Tracking Vehicular Motion-Position Using V2V Communication

by

Zheng Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents the formulation and validation of tracking algorithms for vehicular motion for use in active collision prevention in V2V communications. The main objective is to estimate position and velocity of a vehicle based on update from vehicular wireless network. By using vehicular wireless network, the range of position estimation improves when compare to conventional radars and sensors. On the other hand, from a vehicular wireless network point of view regular measurement information update is more difficult to obtain because of packet losses due to interference between communicating vehicles. Our proposed algorithms are based on methods from position tracking termed alpha-beta trackers in aerospace applications with constant rate of information updates, with some modifications to better solve the problem. We present the main algorithms and provide numerical evidence of their accuracy based on simulation data. The modified filters are shown to be computationally efficient (lightweight) and provide sufficient accuracy for estimation of vehicle positions based on information update in a wireless V2V system.

Acknowledgments

I would like to thank my supervisor Professor Ravi R. Mazumdar for his support and guidance. His consultation and help helped me to stay focused and brought me back when I have become strayed in side details. It has been a great experience to learn and research under his supervision. I would also like to thank Dr. Sayee C. Kompalli for his help in formulating the basic problem model, which becomes the most fundamental part of my research. Last but not least, I would like to thank my family and colleagues who supported me during my research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The need for active collision detection and prevention mechanisms is ever increasing because of the congestion on major highways and the need to maintain smooth traffic flow. With the increasing use and availability of wireless systems in automobiles the use of the wireless capability to develop active collision prevention systems through V2V communications has become a reality. The advantage is that this will allow for the decentralization of resources and remove the need to provide centralized communications and control infrastructure that would be extremely costly and difficult to deploy across the vast highway system.

As the number of cars increase, traffic collisions become a major issue to personal wellbeing and maintaining the efficiency and smooth flow along a highway. It is widely known that driver behavior such as the inability to respond to sudden changing conditions or failing to watch for danger are some of the main causes of accidents. Although there are certain unavoidable accidents, if drivers can receive warnings in advance then at least the major damage and disruption due to chain crashes can be avoided. There are many papers discussing this topic. [5, 13] For instance, in [17] there is a proposal on how crashing might be avoided if the car driving ahead can send emergency break notice through a Vehicular Ad Hoc Networks (VANET) to the cars following it, or sending a red light notice to drivers when a car is approaching intersection. The purpose of this research was to investigate on how inter-vehicle communications could be used to provide a given vehicle a picture of the positions of other vehicles in their neighborhood as well as information on their speeds. The objective was to develop an algorithm to achieve such a goal given that the vehicles would communicate over the VANET which

is prone to packet losses and collisions due to the communication over a wireless medium.

In addition to the safety concern, motion estimation is also important for other applications. In [13] it is said that almost all unicast routing protocols require information regarding the position of vehicles. For instance, when one needs to use the VANET to forward packets using geologically based routing techniques, it is not realistic to know the position of all the vehicles that will act as intermediate forwarding node beforehand. However, positions in near future given by motion estimation may be valuable data for such technology when it needs to make decision.

Of all the traffic accidents caused by human error, there are a large number attributed to existence of blind spot area, the area where a driver cannot see from mirrors and need to directly turn one's head to look at. For various reasons a driver might fail to find cars in the blind spot when one performs a lane change. These reasons may range from where fault lies entirely at the driver, such as assuming no vehicle, to more inevitable reasons, such as lack of visibility, etc. By utilizing the VANET, we could perform motion estimation of another vehicle and eliminate this problem.

The techniques of motion estimation and position tracking have been well developed in the context of aerospace applications such as navigation and air traffic control. Here it is assumed that position data obtained from GPS and inertial navigation systems is relayed at regular intervals upon which the necessary position estimates are computed. The only real issue is that of error mitigation due to observation noise. Collisions are avoided by either maintaining prescribed flight paths or by proximity warning radar systems. These are mostly warning-type systems. The tracking systems are used by air traffic controllers to ensure that planes are maintaining their flight paths. In normal conditions warnings are conveyed by air traffic controllers to the pilots via dedicated communication channels. The position estimation algorithm is based on the dynamics of motion and the technique for noise reduction includes the Kalman filter and other filters such as $\alpha - \beta$ trackers. For instance [9] gives an example of tracking the state of an airplane by Kalman filter used in an automated landing system. In general, Kalman filters are used for applications that require information on the precision of estimate and have better computational power and resources such as the Apollo program. On the other hand, $\alpha - \beta$ trackers are used for cheaper applications that do not have much computational power, including mobile radars that can only update the po-

sition of aircrafts. In this work we show how the above tracking algorithm from the aerospace applications can be modified to be used to estimate vehicle position and velocity.

In V2V systems a centralized system will be too expensive to implement and therefore impractical. The rapid change of node locations and connections established and lost within the network should be considered as well. In addition the probability of collision is much higher due to driver behavior as well as traffic conditions. We also need to have more accurate estimates of position because the distance scales are much smaller and the reaction time is much shorter. However, the motion of vehicles is still governed by the same laws of motion but now we not only need to determine location but also speed and acceleration because drivers rarely maintain steady speeds. Moreover because of the use of a wireless V2V network, packet collisions mean that packets are delayed and/or lost which means the algorithm must be robust enough to account for missing or unavailable information to perform updates. In the following section, we shall explain why we put emphasis on the packet collision nature of V2V network.

1.1 V2V Communication And Its Issues

Vehicular wireless communication is an emerging research area enabled by the improvement of performance and reduced cost of wireless devices. Most of the researches in this area are now focused on the so called Vehicular Ad Hoc Networks (VANET). It is mainly designed for the improvement of traffic safety by extending the already existing sensors and radars, but now encompasses possibility to build a wider range of applications on it. It has similar properties with other wireless networks, but it also has its own specific problems.

As stated in [8], the VANET enables exchange of data among nearby vehicles in order to enhance road safety. Prior to the VANET, there were many projects that allowed a vehicle to detect surrounding environment by sensors and radars, but these data were restricted to the vehicle itself. This means critical safety information is only available to the vehicles that have a ”line-of-sight” to the source of such information. Due to this limitation of information availability we can say that such sensors and radars are only extensions to the driver’s sight and hearing, and cannot prevent accidents that may occur due to obstacles in the ”line-of-sight” of the sensors and radars. One good example can be found in [2] where three vehicles are driving in a single lane. The example shows that without the VANET,

if the vehicle driving in the front applies an emergency breaking, the last vehicle driving might not detect it early enough if it only monitors the vehicle in the middle, and hence crashes onto the vehicle in the middle. On the other hand, when the vehicle in the front and in the end both have the VANET then the vehicle in the front may send warning so the vehicle in the end may apply breaking early enough to avoid crashing. Although this accident example may be attributed to inappropriate driving because the vehicles only have a space of 32 meters, a distance that only takes one second for the vehicles cover, whereas the standard distance is to keep two seconds worth distance between the vehicle in front, a system that tries to improve vehicular safety should take such frequent violation into consideration in order to be successful.

Besides the vehicle safety, researchers are now looking to incorporating a wider range of applications in the VANET. For example, in the FleetNet Project, a project partially supported by the Government of Germany and ended in 2003, suggests three main areas of applications. These are cooperative driver assistance, decentralized floating car data, and user communications and information services. Of these three areas, cooperative driver assistance is more concentrated on achieving safe driving by providing emergency notification and other advanced warnings to assist the driver. In contrast with cooperative driver assistance, decentralized floating car data allows a better driving experience by passing the data collected to the vehicles in the surrounding so that vehicles may find road congestion levels and dangerous situations such as bad weather without direct observation. This type of features do not necessarily prevent the traffic accidents directly, but they provide a way to find routes that are less dangerous and stressful that may keep the drivers more alert on the road. The third type of applications, user communications and information services, is more concentrated on convenience and fun. For instance, the user communications include applications that allow conversation among passengers with nearby vehicles and other network applications for fun. The information services enabled by the FleetNet Project include access to the internet and advertising from road side units such as gas price of the gas station at next service station or next exit on the highway. Other projects suggest that the vehicle status may be monitored via the VANET and towing and repair services may be contacted in advance if there is something wrong with the vehicle.

Development of the VANET involves roughly three types of groups, the governments, the corporations, and the academics. The governments are involved because of the legislation issues such as bandwidth allocation, reduction of con-

gestion and gas emission, and road safety. The corporations working on the development of the VANET include both the car manufacturers and wireless device manufacturers. The academic groups work on the principal theoretic to solve issues from how to efficiently utilize the allocated bandwidth to the architectures of safety applications to the routing of safety information along the flow of traffic.

There are still many issues for the VANET yet to be solved. The resource allocation and management are not yet finalized though the remaining candidates are reduced to DSRC and UTRA TDD. Since the VANET uses no base station, the limited resources need to be managed efficiently in a distributed fashion in order to achieve robust and reliable communication even at high vehicle speed and high node concentration. Routing of packets is also a problem because the forwarding path changes very frequently due to vehicles moving in and out of communication range or vehicles making turning and obsolete the forwarding path. Another issue involving the VANET is the security of communication. Since some applications may need to uniquely identify the vehicles identity theft will be a problem. Similarly violated use of priority packets and illusion attacks may cause serious accidents on the road. There are some authentication mechanisms proposed such as IEEE 1609.2 or [16]. However, the VANET is still more vulnerable to attacks since it requires communication with random vehicles and road side units open to modification by malicious hands.

Since the VANET can be described as a variation of mobile ad hoc networks (MANETs), it shares some properties and problems with MANETs, but also has its own unique problems that may be already solved in other types of wireless network due to the fast motion of its nodes. One such problem that may have serious impact to the vehicular safety application such as our vehicle position estimation by causing packet collision is the hidden node problem. The hidden node problem is a well-known issue to researchers in the area of wireless medium access control. The problem occurs when there are two nodes located close enough to another node or access point to communicate with it, but the two nodes themselves are not in communication range of each other so they cannot communicate directly. Here the two nodes are referred to be hidden from each other. In other type of wireless networks, there are several techniques proposed to solve this problem. There are many solutions studied other than the famous IEEE 802.11 RTS/CTS and Wi-MAX time division. A list of such solutions may be found in [12], namely adjustment of the transmission range, direction of the antennas, improvement of transmission path between the hidden nodes, moving of the nodes, etc. All of the solutions are not ideal to solve the hidden node problem in the VANET. Since

the nodes in the VANET are vehicles running with possibly a very high speed compared to other wireless networks, the topology of the nodes also change very fast. The IEEE 802.11 RTS/CTS method requires synchronicity among the nodes, which will be hard to maintain given the nodes always move in and out of communication range. Similarly, the Wi-MAX time division method requires a central unit to assign the time slots, which may not exist in the current VANET infrastructure. Adjustment of transmission power method aims to reveal hidden nodes by increasing transmission range so that the nodes who could not hear each other can directly communicate, but this solution does not work very well for the VANET because many vehicles may be driving one after another and the power required to cover all vehicles becomes too large. The method involves direction of antennas tries to expose the nearby hidden terminals using Omni-directional antennas but the VANET already uses Omni-directional antennas since the safety information it is transmitting is usually useful for all vehicles in the surrounding. Furthermore if any vehicle wants to point its antenna to a specific object the vehicle needs to estimate the position of the object it is pointing because the relative direction and distance are likely to change while the transmitting vehicle is driving along the road. Improvement of transmission path attempts to discover the hidden nodes by removing obstacles to achieve a shorter and better transmission path. This method again does not work for the VANET because the vehicles themselves hardly have any means to change their surroundings. If there are some separators between the lanes that block the signals between them, there is nothing a vehicle can do to remove them. If a road side unit is placed on such separator then it creates a hidden node problem. Methods that attempt to solve the hidden node problem by moving nodes also attempts to make all nodes hear each other by placing them close enough. Needless to say that in the VANET all nodes are moving according to some other rules that cannot guarantee all participating nodes can hear each other. A simple example would be where the vehicle can also communicate with vehicles driving in the opposing direction. Parts of the nodes are guaranteed to move to edge of communication range and may not be reachable to some other nodes. However, this area is still under research and in the future there may be protocols and methods that can actually solve the hidden node problem for the VANET.

1.2 Vehicle Position Estimation Techniques

There have been studies committed to estimate position of vehicles via update from the VANET. There are GPS, map matching, dead reckoning, cellular localization, image/video processing, localization service, ad hoc localization, just to name techniques that maybe used alone or in combination. Some of the techniques are proposed independently and could be older than the VANET itself, whereas others are more closely related to the characteristics of the VANET.

GPS based techniques utilize the satellite signals provided by the Global Positioning System and the advantage is that these signals can be received almost anywhere on the Earth. In other words, there is no need to build additional infrastructure to use this type of techniques except for certain areas where there are only poor satellite signal reception. Map matching based techniques usually involves at least another type of technique to give a rough idea of the position of the vehicle. The advantage of map matching is that as long as the geographic information database is well maintained this technique can significantly limit the probability distribution of the vehicle over space. Since there are now open source projects such as OpenStreetMap providing geographical information updated every day online all devices that can access the Internet will be able to download such information and apply map matching based on it. Dead reckoning based techniques requires no external infrastructure and gathers all data by the vehicle itself. The preciseness of estimate will depend on the type and measurement error of the sensors, so there is option to trade off cost with estimation precision for every vehicle. Although this type of techniques are not able to estimate the position of any other vehicles, the vehicle may broadcast the parameters it uses to make estimate over the VANET so that other vehicles may use them to predict the position of the transmitter. Cellular localization techniques are also the techniques used in mobile phone tracking. Similar to the GPS, this type of techniques also do not need additional infrastructures and is expected to work in the areas where the satellite signals might be over-degraded but cellular signals remain relatively strong. Image/video processing techniques are generally used to estimate the position of a vehicle by third party. These techniques require setting up the cameras along the road, but when used correctly they can provide much more precise and detailed estimate of the position of the vehicle. Localization service refers to techniques using local infrastructures, either alone or combined, to provide position of a vehicle. These infrastructures could be tuned according to the characteristic of the environment in order to make estimation of the vehicle position and replace

other type of techniques such as GPS in environments where there is poor satellite reception. Ad hoc localization techniques estimate position of other vehicles by measuring relative distances and sharing it via the VANET. One example using the radio signal strength of the VANET to measure inter-vehicle distance can be found in [14]. These relative positions might be converted to global position by matching any one of the vehicle with its GPS readings. More detailed discussion regarding these techniques can be found in [3].

These techniques are typically used in combination. Techniques to find own position by combining GPS with dead reckoning and map matching has been proposed in [11]. Here the vehicle tries to find its own position by processing GPS data with map database and adds data from on-board sensors to it. In [17] there is a vehicle tracking system by using GPS information and status of the vehicle, such as yaw rate, steering, and brake, in other words, dead reckoning data, to make estimation of vehicle position. However, this technique would fail in case an update packet is lost, because the author was trying to reduce amount of wireless transmission as much as possible and only transmits complete updates when the estimator within the transmitting vehicle has detected a large enough error.

In order to make position estimation possible in environment where packet might be lost, we need some form of filter that could be easily set up and can switch between mode that has error correction and mode that does not have error correction quickly. Also, we need to make the filter a computationally inexpensive one so to guarantee that in case there are many vehicles within reception range the filter will not occupy too much of the limited resources in the vehicle on-board unit.

1.3 Organization Of Thesis

In Chapter 2, we provide a simple model we used to formulate a basic problem in order to find appropriate algorithm to make vehicle position estimate based on lossy wireless update, and overviews of Kalman filter and $\alpha - \beta$ trackers as they occur in aerospace applications. In Chapter 3, the actual filter mechanism is explained and some experimental results are discussed. In Chapter 4 we introduce the tools and methods to generate more realistic data and the evaluation result of our algorithms against these data. We conclude with some observations and discussion of our future work in developing robust estimation algorithms.

Chapter 2

Problem Formulation And Trackers

In the following section, we shall introduce a prototype problem we have formulated to find our algorithm. In this problem we shall consider problems related to the motion of vehicle and the nature of the vehicular wireless communication. The details of the algorithms we used to solve this problem is in Chapter 3, and in Chapter 4 we explain how we tested our algorithms against more realistic settings.

2.1 Basic Problem Formulation

We begin by first presenting a simplified scenario in order to develop an appropriate vehicle position and velocity estimation algorithm. We perform estimation of position, velocity, and acceleration of a single vehicle running on a single lane road simulated. The vehicle will broadcast its velocity and position (with some measurement error) periodically to the estimator. This information update packet is assumed to be very small and the transmission time is negligible. Similarly, the propagation delay is also considered very small compared to the time between two estimations, due to the fact that transmission is made in a relatively small range and only single hop communication, and thus transmission delay due to speed of light effects is ignored. The motion and acceleration of a vehicle is assumed to follow a model unknown to the estimator. Our goal is to make regular estimates of vehicle information both when there is an information update and no update. This is because we cannot assume that a vehicle remains at the old location when there is no update or information could have been sent but the information has

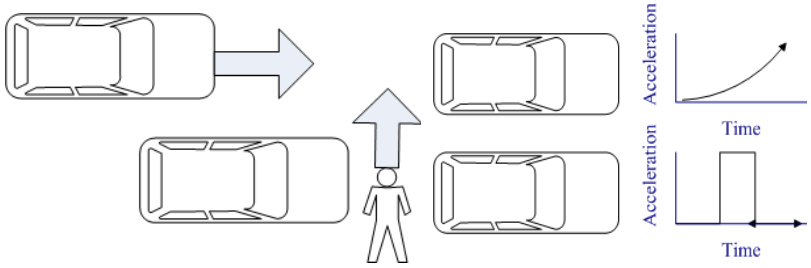
been lost. We must be able to predict where the vehicle is likely to be periodically at sufficiently short intervals of time in order to be able to react to any changes in the location of the other vehicles due to a maneuver or due to acceleration, lane changes, etc.

The motion of the vehicle is described Newton's Law of Motion, that is:

$$\begin{bmatrix} d(t) \\ v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 1 & t_d & \frac{t_d^2}{2} \\ 0 & 1 & t_d \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d(t-t_d) \\ v(t-t_d) \\ a(t-t_d) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{F(t)}{m} \end{bmatrix}$$

where m is the mass of the vehicle, $d(t)$ is the position, $v(t)$ is the velocity, $a(t)$ is the acceleration, and $F(t)$ is force applied to the vehicle at time t. In order to make estimation of vehicle position and speed even for vehicles that can only remain in the communication range of the VANET for a short period, we should not assume that the change of acceleration has a normal distribution. Although when we look at the acceleration of a vehicle in long period it should sum up to zero since it has to start moving and stop moving at some point, it is not true when we are focusing on only the part of the motion when the vehicle joins the communication range of the VANET. It is not possible to model the acceleration using auto-regressive (AR) models since the acceleration depends on the force, which is the result of the driver's action based on one's knowledge about the road condition at time t, and does not depend on previous condition. For instance, when the driver is driving on the road one might be accelerating to reach desired speed, but at the next moment one might be decelerating to avoid danger such as people or other objects that dash out from corner. Also, the behaviors of drivers differ from one another, and one might prefer to change acceleration slowly where another prefer to change rapidly. We may consider that the cruising motion with constant speed as 2D, i.e. $[d(t), \dot{d}(t)]$, motion with constant acceleration as 3D, i.e. $[d(t), \dot{d}(t), \ddot{d}(t)]$, and motion with constant first moment of acceleration as 4D. The 3D case corresponds to the drivers who will change acceleration rapidly in the bottom part of Figure 2.1b, and the 4D case corresponds to the drivers who will change acceleration in a relatively constant pace in the top part of Figure 2.1b. In other words, we are trying to make estimation of vehicle position without knowledge regarding how the motion model might switch from constant speed to constant acceleration, or to constant first moment of acceleration. In order to obtain such knowledge the motion estimation system will require too much information on the surrounding environment and involve private information such as preference of acceleration,

which will make the system over complex and impractical.



(a) Driver may decelerate due to unforeseen condition.

(b) Drivers having completely different preference in accelerating.

Figure 2.1: Possible scenarios for driver's behavior.

The coordinate used for vehicle movement is limited to one-dimension for this simple scenario since we are assuming single lane road. This vehicle is expected to move with a given velocity vector that indicates vehicle velocity at every 0.1 second in simulation, and thus acceleration during this 0.1 second is considered constant. The estimator is assumed to always stay in the range of communication. The extension of the model to a two dimensional model is quite straightforward except that now one must have a good model of the two dimensional force field that is applied by the driver, i.e. if the driver swerves, changes lane, etc.

We assume that there exists a constant packet lost rate (due to both error and interference) for each experiment. In addition, we assume that the measurement noises for position and velocity are Gaussian, and have their own variance.

$$y(t) = Ax(t) + n(t)$$

where $y(t) = [d_m(t), v_m(t)]^T$, $x(t) = [d(t), v(t), a(t)]^T$, and $n(t) = [n_d(t), n_v(t)]^T$. The subscript m denotes the measurement. The variances for the noise on the distance and velocity are based on the errors from the corresponding measurement devices namely the GPS and speedometers and are fixed at $\sigma_d = 5$ m and $\sigma_v = 0.3$ (m/s) (approximately 1km/h measured by the speedometer).

In order to test this simple scenario we have run 8 experiments with different settings. Every experiment consists of 100 simulations with random packet lost. The result of each experiment consists of statistics of all simulations during the

experiment and estimation result of the 100th simulation. In each simulation, the information update packets are filtered randomly based on the previously stated packet lost rate. Only those updates that are considered successful are used in error correction. A small white noise of variance 0.01 to change the velocity, with unit in meter per second, is added to the velocity so that there are no simulations based on exactly same velocity history. In the following chapter we shall introduce widely used vehicle position tracking techniques. Our algorithms and their experimental results are presented in Chapter 3.

2.2 Overview Of Kalman Filter

The Kalman filter is a recursive filter widely used to estimate the state of a dynamic process based on state-space model. Since it has been introduced in [10] it has been widely used in the areas of engineering such as orbit estimation, radar tracking, control systems, and signal processing. It is known to be efficient because it does not need to keep past data and optimum because it minimizes the mean-squared estimation error.

For the original Kalman filter, the system model of the process to be estimated is assumed to have following form:

$$\begin{aligned} x_k &= F_k x_{k-1} + B_k u_k + w_k \\ y_k &= H_k x_k + v_k \end{aligned}$$

where x_k is the state of process at time k, y_k is the observation of the state at time k, F_k is the linear state transition from previous state, $B_k u_k$ is how input $u_k$will affect the state at time k, H_k is the linear observation model, and w_k and v_k are the process noise and observation noise, respectively. The noises are assumed to have zero mean Gaussian distribution with specific covariance Q_{w_k} and Q_{v_k}. Based on these assumptions, the Kalman filter algorithm calculates the predicted state and its covariance, and then makes an update to the estimate when new observation corresponding to the estimated state becomes available.

Apart from the most basic form of Kalman filter, there have been many modified versions developed. One such modification is called Extended Kalman filter that enables prediction of non-linear process models based on assumption that current state is given from a function f, and observation is given from another

function h, and both of them are differentiable functions. Another important derived filter is Ensemble Kalman filter that can process large number of states or particles by approximating the Kalman filter covariance with sample covariance. Other modifications of the Kalman filter include making estimation with incomplete data. In [6], there has been a study in this problem and the solution is to simply not make update when the update y_k is completely missing.

However, there are some problems in applying Kalman filter to our case. One particular problem is that we do not have information regarding the change in vehicle motion models. As stated in Section 2.1, there are three different possible motion models a particular vehicle might use, and the transition between them are highly dependent on the driver's preference and perception of the environment. Another problem that might make Kalman filter solution difficult is the limitation of computation power of on-board unit and the large number of vehicles needed to be estimated.

2.3 Overview Of $\alpha - \beta$ Trackers

The origin of $\alpha - \beta$ trackers is from aerospace applications where there is the need to estimate position and velocity from noisy observations of the current position that is obtained from radar. [1] It could be said that $\alpha - \beta$ trackers are special version of Kalman filter simplified by exploiting the characteristics of the model it is trying to estimate. It is possible to achieve optimality when designed carefully [18] but sub-optimal implementation usually yields good performance and requires less computation.

An $\alpha - \beta$ tracker, also known as an $\alpha - \beta$ filter[1], is a filter that does not require detailed knowledge of the system it is observing. This form of filter is widely used in radars that has limited resource and assumes constant velocity. In order to have better tracking result on objects that have constant acceleration instead of constant velocity, a modified version of the tracker is used, called the $\alpha - \beta - \gamma$ tracker[2] (or filter). Although such a tracker is usually suboptimal when compared to Kalman filters, it has a surprisingly good performance when one considers its simplicity compared to Kalman filter.

[1] Another name is the g-h filter.

[2] There also exists a g-h-k filter which is a slightly modified version of an $\alpha - \beta - \gamma$ tracker. The difference is $k = 2\gamma$.

These filters remain to be one of the important tracking techniques in the radar industry.

The $\alpha - \beta$ filter tracks an object in two stages. These two stages are called prediction stage and estimation stage. In the prediction stage, the position and velocity of the object is predicted based on previously estimated result:

$$\begin{aligned} d_{p(n+1)} &= \hat{d}_n + T\hat{v}_n \\ v_{p(n+1)} &= \hat{v}_n \end{aligned}$$

where $\hat{d}_n$ and $\hat{v}_n$ denote the estimated values at the n-th sampling instant, the instants being T seconds apart. The subscript p denotes the prediction for the $(n+1)$-th sampling instant.

In the estimation stage, the tracker tries to correct prediction error using measured position with α and β (both could be constant values) as multiplier of the difference between measurement and prediction and is given by:

$$\begin{aligned} \hat{d}_n &= d_{pn} + \alpha(d_{mn} - d_{pn}) \\ \hat{v}_n &= v_{pn} + \frac{\beta}{T}(d_{mn} - d_{pm}) \end{aligned}$$

where the terms with the m subscript denotes the observed data and the subscript p denotes the predicted data. The values α and β weigh the correction between the observed data and the predicted data, smaller values imply that the prediction error has little influence which might be the case when the noise variance is large. Larger values enable faster tracking or account for greater weight to the prediction error which is typically the case when the error due to noise is small. Typically the values are chosen in $(0, 1)$.

Clearly, the two steps above do not require knowledge of how motion evolves or how much error is there in the measurement. The convergence of $\alpha - \beta$ trackers is only guaranteed with constant coefficients when the velocity is constant.

Chapter 3

Algorithms For Vehicular Motion-Position Estimation

3.1 $\alpha - \beta - \delta$ Tracker

In this section we discuss the algorithms that we propose for estimation tailored to the information pattern in the context of vehicular position-motion estimation that takes into account the particular constraints imposed by the V2V paradigm.

There exists a significant difference in the traditional $\alpha - \beta$ tracker setting and our problem. In traditional $\alpha - \beta$ trackers, only the position of the object is available as information update, but thanks to V2V communication we can use velocity information as well. Using this information, we decided to filter the update of velocity information using $\alpha - \beta$ tracker, and another independent parameter δ for the error correction of position. The equations for such a tracker, assuming we are making estimate at every Δt seconds at the observer, are given by:

Prediction step:

$$\begin{aligned} d_{p(n+1)} &= \hat{d}_n + \Delta t \hat{v}_n + \frac{(\Delta t)^2}{2} \hat{a}_n \\ v_{p(n+1)} &= \hat{v}_n + \Delta t \hat{a}_n \\ a_{p(n+1)} &= \hat{a}_n \end{aligned}$$

where a_n denotes the acceleration at the n-th observation point.

Estimation step:

$$\begin{aligned}
\hat{d_n} &= d_{pn} + \delta(d_{mn} - d_{pn}) \\
\hat{v_n} &= v_{pn} + \alpha(v_{mn} - v_{pn}) \\
\hat{a_n} &= a_{pn} + \frac{\beta}{\Delta t}(v_{mn} - v_{pn})
\end{aligned}$$

We call this filter $\alpha - \beta - \delta$ tracker. Now we have equations for our filter when there is regular information update. However, we know that there exists a non-zero packet loss rate due to packet collisions or due to the delays in receiving packets by the update time. Therefore some of the information packets will not reach the observer at time of estimation. This means from time to time we need to be able to estimate without measurement. In order to make some estimate at times when there is no measurement information update, we treat the prediction as our estimation, and let the error accumulate until it can be corrected with an information update. Same technique has been proposed for Kalman filter as well in [6].

Another problem that may occur and makes error correction in estimation stage difficult is that the information update might arrive in between two estimates, not exactly at the estimation time. In order to solve this problem we treat the update packets as following:

1. If the packet has arrived at exactly the same time estimation is taking place, it is used at the same estimation. In Figure 4 this corresponds to the packet T_{n-1}. The information contained in T_{n-1} is used for the estimation at time $(m-3)\Delta t$.

2. If the packet has arrived in between two estimation points, it is used at the immediate following estimation point. In Figure 4 this corresponds to the packets T_n and T_{n+1}. The information contained in T_n is used for the estimation at time $(m-1)\Delta t$.

3. If the packet has been lost, like the one transmitted between T_n and T_{n+1} in Figure 4, it will be ignored and the filter will simply regard the estimation as any other estimation that does not have update.

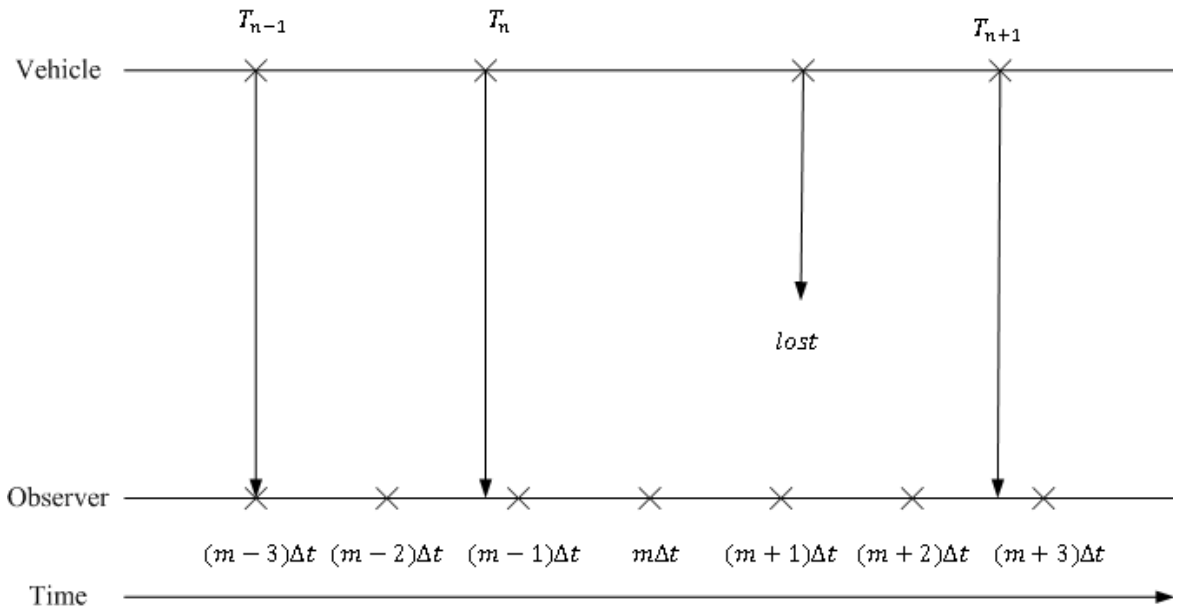


Figure 3.1: Possible measurement information update packet arrival scenario.

Now, knowing that the time between two information updates can be larger than Δt, the inter-estimation time, we need to take this into consideration for the error correction part where Δt is used. After all, the reason for dividing by Δt in the estimation part is to adjust the error from velocity to acceleration, since velocity is the integration of acceleration over time. If we do not adjust this division then we will contribute all velocity errors from several past estimations where update was not available to the current error correction. Based on this insight, the previously proposed estimation stage for a_n is modified as follows:

$$\hat{a_n} = a_{pn} + \frac{\beta}{N\Delta t}(v_{mn} - v_{pn})$$

where N indicates how many multiples of Δt seconds have passed since last estimation with a valid information update. This allows us to discount the prediction error from the measurements so that if a large number of packets are lost we disregard the error.

The following is a pseudo code for $\alpha - \beta - \delta$ tracker:

```
Estimation_timer = Δt;
Last_update_time = 1000;
Received_update = false;
for (t = 0;true;t = t + δt) {
  Last_update_time = Last_update_time + δt;
  Estimation_timer = Estimation_timer - δt;
  If (New_update) {
    Current_update = update;
    Received_update = true;
  }
  If (Estimation_timer ≤ 0) {
    If (Received_update) {
      Generate α − β − δ using Last_update_time;
      Perform prediction error correction using Current_update;
      Received_update = false;
      Last_update_time = 0;
    } Else {
      Assign predicted value to estimate value;
    }
    Estimation_timer = Δt;
  }
}
```

3.2 The Parameters α, β, δ

So far we have discussed how we would make prediction and then obtain the required estimates. We now discuss how we actually determine the values for α, β and δ. The position measurement is made through GPS, which has error variance of 5 meter, and velocity measurement is made by the vehicle itself so the error variance is much smaller. This means if the difference between the measurement and prediction is relatively small then it is probably all measurement error and if it is large then there is error that needs to be corrected. From these observations, we chose to determine the parameters α, β and δ based on the errors $d_{mn} - d_{pn}$ and $v_{mn} - v_{pn}$.

To set the appropriate values for the parameters α and β we use the equations for critically damped g-h filter in [4]

$$\begin{aligned} \alpha &= 1-\theta^2 \\ \beta &= (1-\theta)^2 \end{aligned}$$

.

The parameter θ is obtained from the relative values of the observation noise to the total uncertainty and we define it as:

$$\theta = \frac{\sigma_v}{|v_{mn} - v_{pn}| + \sigma_v}$$

The value of δ is obtained from:

$$\delta = 1 - \frac{\sigma_d}{|d_{mn} - d_{pn}| + \sigma_d}$$

The graph below shows the changes in α, β and δ as the error changes:

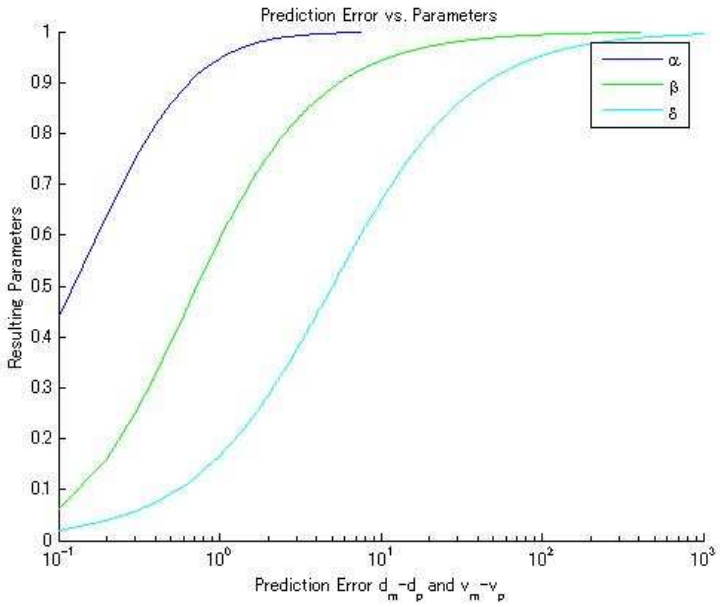


Figure 3.2: Prediction error versus α, β and δ

3.3 A 4 Parameter Tracker

In the equations for the 3-parameter α, β, δ tracker it was assumed that the acceleration was constant during the inter-estimate interval Δt. However because of packet loss or variable delays, the acceleration could change between the parameter estimation epochs. A similar algorithm can be easily created with assumption that acceleration $a(t)$ will change during the inter-estimate period, but its derivative with respect to time $\frac{da}{dt}$ remains constant.

This new algorithm performs better than the 3-parameter tracker in cases where there is high correlation between accelerations within a small period of time. The prediction and estimation stages are given below.

Prediction step:

$$
\begin{aligned}
d_{p(n+1)} &= \hat{d}_n + \Delta t \hat{v_n} + \frac{\Delta t^2}{2} \hat{a_n} \\
v_{p(n+1)} &= \hat{v_n} + \Delta t \hat{a_n} + \frac{\Delta t^2}{2} \hat{\dot{a}_n} \\
a_{p(n+1)} &= \hat{a_n} + \Delta t \hat{\dot{a}_n} \\
\dot{a}_{p(n+1)} &= \hat{\dot{a}_n}
\end{aligned}
$$

Estimation step:

$$
\begin{aligned}
\hat{d_n} &= d_{pn} + \delta(d_{mn} - d_{pn}) \\
\hat{v_n} &= \alpha(v_{mn} - v_{pn}) \\
\hat{a_n} &= a_{pn} + \frac{\beta}{N\Delta t}(v_{mn} - v_{pn}) \\
\hat{\dot{a}_n} &= \dot{a}_{pn} + \frac{\gamma}{N\Delta t^2}(v_{mn} - v_{pn})
\end{aligned}
$$

We refer to this filter as $\alpha - \beta - \gamma - \delta$ tracker. Again using equations from [4] that gives a critically damped g-h-k filter, we obtain relationship between prediction error and the 4 parameters and this relationship is shown in the following plot.

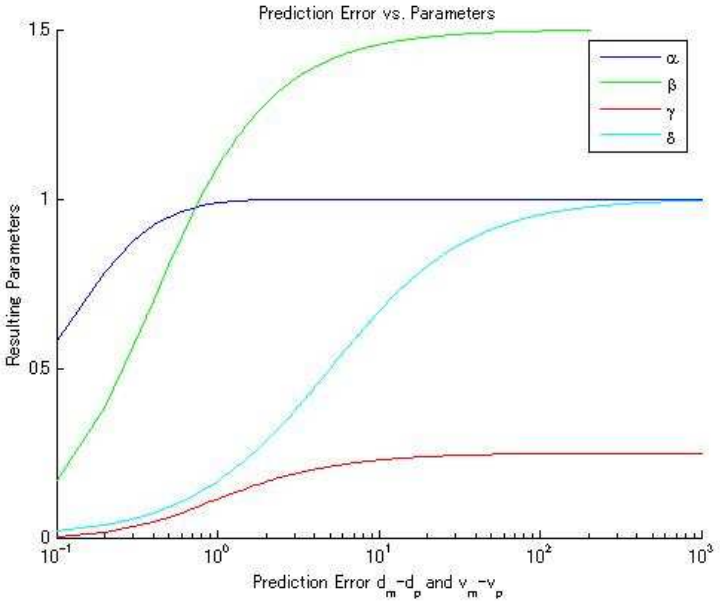


Figure 3.3: Prediction error vs. α, β,γ, and δ.

3.4 Experimental Results

8 experiments are performed under different settings, and the results can be found in Appendix A. Adjusted parameters include correlation among accelerations within a small period, information update packet loss rate, frequency of estimation, frequency of information update, and if both estimation and information update are done with certain frequency, the phase shift between them. Note that our simulation used 0.1 second as smallest interval on the *real time* discrete clock. The resulting mean and variance of position, velocity, and acceleration errors are attached to the end of this report, in the Appendix.

The results 1-4 show that the $\alpha-\beta-\delta$ tracker performs better than the 4-parameter tracker in normal driving conditions, and $\alpha-\beta-\gamma-\delta$ tracker performs better when there is high correlation among accelerations and high packet loss.

When the estimation frequency is reduced from real time to 0.5 seconds and frequency of information update remains 0.1 seconds, both of the trackers have degraded performance in general, as it can be found from result 5. This is because

of the relative long interval between updates.

On the other hand, when the estimation frequency remains 0.1 seconds and frequency of information update is reduced from 0.1 seconds to 0.5 seconds, which is the setting for result 6, most of the case the mean of the errors only increased slightly when compared to result 3. There are some exceptions where the error has a sharp increase, probably due to successive failure in information update that caused prediction error to build up in between updates. Note that here that the 4-parameter tracker performs better than the 3-parameter tracker because of the high correlation among accelerations.

In result 7, where both the estimation frequency and frequency of information update are set to 0.5 seconds, the mean of the errors again only increased slightly when compared to result 3. The spikes in result 6 no longer exist because there are much fewer estimations that do not have updates.

In result 8, there are much more error due to the fact that all estimations are using data that are 0.4 seconds old. This problem could be solved if we resynchronize the phase between update and estimation.

Chapter 4

Evaluation With More Realistic Data

In Chapter 3 we have verified that our algorithms will work for one dimensional motion. In this chapter we shall discuss further evaluation of our algorithms against data extracted from widely used traffic simulators.

4.1 Tools And Simulators

There are numbers of traffic simulators that can model motion of multiple vehicles based on user provided conditions available in both open source and commercial software. In evaluation of our algorithms, we used the OpenStreetMap project to extract road network information, eWorld to adjust and convert the road network into XML format, SUMO to read and simulate vehicle motion based on road network XML and traffic flow definition, and OMNeT++ to run application on vehicle on-board unit while communicating with SUMO via TraCI interface. Figure 4.1 gives an overview of how data are generated using these tools and simulators. In this figure, the square blocks OpenStreetMap and Vehicle flow definition are the source of data, and blocks with rounded corners are the processes that convert the data.

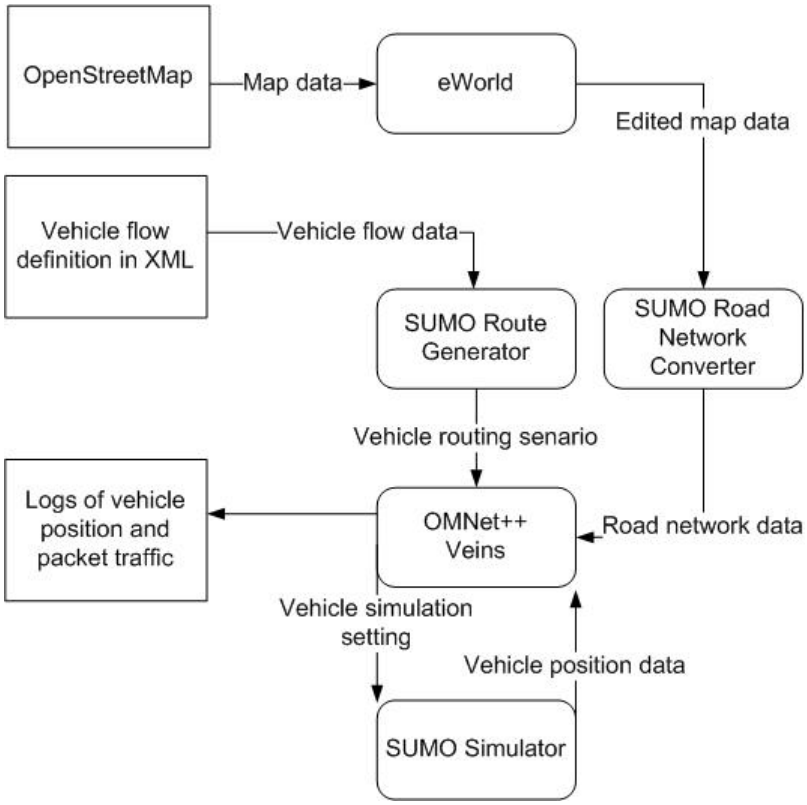


Figure 4.1: From map data and flow definition to simulation

4.1.1 OpenStreetMap

OpenStreetMap, or OSM in short, is an open source project currently hosted by VR Center for the Built Environment of University College London. The map data of this project has been built from ground survey result initially, but now there are contributions from both the government such as the United States and United Kingdom and commercial companies such as Automotive Navigation Data and Yahoo! aside from personal contributors. The data from this project is currently under Creative Commons license but the project is moving on to the Open Database License, both allow the data to be used freely by the public. For evaluation of our algorithms we have used the map data west of the junction of Interstate 80 and U.S. Route 209 in Pennsylvania, centered approximately at east longitude 75.2637 and north latitude 40.9871.

Figure 4.2: Region used for simulation

4.1.2 eWorld

eWorld is an open source project to provide a framework to convert map data from sources including OSM to formats that can be used for road traffic simulators including SUMO. The advantage in using this software is that it allows user to edit road information such as speed limit and number of lanes of each edge, presence of traffic lights and how they would be turned on and off. We used this software to change the speed limits of the highways I-80 and Route 209, which were not set in the OSM data and had been assumed as 130 km/h initially, to 100 km/h and exported the resulting road network to SUMO in XML format. The version used for this evaluation is release 0.8.3.

4.1.3 SUMO

Simulation of Urban MObility, or SUMO in short, is an open source road traffic simulator package based on microscopic movement model developed by Stefan Krauβ.[7] This simulator package allows user to import road network set-

tings from both XML files and commercial simulator packages. Simulation is performed on discrete time steps, and user can specify vehicle movement by explicitly defining it, specify vehicle flow from source edge to destination edge, or in random source and destination. This simulator also supports many other features such as different types of vehicles that have different gas emission and acceleration/deceleration, or whether a vehicle has a routing device. There exist several extensions to this simulation package, some are now obsolete while others are kept up-to-date. For evaluation of our algorithms, we used SUMO release 0.11.1, which simulates in one second per simulation step. We used the tools that came with this simulator to convert road network files and vehicle flow scenario, and used this simulator to generate position and velocity of vehicles, which are sent to OMNeT++ via TraCI interface.

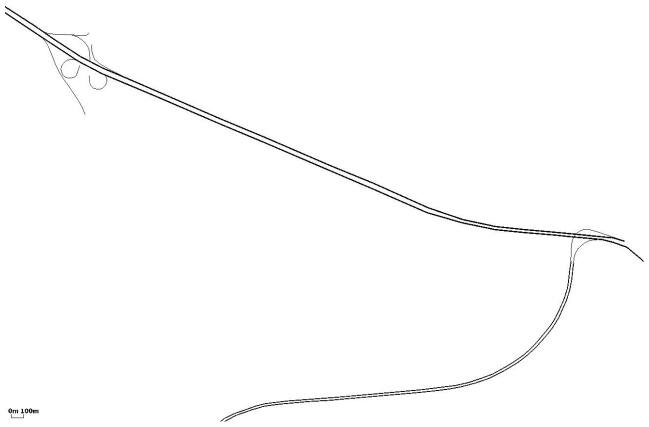


Figure 4.3: Road network used in simulation

4.1.4 OMNeT++

OMNeT++ is an open source simulation framework based on C++ and free for academic use. It comes with a graphical IDE and supports graphical simulation that allows user to view flow of packet among network applications. This framework has extensions that allow simulation of specific communication network, and there is now a project called MiXiM that combines several mobile and wireless simulation extensions together. In particular, we used the extension developed by Christoph Sommer, a project with the name Veins, which stands for Vehicles

in Network Simulation. This extension provides a skeleton for simulating vehicle wireless network by outsourcing the simulation of vehicle motion to SUMO and interacts with it via TraCI interface. By using this extension user may program vehicle on-board unit as desired without worrying about how to simulate and retrieve vehicle data from SUMO simulation. The version used for this evaluation is OMNeT++ version 4.0 and Veins tracidemo branch released on March 13th, 2010.

4.2 TraCIBroadCast

In addition to the above stated tools and simulators, we have written an application called TraCIBroadCast to be executed by the on-board unit of all vehicles in the simulation to collect data required to perform evaluation. This application would randomize the first position broadcast when the vehicle enters simulation. This is very important because if all vehicles make position update at exactly same time then no vehicles would receive any update due to packet collision. Also, this application would generate a random packet size larger than 100 using the positive half of Gaussian distribution, with σ equals to half of the default wireless LAN MAC RTS threshold to represent other periodically generated messages sent together with position update. Last but not least, this application would record the TraCI updates, i.e. position update of the vehicle running the application, and the packets successfully received by this vehicle to text files that will be transformed to matrices and feed into Matlab.

Figure 4.4 gives an overview of how the simulation of vehicle motion and wireless transmission are simulated using SUMO, OMNeT++, and the application programmed by us.

Within the Veins extension, TraCIBroadCast sends/receives packets using messages sent to and received from the scheduler within OMNeT++ like any other timed events. The messages are distinguished by checking whether it is a message generated by the application itself. This type of message includes timers and notifications from other part of the application. If the message is not generated by the application itself then it is considered as packet sent from another application, in our case, another vehicle running the TraCIBroadCast application. The update from TraCI regarding the GPS vehicle position is processed as change notification, which is sent to notification board within the Veins extension instead

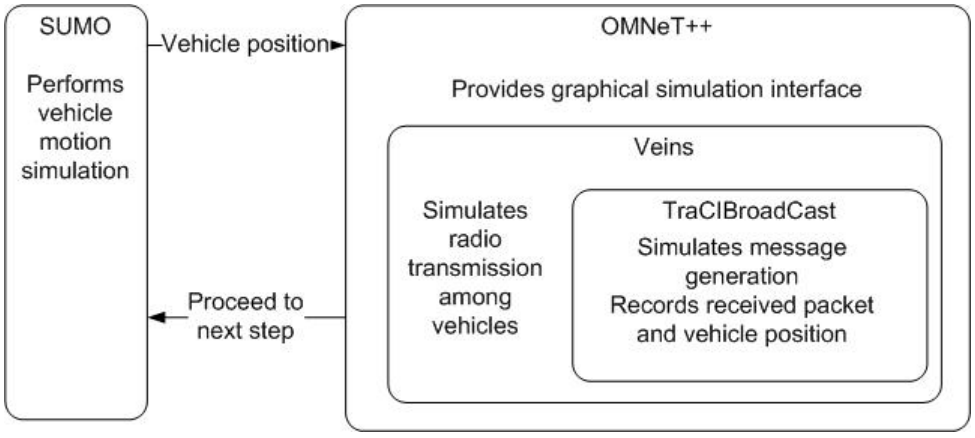


Figure 4.4: Overview of vehicle motion and wireless transmission simulation

of scheduler of OMNeT++. Figure 4.5 shows the exchange of messages among applications and components in an illustration.

The output generated by TraCIBroadCast are saved with file names "tracelog", which we will refer as trace log file, and vehicle ID + ".log", which we will refer as packet traffic files. The trace log file includes the position, velocity, and direction of each vehicle sampled at every TraCI update interval. The packet traffic files records the packet received at each vehicle, recording the sender, sending time, and receiving time. For example the file "host[0].log" logs all packets sent to the vehicle with ID "host[0]" during simulation without collision or any other errors. The formats of these files are shown in Table 4.1 and 4.2. These outputs are converted to matrices to be read by Matlab. In upcoming section we shall discuss details of our simulation settings.

4.3 Road Traffic Simulation Settings

In this section we shall discuss the settings for the simulation of road traffic simulation.

We defined 11 traffic flows for evaluation, all emitting 10 vehicles one after another with randomized intervals within the first 1000 simulation steps. Since we are using 0.1 second per simulation step, this means for each flow the 10

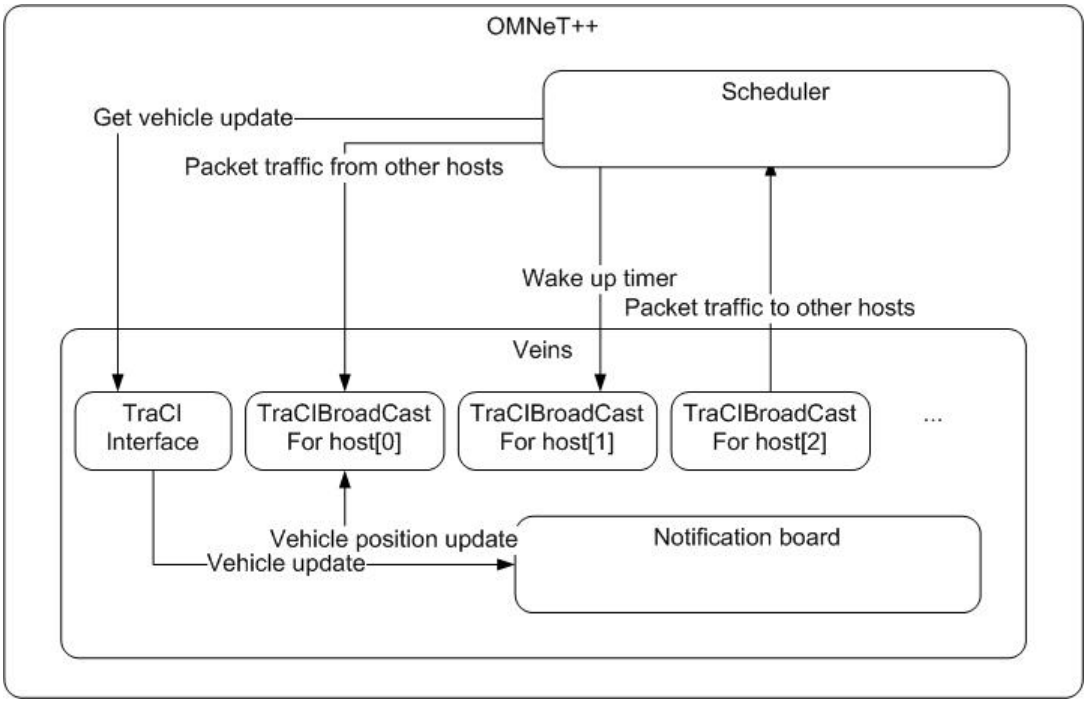


Figure 4.5: Exchange of messages within OMNeT++

vehicles are injected randomly in the first 100 seconds of simulation. The reason for limiting to 10 vehicles is to reduce the amount of data, since there will be more radio transmission to be recorded if the number of vehicles with in transmission range increase. The reason for limiting their departure time to the first 100 seconds in simulation is to make sure the vehicles within same flow are relatively grouped together, so that we can expect more information updates from vehicles running in same direction and increase chance of collision of packets and busy channel event when we perform the simulation. The positions of sources and destinations on the SUMO road network are indicated in Figure 4.6. The combinations of vehicle flows are listed in Table 4.3. The vehicles are assigned name "host[#]", where # is the order of injection into the simulation and has range from 0 to 109.

As stated in the brief introduction to SUMO, the version we are using performs simulation based on one second per simulation step. Since we are assuming that it is possible to have more frequent update of position and velocity data, we needed to reduce the amount of acceleration/deceleration to one hundredth, highest speed reachable by the vehicle and similarly speed limit of highway to one tenth in

	Trace log file	
Format	Vehicle ID	string
	Time stamp	float
	x	10 digit float
	y	10 digit float
	Speed	10 digit float
	Direction in radian	10 digit float
	Separator	:
Example	1 host[0]:1:x:1604.189941:y:7452.499512:v:0.5:r:0.9776797295 2 host[0]:1.1:x:1604.592773:y:7451.901611:v:0.7209999561:r:0.9776797295 3 host[0]:1.2:x:1605.109863:y:7451.134766:v:0.924761951:r:0.9776797295 4 host[0]:1.3:x:1605.731934:y:7450.212158:v:1.112630486:r:0.9776797295 5 host[0]:1.4:x:1606.450684:y:7449.145996:v:1.28584528:r:0.9776797295 6 host[0]:1.5:x:1607.258301:y:7447.947266:v:1.445549369:r:0.9776797295 7 host[0]:1.6:x:1608.148926:y:7446.626709:v:1.592796564:r:0.9776797295 8 host[0]:1.7:x:1609.114746:y:7445.193359:v:1.728558421:r:0.9776797295 9 host[0]:1.8:x:1610.150879:y:7443.65625:v:1.853730917:r:0.9776797295 10 host[0]:1.9:x:1611.251465:y:7442.023438:v:1.969139934:r:0.9776797295 11 host[0]:2:x:1612.411621:y:7440.302246:v:2.07554698:r:0.9776797295 12 host[1]:2:x:25:y:50.109375:v:0.5:r:-1.170312524 13 host[2]:2:x:7429.970215:y:4248.47998:v:0.5:r:2.854291916 14 host[0]:2.1:x:1613.626953:y:7438.5:v:2.173654318:r:0.9776797295 15 host[1]:2.1:x:25.28100586:y:50.7734375:v:0.7209999561:r:-1.170312524 16 host[2]:2.1:x:7429.278809:y:4248.275879:v:0.7209999561:r:2.854291916	

Table 4.1: Format of trace log file

order to simulate position and velocity update of 0.1 second. This means when we perform position estimation using these data we need to multiply the speed by ten times to get the actual speed. Table 4.4 shows the acceleration, deceleration, highest speed and speed limit used.

For the wireless channel setting, we used the values in the sample simulation within tracidemo branch of Veins. We changed the TraCI update interval setting, the interval to get next step of SUMO simulation, in the sample to 0.1 second. This means after OMNeT++ finish simulating the wireless events within 0.1 second it will use the TraCI interface to tell SUMO to proceed to next simulation step. The details of wireless settings are listed in Table 4.5.

The vehicle on-board unit is programmed to run TraCIBroadCast that accepts three options. These options are broadcast interval, early broadcast rate, and packet error rate. They are set to 0.5 second, 1%, and 10% for high packet loss, 1% for low packet loss respectively. This means that normally the position update would be broadcasted 0.5 second after last broadcast, but with 1% proba-

	Packet traffic files	
Format	Sender ID	string
	Reception time stamp	10 digit float
	Transmission time stamp	10 digit float
	Separator	:
Example	1 host[29]:r:138.171883632179:t:138.171566745253 2 host[29]:r:138.671886813258:t:138.671566745253 3 host[29]:r:139.171885813034:t:139.171566745253 4 host[29]:r:139.671888177522:t:139.671566745253 5 host[35]:r:140.044662742453:t:140.044344937452 6 host[29]:r:140.171886272484:t:140.171566745253 7 host[35]:r:140.544662923692:t:140.544344937452 8 host[29]:r:140.671888925854:t:140.671566745253 9 host[35]:r:141.044663469148:t:141.044344937452 10 host[29]:r:141.171887825713:t:141.171566745253 11 host[35]:r:141.544665015799:t:141.544344937452 12 host[41]:r:141.606492231268:t:141.606173443289 13 host[29]:r:141.671885530111:t:141.671566745253 14 host[35]:r:142.04466383823:t:142.044344937452 15 host[41]:r:142.10649559413:t:142.106173443289 16 host[29]:r:142.171884888382:t:142.171566745253	

Table 4.2: Format of packet traffic files

Flow No.	1	2	3	4	5	6	7	8	9	10	11
Source	B	B	B	G	G	I	I	C	D	I	L
Destination	F	H	J	H	J	E	A	A	A	K	J

Table 4.3: Flows of vehicles

bility the next broadcast would happen before the 0.5 second timer expires. This is to represent other non-periodic messages such as brake warning, etc that might also transmit the position and velocity of vehicle with it. Once the early broadcast happened, the following broadcast would come 0.5 second after it. The packet error rate is to simulate packet collision with transmissions that are generated by vehicles and other sources not covered in this simulation and other possible communication error due to motion of the vehicle.

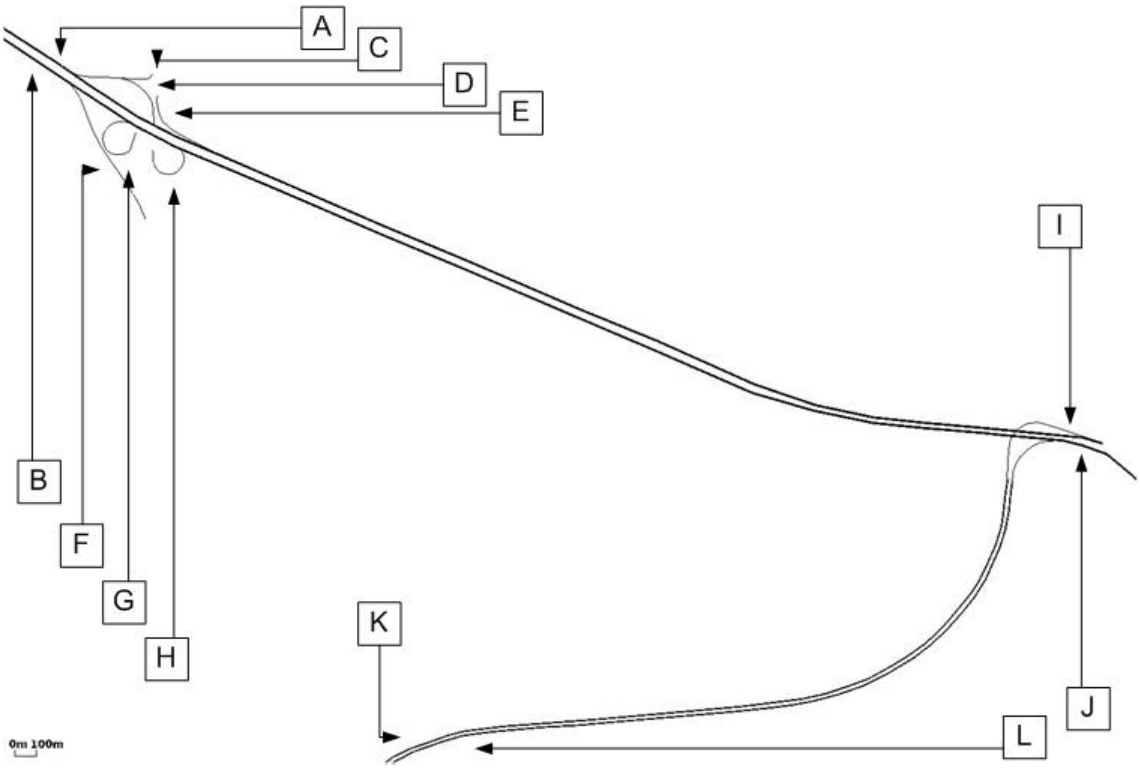


Figure 4.6: Sources and destinations

4.4 Algorithms For More Realistic Conditions

Since in this evaluation we are using vehicle position on a real map, we are facing some new problems, namely the problem of prediction in 2D. In order to maintain the simplicity of our filter, we decide to not include any coordinate conversion system regarding the position and simply use x-y coordinate as supplied by the SUMO simulation. In order to use common coordinate system for estimation so that we can add independent noises to x and y direction, we need to divide speed and acceleration into x direction and y direction as well and treat them independently. The reason for not using polar coordinate for speed is that the error correction will need to have a function to correct error in the polar direction, which may need a more complex motion model than we have assumed in the beginning. Although there is benefit in using polar representation for speed, we prefer to maintain the simplicity of our filter. Since SUMO provides speed of the

Acceleration	Deceleration	Max Speed	Speed Limit
0.026 m/s^2	0.045 m/s^2	3.33 m/s	3.33 m/s

Table 4.4: SUMO vehicle status

Transmitter Power	Radio Bitrate	Thermal Noise
2mW	11Mbps	-110dBm

Path Loss α	SNIR Threshold	Radio Sensitivity
1.9	3dB	-85mW

Table 4.5: Wireless channel settings

vehicle with the direction it is facing, the division will be done using sine and cosine.

$$v_x = v_o \cos(angle\_in\_rad)$$
$$v_y = v_o \sin(angle\_in\_rad)$$

In previous description of our algorithms in 3.1, the update is assumed to be created at the time the packet is transmitted, so we can assume there is no time difference for the update information when we make error correction. However, as stated in previous section, the position and motion information are only available at every 0.1 second, not real time as assumed before. Therefore, we will have to treat the update information as constant for the 0.1 second until new update is available. This indicates that when we make error correction, we need to take into account the delay in the information. That is, we need to make prediction at the time the update information has been generated and make error correction at that point, and then make another prediction up to current time when we receive update. In other words, we are forced to make prediction of about 1 to 2 steps even when we have update information.

In addition, we have discussed cases where there might be delays and losses of packets during estimation, but we have not yet covered the case where the estimation would be terminated due to long period of inactivity in the position update. That is, we need to alter our algorithms so that they will stop estimation once certain amount of time had passed since last position update has been received. The following is a pseudo code for our modified $\alpha - \beta - \delta$ tracker, where Δt rep-

resents the interval between estimation, δt represents the smallest interval on the *real time* discrete clock, and threshold represents the time limit for inactivity in the position update:

```
Estimation_timer = Δt;
Last_update_time = 1000;
Received_update = false;
for (t = 0;true;t = t + δt) {
  Last_update_time = Last_update_time + δt;
  Estimation_timer = Estimation_timer - δt;
  If (New_update) {
    Current_update = update;
    Received_update = true;
  }
  If ((Estimation_timer ≤ 0 && Last_update_time ≤ threshold)
        || Received_update) {
    If (Received_update) {
      Generate α − β − δ using Last_update_time;
      Perform prediction error correction using Current_update up to the
                                                  time update was generated;
      Perform prediction up to current time;
      Received_update = false;
      Last_update_time = 0;
    } Else {
      Assign predicted value to estimate value;
    }
    Estimation_timer = Δt;
  } Else if (Last_update_time > threshold) {
    Last_update_time = 1000;
  }
}
```

For evaluation of our algorithms, we use 0.5 second for estimation timer and 0.1 second as smallest interval on the *real time* discrete clock. The threshold for inactivity in the position update is set to 20 discrete intervals, that is, 2 seconds in real time.

4.5 Evaluation Process

The output data given example in Table 4.2 are converted to matrices so that for each of the vehicles there is a matrix that gives the time stamps of the packets it received and all the position data are converted to 4 position matrices representing the x coordinates, y coordinates, speeds, and directions in radian. The packet traffic matrices are formulated such that every two rows represent the packets from one of the vehicles. The first row represents the receiving time and the second row represents the transmission time. Since we need to match the length of each row, we group the valid time stamps to the columns to the left and pad all rows with -1 except the longest rows to match the length. By doing so, we can quickly count the number of packets and skip the empty period where there is no packet traffic using Matlab. The position matrices are formatted so that each row represents one vehicle and the column indices imply the simulation step. The locations where the vehicles are not injected yet or exited the simulation are filled with -1. This way, we can find the position of vehicle at specific time by simply specifying the coordinate.

After loading these matrices into Matlab we add noises to the data and apply our tracker to the degraded data. The tracker is assumed to be on all the vehicles so we perform tracking on all vehicles against any vehicles that have sent at least two packets to the vehicle. That is, from the packet matrix of one particular vehicle X we make a list of vehicles that have successfully transmitted at least two packets to vehicle X, and we used this packet availability information to try to track the vehicles on the list. The detailed procedure is explained in the following paragraph.

First of all, we multiply the transmission time and receiving time with factor of 10 to convert it to same unit (0.1 second) as the index of the columns for the position matrices. Then, we use ceiling function to round up the receiving time of the packets to make it equal to the discrete time when we are making prediction and correction, and use floor function to round down the transmission time of the packets to make it equals to the discrete time the position has been updated. When the packet transmission time is determined, a new set of data are created by copying the x and y coordinates and generating speed in x and y direction using equations in 4.4. Gaussian noises with specified sigma values are created independently and added to the new set of data at indices where the packet has been transmitted. This set of data, the packet transmission and reception time, and the sigma values for the noises are passed to our modified algorithms to generate

tracking result. When the tracking has been done, the position and velocity errors are measured by taking average and variance in both x and y direction.

This procedure is repeated for all vehicles that have sent at least two packets to vehicle X, and the errors are plotted in a graph for this particular vehicle. Since we have a total of 110 vehicles we will generate 110 such graphs every time we make a complete tracking experiment.

On side note there is a problem with the data generated by SUMO. That is, there exists an error between the simulated positions of the vehicle and the theoretical positions.

$$d_x(n+1) \neq \begin{cases} d_x(n) + v_x(n)\Delta t \\ d_x(n) + v_x(n)\Delta t + \frac{\Delta t^2}{2}(v_x(n+1) - v_x(n)) \end{cases}$$
$$d_y(n+1) \neq \begin{cases} d_y(n) + v_y(n)\Delta t \\ d_y(n) + v_y(n)\Delta t + \frac{\Delta t^2}{2}(v_y(n+1) - v_y(n)) \end{cases}$$

We have observed that this error is accumulative, and in some cases the displacement between two time steps was even larger than the sum $v_y(n+1)\Delta t + \frac{\Delta t^2}{2}(v_y(n+1) - v_y(n))$ when the vehicle is still accelerating. This is physically impossible and against the law of motion. This error can be the result of simulating the GPS error or can be error measuring the speed (though not very likely), but since we need a set of data clean of errors for comparison of performance of our filter algorithms, we regenerate the set of position data by taking the first valid position as error free initial condition and overwrite all the subsequent position data using constant acceleration within each simulation step.

4.6 Evaluation Results

Two tracking experiments with ideal conditions are performed to verify the adjusted algorithms. Both have the settings that there is no packet loss and infinite range of transmission, and one of the simulation has 1 meter for sigma of Gaussian position error , 0.06 m/s for sigma of Gaussian velocity error and the other has 5 meters for sigma of Gaussian position error, 0.3 m/s for sigma of Gaussian velocity error. All errors are applied for both the x axis and y axis. The average overall errors of the adjusted $\alpha - \beta - \delta$ tracker for the simulation with smaller

noise are 0.1198 meters for the position and 0.1334 m/s for the velocity, and for the larger noise are 0.3724 meters for the position and 0.1655 m/s for the velocity. The errors of the $\alpha-\beta-\gamma-\delta$ tracker are of similar values but somewhat larger.

Four additional simulations are performed using the data generated with 10% packet loss rate and 1% packet loss rate. The resulting errors are listed in Table 4.6. Overall, the average position errors are about half of the sigma values. The velocity errors tend to be larger than the sigma of the velocity noises because the absolute values of the errors are not very large so the corrections tend to be processed by an over-damped filter. Note that the large noise setting has same sigma values as the experiment done in chapter 3 and the resulting position error is almost the same.

	$\alpha-\beta-\delta$ tracker	$\alpha-\beta-\gamma-\delta$ tracker
Avg pos error	0.5135m	0.5328m
Avg vel error	0.1892m/s	0.2693m/s

(a) 10% packet loss rate with smaller noise

	$\alpha-\beta-\delta$ tracker	$\alpha-\beta-\gamma-\delta$ tracker
Avg pos error	0.5029m	0.5203m
Avg vel error	0.1634m/s	0.2323m/s

(b) 1% packet loss rate with smaller noise

	$\alpha-\beta-\delta$ tracker	$\alpha-\beta-\gamma-\delta$ tracker
Avg pos error	2.3832m	2.3962m
Avg vel error	0.2918m/s	0.3599m/s

(c) 10% packet loss rate with larger noise

	$\alpha-\beta-\delta$ tracker	$\alpha-\beta-\gamma-\delta$ tracker
Avg pos error	2.3862m	2.3992m
Avg vel error	0.2671m/s	0.3261m/s

(d) 1% packet loss rate with larger noise

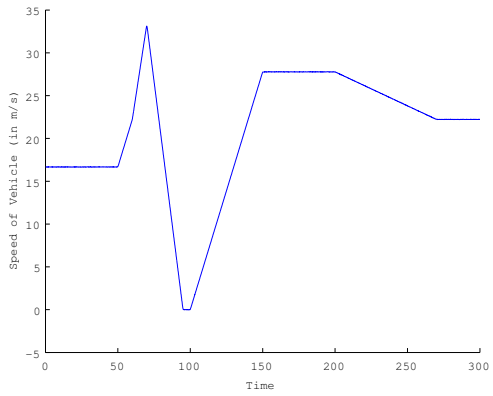
Table 4.6: Estimation errors

Here the average errors of our 4 parameter tracker are larger than our 3 parameter tracker in general. This is because the speed generated by our simulation has shape similar to the cases in experiments we have where the 3 parameter tracker

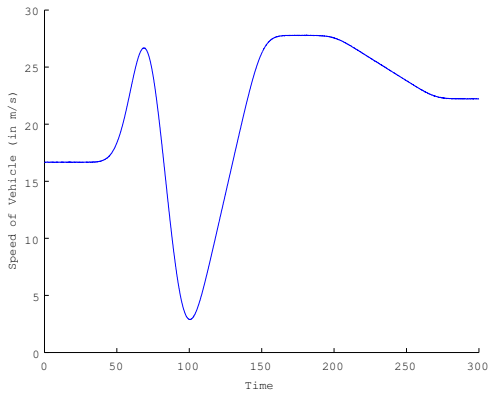
performed better than 4 parameter tracker. Figure 4.7c shows the speed versus time of one of the vehicle in our evaluation. Figure 4.7a and Figure 4.7b are from the experiment against our basic algorithms in chapter 3. Figure 4.7a shows the speed versus time where 3 parameter tracker perform better, and Figure 4.7b shows the speed versus time where 4 parameter tracker perform better. As it can be seen from the enlarged view in Figure 4.7d, the speed of the vehicles in our realistic data tend to change impulsively when compared to Figure 4.7b. Here we have sampled the speed by every 0.1 second so if the speed has changed more smoothly then there should have been a concaved curve between 49.8 second and 50.1 second. The reason for having this type of speed can be attributed to the fact that we do not have the option to simulate the motion of vehicle by 0.1 second time step. As stated in the introduction of SUMO, the vehicle motion simulator we have used, there is only the option to simulate vehicle motion in 1 second time step. In addition, we are simulating in 2D for this evaluation so although we have modified the vehicle acceleration and deceleration ability to 1/100, there is some possibility that the vehicle's turning motion within this 1 second time step can affect the speed when we divide it into x and y components. For this reason we conclude that the 3 parameter tracker should perform better than the 4 parameter tracker in general for this evaluation.

Frequent large gap between two updates compare to the total number of packets received is the dominating reason for error. This will be especially damaging when the vehicles are making turns, for example when they are on the ramps at entrance G or exit H on Figure 4.6. This can be found on the tracking result of host[8] in Figure B.3 at vehicle number 4 on the graph. Here the number of received packet from vehicle number 4 is relatively small and yet there was a relatively large average interval between the updates. It is likely that the vehicle is accelerating when the gap happens, or it is still accelerating when the last update is received, and so our trackers keep adding the acceleration without error correction. On the other hand, if the vehicle is driving on the smoother part, such as the part from entrance I to exit E, the errors can still be bounded in relatively small region. This can be found in the case of host[0] in Figure B.1. Here although vehicle 9 has the largest interval without update, the position errors are still well bounded when compare to the error of vehicle 11. In general, the amount of error in position depends on how many updates are received, and the amount of error in velocity depends on how close the update intervals are. By looking at Figure B.2, the position errors of vehicles 1, 2, 4-5 are closer to the average of overall position errors, and the variances of these vehicles are also small. However, for other

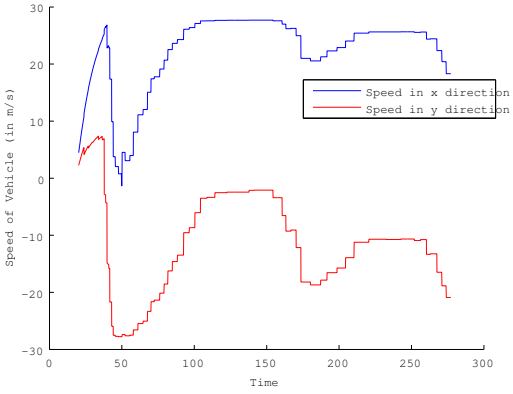
vehicles the number of packets is much smaller so the errors tend to have more variations. Vehicle 6 is an exception because it is making a turning motion when the gap happens, and it is very hard for current algorithms to make prediction of continuous turning motion when there is no update at the moment of turning. On the same graph, the errors of the speeds tend to be larger for those who have large interval between updates, that is, vehicle 2 and 6. Looking at the graph one may notice that the velocity errors for vehicles 7 and larger are very small even when the total number of packets are limited. This can be the result of vehicles running on the opposing direction of the highway and having reached steady speed.



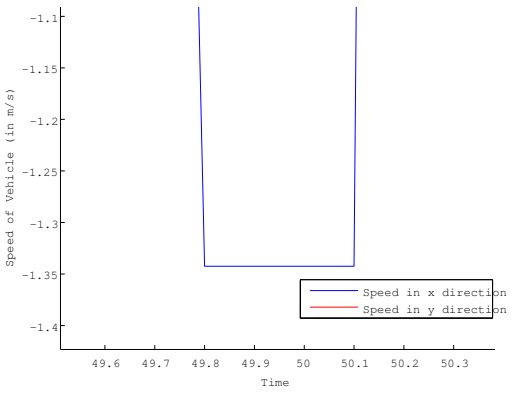
(a) Vehicle speed with low correlation in acceleration change



(b) Vehicle speed with high correlation in acceleration change



(c) Speed of one of the vehicle generated in more realistic setting



(d) Enlarged view of 4.7c

Figure 4.7: Comparison of Vehicle Speed

Chapter 5

Conclusions And Future Work

In this thesis, we have defined a problem in estimating vehicle position with a view point from the wireless network perspective. Two trackers based on the techniques of $\alpha - \beta$ trackers are developed and tested in simulation. From the experimental results we notice that these trackers perform very well considering its simplicity. We have seen that it can tolerate certain amount of packet losses and low information update rate, as well as under specific type of setting one of the two trackers may perform better than the other. One important observation from the experimental results is the importance of synchronization of information update and estimation, so that there will be less estimation without correction and guarantee of fresh information for correction.

In addition, we have used widely used traffic simulators to generate data sets in order to evaluate the performance of our tracker against more realistic condition. From these data we perform experiments using slightly adjusted versions of our trackers. In these adjusted versions we are forced to make predictions of about 0.1 to 0.2 seconds after we make correction, due to the way the position information is sampled and the transmission. In general, we achieve same performance for the position error when the noise settings are equivalent, and showed that if the noises to the observed data can be reduced then our tracker can perform even better. This means if the GPS readings and speed measurements can be processed by the transmitting vehicle then the receiving vehicle need not have a very complex tracker system to predict the future position in condition where packet loss may happen. On the other hand, our adjusted version of tracker converted the polar representation of speed and direction to x and y components, which is shown to

have trouble estimating turning motion, such as the entrance ramp G on Figure 4.6. This is because we only have accelerations in x-y coordinate so when there is no update our estimation result is the vehicle driving more or less in a straight line when the vehicle is actually making a turn. This weakness can be resolved in at least two ways, by converting the coordinate system or by using data fusion method.

5.1 Coordinate Conversion System

Further development for our trackers may include coordinate conversion system to improve performance in 2D or even 3D tracking of the vehicle. To do so we will develop new state-space equations for the motion of vehicle possibly in polar coordinate. This means each vehicle are assumed to provide a unique set of coordinate that the vehicle itself finds best to estimate its own motion, and transmit this coordinate with offsets to convert back to global coordinates. For instance, the vehicles in Chapter 4 used polar coordinates for their speeds and directions, so we expect to have a more precise estimation result when we use a improved tracker that can estimate the polar speeds and accelerations and apply the result to position estimate. We will also need to find new method to complement the errors in different coordinates, but this approach shall give us a tracker that can estimate continuous turning motion more accurately.

5.2 Data Fusion

Another direction of our trackers can go to is the data fusion techniques. This means we will develop methods to add data to the processors of the transmitting vehicle and possibly reduce our tracker to 1D + α only. For example, we can make the transmitting vehicle to recognize which road it is on and which lane it is on, so that it will only need to transmit the ID of the edge it is on the road network with the 1D position relative to the end of the edge, speed along the edge, and the probability it will switch to another lane based on the yaw rate of the vehicle. This approach will allow us to estimate the vehicles as if they are mostly running on a single lane with speed changes only. The lane changing part can be estimated as if the vehicle is driving across two lanes and how much the vehicle has shifted between the two lanes may be estimated as a ratio.

APPENDICES

Appendix A

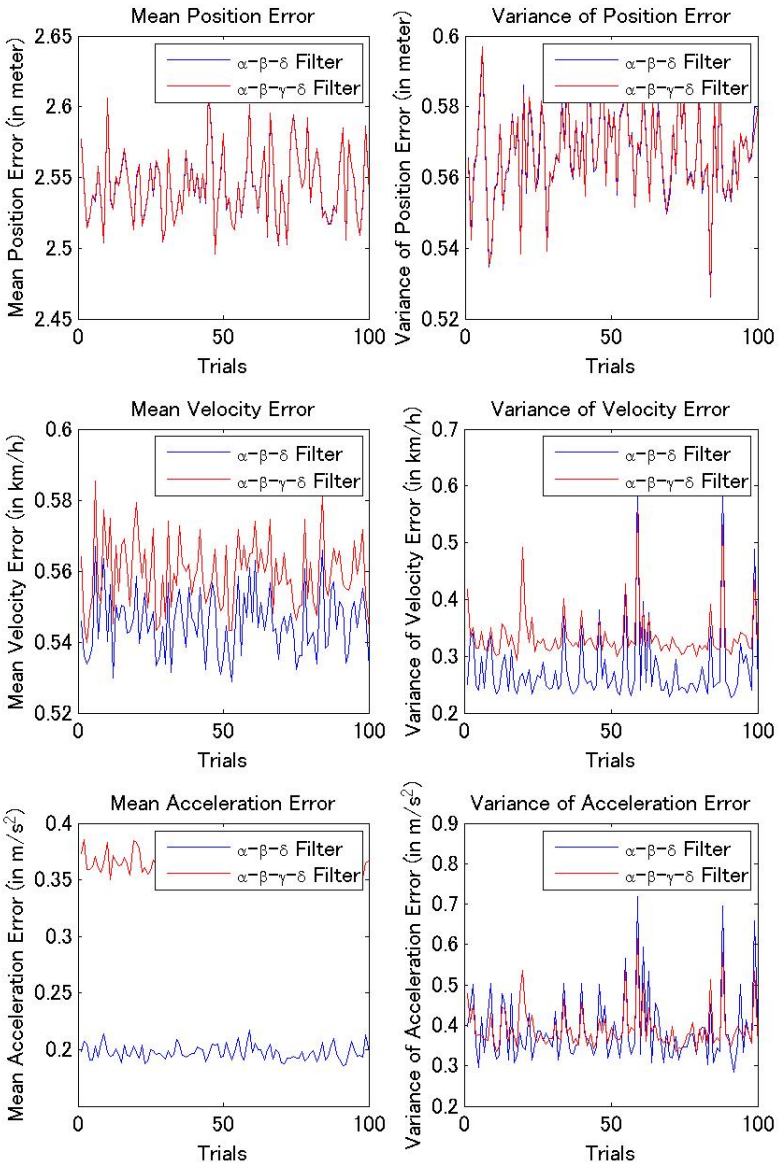


Figure A.1: Simulation result of low correlation, packet loss rate: 0.25, estimation made every 0.1 seconds, information update attempted every 0.1 seconds, no phase shift between estimation and update.

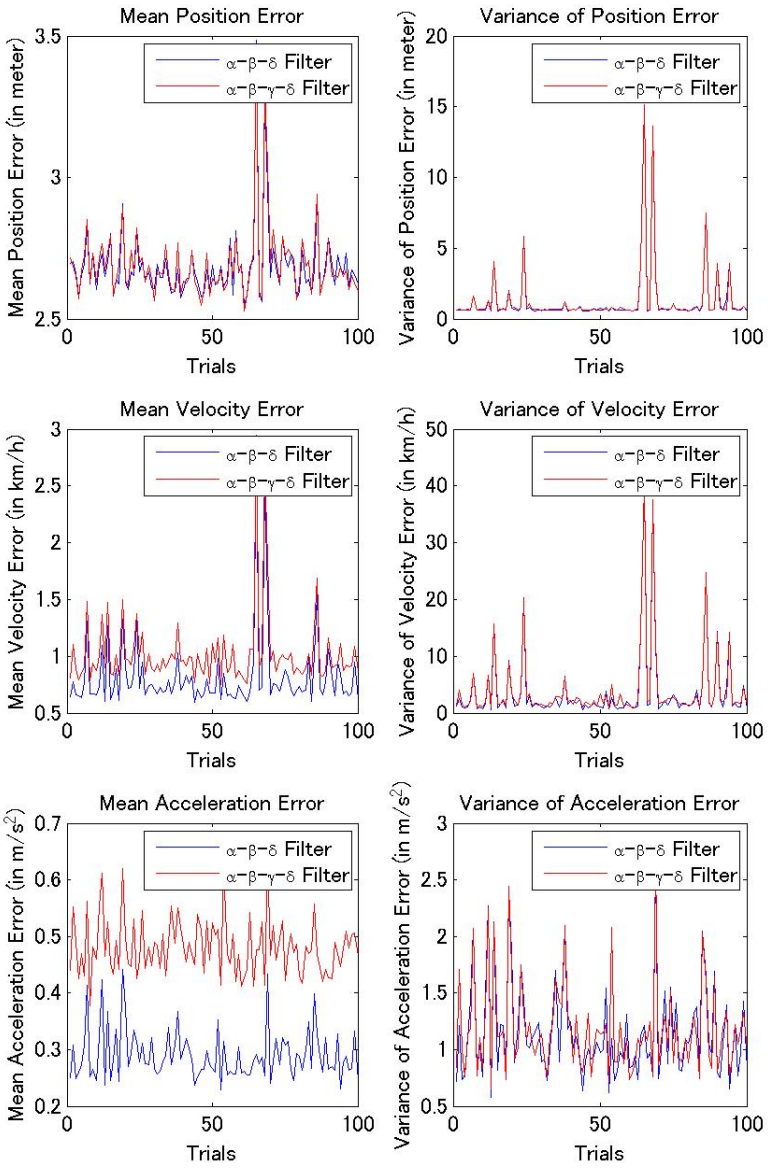


Figure A.2: Simulation result of low correlation, packet loss rate: 0.75, estimation made every 0.1 seconds, information update attempted every 0.1 seconds, no phase shift between estimation and update.

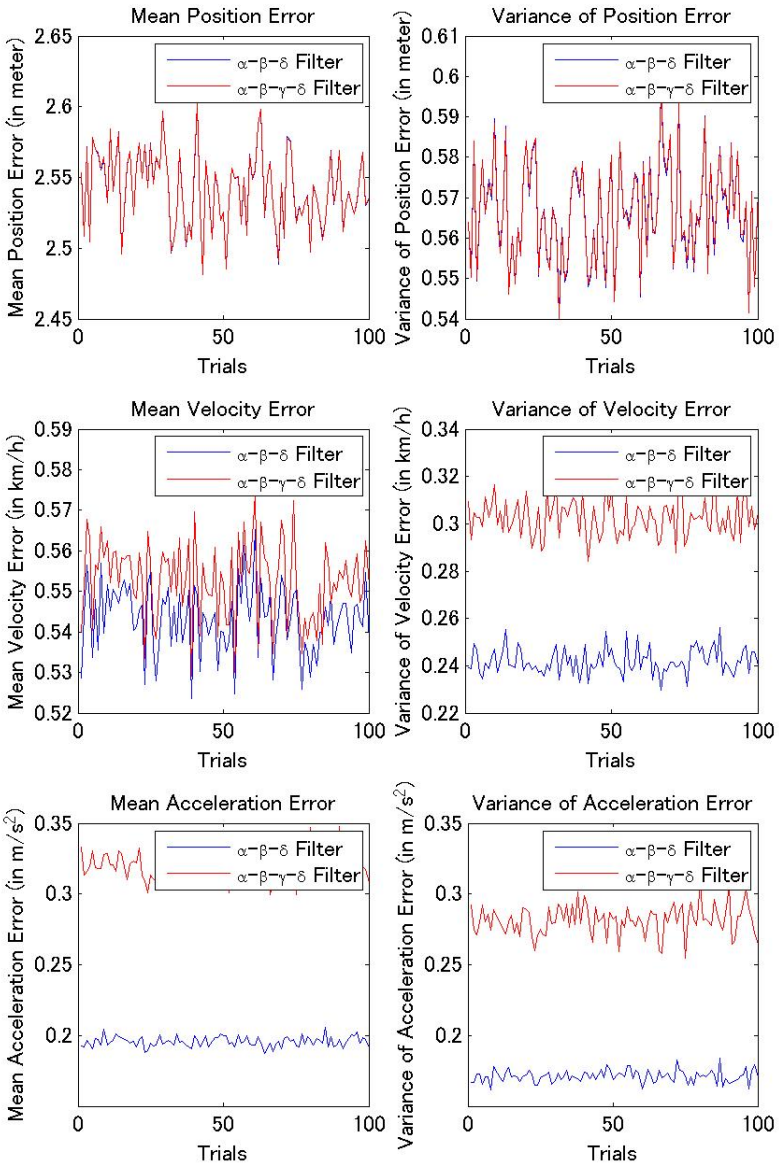


Figure A.3: Simulation result of high correlation, packet loss rate: 0.25, estimation made every 0.1 seconds, information update attempted every 0.1 seconds, no phase shift between estimation and update.

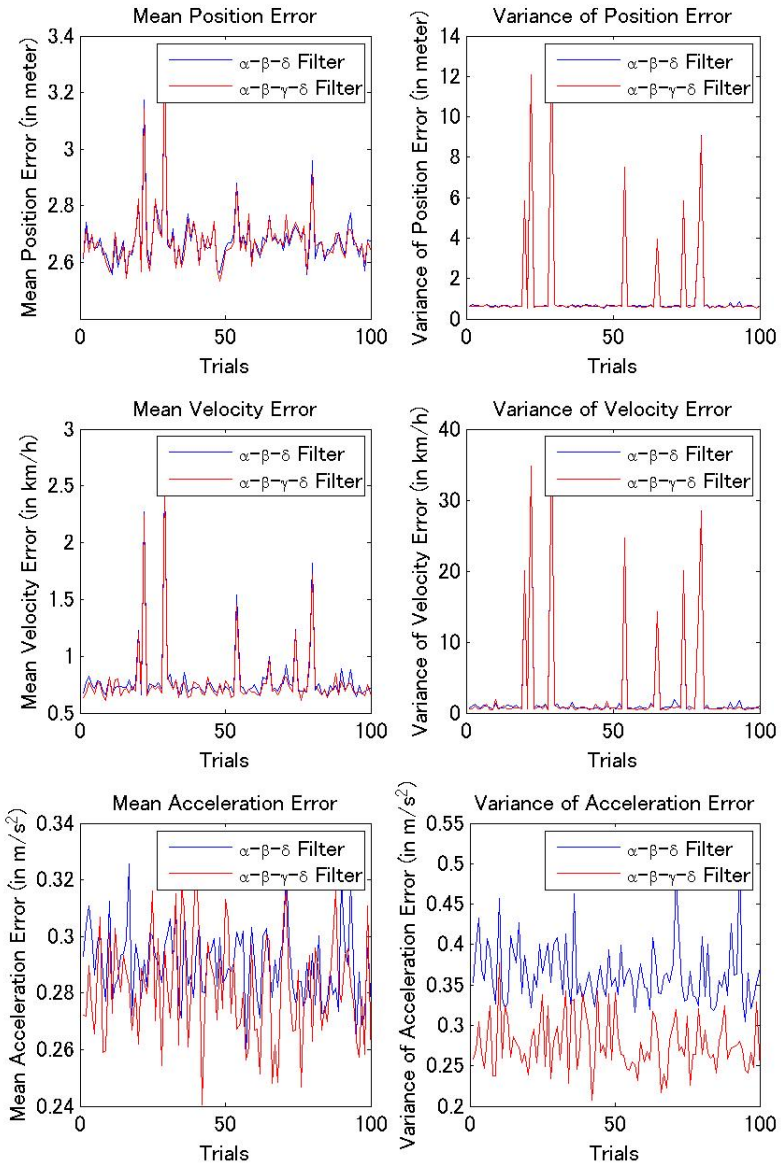


Figure A.4: Simulation result of high correlation, packet loss rate: 0.75, estimation made every 0.1 seconds, information update attempted every 0.1 seconds, no phase shift between estimation and update.

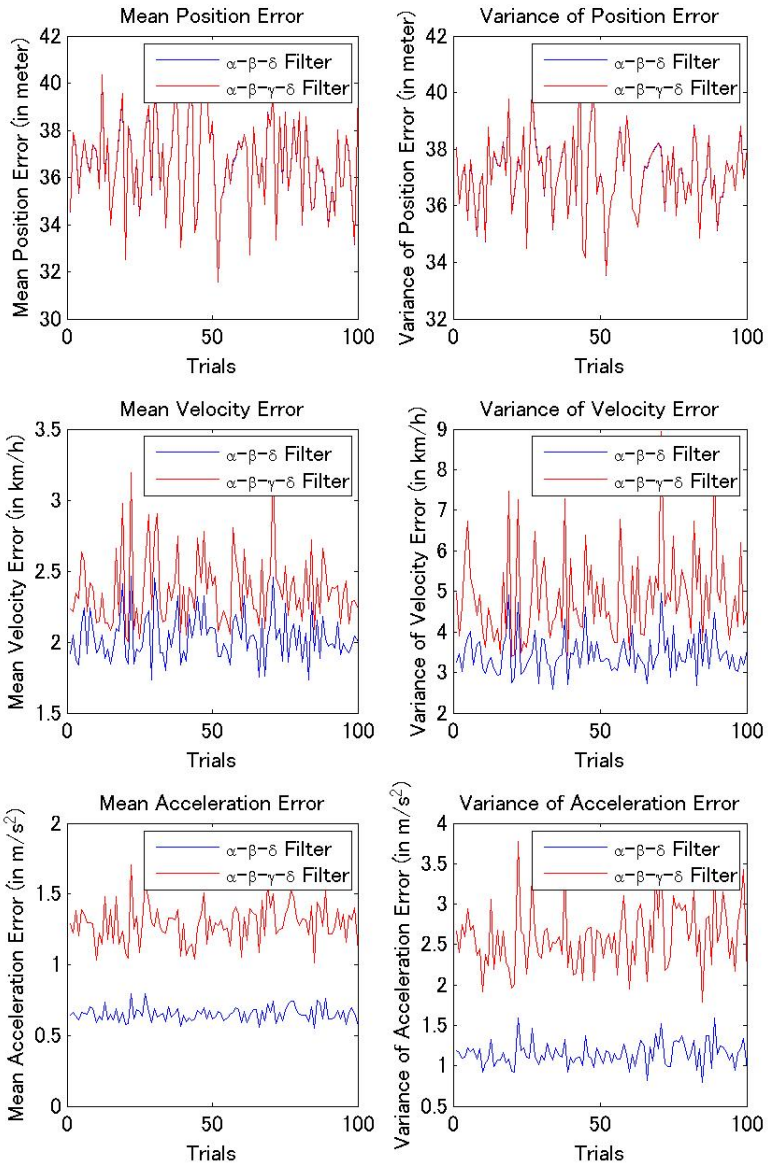


Figure A.5: Simulation result of high correlation, packet loss rate: 0.25, estimation made every 0.5 seconds, information update attempted every 0.1 seconds, no phase shift between estimation and update.

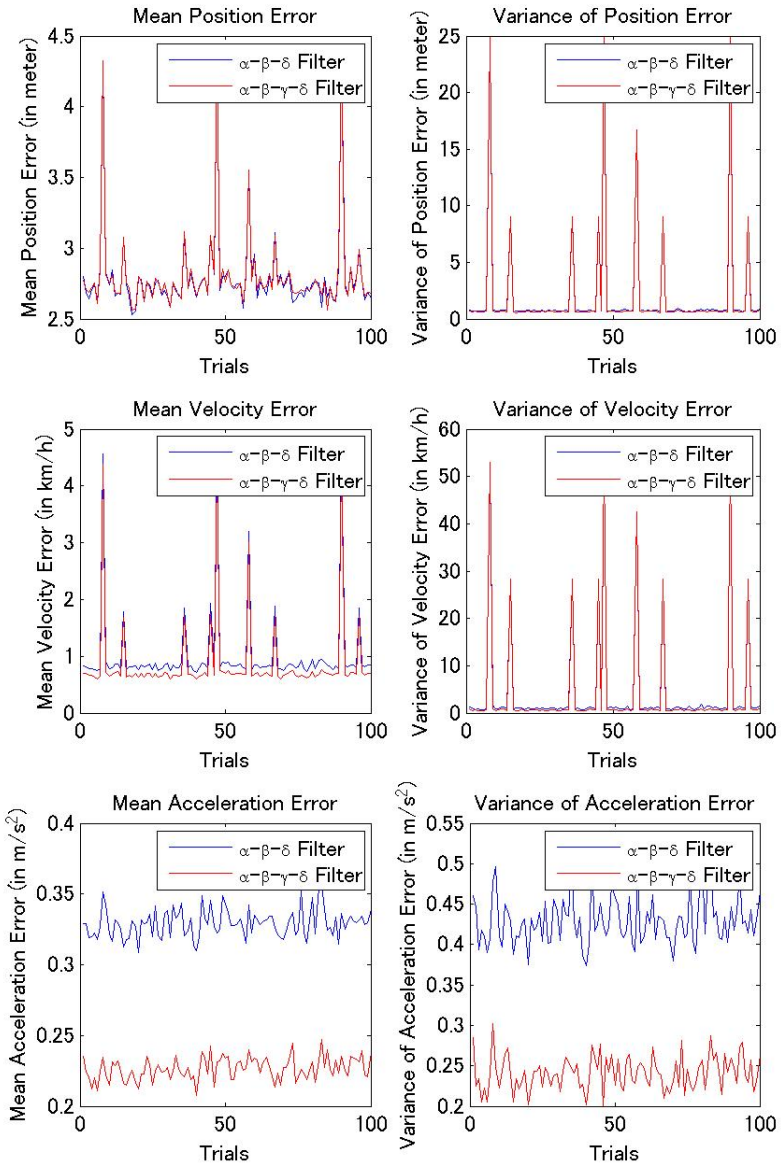


Figure A.6: Simulation result of high correlation, packet loss rate: 0.25, estimation made every 0.1 seconds, information update attempted every 0.5 seconds, no phase shift between estimation and update.

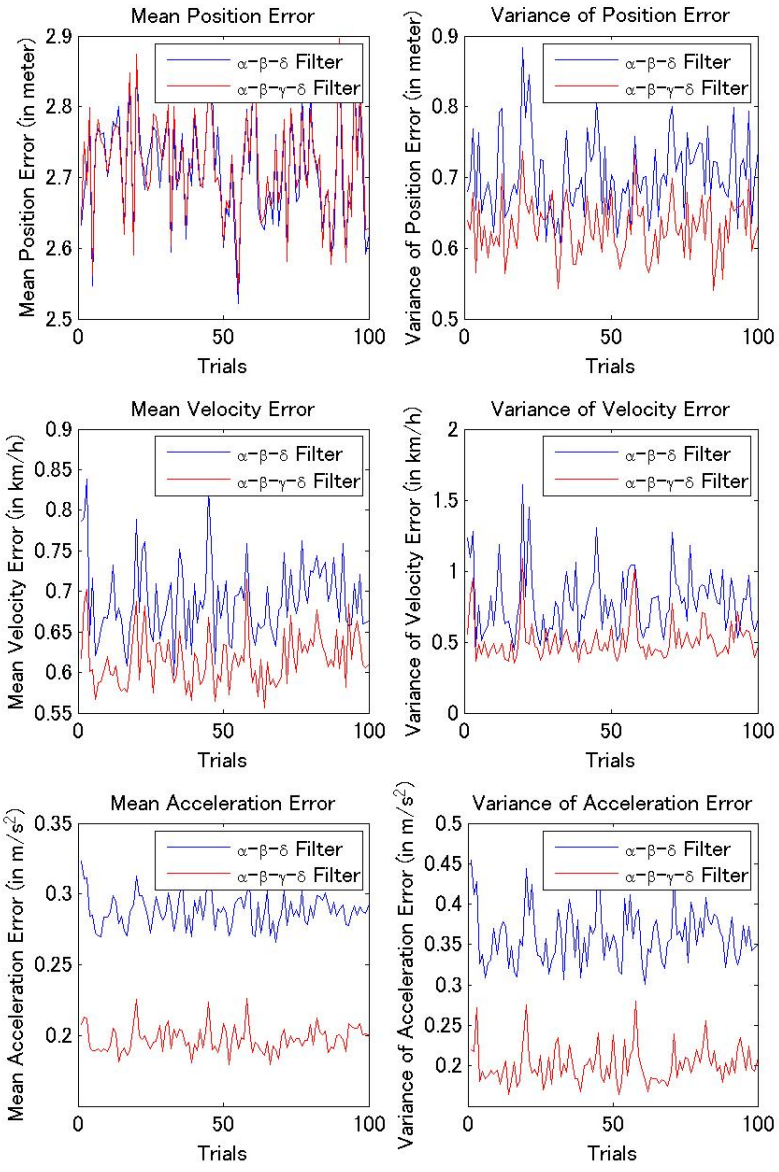


Figure A.7: Simulation result of high correlation, packet loss rate: 0.25, estimation made every 0.5 seconds, information update attempted every 0.5 seconds, no phase shift between estimation and update.

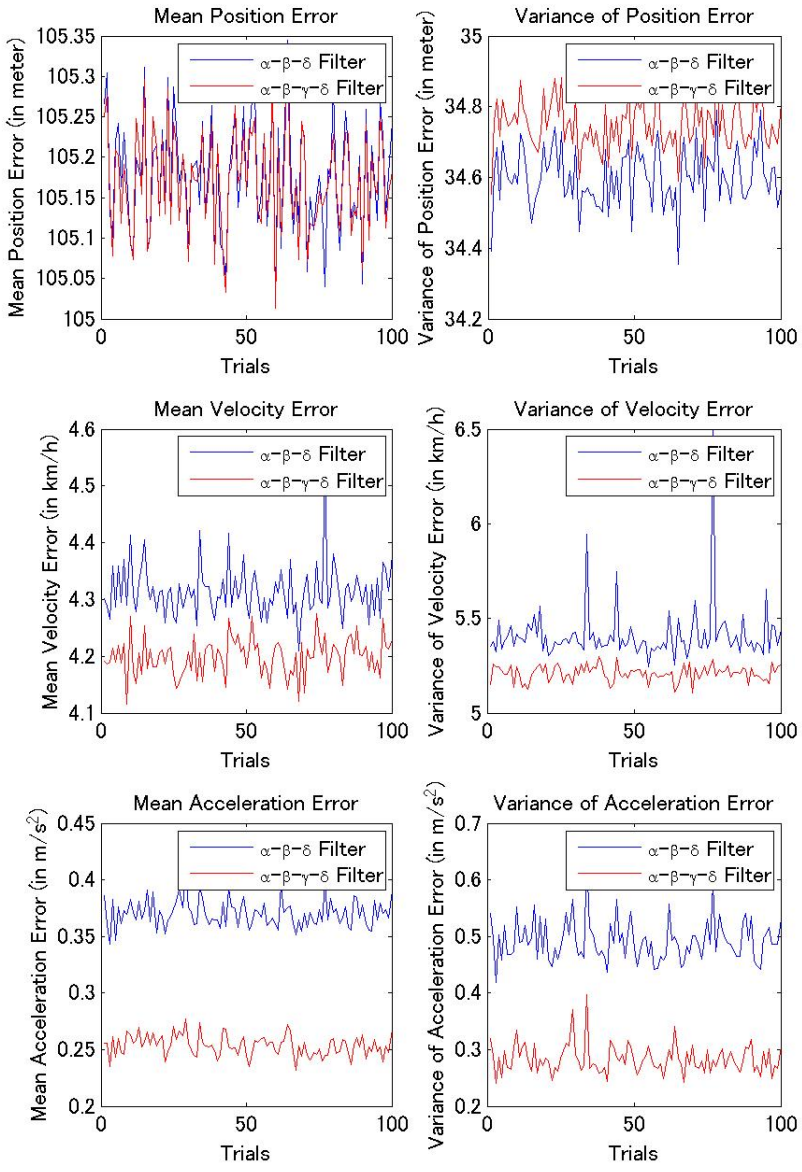


Figure A.8: Simulation result of high correlation, packet loss rate: 0.25, estimation made every 0.5 seconds, information update attempted every 0.5 seconds, estimation leads information update by 0.1 seconds.

Appendix B

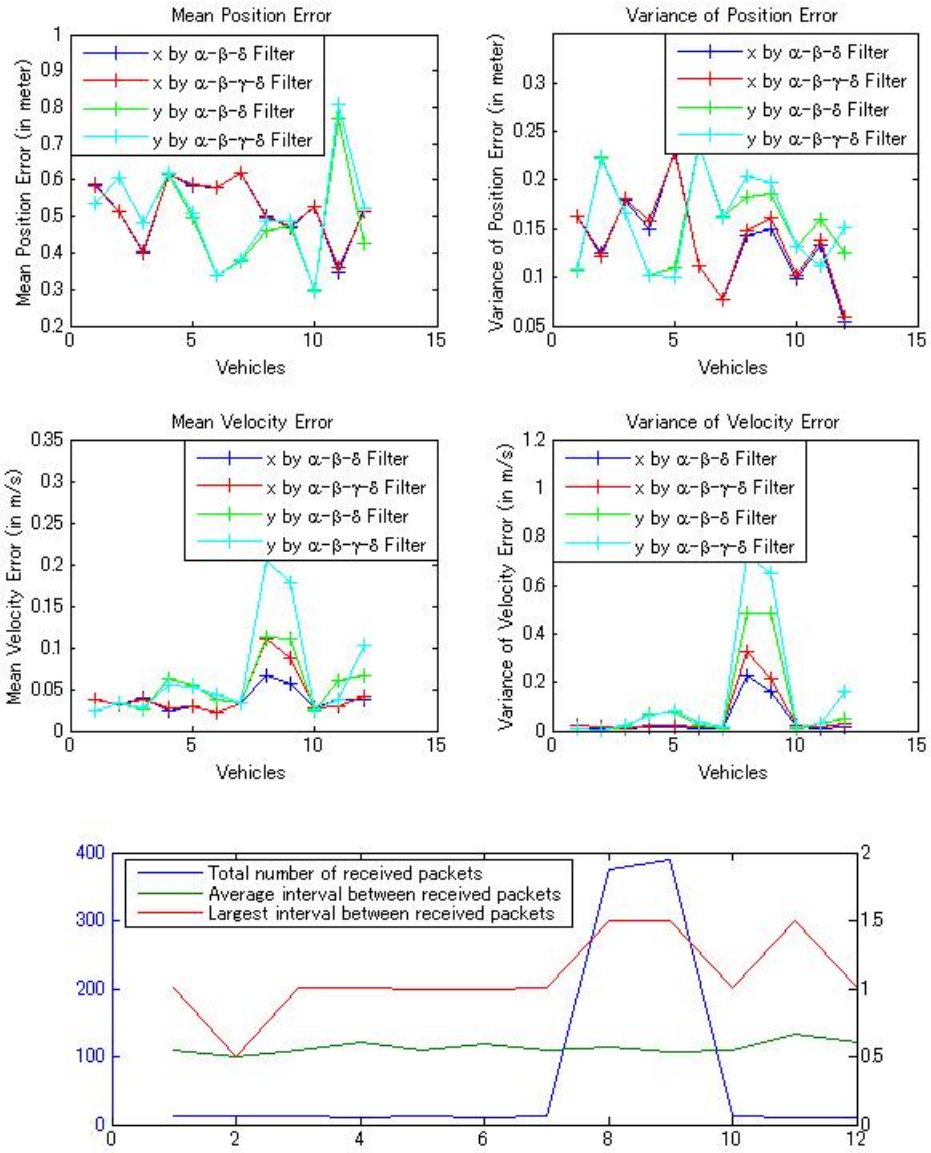


Figure B.1: Host[0] 10% packet loss rate with smaller noise

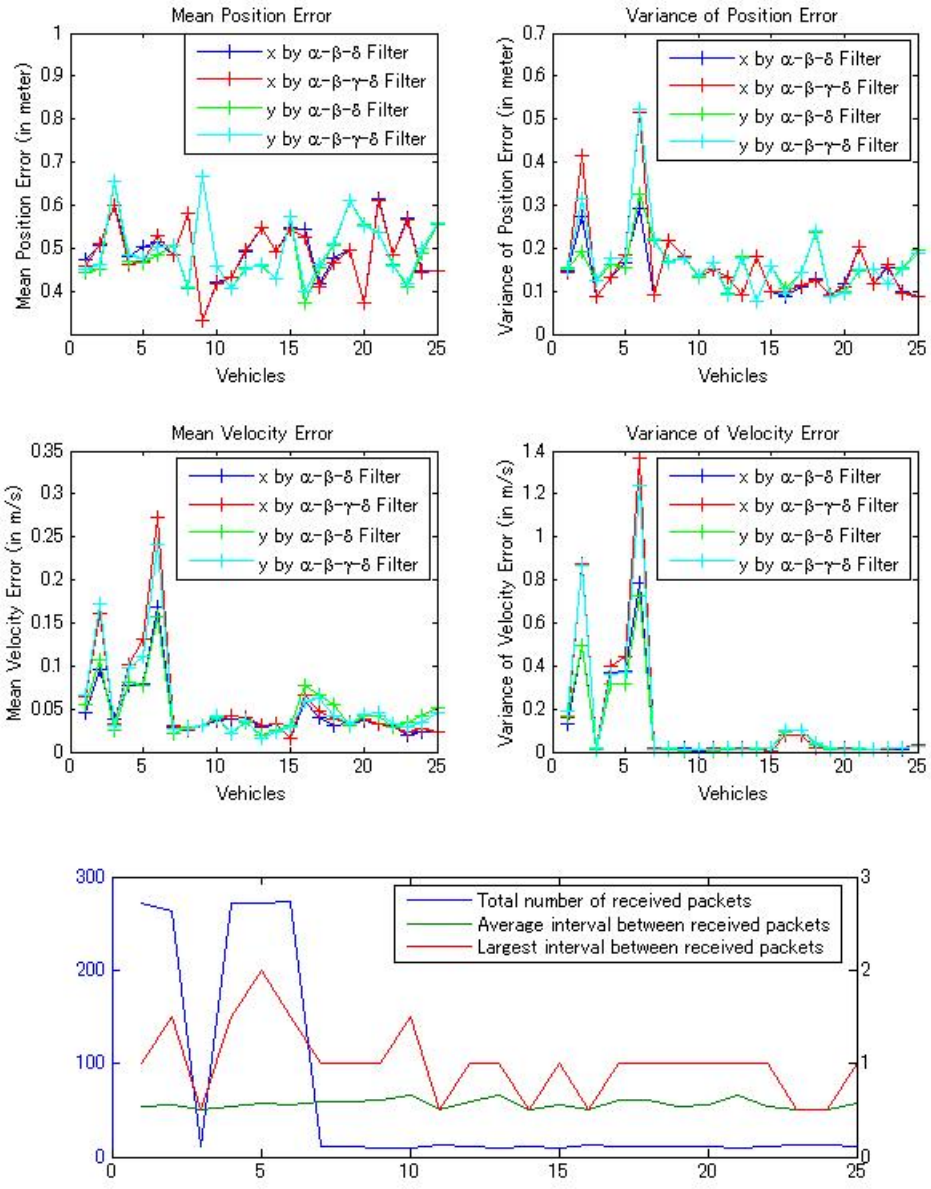


Figure B.2: Host[1] 10% packet loss rate with smaller noise

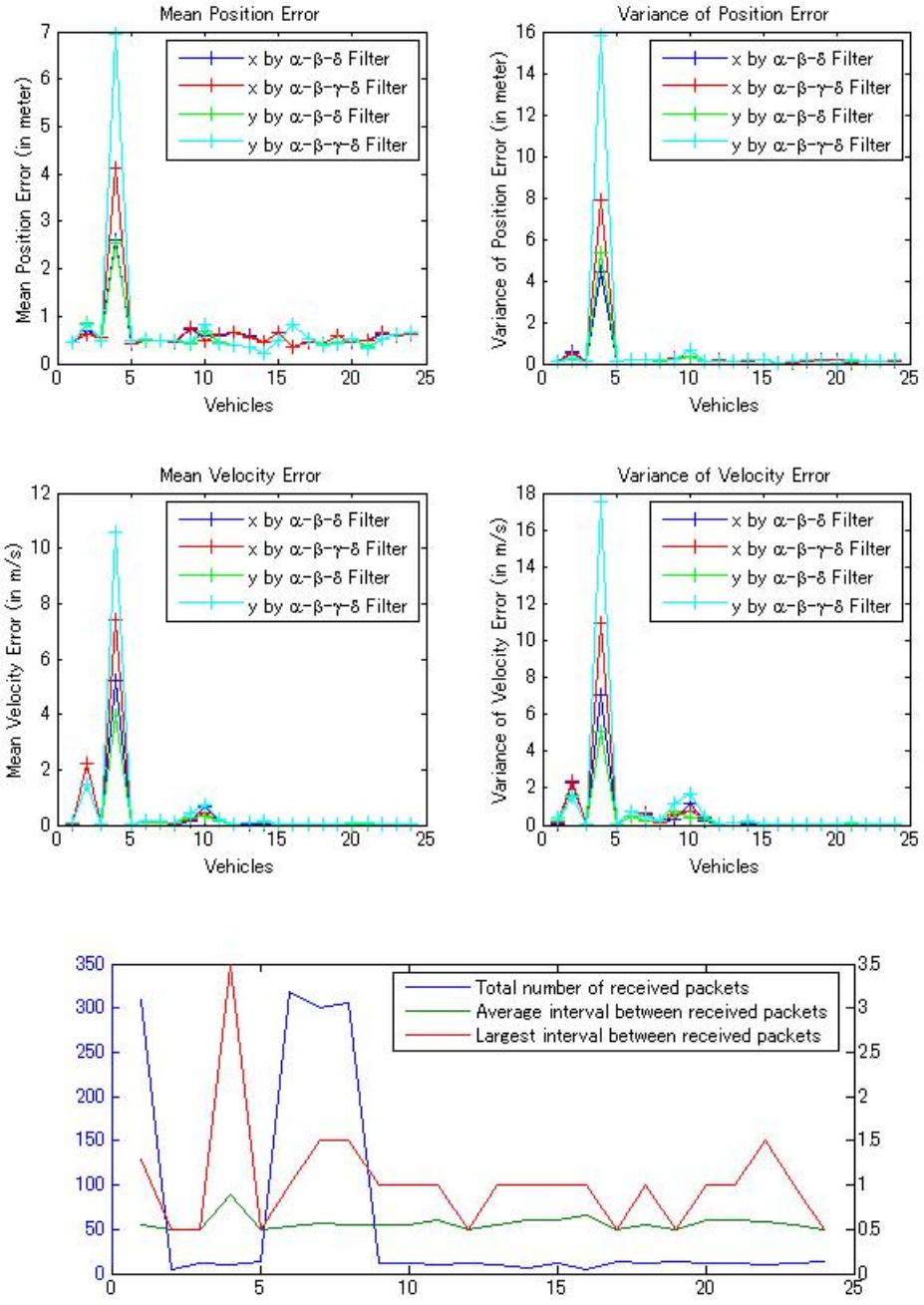


Figure B.3: Host[8] 10% packet loss rate with smaller noise

Bibliography

[1] T.R. Benedict and G.W. Bordner. Synthesis of an optimal set of radar track-while-scan smoothing equations. pages 27–32, 1962.

[2] S. Biswas, R. Tatchikou, and F. Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *Communications Magazine, IEEE*, 44(1):74–82, Jan. 2006.

[3] Azzedine Boukerche, Horacio A.B.F. Oliveira, Eduardo F. Nakamura, and Antonio A.F. Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Computer Communications*, 31(12):2838 – 2849, 2008. Mobility Protocols for ITS/VANET.

[4] Eli Brookner. *Tracking and Kalman Filtering Made Easy*. Wiley-Interscience, April 1998.

[5] Arthur A. Carter. The status of vehicle-to-vehicle communication as a means of improving crash prevention performance. *NHTSA, Tech. Rep*, 2005:01–19, 2005.

[6] T. Cipra and R. Romera. Kalman filter with outliers and missing observations. *TEST*, Volume 6:379 –395, 1997.

[7] G. Hertkorn P. Mieth C. Rossel J. Zimmer D. Krajzewicz, M. Hartinger and P. Wagner. Using the road traffic simulation "sumo" for educational purposes. *Traffic and Granular Flow '03*, pages 217–222, 2005.

[8] A. Festag, H. Fußler, H. Hartenstein, A. Sarma, and R. Schmitz. Fleetnet: Bringing car-to-car communication into the real world. *Computer*, 4(L15):16, 2004.

[9] M. Ben Ghalia and A.T. Alouani. Robust control design of an autolanding system. *Proceedings SSST '93. Twenty-Fifth Southeastern Symposium on System Theory*, pages 248 – 252, 1993.

[10] Kalman, Rudolph, and Emil. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[11] Wei-Wen Kao. Integration of gps and dead-reckoning navigation systems. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 635 – 643, 20-23 1991.

[12] Viral V. Kapadia, Sudarshan N. Patel, and Rutvij H. Jhaveri. Comparative study of hidden node problem and solution using different techniques and protocols. *CoRR*, abs/1003.4070, 2010.

[13] Jun Luo and Jean-Pierre Hubaux. A survey of inter-vehicle communication. Technical report, School of Computer and Communication Sciences, EPFL, 2004.

[14] R. Parker and S. Valaee. Cooperative vehicle position estimation. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 5837 –5842, 24-28 2007.

[15] Robert Penoyer. The alpha-beta filter. *C Users J.*, 11(7):73–86, 1993.

[16] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5:2002, 2002.

[17] C. L. Robinson and L. Caminiti D. Caveney. Efficient coordination and transmission of data for cooperative vehicular safety applications. In *the Proc. of the 3rd international*, pages 10–19, 2006.

[18] D. Salmond. Target tracking: introduction and kalman tracking filters. *In Proceedings of IEE target tracking: algorithms and applications. Workshop*, Vol. 2, 2001.

[19] Michele C. Weigle Stephan Olariu, editor. *Vehicular networks : from theory to practice*. Boca Raton : CRC Press, 2009.