance of initial states $P_{0|0}$ in (3.9) are shown in Table: 4.2.

Standard Derivation of Initial States Error Covariance of $P_{0\|0}$	Value (unit)
σ_{0x_i}	$7\ m$
$\sigma_{0v_{xi}}$	$0.8\ m/(0.1s)$
σ_{0y_i}	$7\ m^2$
$\sigma_{0v_{y_i}}$	$0.8\ m/(0.1s)$
$\sigma_{0a_{xi}}$	$0.5\ m/(0.1s)^2$
$\sigma_{0a_{y_i}}$	$0.5\ m/(0.1s)^2$

Table 4.2: Initial Position Error Variance

These values adjust the initial state X_0 by the algorithm automatically. It is critical to choose values in initial states error covariance. For the initial positions value, it is good to choose twice than variance of error of GPS sensor in initialization timestep. This makes cooperative algorithm more flexible to fetch an accurate relative initial positions. It is easier to converge vehicles to the right path even in a bad initial position estimate. In Table: 4.3, it reveals the error variances in all kind of sensors.

Sensors standard derivation	Value (unit)
GPS	30 m
Distance	1 m
Velocity	0.3 $m/(0.1s)$

Table 4.3: Sensor Noise Parameter

The standard derivation of GPS error in the urban scenario is set to be 30 m according to a survey in [27]. The distance error comes from the inaccuracy synchronization of broadcast protocol and multi-path signals arrival. Assuming it has an accurate time synchronization and an accurate high-speed hardware calculation, the distance measurement error can shrink down to one meter. In order to decrease the payload of network, the acceleration sensor does not use in the thesis.

Error Estimation

There are two methods to represent the algorithm performance. One is the root-mean-square-error (RMSE) method, which is called position error in the thesis. All position errors in x and y directions are taken into account by the following equation:

$$E_{position} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i - R_i)^2}, \tag{4.1}$$

where A_i represents the position estimates from algorithm including both x and y direction in each vehicle, R_i is the same as A_i but represents real position and N is the total number of collecting data.

The other way is the distance error. It is more in perceptual intuition, showing in the following equation:

$$E_{distance} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(X_i - Rx_i)^2 + (Y_i - Ry_i)^2}, \tag{4.2}$$

where X is position estimated by global system in X coordinate and Y is the Y coordinate. Rx is the real position in X direction and Ry is in Y direction. Calculate the distance error in each vehicle in every timestamp and then average them to get a more accurate value. However, there is only a small difference between these two methods.

4.2 Simulation in Different Scenarios

In the thesis, several scenarios are generated to verify the robustness of cooperative algorithm in the global system. The global system is that each vehicle collects

all the information from around vehicles and then utilizes all these information for position estimates.

Scenario Selection

Figure 4.1 is obtained from open street map randomly to reveal the real road network. Utilize SUMO to place some sceneries around the road in the open-street-map, which includes trees, lakes, buildings and pedestrians. They are shown as the color region displayed in Figure 4.1. All the simulations and the analysis are based on theses road network in crossroad scenario, T-crossroad scenario and highway scenario.

Figure 4.1: Open Street Map for simulation use

4.2.1 Scenario 0: Four vehicles meet at the crossing

Crossroad is the most typical traffic in the urban scenario along with the traffic light. Therefore, it is meaningful to make a research on this scenario. The crossroad scenario is grabbed in open-street-map in Figure 4.1 and afterward is shown in Figure 4.2. Make four vehicles run straightly and meet in an intersection as shown in Figure 4.3.

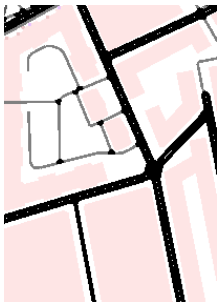

Figure 4.2: Road Condition of Crossroad

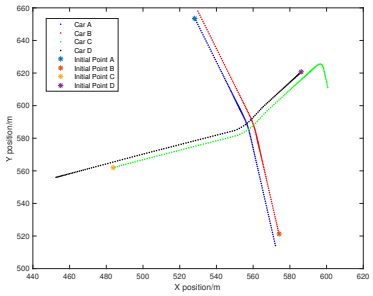

Figure 4.3: Four Vehicles' route at Crossroad

In the Figure 4.3, the stars are the initial points of each vehicle and dots are vehicles' real running-path for the simulation. As Table: 4.3 mentioned, GPS error in the urban is approximately equaling to 30 meters, which is a large and unacceptable error for positioning shown in Figure 4.4. The dots cover mostly the whole maps that even the route of each vehicle can not be recognized.

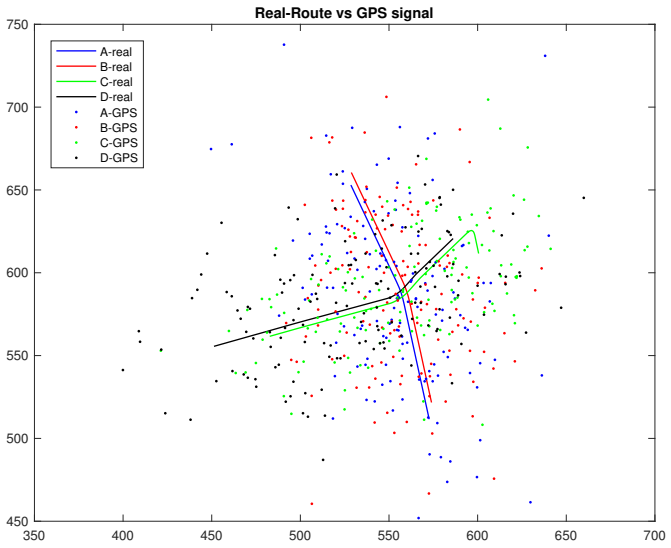

Figure 4.4: Only GPS used in positioning

The solid lines represent the real route for each vehicle and dots represent positions collecting from GPS sensor. Different vehicles are distinguished by different colors. From the figure, the dots are distributed around the real route. Some of them are far away from their real positions. Many predecessors utilize the Kalman filter to improve the accuracy of GPS positioning, but rarely use cooperative method especially using distance as input measurement value. This idea is also the spotlight in the thesis. Considering distance as the input measurement value, radio device based on IEEE 802.11p needs to be placed in each vehicle

to estimate distance and collect vehicles' states among the surrounding vehicles. Other vehicles positions information are also required to estimate the distance through distance equation:

$$Distance = \sqrt{(X_I - X_N)^2 + (Y_I - Y_N)^2}, \tag{4.3}$$

where X_I and Y_I are the X and Y positions of the local vehicle respectively, X_N and Y_N are the X and Y coordinate of the remote vehicle respectively. In order to compare the principle result in the same level, it is necessary to keep the random seed same to generate noise in MATLAB. Here, we compare four different ways that improve the performance of positions.

Position with Distance Implementation

Since V2X system is published in the standard IEEE 802.11p, the vehicles in distance measurement has much more improvement than the method using multipath component to measure distance. The accuracy of distance measurement improves down to one meter. Therefore, the distance measurement will be widely used in the cooperative positioning method. DSRC module is a distance measurement sensor fulfilling the standard IEEE 802.11p requirement. It broadcasts information at a fixed time and calculates the distance by the arrival time of signal. The distance calculation is proportional to the signal flight time in the air. While with a bad GPS estimate, using distance measurement is a possible method to accurate the position estimates. In the Figure 4.5, it shows the simulation result from cooperative algorithm SCKF with only distance measurement value as input. The initial positions and variance of initial positions are assumed to be known by the algorithm. For example, they can obtained by GPS sensor in the previous timestamp.

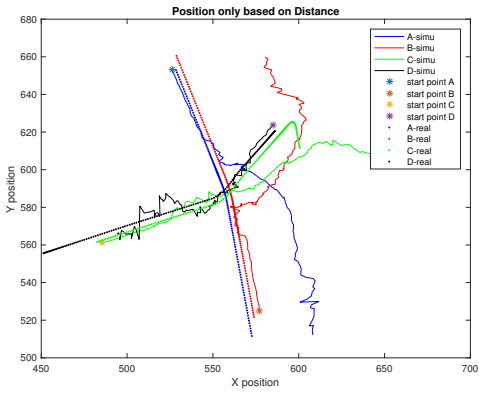

Figure 4.5: Simulation result with Distance

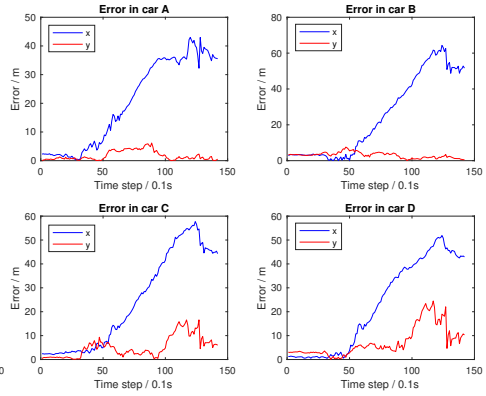

Figure 4.6: Positioning error for each vehicle

The stars represent the initial points of each vehicle, dots are the real route that the vehicles run in simulation, solid lines are the vehicle route estimated from

cooperative positioning algorithm. In the Figure 4.5, vehicles are on their own running path with small variation before all vehicles meeting at the crossing. In this period, it is good to see that the vehicles do not fluctuate a lot and run the same direction as they should be. This is fairly acceptable before encountering at the cross. The cooperative algorithm no longer works after intersecting since it only can estimate the positions of each vehicle, which they could lead to vehicle running in the wrong direction. When the vehicles are off-track from real path, they faces the challenge about tracking themselves back to the real path as Figure 4.6 shown. It indicates that how much errors of each vehicle are accumulated from time to time both in X and Y directions. In order to track the vehicles back, there are two methods that can be used to improve the prediction of positioning. One is to use GPS data from sensor directly instead of the position estimates from algorithm at the certain time, so that the error accumulation by the algorithm estimates can be eliminated through redoing the algorithm. This method is not very suitable because GPS sensor is not going to be well-performed in the urban scenario. The other way is to use GPS sensor data as measurement value in SCKF to adjust the positions. The second method would be discussed on the following section.

Position with GPS-Distance Implementation

In order to relocate after off-track, GPS is introduced to assist to accurate vehicles positions. In the WiFi 802.11p standard, the vehicles in the global system share information with each other. The information includes GPS positions and distance measurement. Each vehicle not only implements itself information but also implements other vehicle information into the SCKF algorithm to estimate the positions. The simulation results of GPS-distance implementation is shown in the Figure 4.7. Both measurement errors from sensors are listed in Table 4.3.

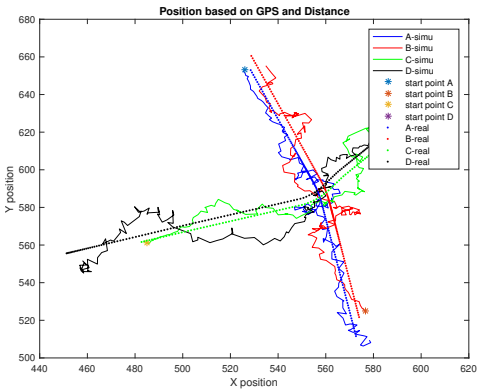

Figure 4.7: Simulation result with GPS and Distance

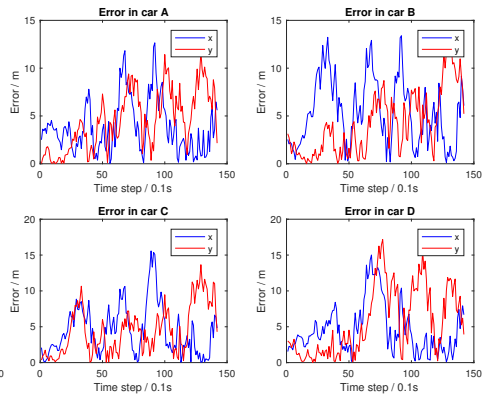

Figure 4.8: Position error for each vehicle

In Figure 4.7, the algorithm indeed solves the problem about the estimates of vehicle positioning in the wrong direction when coming to a crossing. However,

GPS sensor data is unstable with a huge error that it affects the positioning estimate from SCKF algorithm seriously. Compared with the Figure 4.7, Figure 4.5 shows that the position estimates of the vehicle is quite smooth before meeting each other at the crossing. The reason for these two differences is that the GPS sensor is in the bad performance. With GPS-distance implementation, the simulated positions are fluctuating along their own real positions and very unstable with distance error 7.4 meters. This model is not acceptable for our determination even though it improves a little bit better performance compared to demonstration only GPS measurement used.

Position with GPS-Distance-Velocity Implementation

As the discussion above, the model with GPS-distance implementation provides better position estimates than the model only with GPS. But the performance of positioning are influenced by the error variance of GPS sensor. In this section, velocity information is implemented to restrict the position estimates. Since the error variance of velocity is smaller than variance error from GPS sensor, the filter algorithm have much more confidence on the velocity measurements instead of GPS. In the SCKF algorithm, high variance of measurement noise means a less confidence on it. Hence, all sensors data from GPS, velocity and distance are packeted to send to other vehicles through DSRC module. Implement all these information into cooperative positioning algorithm to check the positioning performance. The simulation outcomes are shown in Figure 4.9 and 4.10.

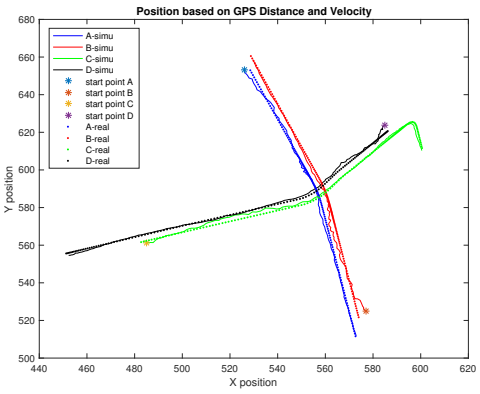

Figure 4.9: Simulation result with GPS, Distance and Velocity

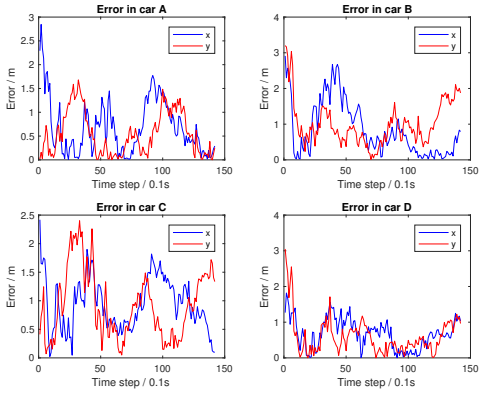

Figure 4.10: Position error for each vehicle

In Figure 4.9, it seems that the GPS-distance-velocity implementation has solved the error estimate boosting problem when they come to crossing as Figure 4.5 shown. It also solves the problem that the error position estimate has a large variance all the time as Figure 4.7 shown. Even having the poor initial positions, the SCKF algorithm can still track the vehicle back to its real path quickly. The vehicles move exactly along their own path with only 1.2 meters distance error. The drawback of this implementation is that so many information should be shared

each other through the limited bandwidth. The payload is more than heavy when an increasing vehicles join in the global system.

Positioning with Distance-Velocity (without GPS) Implementation

The goal of thesis is to solve the problem of positioning in the urban scenario, where dense high buildings and crowed pedestrians perhaps block the GPS signal. When GPS is not well received, we also test the algorithm without GPS, barely using velocity and distance measurements as observation values. The following Figures 4.11 and 4.12 show the simulation outcomes from cooperative algorithm. The initial positions of vehicle are assumed to be known by the GPS in the previous time stamp.

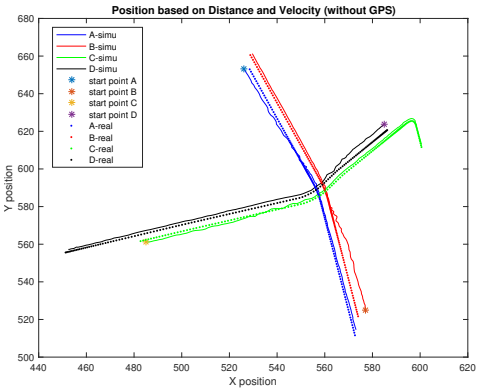

Figure 4.11: Simulation result with Distance and Velocity

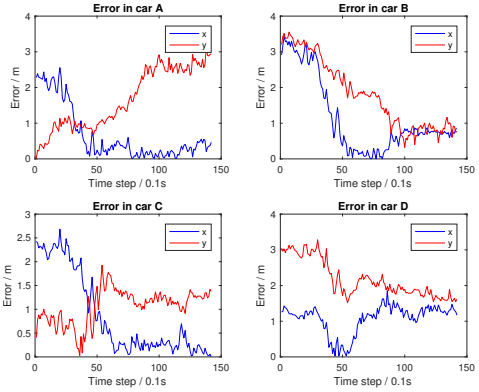

Figure 4.12: Position error for each vehicle

In Figure 4.11, the initial positions of the vehicles derivate from the real positions slightly. The movement of vehicles are very smooth along with the real path, since the error from velocity sensor and distance sensor are small. The distance error in this implementation is approximate 2.1 meters, which is suboptimal to GPS-distance-velocity implementation. There is still a problem. If all the vehicles have poor initial positions, the outcomes simulation suffer from the tracking-back problem unavoidable. Both velocity error and distance error are accumulated. There is no way to relocate the vehicles back to its real path. Even they are only a little bit derivation from the real path. Nevertheless, the outcome of positioning from SCKF algorithm with distance-velocity implementation is also acceptable even the GPS signal is totally blocked.

In conclusion, Table 4.4 shows the all simulation result in each implementation with single random seed in MATLAB to generate noise. The input information shows what kind of data is used. All the GPS error as input information is about 30 m.

Implementation used in SCKF Algorithm	Distance error (m)
GPS	12.6
GPS-Distance	7.4
GPS-Distance-Velocity	1.2
Distance-Velocity	2.1

Table 4.4: Comparison among different Input Information

Through Table 4.4, any combinations with additional sensors information as observation values will improve the performance than the implementation only GPS used. In comparison, there is no doubt that the GPS-distance-velocity implementation has the best performance than others. It is also acceptable that input all measurement values without GPS value into SCKF to obtain the positions, whose error is only slightly smaller than the best implementation. However, these analysis are based on the fixed seed to generate the noise in MATLAB. It can not represent the noise in reality very well. So that the models with distance-velocity implementation and GPS-distance-velocity implementation are discussed in statistic point of view in different scenarios.

4.2.2 Particle Filter

The particle filter is widely used in indoor positioning with its good performance in tracking devices back to the real positions when they are in the wrong places. It means that the error does not accumulate. In this section, particle filter algorithm is introduced to test whether it has a better performance or not in high-speed movement positioning. The particle filter is good at handling the wrong initial position estimates because they produce many state particles giving a high weight on the valuable state particles and a low weight on less valuable state particles. The particle filter extracts good state particles and abandons bad particles with resampling method. After resampling, the valuable particles create more particles based on their parent states and subtract more valuable state particles from them. In order to compare particle filter algorithm with SCKF algorithm, all the parameters are the same in both algorithms, shown in Table 4.1, 4.2 and 4.3. All theses information from vehicles are implemented in the particle algorithm. Due to the high dimensional states, each states create more than 500 state particles to assure accurate positioning. As 24-dimensional states in the global system model in the thesis, particle filter would iterate 12000 times to get the present positions in every timestep. The results are shown in Figure 4.13 and 4.14.

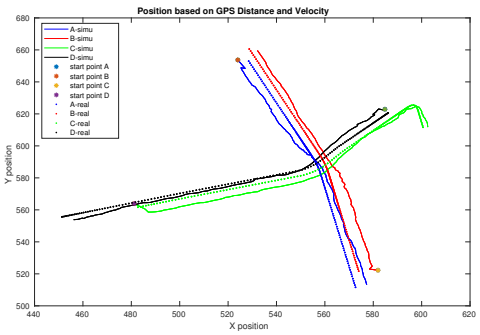

Figure 4.13: Particle Filter Simulation result with GPS, Distance and Velocity

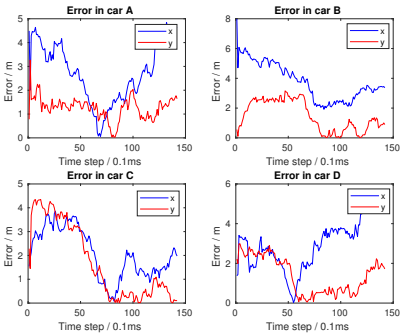

Figure 4.14: Position error for each vehicle

The distance error is around 3.4 meters, which is worse than the distance error in SCKF algorithm simulation with same GPS-distance-velocity implementation. This could happen when the number of new state particles are not enough to trace vehicle back in high-dimensional states. However, the resampling method is always applied while the present states create more particles. The chosen particles will be given to the same weight. This is the same principle as SCKF algorithm does, who selects particles intersecting with the axis and applies the same weights to them. The difference between them is that SCKF algorithm picks up the domain particles, while particle filter algorithm picks up both domain particles and less domain particles at the same time giving them the same weights. This is the reason why particle filter works worse than SCKF algorithm in high-dimensional stats situation. Meanwhile, a large number of particles take more time in calculation, which does not suit for the vehicle-to-vehicle communication limiting 0.1 s to finish the process. Therefore, in the rest sections, deep research about SCKF is made instead of analyzing both algorithms.

4.2.3 Statistic Analysis

The analysis so far are based on the single random seed to generate the noise in MATLAB. This section introduces a better way to analyze them in the statistic point of view. The algorithm simulation in MATLAB utilizes different seeds to simulate complicated noise for analysis of distribution. From Table 4.4, the third and the fourth models have better performance than the first and the second models. The third and the fourth models will be discussed in statistics point of view in next section.

As for analyzing the distribution, one mathematic method called Nakagami distribution [28] is introduced to analyze the statistic distribution. Nakagami distribution is used to fit the phenomenon in telecommunication area, which the signal transmits between Non-LOS and LOS. Non-LOS is matched by Rayleigh distribution and LOS is matched by Rician distribution. Nakagami distribution is one of the members in the probability family with shape parameter m [29] and control-

ling spread ω parameter. Shape parameter is neither location parameter shifting the entire distribution left or right, nor scale parameter compressing or stretching entire distribution. Nakagami distribution changes the shape of distribution in other ways.

Positioning with GPS-Distance-Velocity Implementation

Instead of generating the noise with one random seed in MATLAB, the random seeds are varied from 1 to 100 with intercepting one to generate different Gaussian noise and keeping other parameters the same. In GPS-distance-velocity implementation, the distribution of distance error and position error are shown in Figure 4.15 and 4.16 respectively.

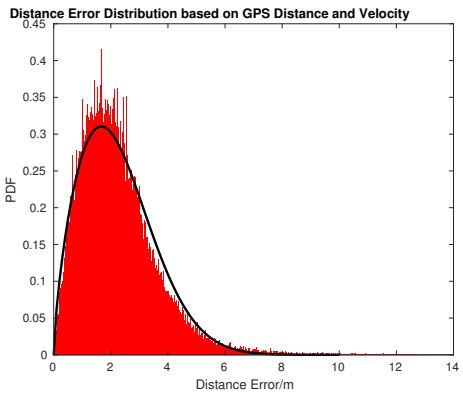

Figure 4.15: Distribution of distance error with GPS, Velocity and Distance

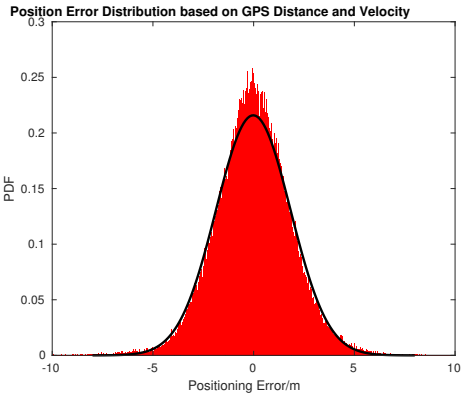

Figure 4.16: Distribution of position Error with GPS, Velocity and Distance

In Figure 4.15, the distance error is modeled as Nakagami distribution. The mean value is 2.3 m and the variance value is about 1.7 m^2. Form Figure 4.16 shown, positions distribution is more likely to be the Gaussian distribution because both process noise and measurement noise in SCKF algorithm are assumed to be the Gaussian distribution. As for every parameter in the states such as GPS, velocity and acceleration, they have almost the same characteristics. The reason why the distribution not exactly fit with the Gaussian distribution is that distance measurement value is implemented into SCKF algorithm as well. The distance measurement introduces the non-linear factor to the cooperative positioning. However, the position error distribution is the Gaussian distribution with zero mean and 1.8 m^2 variance. This is a quite good result among all the simulation discussed above.

Positioning with Distance-Velocity Implementation (without GPS)

As for the situation that GPS is blocked due to very high buildings and crowded pedestrians, GPS data is dropped from the sensor unfortunately. Only velocity

and distance measurement can be obtained from their sensors. The simulation of distance error and position error are shown in Figure 4.17 and 4.18 respectively with velocity-distance implementation.

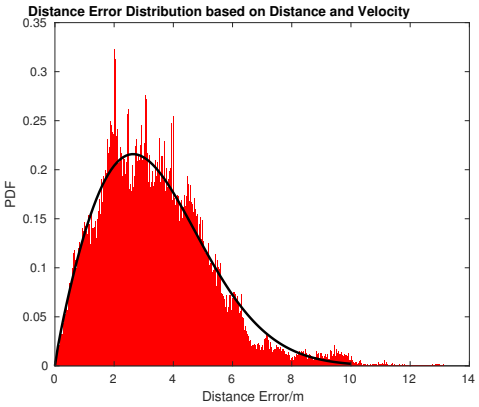

Figure 4.17: Distribution of distance error with GPS and Velocity

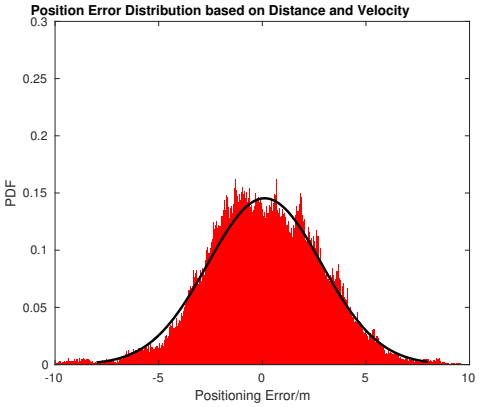

Figure 4.18: Distribution of position Error with GPS and Velocity

Even without GPS, the cooperation positioning algorithm still has a good performance. The mean value of distance error distribution is 3.4 m with its error variance 3.4 m^2. The distribution of position error is assumed to be the Gaussian distribution with zero mean and 2.7 m^2 error variance. The result is still a little bit worse than the implementation with GPS-distance-velocity measurement.
In conclusion, the Table 4.5 shows the entire simulation results based on different implementations. These results include mean distance error, variance error of distance and variance error of position. The mean position error is about zero so it is neglected in the chart. The error from raw GPS sensor is set to be 30 m.

Implementation	Mean distance error(m)	Variance distance error(m^2)	Variance positioning error(m^2)
GPS-Distance	6.6	13.6	5.3
GPS-Distance-Velocity	2.3	1.7	1.8
Distance-Velocity	3.4	3.4	2.7
GPS-Velocity	4.0	4.9	3.2

Table 4.5: Distribution comparison among different Input Implementation

Compared with mean distance error in the Table 4.4, Table 4.5 shows the worse simulation results. But the results in Table 4.5 are more reliable because the noise in our real environment is quite random to predict. The GPS-Distance and GPS-velocity implementations are also simulated in the same way showing the results in Figure A.1, A.2 and Figure A.3, A.4 respectively. The variances of distance error in Table 4.5 shows how the distance error varies around its mean distance error. A smaller variance implies more stable performance in positions. As for table shown above, the full implementation has the smallest mean distance error and the smallest variance of both distance error and position error. In the simulation, even without GPS sensor, the implementation with distance-velocity still has a good performance. This result is what we expect in our entire simulation. It shows that distance cooperation measurements helps vehicles to locate themselves, especially the GPS running out. All these simulations are only operated in four vehicles meeting in a crossing. In order to verify that distance measurement still helps in different road networks, several other scenarios are simulated in a statistical way to see their behaviors.

4.2.4 Simulation in more Scenarios

Scenario 1: Four vehicles intersect at a crossing while three vehicles go straightly and one vehicle makes a turn

The simulation of four vehicles are going straightly when intersecting in a crossing. The algorithm need to be test when some of the vehicles make the turn and some of them still go straightly. In Figure 4.19, it indicates that three vehicles go ahead and one vehicle makes a turn at the cross.

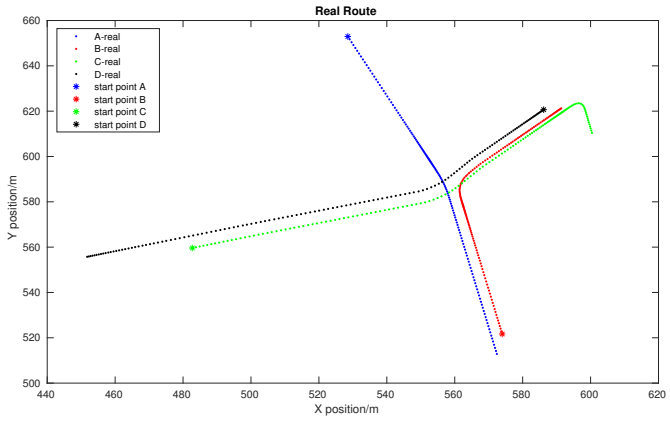

Figure 4.19: Real route for each vehicles

The dots represent the actual route for each vehicle, one color for one vehicle. Every interval between two adjacent dots is $100ms$ in each step. Stars are the initial positions of each vehicle. The red line and green line are overlapping. The vehicle B turns right when it comes to cross and follows after vehicle C until the end of simulation. Keeping all parameters the same as above is important both in this scenario and in following scenarios simulation. In the thesis, only two best implementations depending on the given analysis result are simulated. One is SCKF with GPS-distance-velocity implementation, the other is SCKF with distance-velocity implementation. The simulation results form SCKF with GPS-distance-velocity implementation are shown in appendix. Figure A.5 and A.6 are the simulation results, which utilize a single random seed to generate noise in MATLAB. Figure A.7 and A.8 are in terms of distribution point of view. In this scenario, mean distance error estimate form the Nakagami model is about 2.3 m with variance 1.9 m^2 and mean position error is about zero but with variance 1.9 m^2. As for distance-velocity implementation, the simulation analysis plots in Figure A.9 and A.10 in a fix random seed. The Figure A.11 and A.12 are in distribution point of view. The mean distance error is approximately 3.5 m and its variance is about 3.3 m^2. The mean position error is zero but with the variance of position error to be 2.8 m^2.

Scenario 2: Four vehicles intersect at the crossing while two vehicles go straightly and two vehicles make turn

In this scenario, four vehicles are simulated to run in different ways. Two vehicles go straightly and two other vehicles make turn when they are at the cross. The entire road network is shown in Figure 4.20, where the dots are the real route for each vehicle, stars are the initial positions of each vehicle.

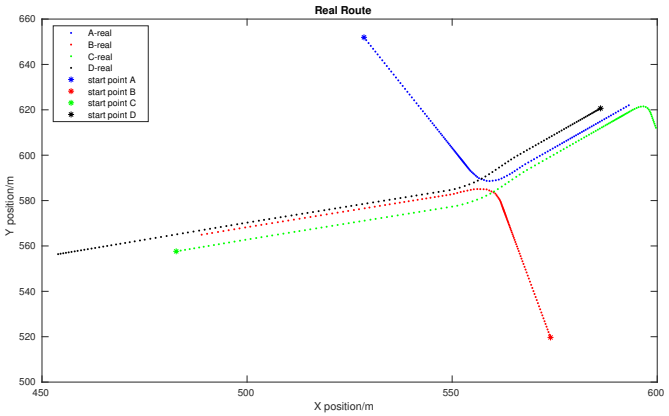

Figure 4.20: Real route for each vehicles

The simulation results of SCKF algorithm with full implementations are separately shown in Figure A.13 and A.14 with single seed for noise generation in MATLAB. The Figure A.15, A.16 are in terms of statistic distribution. In statistics point of view, the mean distance error is roughly 2.2 m and its variance is about 1.6 m^2. The mean position error is zero with its variance error of position 1.8 m^2. When only velocity and distance measurements are implemented into SCKF algorithm, Figure A.17 and Figure A.18 are plotted to show the analysis results with single seed for noise simulation. Figure A.19 and A.20 are shown in distribution point of view. According to the implementation in distribution analysis, the mean distance error is approximate 3.4 m and its variance is about 3.0 m^2. The mean position error is zero with 2.7 m^2 variance of position error.

Scenario 3: Four vehicles intersect in a crossing while one vehicle goes straightly and three vehicles make turn

Based on Scenario 2, one more vehicle makes a turn when they are meeting at a crossing. The scenario is shown in Figure 4.21.

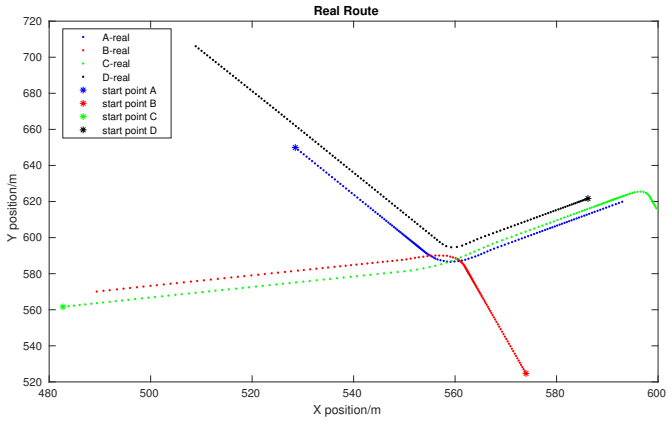

Figure 4.21: Real route for each vehicles

The dots represent the real path for each vehicle and stars represent the initial positions. Do the same analysis as above, getting some results shown in the following figures. With GPS-distance-velocity implementation, position estimates from SCKF algorithm are shown in Figure A.21 and A.22. When analyzing the demo in a statistic point of view, the results are shown in Figure A.23 and A.24 representing the distance error and position error separately. The mean distance error is approximate 2.2 m and variance of distance error is about 1.6 m^2. The mean position error is zero with 1.8 m^2 variance of position error. When only velocity and distance are implemented into SCKF algorithm, the simulation results with the single seed for noise generation are shown in Figure A.25 and A.26. They display how the vehicles position varies around the real route and what the position error it is for each vehicle. The distance error is 3.5 m. In statistic point of view, the analysis in different seeds are shown in Figure A.27 and A.28. The distance error is 3.4 m distance error with variance 3.0 m^2 and the mean position error is approximately equal to zero with variance 2.7 m^2.

Scenario 4: Four vehicles intersect in a crossing while all vehicles make turns

At the end, the scenario are simulated that all the vehicles make turns when they come across at crossroad. The scenario is shown in Figure 4.22.

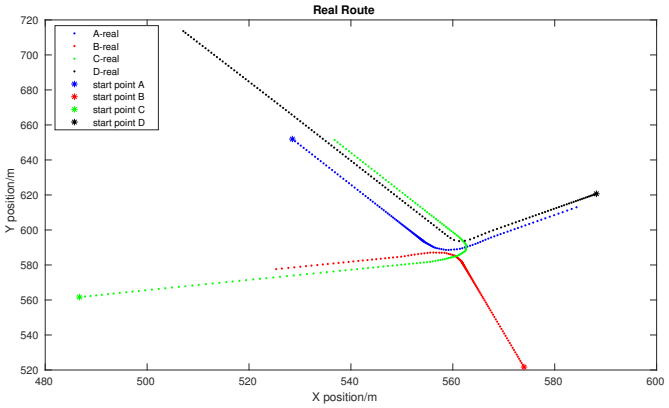

Figure 4.22: Real route for each vehicle

When using the single random seed to generate noise in simulation, the results are plotted in the following figures. Figure A.29 and A.30 display the analysis results when implementing GPS velocity and distance measurement into algorithm. The distance error in this implementation is about 2.0 meters. The Figure A.33 and A.34 show the simulation outcome with distance-velocity implementation in the algorithm. The distance error is about 1.9 meters. When simulating the algorithm in different seeds to generate noise, the distribution results are shown in Figure A.31 and A.32 with fully information as observation values. The mean distance error is approximate 2.2 m and variance of distance error is about 1.6 m^2. The mean position error is zero with 1.8 m^2 variance. The Figure A.35 and A.36 display the distribution analysis using distance-velocity implementation. The mean distance error is 3.3 m with variance 2.9 m^2 and the mean position error approximately equals to zero with variance 2.6 m^2.

Scenario 5: Four vehicles intersect in T-cross

Having discussed all the scenarios at the crossroad, one more scenario is simulated in the T-cross shown in Figure 4.23 showing how the four vehicles move in T-cross.

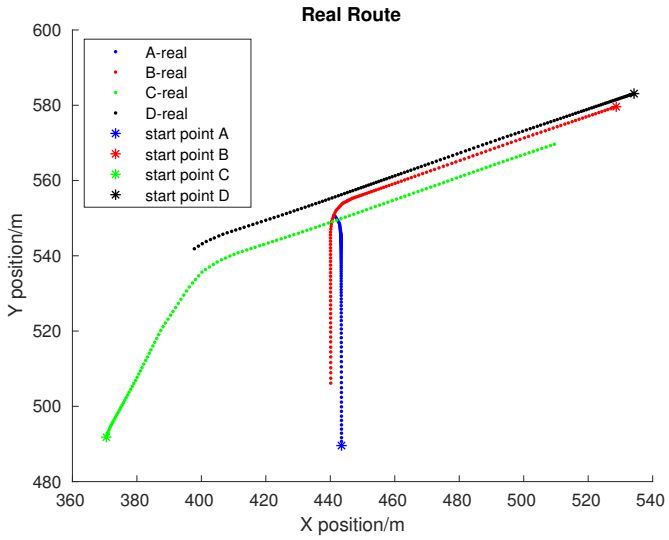

Figure 4.23: Real route for each vehicle

The dots and stars are the same description as above discussion. The simulation results of SCKF algorithm with fully implementations are separately shown in Figure A.37 and A.38 with single random seed in noise generation in MATLAB. Figure A.39 and A.40 are the simulation results in terms of distribution with GPS-distance-velocity implementation. In statistics point of view, the mean distance error is approximately 2.4 m and variance of distance error is about 1.7 m^2. The mean position error is zero with 1.9 m^2 variance error of position. When SCKF algorithm uses distance-velocity implementation, Figure A.41 and A.42 are plotted to show the analysis results in the single seed for noise generation in MATLAB. Figure A.43 and A.44 show the distribution analysis, which the mean distance error is approximate 3.6 m and its variance is about 3.4 m^2. The mean position error is zero with 2.8 m^2 variance of position error.

4.3 Simulation Results

Collecting all the simulation results from above, Table 4.6 and 4.7 show the mean distance error, variance of distance error and variance of position error both in GPS-distance-velocity implementation and distance-velocity implementation. Both tables neglect mean position error because they are almost close to zero in the Gaussian noise assumption model. Scenario 0 is that four vehicles all go straightly when they come to cross. Scenarios from 1 to 5 are described as the titles in our discussion above.

Scenario	Mean distance error(m)	Variance distance error(m^2)	Variance position error(m^2)
0	3.4	3.4	2.7
1	3.5	3.3	2.8
2	3.4	3.0	2.7
3	3.4	3.0	2.7
4	3.2	2.9	2.6
5	3.6	3.4	2.8
Average	3.4	3.2	2.7

Table 4.6: Comparison among different scenarios in distance-velocity implementation

Scenario	Mean distance error(m)	Variance distance error(m^2)	Variance position error(m^2)
0	2.3	1.7	1.8
1	2.3	1.9	1.9
2	2.2	1.6	1.8
3	2.2	1.6	1.8
4	2.2	1.6	1.8
5	2.4	1.7	1.9
Average	2.3	1.7	1.8

Table 4.7: Comparison among different scenarios in GPS-distance-velocity implementation

In Table 4.6, the average of mean distance error is about 3.4 m with 3.4 m^2 error variance among six different scenarios when SCKF algorithm uses distance-velocity implementation. While in Table 4.7, the average of mean distance error is about 2.3 m with GPS-distance-velocity implementation. From the simulation results, the distance-velocity implementation is about only one meter less accurate than GPS-distance-velocity implementation. In other words, as normal GPS error with 30 meters assumed in the thesis, the implementation with GPS measurement barely improves positioning. But it increases the payload in wireless network heavily. Every vehicle share one more GPS measurements to other vehicles. If the GPS error increases, the distance error in GPS-distance-velocity implementation increase accordingly. Figure 4.24 shows the analysis result that keeping all parameters same as Table 4.1, 4.2 and 4.3 shown, but varies the standard derivation of GPS measurement noise from 30 to 2000 meters. The simulation result is shown:

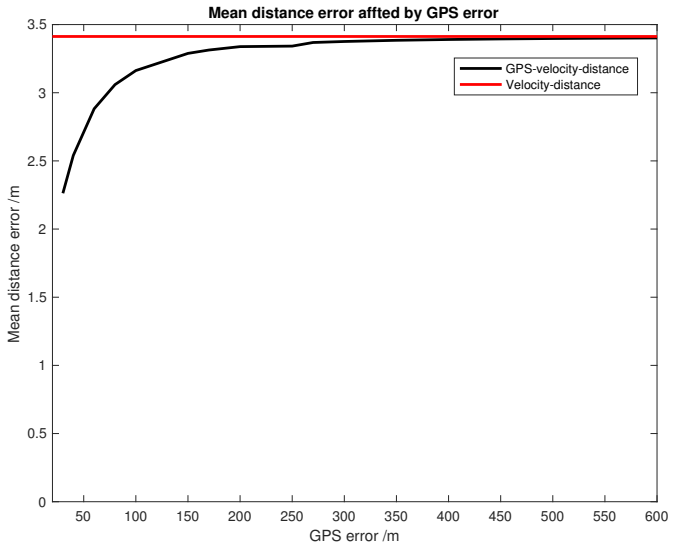

Figure 4.24: Distance error affected by GPS error

X axis is the standard derivation of GPS error and the Y axis is the distance error from SCKF algorithm. The straight line in red and curve line in black represent the simulation result of the cooperative algorithm with distance-velocity implementation and GPS-distance-velocity implementation respectively. Since distance-velocity implementation does not suffer from GPS error, the distance error is a constant no matter how the GPS error changes. In the GPS-distance-velocity implementation, the error of simulation result increases along with the increment of GPS error. From Figure. 4.24, the black line tends to close to the red line as GPS error increasing. It means that the difference of distance error between these two implementations will be eliminated. It also means that GPS-distance-velocity implementation is only one meter more accurate than distance-velocity implementation at most time in the urban scenario.

In conclusion, utilizing the cooperative algorithm SCKF can estimate position of vehicles in the urban scenario without GPS measurements. The SCKF algorithm with cooperative measurement works quite good in most urban scenario, because the mean distance error and the mean position error slightly variate from their average values as shown in Table 4.6 and 4.7. The algorithm will not diverge even the vehicles run on the different ways. The position estimates with distance-velocity implementation performed at most one meter worse than GPS-distance-velocity implementation. To positioning, this implementation improves a lot than only GPS measure.

Chapter 5
Conclusion

5.1 General Conclusion

In conclusion, the cooperative algorithm SCKF with vehicle information is a good tool to help vehicle to estimate the position in the urban scenario. As WiFi technology being developed in the high-speed movement area, this method is an economical solution when GNSS is not working properly. The Kalman filter improves positioning by using the cooperative positioning with the distance measurement, which is implemented in the Kalman filter. In the good positioning estimates from GPS sensor, the Kalman filter with cooperative positioning will get a much better positioning with less position error and less position error covariance than the GPS sensor. When the vehicle share more information with others, the positioning of vehicles in this global system will getting much better. From the thesis, the way to positioning without GNSS in the urban scenario is achievable with our cooperative positioning method. As more and more kinds of vehicle information sharing with each other, the cooperative positioning method will help vehicle get better positions even without using GNSS.

5.2 Future Work

A real road network is so complicated that the Kalman filter to update vehicles positions seems not good enough in the single model. Based on the complicated road network scenarios, the multi-models combining many physical models can be picked up by vehicles automatically. For example, the single model with small process noise is chosen to estimate position in the high way scenario when the vehicle moves in relative steady distance and velocity. While in the urban road network, vehicles suffer from many types of road network, so that multi-models can be picked up to against the unstable road network.
As for the measurement noise implementation, a fixed value is hard to represent the real measurement process precisely. In this thesis, the measurement noise is set to a little bit larger than the real measurement noise variance to against the complex and highly dynamic road networks. In the future work, these error variance of sensors can be estimated by adding a pilot to their noise measurement off-line and the measurement noise is updated to the Kalman algorithm from

time to time. With a very good estimate in the measurement noise, the algorithm performs a better way to estimate position.
To improve the positioning, one possible way is to improve the accuracy of distance sensors and velocity sensors. It stabilizes the algorithm and gets a better performance in positioning. The other way is to combine more sensor measurements such as acceleration sensors to restrict the algorithm and to limit the estimates of position.
All the position estimates are based on simulation software. The algorithm should be tested in the real system. The vehicle should use the measurements obtained from vehicle sensors and uses 802.11p to share the information with others.

Bibliography

[1] D.Dardari, E.Falletti, and M.Luise,*"Satellite and Terrestrial Radio Positioning Techniques-A Signal Processing Perspective"*. London, U.K.:Elsevier, 2011, p.464.

[2] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu *"Survey of Wireless Indoor Positioning Techniques and Systems"*, published in IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) p.1067 - 1080, Nov. 2007.

[3] Sven Fischer *"Observed Time Difference Of Arrival (OTDOA) Positioning in 3GPP LTE"* in 2014 Qualcomm Technologies, INC, June,6 2014.

[4] Davide Dardari, Pau Closas, and Petar M.Djuric *"Indoor Tracking: Theory, Methods, and Technologies"* published in IEEE Transactions on Vehicular Technology, Volume:64, Issue:4, April 2015.

[5] J.M.Castro-Arvizu, P.Closas and J.A.Fernandez-Rubio, *"Cramer– Rao lower bound for breakpoint distance estimation in a path-loss model"*, published in Proc.IEEE ICC Workshop Adv.Netw. Localization Navigat, Sydney, Australia, Jun. 2014.

[6] M.Bolic, M.Rostamian and P.M.Djuric, *"Proximity detection with RFID: A step toward the Internet of things"*, pubished in IEEE Pervasive Computing, Volume:14, Issue:2, Apr.-June 2015.

[7] J.Chung, *"Indoor location sensing using geo-magnetism,"* in Proc.9th Int. Conf. Mobile Syst., Appl., Services., ACM, 2011, pp. 141–154.

[8] Ihn-Sik Weon, Soon-Geul Lee, Sang-Chan Moon *"Precise localization of a vehicle within a driving lane by combining the vehicle trajectory with vision information"*, published in Control, Automation and Systems (ICCAS), 2016 16th International Conference. 16-19 Oct. 2016.

[9] Jianmin Duan, Zhixue Song, Changren Wang, Dan Liu *"Inertial Navigation Algorithm Based on Modified Kalman Filter and Wavelet Technique for Intelligent Vehicle"*, published in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2016 8th International Conference, 27-28 Aug. 2016.

[10] Gary Bishop and Greg Welch, *"An Introduction to the Kalman Filter"*, Department of Computer Science University of North Carolina at Chapel Hill Chapel Hill, NC 27599-3175, July 24, 2006.

[11] Ienkaran Arasaratnam and Simon Haykin, *"Cubature Kalman Filter"*, published in IEEE Transactions on Automatic Control. Volume:54, Issue:6, page:1254 - 1269, June 2009.

[12] Eric A.Wan and Rudolph vander Merwe *"The Unscented Kalman Filter for Nonlinear Estimation"*, published in Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, 4-4 Oct. 2000.

[13] Emin Orhan *"Particle filtering"* Department of Brain & Cognitive Sciences University of Rochester, NY14627, USA, August 9, 2012.

[14] Brown,R.G.and Hwang,P.Y.C. *"Introduction to Random Signals and Applied Kalman Filtering"*, Second Ed., Wiley, New York, 1992.

[15] Feng Shen and Guanghui Xu, *"An Enhanced UWB-Based Range/GPS Cooperative Positioning Approach Using Adaptive Variational Bayesian Cubature Kalman Filtering"*, published in Mathematical Problems in Engineering. College of Automation, Harbin Engineering University, 26 December 2014.

[16] R.Cools, *"Constructing cubature formulas: The science behind the art"*, Act a Numerica, vol.6, pp.1–54, 1997.

[17] Paul D.Groves *"Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems"* published in book Acta Numerica, Cambridge, Page74, January 1997.

[18] M.Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*, published in IEEE Transactions on Signal Processing. Volume:50, Issue:2, Feb 2002.

[19] A.Doucet *"On sequential Monte Carlo methods for Bayesian filtering"* Dept. Eng., Univ. Cambridge, UK, Tech. Rep., 1998.

[20] R.vander Merwe, A.Doucet, J.F.G.deFreitas, and E.Wan, *"The unscented particle filter"*, Adv. Neural Inform. Process. Syst., Dec. 2000.

[21] G.Kitagawa, *"Monte Carlo filter and smoother for non-Gaussian non- linear state space models"*, J.Comput. Graph.Statist., vol.5, no.1, pp.1–25, 1996.

[22] Huiting Cheng, Yasushi Yamao*"Performance Analysis of ITS V2V Broadcast Communication Using CSMA/CA and a Roadside Relay Station at Intersections"* published in Performance Analysis of ITS V2V Broadcast Communication Using CSMA/CA and a Roadside Relay Station at Intersections. October 10, 2012.

[23] Mariam Elazab, Aboelmagd Noureldin and Hossam S.Hassanein *"Integrated Cooperative Localization for Connected Vehicles in Urban Canyons"*, published in Global Communications Conference (GLOBECOM), 2015 IEEE. Electrical and Computer Eng. Dept., Queen's University, Canada, Dec. 2015.

[24] SUMO official website *http://www.sumo.dlr.de/daily/userdoc/Downloads.html* fetched at June, 2017.

[25] OSM official website *http://www.openstreetmap.org* fetched at June, 2017.

[26] NS3 official website *https://www.nsnam.org/ns-3-dev* fetched at June, 2017.

[27] URL:*"http://geoawesomeness.com/how-accurate-is-your-smartphones-gps-in-an-urban-jungle/"*, fetched at June, 2017.

[28] URL:*https : //en.wikipedia.org/wiki/Nakagami_distribution*, fetched at June, 2017.

[29] URL:*https : //en.wikipedia.org/wiki/Shape_parameter*,fetched at June, 2017.

[30] URL:*https://www.cnet.com/news/south-korea-aims-more-accurate-gps-at-navigation-systems/* fetched at June, 2017.

Appendix A

Appendix

A.1 The Kalman Filter Algorithm

Time Update:
1) Assume old posterior density function is known at time step k:

$$X_{k|k-1} = F\hat{X}_{k-1|k-1} + Bu_k \tag{A.1}$$

2) Error covariance

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_{k-1} \tag{A.2}$$

Measurement Update:
1) Kalman Gain

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{A.3}$$

2) Updated State

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - HX_{k|k-1}) \tag{A.4}$$

3) New Posterior covariance

$$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{A.5}$$

A.2 The CKF Algorithm

Time Update:
1) Assume old posterior density function is known at time step k:

$$p(X_{k-1|k-1}|Z_{1:k-1}) = N(\hat{X}_{k-|k-1}, P_{k-1|k-1}) \tag{A.6}$$

Factorize the old posterior covariance error:

$$P_{k-1|k-1} = S_{k-1|k-1}S_{k-1|k-1}^T \tag{A.7}$$

2) Cubature points

$$X_{i,k-1|k-1} = \sqrt{\frac{2n}{2}}S_{k-1|k-1}[1]_i + \hat{X}_{k-1|k-1} \tag{A.8}$$

where i= 1,2,$\cdots$,2n.
3) Propagated Cubature points

$$X^*_{i,k|k-1} = f(X_{i,k|k-1}) \tag{A.9}$$

4) Estimate predicted state

$$\hat{X}_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} X^*_{i,k-1|k-1} \tag{A.10}$$

5) Predicted error covariance

$$P_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} X^*_{i,k|k-1}X^{*T}_{i,k|k-1} - \hat{X}_{k|k-1}\hat{X}^T_{k|k-1} + Q_{k-1} \tag{A.11}$$

Measurement Update:
1) Factorize the predicted error covariance

$$P_{k|k-1} = S_{k|k-1}S^T_{k|k-1} \tag{A.12}$$

2) Cubature points

$$X_{i,k|k-1} = \sqrt{\frac{2n}{2}}S_{k|k-1}[1]_i + \hat{X}_{k|k-1} \tag{A.13}$$

3) Propagated Cubature points

$$Z_{i,k|k-1} = h(X_{i,k|k-1}) \tag{A.14}$$

4) Predicted Measurement

$$\hat{Z}_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} Z_{i,k|k-1} \tag{A.15}$$

5) Innovation covariance

$$P_{zz,k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} Z_{i,k|k-1}Z^T_{i,k|k-1} - \hat{Z}_{k|k-1}\hat{Z}^T_{k|k-1} + R_k \tag{A.16}$$

6) Cross-covariance matrix

$$P_{xz,k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} X_{i,k|k-1}Z^T_{i,k|k-1} - \hat{X}_{k|k-1}\hat{Z}^T_{k|k-1} \tag{A.17}$$

7) Kalman Gain

$$K_k = P_{xz,k|k-1}P^{-1}_{zz,k|k-1} \tag{A.18}$$

8) Updated state

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k(Z_k - \hat{Z}_{k|k-1}) \tag{A.19}$$

9) New posterior covariance

$$P_{k|k} = P_{k|k-1} - K_k P_{zz,k|k-1}K^T_k \tag{A.20}$$

A.3 SCKF Algorithm

1) Cubature points

$$X_{i,k-1|k-1} = \sqrt{\frac{2n}{2}} S_{k-1|k-1}[1]_i + \hat{X}_{k-1|k-1} \tag{A.21}$$

where i=1,2,$\cdots$,2n.
2) Propagated cubautre points

$$X^*_{i,k|k-1} = f(X_{i,k-1|k-1}) \tag{A.22}$$

3) Predicted state

$$\hat{X}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} X^*_{i,k|k-1} \tag{A.23}$$

4) Square-root factor of Predicted error covariance

$$S_{k|k-1} = Tria([X^*_{k|k-1}\, S_{Q,k-1}]) \tag{A.24}$$

where

$$Q_{k-1} = S_{Q,k-1} S^T_{Q,k-1} \tag{A.25}$$

and weighted, centered matrix:

$$X^*_{k|k-1} = \frac{1}{\sqrt{2n}} [X^*_{1,k|k-1} - \hat{X}_{k|k-1}\, X^*_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X^*_{m,k|k-1} - \hat{X}_{k|k-1}] \tag{A.26}$$

and *Tria* is general triangularization algorithm (such as QR decomposition), S=Tria[A]. In order to get the lower triangular matrix using QR decomposition. Letting upper triangular matrix R be the QR decomposition of A^T, so that the lower triangular matrix would be $S = R^T$.

Measurement Update:
1) Cubature points

$$X_{i,k|k-1} = \sqrt{\frac{2n}{2}} S_{k|k-1}[1]_i + \hat{X}_{k|k-1} \tag{A.27}$$

2) Propagated Cubature points

$$Z_{i,k|k-1} = h(X_{i,k|k-1}) \tag{A.28}$$

3) Predicted Measurement

$$\hat{Z}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} Z_{i,k|k-1} \tag{A.29}$$

4) Square-root of the innovation covariance

$$S_{zz,k|k-1} = Tria([Z_{k|k-1}\, S_{R,k}]) \tag{A.30}$$

where $S_{R,k}$ is a square-root of R_k that $R_k = S_{R,k}S_{R,k}^T$. And weighted, centered matrix

$$Z_{k|k-1} = \frac{1}{\sqrt{2n}}[X_{1,k|k-1} - \hat{X}_{k|k-1}\, X_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X_{m,k|k-1} - \hat{X}_{k|k-1}] \quad \text{(A.31)}$$

5) The cross-covariance matrix

$$P_{xz,k|k-1} = X_{k|k-1}Z_{k|k-1}^T \quad \text{(A.32)}$$

where the weighted, centered matrix

$$X_{k|k-1} = \frac{1}{\sqrt{2n}}[X_{1,k|k-1} - \hat{X}_{k|k-1}\, X_{2,k|k-1} - \hat{X}_{k|k-1} \cdots X_{m,k|k-1} - \hat{X}_{k|k-1}] \quad \text{(A.33)}$$

6) The Kalman gain

$$K_k = (P_{xz,k|k-1}/S_{zz,k|k-1}^T)/S_{zz,k|k-1} \quad \text{(A.34)}$$

7) Updated State

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k(Z_k - \hat{Z}_{k|k-1}) \quad \text{(A.35)}$$

8) Square-root factor of the new posteriori error covariance

$$S_{k|k} = Tria([X_{k|k-1} - K_k Z_{k|k-1}\, K_k S_{R,k}]) \quad \text{(A.36)}$$

A.4 The Structure of Generic Particle Filter

1) Creating particles based on Gaussian distribution.
Draw:

$$x_k^i \sim q(x_k|x_{k-1}^i, z_k) \quad \text{(A.37)}$$

where i is the i_{th} particles, which it is set to up to N_s particles totally.
Assign the particle weight w_k^i

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \quad \text{(A.38)}$$

Calculate the total weight and normalized them

$$w_k^i = \frac{w_k^i}{\sum_{i=1}^{N_s}(w_k^i)} \quad \text{(A.39)}$$

Calculate the effective sample size N_{eff}

$$N_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \quad \text{(A.40)}$$

If $N_{eff} < N_T$
Resample the particles using the following algorithm:
Initial the CDF:

$$C_1 = 0$$

for i=2:N_s construct CDF:

$$c_i = c_{i-1} + w_k^i \quad \text{(A.41)}$$

A.5 Figure

A.5.1 Scenario 0

Four go straightly and none vehicle makes a turn in the crossing.

GPS-Distance Implementation

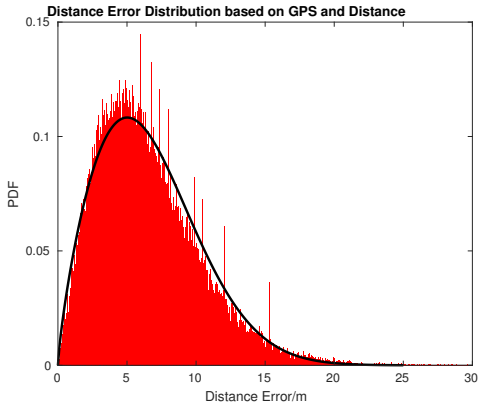

Figure A.1: Distribution of distance error with GPS and Velocity

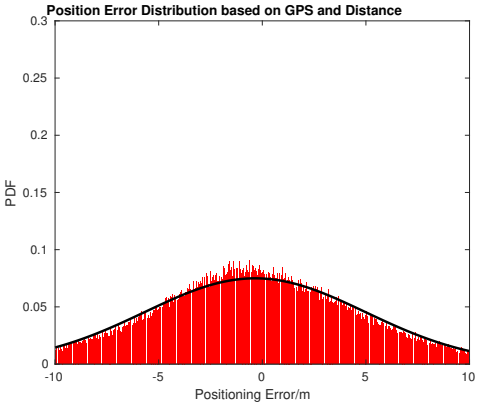

Figure A.2: Distribution of position Error with GPS and Velocity

GPS-Velocity Implementation

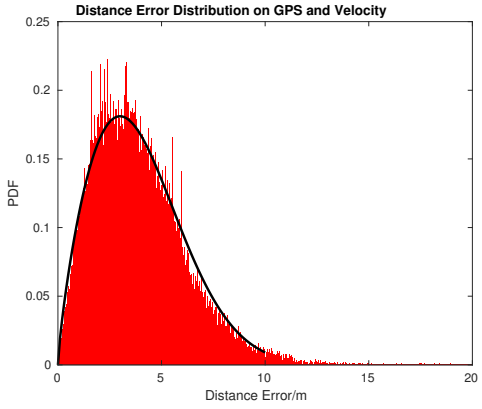

Figure A.3: Distribution of distance error with GPS and Velocity

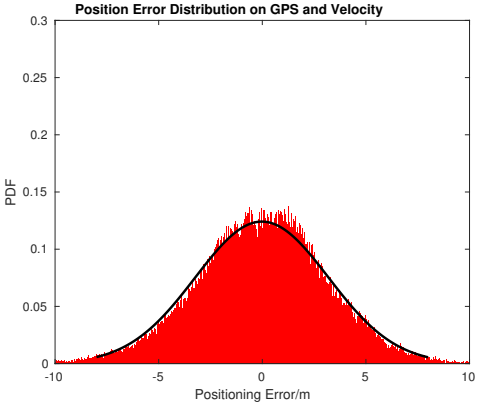

Figure A.4: Distribution of position Error with GPS and Velocity

A.5.2 Scenario1

Three vehicles go straightly and only one vehicle make turn at the crossing.

GPS-Distance-Velocity Implementation

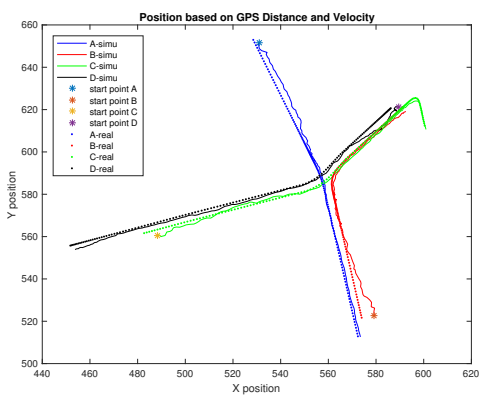

Figure A.5: Simulation result with GPS, Distance and Velocity

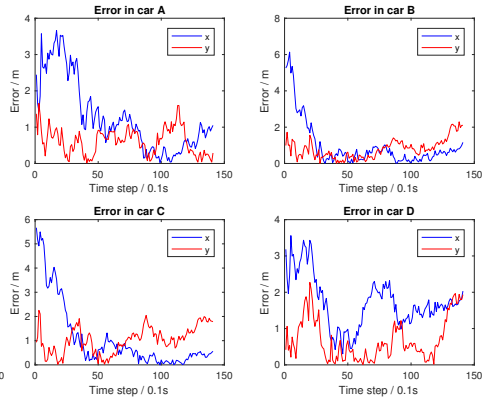

Figure A.6: Position error for each vehicle

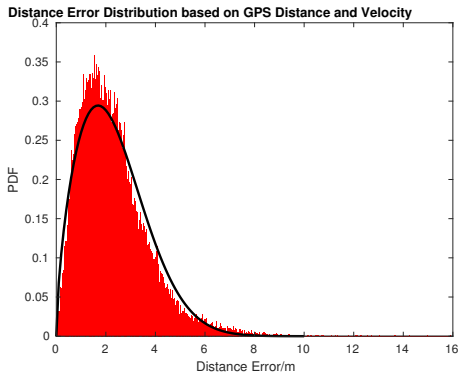

Figure A.7: Distribution of distance error with GPS, Velocity and Distance

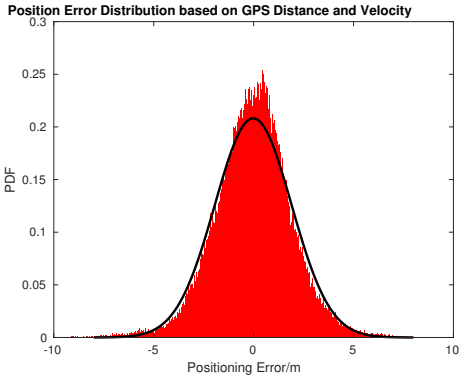

Figure A.8: Distribution of position Error with GPS, Velocity and Distance

Distance-Velocity Implementation

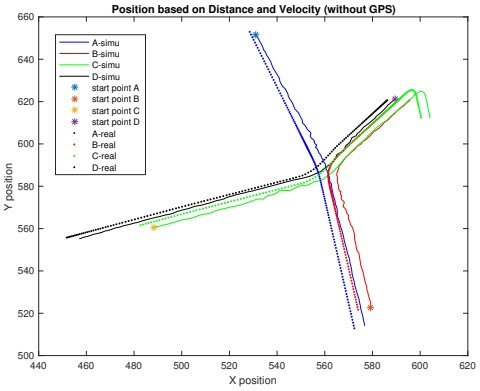

Figure A.9: Simulation result with Distance and Velocity

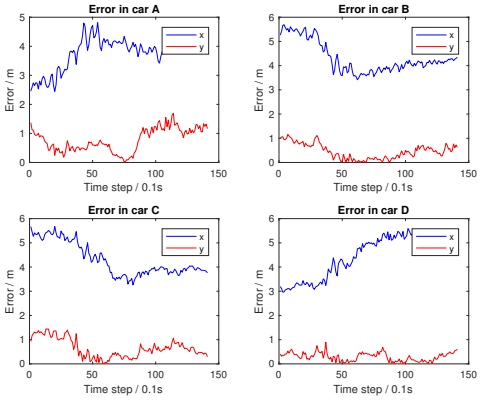

Figure A.10: Position error for each vehicle

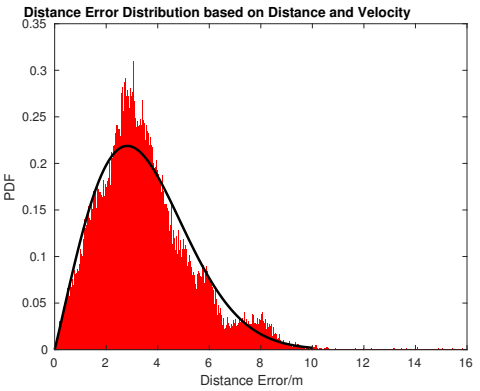

Figure A.11: Distribution of distance error with Velocity and Distance

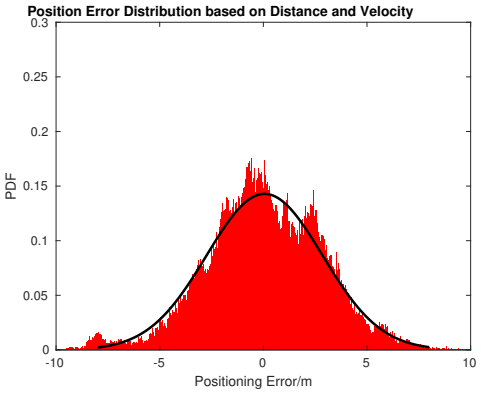

Figure A.12: Distribution of position Error with Velocity and Distance

A.5.3 Scenario2

Two vehicles go straightly and two vehicles make turn at the crossing.

GPS-Distance-Velocity Implementation

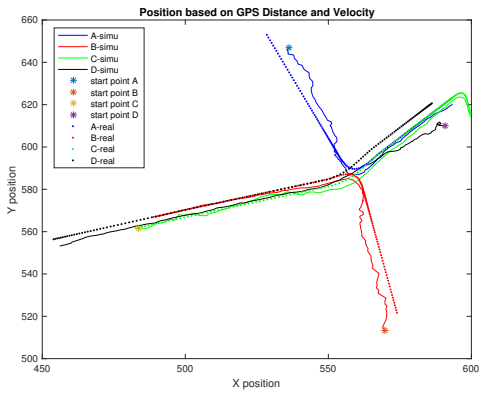

Figure A.13: Simulation result with GPS, Distance and Velocity

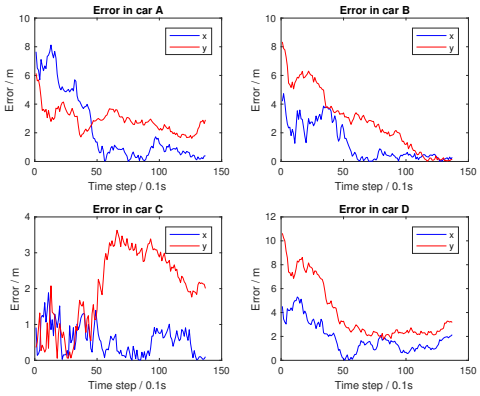

Figure A.14: Position error for each vehicle

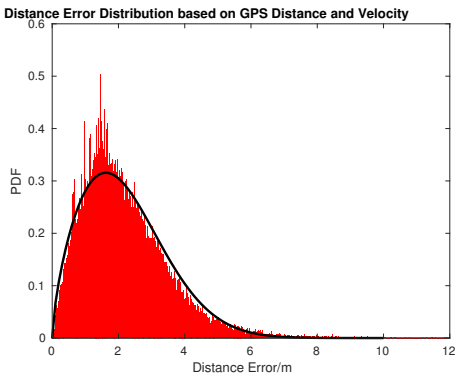

Figure A.15: Distribution of distance error with GPS, Velocity and Distance

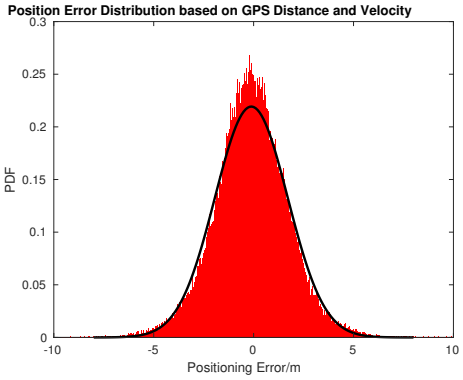

Figure A.16: Distribution of position error with GPS, Velocity and Distance

Distance-Velocity Implementation

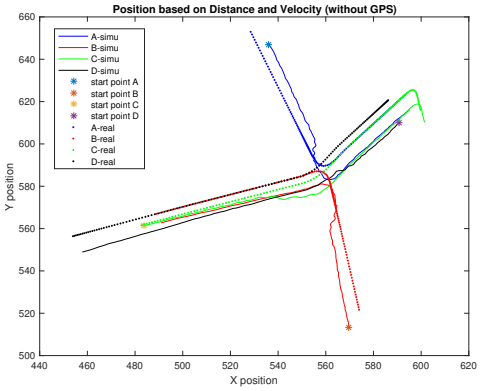

Figure A.17: Simulation result with Distance and Velocity

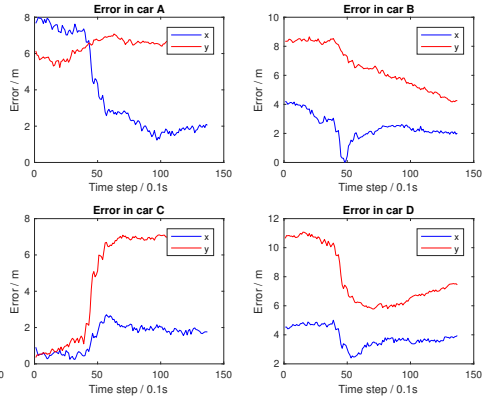

Figure A.18: Position error for each vehicle

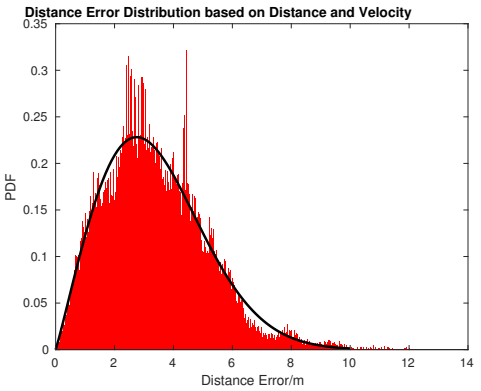

Figure A.19: Distribution of distance error with Velocity and Distance

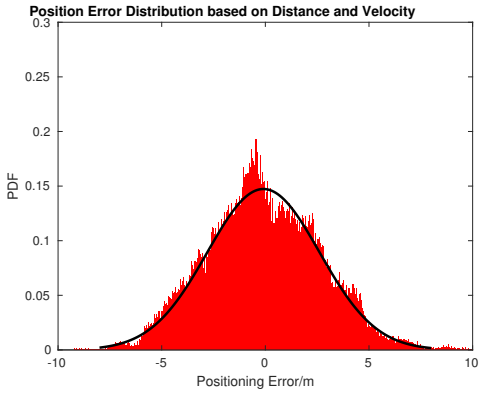

Figure A.20: Distribution of position error with Velocity and Distance

A.5.4 Scenario3

Only one vehicle go straightly and three vehicles make turn at the crossing.

GPS-Distance-Velocity Implementation

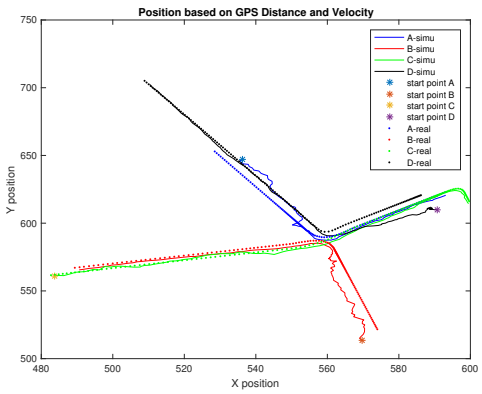

Figure A.21: Simulation result with GPS, Distance and Velocity

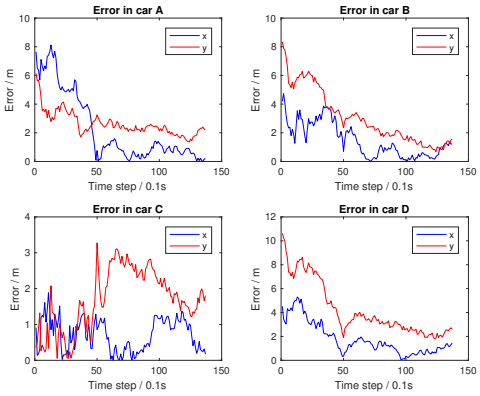

Figure A.22: Position error for each vehicle

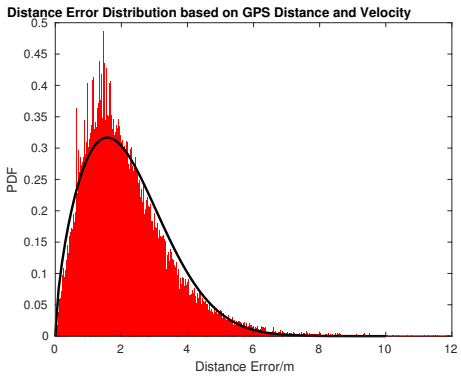

Figure A.23: Distribution of distance error with GPS, Velocity and Distance

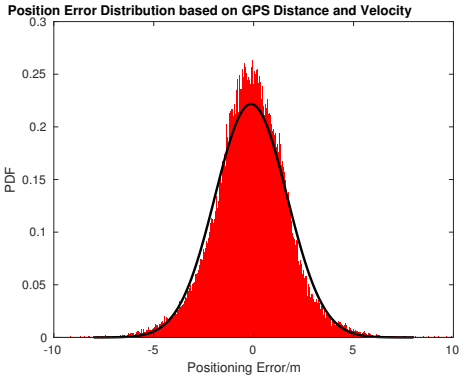

Figure A.24: Distribution of position error with GPS, Velocity and Distance

Distance-Velocity Implementation

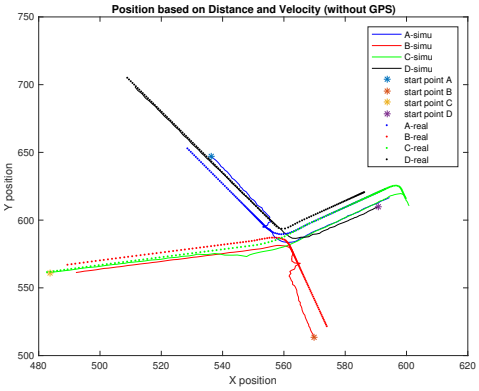

Figure A.25: Simulation result with Distance and Velocity

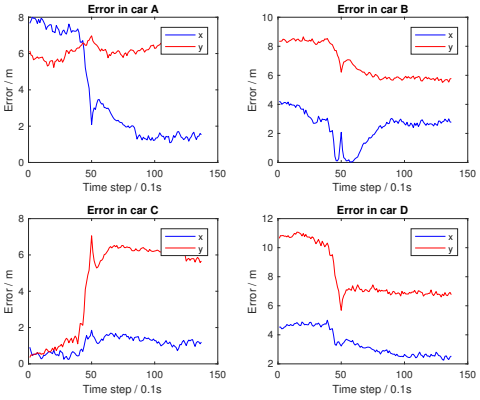

Figure A.26: Position error for each vehicle

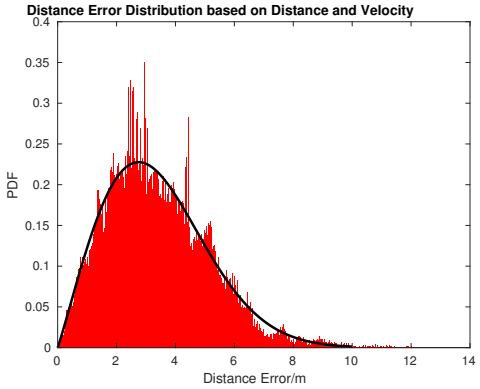

Figure A.27: Distribution of distance error with Velocity and Distance

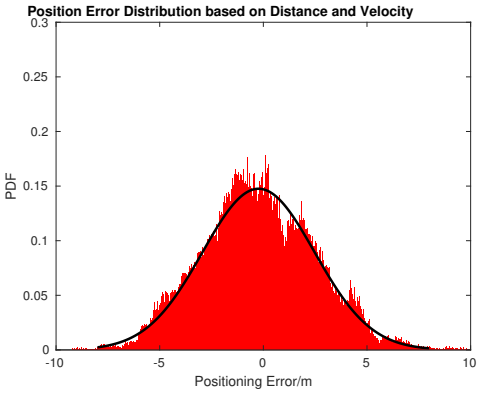

Figure A.28: Distribution of position error with Velocity and Distance

A.5.5 Scenario4

None vehicle go straightly and all vehicles make turn at the crossing.

GPS-Velocity-Distance Implementation

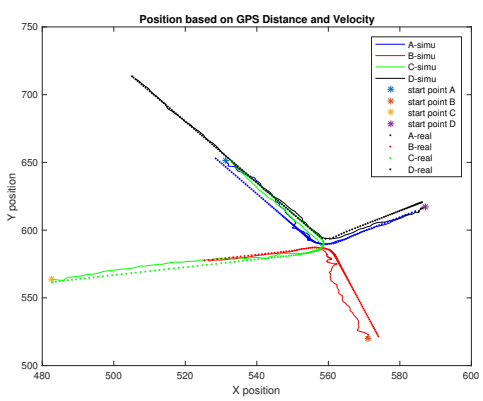

Figure A.29: Simulation result with GPS Distance and Velocity

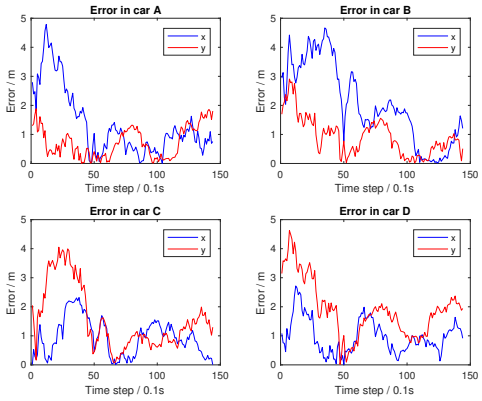

Figure A.30: Psition error for each vehicle

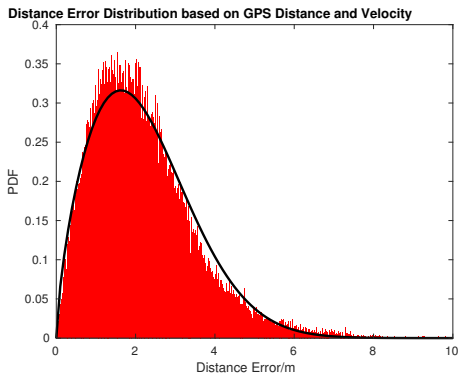

Figure A.31: Distribution of distance error with GPS, Velocity and Distance

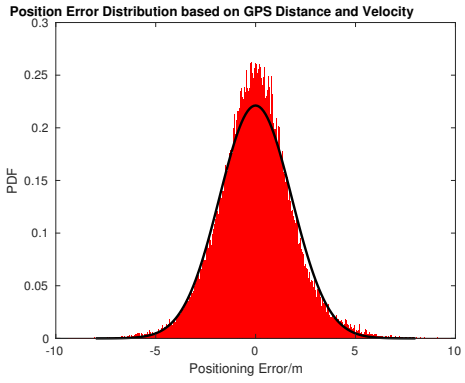

Figure A.32: Distribution of position error with GPS Velocity and Distance

Distance-Velocity Implementation

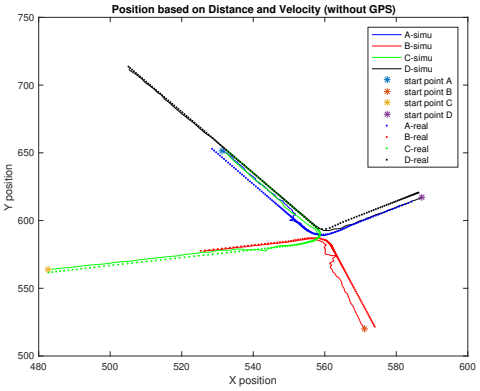

Figure A.33: Simulation result with Distance and Velocity

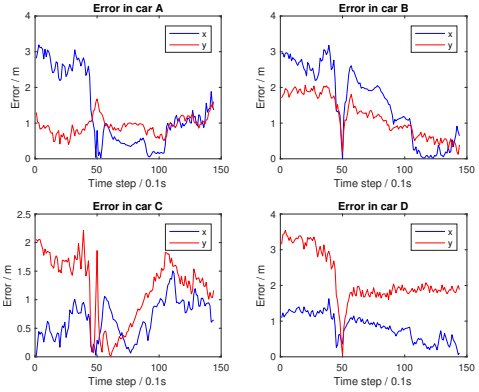

Figure A.34: Position error for each vehicle

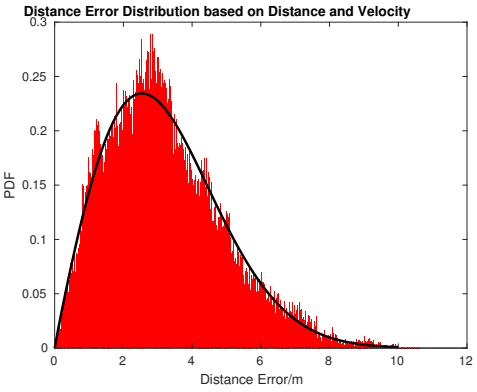

Figure A.35: Distribution of distance error with Velocity and Distance

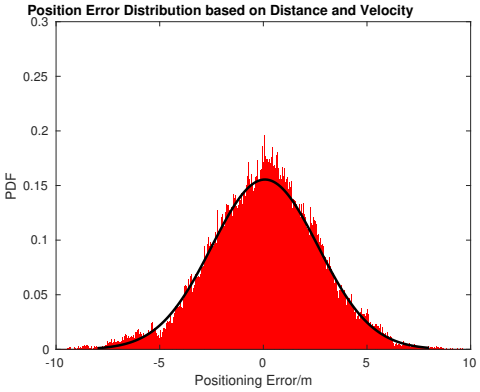

Figure A.36: Distribution of position error with Velocity and Distance

A.5.6 Scenario 5

All four vehicles meet in T-cross while two vehicles go straightly, one vehicle makes turn and one vehicle stop at the T-cross.

GPS-Velocity-Distance Implementation

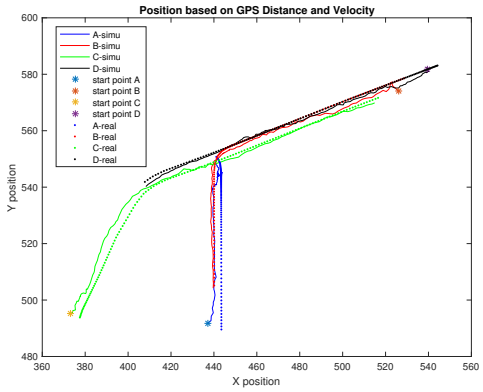

Figure A.37: Simulation result with GPS, Distance and Velocity

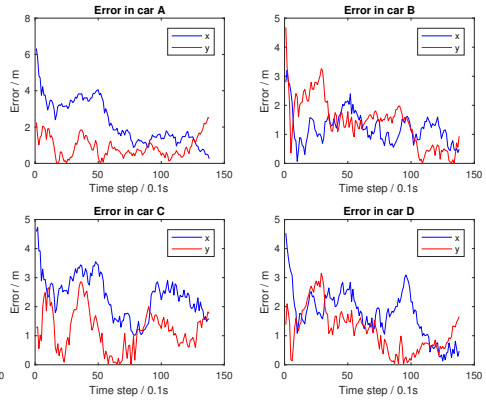

Figure A.38: Position error for each vehicle

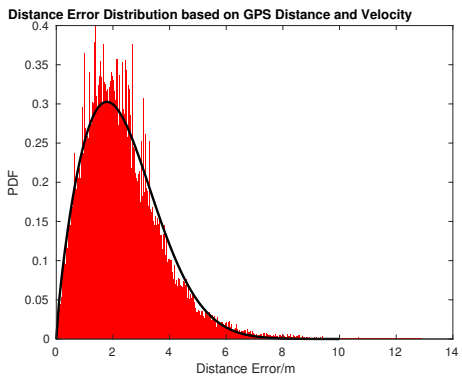

Figure A.39: Distribution of distance error with GPS, Velocity and Distance

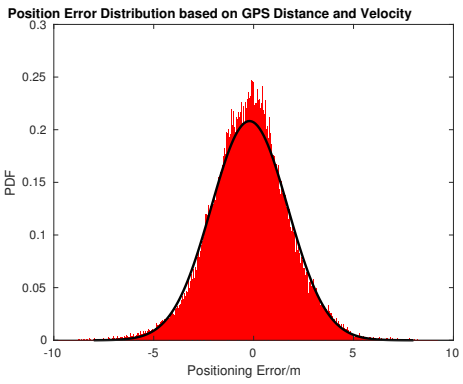

Figure A.40: Distribution of position error with GPS, Velocity and Distance

Distance-Velocity Implementation

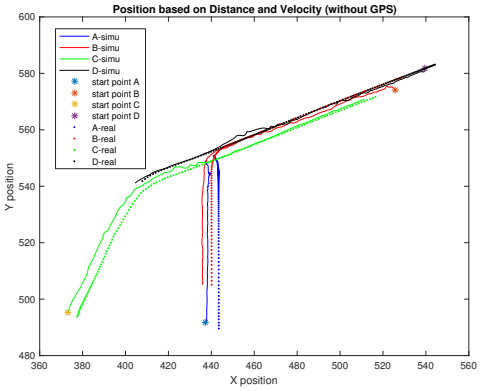

Figure A.41: Simulation result with Distance and Velocity

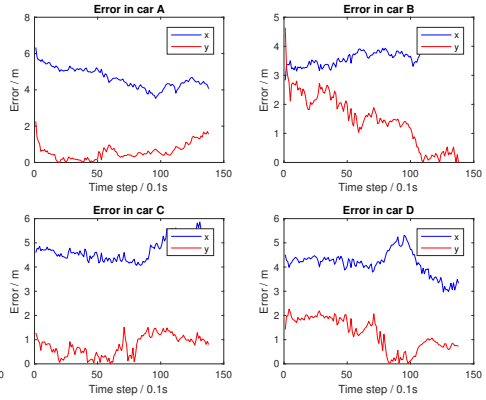

Figure A.42: Position error for each vehicle

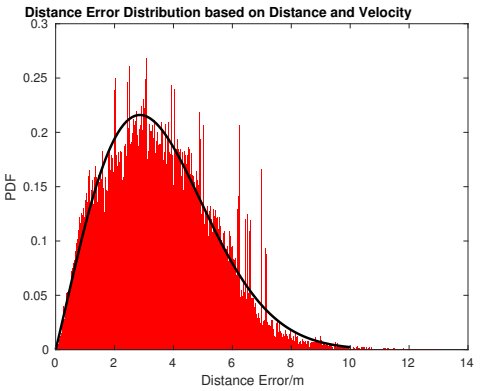

Figure A.43: Distribution of distance error with Velocity and Distance

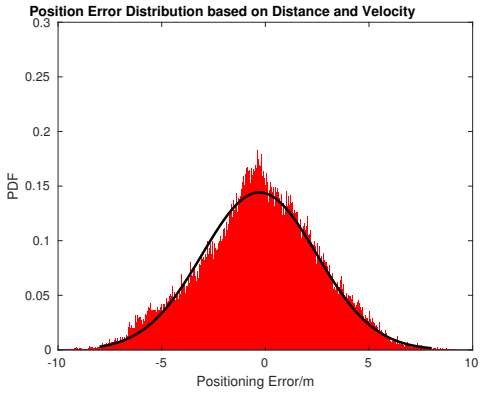

Figure A.44: Distribution of position error with Velocity and Distance