2015

Predicting Trajectory Paths For Collision Avoidance Systems

Cesar Barrios
University of Vermont

PREDICTING TRAJECTORY PATHS FOR COLLISION AVOIDANCE SYSTEMS

A Dissertation Presented

by

Cesar Barrios

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Electrical Engineering

January, 2015

Defense Date: November 5, 2014
Dissertation Examination Committee:

Walter Varhue, Ph.D., Advisor
Dryver Huston, Ph.D., Chairperson
Yuichi Motai, Ph.D., Research Advisor
Stephen Titcomb, Ph.D.
Tian, Xia, PhD.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

ABSTRACT

This work was motivated by the idea of developing a more encompassing collision avoidance system that supported vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communications. Current systems are mostly based on line of sight sensors that are used to prevent a collision, but these systems would prevent even more accidents if they could detect possible collisions before both vehicles were in line of sight.

For this research we concentrated mostly on the aspect of improving the prediction of a vehicle's future trajectory, particularly on non-straight paths. Having an accurate prediction of where the vehicle is heading is crucial for the system to reliably determine possible path intersections of more than one vehicle at the same time. We first evaluated the benefits of merging Global Positioning System (GPS) data with the Geographical Information System (GIS) data to correct improbable predicted positions. We then created a new algorithm called the Dead Reckoning with Dynamic Errors (DRWDE) sensor fusion, which can predict future positions at the rate of its fastest sensor, improving the handling of accumulated error while some of the sensors are offline for a given period of time. The last part of our research consisted in the evaluation of the use of smartphones' built-in sensors to predict a vehicle's trajectory as a possible intermediate solution for a V2V and V2I communications, until all vehicles have all the necessary sensors and communication infrastructure to fully populate this new system.

For the first part of our research, the actual experimental results validated our proposed system, which reduced the position prediction errors during curves to around half of what it would be without the use of GIS data for prediction corrections. The next improvement we worked on was the ability to handle change in noise, depending on unavailable sensor measurements, permitting a flexibility to use any type of sensor and still have the system run at the fastest frequency available. Compared to a more common KF implementation that would run at the rate of its slowest sensor (1Hz in our setup), our experimental results showed that our DRWDE (running at 10Hz) yielded more accurate predictions (25-50% improvement) during abrupt changes in the heading of the vehicle. The last part of our research showed that, comparing to results obtained with the vehicle-mounted sensors, some smartphones yield similar prediction errors and can be used to predict a future position.

CITATIONS

Material from this dissertation has been published in the following form:

Barrios, C., Motai, Y.. (2011). Improving Estimation of Vehicle's Trajectory Using the Latest Global Positioning System with Kalman Filtering. Transactions on Instrumentation and Measurement, IEEE, 60, 12, 3747-3755.

AND

Material from this dissertation will be submitted for publication to IEEE Transactions on Industrial Informatics, Special Issue on Secure Detection, Estimation, and Control in Cyber Physical Systems by January 31st, 2015 in the following form:

Barrios, C., Motai, Y., Huston, D.. Asynchronous Heterogeneous Sensor Fusion using Dead Reckoning and Kalman Filters. Transactions on Industrial Informatics, IEEE.

AND

Material from this dissertation will be submitted for publication to IEEE Transactions on Industrial Electronics and IEEE Transactions on Industrial Informatics, Joint Special Issue on Connected Vehicles – Advancements in Vehicular Technology and Informatics by November 30th, 2014 in the following form:

Barrios, C., Motai, Y. , Huston, D.. Can Smartphones Fill in the V2V/V2I Implementation Gap?. Transactions on Industrial Informatics, IEEE.

DEDICATION

I dedicate this to my wife, Raquel Sanchez-Ferrer Álvarez, for keeping the flame burning to get me here despite all the curve balls life throws at you.

ACKNOWLEDGEMENTS

This work was funded in part by the United States Department of Transportation through the University of Vermont Transportation Research Center.

I want to take the opportunity to acknowledge Dr. Yuichi Motai for his unfailing support, guidance and patience getting me through this degree.

I also want to thank Dr. Dryver Huston for being willing to review the journal articles before they were submitted, and for being the chairperson in my defense committee.

And last, but not least, many thanks to Dr. Walter Varhue, Dr. Steve Titcomb , and Dr. Tian Xia for also being part of this chapter in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Prediction of the trajectory path of a vehicle into the future is a difficult task, and even more so during non-straight paths, as observed in some of the research studied. Many times the predicted future position of where the vehicle will be 3 seconds later in time falls outside of a physical road, making this prediction highly improbable. For the first part of the research, the assumption is made that the driven vehicle will remain on a road at all times, and any prediction that falls outside of a road will be considered incorrect. Through the use of a road mapping technique, it will be shown that this error correction greatly reduces the prediction errors during non-straight paths.

Another problem observed when predicting a future position of a vehicle is that, when using multiple sensors, most of the time they are asynchronous. Some research reviewed described a solution of running the system at the rate of its slowest sensor, and, therefore, solving the problem of asynchronous data. Other research reviewed used previously estimated measurements to fill in the missing data form offline sensors. A vehicle is a large object that cannot change its spatial dynamics very quickly, but running a prediction system at a slow rate can slow down the detection of these spatial changes. For this research the system is run at the rate of its fastest sensor, but, missing measurements are calculated based on measurements obtained from online sensors suing a dead reckoning approach. A technique was developed to properly handle error accumulation from missing data from offline sensors, and that running the system at the fastest rate possible greatly reduces the prediction errors during non-straight paths.

The last part of this research looked into a possible solution to advance the usability of a vehicle-to-vehicle (V2V) system on its initial stages. The National Highway Safety Administration announced its decision to begin taking the next steps toward implementing V2V technology in all new cars and trucks. After all vehicle manufacturers are required to support this technology, it will still take many years until the V2V system is fully populated and most vehicles can contribute. Until that point is reached, the V2V technology will not be taken advantage of, unless a temporary solution is achieved to enable older vehicles to participate in the V2V system as well. Smartphones are readily available and already have many built-in sensors and good processing power, so in this part of the research the possibility of using smartphones to predict the trajectory path of a vehicle will be used. It will be shown that some types of smartphones yield similar prediction errors as predictions calculated using vehicle-mounted sensors.

CHAPTER 1: Improving Estimation of Vehicle's Trajectory Using Latest Global Positioning System with Kalman Filtering

1.1. Introduction

Accurately predicting the future location of a vehicle is a very important and relatively difficult topic in the Intelligent Transportation System (ITS). It can be effectively used in obstacle avoidance systems for vehicles or robots.

Many of the existing obstacle avoidance systems currently being researched are limited to line-of-sight sensors, such as those described in [1-9], using sensors around the vehicles to detect nearby objects. For a long-range obstacle avoidance system, other types of sensors need to be implemented such as those presented in [10, 11].

Researches like the one at the Kansai University of Japan [10] or the one by Miller and Oingfeng [11] investigate the option of using Global Positioning System (GPS) data collected from the different vehicles to predict the future location of each vehicle. The methods used to make these predictions are somewhat simple and do not give very accurate results in scenarios such as curves [Figure 1 and Figure 2] where the estimated future position of the vehicles will not be a straight path.

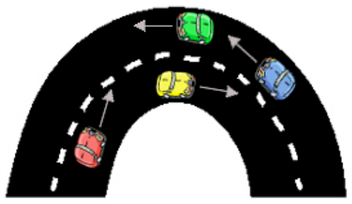


Figure 1: "C" crossing.

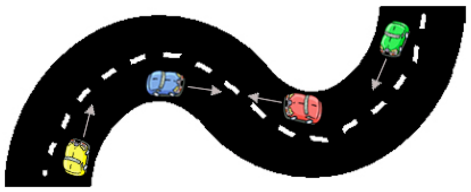


Figure 2: "S" crossing.

It is clear from current research that that what is needed is a more accurate way to predict the trajectory of the vehicles in all different scenarios. This is where the Kalman Filter (KF) comes into play. The KF has a long history of accurately predicting future states of a moving object and has been applied to many different fields, which is why it has been chosen for this research [12-15].

The contribution of this chapter is to investigate the viable idea of using the Geographic Information System (GIS) to reduce error in the prediction of the future location of an automobile, particularly during curves. The system implemented in this chapter consists of an Interacting Multiple Model (IMM) with different Kalman Filters (KF) using the Global Positioning System (GPS) to get a vehicle’s spatial information.

There are a number of existing studies concerning the best methods to take spatial coordinates that fall outside of a defined road and to estimate where they would fall on an actual road, also known as map-matching. For example, in [16-19] the authors go into a lot of detail to explain the different errors that need to be accounted for when using a GPS sensor (among others) and data for road maps (GIS), and how the GPS bias can be utilized to improve the map-matching accuracy. Other researchers, such as [20] and [21], look into the problem of GPS outages, and how the vehicle’s position can be estimated during the outage through the use of KF and map-matching techniques. This study compares experimental results of predictions done with and without our GIS error correction algorithm, and does not consider the problem of GPS outages since other researchers are working solely on this issue.

This research is based on the use of a GPS receiver to obtain location information and to be able to estimate the projected path for a vehicle. There are many factors that can degrade the GPS signal and thus affect its accuracy, but there are also some innovative ways of correcting these errors. The Holux GR-213 1Hz GPS receiver used in this research is Wide Area Augmentation System (WAAS) enabled.

The WAAS is a system developed for civil aviation by the Federal Aviation Administration (FAA) in conjunction with the United States Department of Transportation (USDOT). It is a nationwide differential GPS system where base stations with fixed receivers calculate and transmit the GPS error to the geostationary satellites in its view, which in turn broadcast the corrections that can be used by individual WAAS-capable GPS receivers. Its accuracy is less than 3 meters 95% of the time, and the GPS receiver used claims to have an accuracy of less than 2.2 meters [22].

Similar systems designed to predict a vehicle's trajectory implement the use of other types of sensors to be able to get an accurate estimation, but this research looks into the possibility of using a commercially available, inexpensive but accurate GPS receiver to do a similar task already implemented in some areas [12, 14, 15, 23-25]; and it takes advantage of using a location-based system, such as knowing where on a road map the vehicle is located.

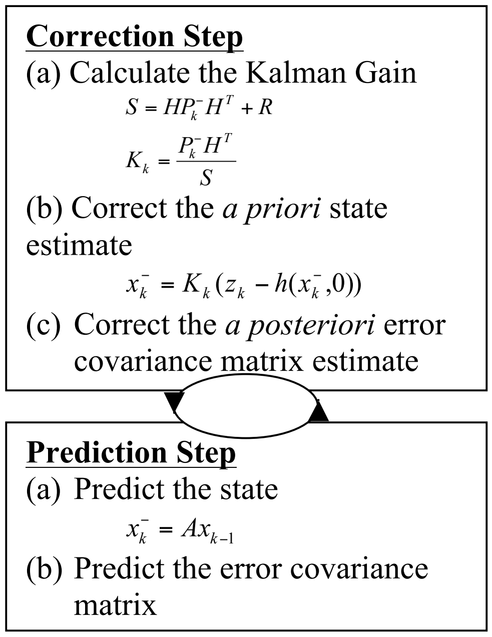
To be able to predict a vehicle's future location, the Kalman Filter (KF) was used. The KF is a set of mathematical equations that provides an efficient computational (recursive) method to estimate the future state of a process. The filter is very powerful because it supports estimations of past, present and even future states, and it can do so even when the precise nature of the modeled system is unknown [25-36].

The multiple KF models approach was chosen over one complex model because setting up multiple smaller models for each different scenario would be simpler than defining one complex model that can be accurate in many different scenarios. Each simple model is good for one specific set of conditions, so several models need to be defined to be able to cover most, if not all, possible scenarios in which a vehicle can be found. For this setup, four models have been identified to cover most of the vehicles' behaviors: a vehicle not moving; a vehicle traveling at constant velocity; or with constant acceleration; or with constant jerk (constant change in acceleration). These models provide a mathematical set of equations that can be used to predict the vehicle's future location after a set amount of time (Δk).

This study researches trajectory estimation at 3 seconds ahead in time, based on the average 1.5 second human reaction time to stop a vehicle [37]. The 3 seconds ahead in time was chosen as a reference point that is double the reaction time of an average human being. In reality, this number will probably vary in relation to the speed and type of the vehicle, since a faster or heavier vehicle will need more time to slow down. The fastest data rate of the GPS receiver used is one second ($\Delta k = 1$), so that is the rate the system will run at, which is set up to estimate the location of the vehicle one second later in time. To be able to obtain an estimation for the location of a vehicle three seconds later in time, the researchers needed to run the prediction steps of the KF system with Δk set to 3 seconds, and use the IMM to obtain the prediction. This extra step to estimate the 3 seconds ahead location adds very little runtime to the overall system, since it is only used to predict the location and no correct steps are needed.

1.2. Kalman Filter

The Kalman Filter (KF) estimates a process by using a form of feedback control loop: the filter estimates the process state at some time, and then obtains feedback in the form of (noisy) measurements, and then it repeats (see Figure 3). As such, the equations for the KF fall into two groups: what we have called "prediction step" and "correction step." The prediction step equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The correction step equations are responsible for the feedback—i.e., for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.


Correction Step

(a) Calculate the Kalman Gain

$$S = HP_k^- H^T + R$$

$$K_k = \frac{P_k^- H^T}{S}$$

(b) Correct the *a priori* state estimate

$$x_k^- = K_k(z_k - h(x_k^-, 0))$$

(c) Correct the *a posteriori* error covariance matrix estimate

Prediction Step

(a) Predict the state

$$x_k^- = Ax_{k-1}$$

(b) Predict the error covariance matrix

Figure 3: Extended Kalman Filter.

Notation:

x	state estimate
z	measurement data
A	Jacobian of the system model with respect to state
H	Jacobian of the measurement model
Q	process noise covariance
R	measurement noise covariance
K	Kalman Gain
P	estimated error covariance
σ_p	prediction noise
σ_m	measurement noise

For our system the state vector for this system consists of two parameters obtained from the GPS sensor, each one decomposed into its *x* and *y* components. The general form of the state estimate matrix is shown in (1).

$$x = \begin{bmatrix} x_v \\ v_v \end{bmatrix} = \begin{bmatrix} Position - of - vehicle \\ Velocity - of - vehicle \end{bmatrix} \quad (1)$$

The elements of the state vector in (1) were selected to account for all the measurements available from the GPS sensor, and from them any other variables needed for the KF models were derived. Keep in mind that each of the components of the state estimate in (1) has an *x* and *y* component to it. So for every x_k represented in the equations there will be an x_{kx} and an x_{ky}.

The error covariance matrix is a dataset that specifies the correlations in the observation errors between all possible pairs of vertical levels. The error covariance for each KF was approximated by running the filters on their own, but its value gets adjusted every 1 second in our setup.

The estimated error covariance P is used together with the Jacobian matrix H and the measurement noise covariance (R) to calculate the Kalman Gain (K) as shown in Figure 3.

$$P = \begin{bmatrix} x_v x_v & x_v v_v \\ v_v x_v & v_v v_v \end{bmatrix} \quad (2)$$

$$h(x,v) = \begin{bmatrix} x_v + v \\ v_v + v \end{bmatrix} \quad (3)$$

$$H = \left[\frac{\partial}{\partial x} h(x,0) \right]_{\substack{x=x(k-1) \\ v=0}} \quad (4)$$

$$R = \sigma_m^2 \cdot \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad (5)$$

Once the Kalman Gain (K) is calculated, the system looks at the measured data (Z) to correct the predicted position and also the covariance error. Since this system only obtains measurements from a GPS receiver, only location, speed, and heading angle can be obtained; therefore, the other two parameters need to be calculated from the measured data. The acceleration is calculated from the velocity difference between the current and previous reading, and similarly, the jerk is calculated from the acceleration difference between the current and previous values. For this experiment, instead of using the current speed and heading from the GPS sensor, the average speed parameter was also similarly derived from the location difference between the current and previous values.

Another important item to point out is that this research does not look into solving the GPS measurement errors that are due to many factors. One of these error contributors is the "Signal Multipath" problem, where the signal reflects off large objects. In this research, it is assumed these errors are minimal since the experiment is done in a very rural area. Also, signal delays (ionosphere and troposphere) can cause the location readings from the GPS to bounce around and imply movement when the vehicle is not even moving. There are many error contributors to the GPS receivers, but we will assume them negligible in this research to concentrate on the main objective of this system.

After correction of the previously predicted values, the system is ready to predict the next position by using the state vector equations. The filter also estimates the error covariance of the estimated location by using the Jacobian matrix A together with the Process noise covariance (Q).

$$A = \left[\frac{\partial}{\partial x} f(x,w)\right]_{\substack{x=x(k-1)\\ w=0}} \qquad (6)$$

To obtain an accurate prediction of the vehicle's future location, four adaptive prediction algorithms are defined to account for the possible scenarios. The state equations are very different between the models. The following four models account for most, if not all, possible situations in which a vehicle could be found.

Constant Location Model (CL)

$$\begin{aligned} x_v(k) &= x_v(k-1) + w \cdot \Delta k \\ v_v(k) &= 0 \end{aligned} \qquad (7)$$

Constant Velocity Model (CV)

$$\begin{aligned} x_v(k) &= x_v(k-1) + (v_v(k-1) + w)\Delta k \\ v_v(k) &= v_v(k-1) + w \end{aligned} \qquad (8)$$

Constant Acceleration Model (CA)

$$\begin{aligned} x_v(k) &= x_v(k-1) + v_v(k-1)\Delta k + \frac{1}{2}(a_v(k-1) + w)\Delta k^2 \\ v_v(k) &= v_v(k-1) + (a_v(k-1) + w)\Delta k \end{aligned} \qquad (9)$$

Constant Jerk Model (CJ)

$$x_v(k) = x_v(k-1) + v_v(k-1)\Delta k + \frac{1}{2}a_v(k-1)\Delta k^2 + \frac{1}{6}(j_v(k-1)+w)\Delta k^3$$
$$v_v(k) = v_v(k-1) + a_v(k-1)\Delta k + \frac{1}{2}(j(k-1)+w)\Delta k^2 \qquad (10)$$

In equations (7) through (10) Δk represents the period of time passed, so the variables at $k-1$ represent the data from 1 period of time ago. In this setup, the period of time is driven by the data rate of the GPS (1 second). The process noise covariance for each of the models (w) is based on the constant term only. For example, for the CV model, the process noise covariance is based on the velocity term only, and it can be derived from the measured data by applying the CV model to it.

Equations (7) through (10) represent the four states in which a vehicle can be found: at rest; moving at constant velocity; moving at constant acceleration; or moving at constant jerk. Each of these models consists of four state equations used to calculate each component of the state estimate matrix defined in (1). These models are very important as they are the heart of the prediction system. They need to cover most, if not all, of the possible scenarios or the predictions will contain more errors.

For more details on how to setup a KF and a detailed explanation of all required mathematical equations, please refer to publications such as [25, 27, 38].

1.3. Interacting Multiple Models Estimation

Because the dynamics of automobiles can vary over time, the state equations (7-10) are already defined to capture the different states in which a vehicle can be found, but these independent state equations need to be merged to produce only one prediction.

There are several algorithms that exist to modify the stochastic information, and they are well known for their ability to automatically adapt the filter in real time to match any variation of the errors involved.

The Interacting Multiple Models Estimation (IMM) algorithm calculates the probability of occurrence for each of the individual filters and uses that information to identify which of the filters will be predominant. This algorithm continues recalculating the probability for each iteration throughout the whole run, weighting the new probability values against the probability values calculated in the previous iteration.

The IMM filter calculates the probability of success of each model at every filter execution, providing combined solution for the vehicle behavior. These probabilities are calculated according to a Markov model for the transition between maneuver states, as detailed in [28]. To implement the Markov model, it is assumed that at each execution time there is a probability *pij* that the vehicle will make a transition from model state *i* to state *j*. Equation (11) shows the transition matrix for the four defined KF models defined in section 2.2.

$$p_{ij} = \begin{bmatrix} CL \to CL & CL \to CV & CL \to CA & CL \to CJ \\ CV \to CL & CV \to CV & CV \to CA & CV \to CJ \\ CA \to CL & CA \to CV & CA \to CA & CA \to CJ \\ CJ \to CL & CJ \to CV & CJ \to CA & CJ \to CJ \end{bmatrix} \quad (11)$$

The IMM can be described as a recursive suboptimal algorithm that consists of five core steps:

• **Step 1)** Calculation of the mixing probabilities

• **Step 2)** Mixing

• **Step 3)** Mode matched filtering

• **Step 4)** Mode probability update

• **Step 5)** Estimate and covariance combination

As in any recursive system, the IMM algorithm first needs to be initialized before it can start its four-step recursion. The number of filters used is 4.

• **Step 1)** Calculation of the mixing probabilities

The probability mixing calculation uses the transition matrix (11) and the previous iteration model probabilities (16) to compute the normalized mixing probabilities (12). The mixing probabilities are recomputed each time the filter iterates before the mixing step.

$$\lambda_k(i \mid j) = \frac{p_{ij}\lambda_{k-1}(i)}{\sum_{i=1}^{N} p_{ij}\lambda_{k-1}(i)} \qquad (12)$$

• **Step 2)** Mixing

The mixing probabilities are used to compute new initial conditions for each of the *N* filters, four in this case. The initial state vectors are formed as the weighted average of all the filter state vectors from the previous iteration (13). The error covariance corresponding to each of the new state vectors is computed as the weighted average of the previous iteration error covariance's conditioned with the spread of the means (14).

$$x_{k-1}^{oj} = \sum_{i=1}^{N} \lambda_{k-1}(i \mid j)\hat{x}_{k-1}^{i} \tag{13}$$

$$P_{k-1}^{oj} = \sum_{i=1}^{N} \lambda_{k-1}(i \mid j) \times \left\{ P_{k-1}^{i} + \left[\hat{x}_{k-1}^{i} - \hat{x}_{k-1}^{0j}\right]\left[\hat{x}_{k-1}^{i} - \hat{x}_{k-1}^{0j}\right]^{T} \right\} \tag{14}$$

• **Step 3)** Mode matched filtering

Using the calculated $\hat{x}_{k-1}^{0j}$ and P_{k-1}^{0j} the bank of 4 Kalman filters produce outputs $\hat{x}_{k}^{j}$, the covariance matrix P_{k}^{j} and the probability density function $f_n(z_k)$ for each filter (*n*) in equation (16), according to the equations for the KF. The covariance for each filter is represented by S_k in (15) and (18).

$$S_k = H \cdot P \cdot H^T \tag{15}$$

$$f_n(z_k) = \frac{1}{\sqrt{(2\pi)^{\frac{4}{2}}\left|S_k\right|}} e^{-\left(\frac{1}{2}\right)\cdot(V^T \cdot S^{-1} \cdot V)} \tag{16}$$

• **Step 4)** Mode probability update

Once the new initial conditions are computed, the filtering step (step 3) generates a new state vector, error covariance and likelihood function for each of the filter models. The probability update step then computes the individual filter probability as the normalized product of the likelihood function and the corresponding mixing probability normalization factor (17).

$$\lambda_k(j) = \frac{f_n(z_k)}{\sum_{i=1}^{N} f_n(z_k)} \sum_{i=1}^{N} p_{ij}\lambda_{k-1}(i) \tag{17}$$

• **Step 5)** Estimate and covariance combination

This step is used for output purposes only; it is not part of the algorithm recursions.

$$\hat{x}_k = \sum_{j=1}^{N} \lambda_k^j \cdot \hat{x}_k^j \tag{18}$$

$$P_k = \sum_{i=1}^{N} \lambda_k^j \cdot \left\{ P_k^j + \left[\hat{x}_k^j - \hat{x}_k \right] \left[\hat{x}_k^j - \hat{x}_k \right]^T \right\} \tag{19}$$

1.4. Geographical Information System

A geographic information system (GIS) is a system for capturing, storing, analyzing, and managing data and associated attributes which are spatially referenced to the earth. It is a tool that allows users to create interactive queries (user created searches), analyze the spatial information, edit data and maps, and present the results of all these operations. In this research we extracted the road information from the maps being used to display the vehicle's location. It is not a very accurate map, but it is enough to demonstrate if the implementation of GIS information with the IMM system improves the prediction of the vehicle's future location or not.

The idea of using GIS data to correct an invalid estimation came about looking at simulations during curves. When the vehicle enters a turn, the prediction of its future locations is very erroneous, many times outside of a road. If the system had a way of knowing the direction of the road ahead, and whether the estimated future location was on an actual road or not, it would be able to correct its estimation and improve its reliability. This is where GIS comes into play, with the assumption that the vehicle will always remain on the road. It is also assumed the driver is handling the vehicle properly and awake for this GIS correction to be practical. These assumptions, though restrictive, still allow the correction to be useful in scenarios such as road intersections.

When a road is designed, the radius of curvature is known, but this information is not available with the GIS data; therefore, a new method is needed to be able to project the estimation outside of the road back in the road.

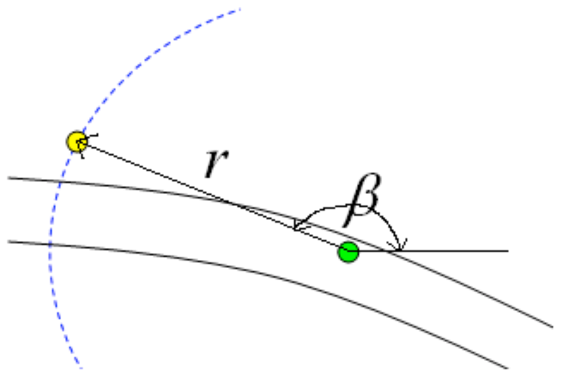


Figure 4: Displaying parameters used in the method to estimate position on the road.

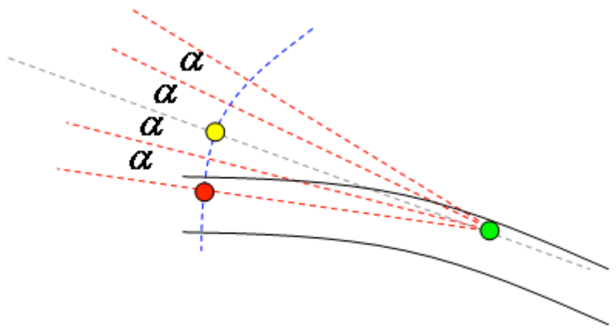


Figure 5: Geometry used to map estimated future location outside the road to a location inside the road.

Because of the limitation of the mapping software used during this research (MapPoint), the only available function to interact with GIS data was to check whether the specific location was on the road or not. A function that provided the distance from the current location to the nearest road would have worked better, but it was not available in MapPoint.

To overcome the limitation described earlier, a method to map the estimated future location outside of the road to an accurate location inside a road had to be designed. From the current GPS location the distance *r* and the angle β shown in Figure 4 are calculated. The angle β varies with the direction of the movement and calculated from East being zero degrees. The *r* is the distance between the current location and the estimated location.

$$count = \frac{circumference}{arc} \qquad (20)$$

$$\alpha = \frac{360\deg}{count} \qquad (21)$$

The variable *arc* used in (20) is the predefined distance between points in the circumference. The smaller this value is, the smaller the increments between check points in the circumference are, and the more accurate the measurement will be. Because the smaller the *arc* value, the more points that need to be checked, it required more CPU processing time so for this research *arc* has a value of 0.6 meters. This value was selected because the smallest road, even if only a one-way lane, cannot be less than 2 meters wide. If we used a value bigger than 2 meters, we could have the possibility of missing a road between checkpoints, so we chose a significantly smaller value. The angle α calculated in (21) is the actual angle increment needed to match the predefined *arc* distance on the circumference.

With the angle β shown in Figure 4 and the angle α calculated in (21), running through the checkpoints of the circumference was started. The estimated location is found at angle β and since this estimated location cannot be too far from the actual road, checking was started from this angle β. The system will check both clockwise and counterclockwise increments of α until a point is found on the road. Figure 5 provides a graphical view of the GIS error checking implemented. The clockwise and counterclockwise increments will continue to occur until either a road is found and a correction on the estimated future location is made, or a maximum number of increments is reached, and no correction is made. If a correction is made, the new estimated future location will still be the same distance away *r,* the only difference is its location coordinates.

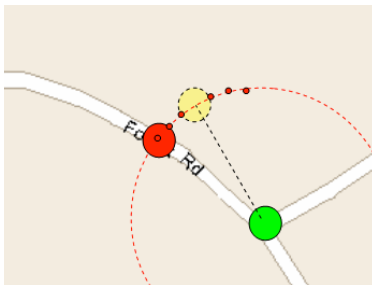


Figure 6: GIS error correction in MapPoint.

In Figure 6, in MapPoint, the current location is a green dot, the predicted future location is a yellow dot, and the GIS corrected data is a red dot. The smaller red dots are the clockwise and counterclockwise increments described earlier. Visually, in Figure 6, the estimated future location is probably incorrect as there is no road in that location. Using GIS data to locate the road, the predicted location can be adjusted to be on the road at the same distance away, as the velocity will probably not change significantly under normal circumstances. The result is a more accurately predicted future location. This method seems to work well during curves, but, as stated earlier, it requires several restrictive assumptions. Therefore, this system could only be useful as a part of a larger and more robust collision avoidance system that took into account some of the scenarios not covered by our proposed method.

1.5. Experimental Results

The experimental setting for testing the models described in section 1.1.2 needs a log file of GPS data that contains different scenarios, especially those currently causing problems in existing systems (Figure 1 and Figure 2). Figure 7 shows the trajectory recorded for this research. It has many turns and contains various changes in speed and direction. Because in trying to improve the trajectory estimation during curves, Figure 7 also shows the curve selected for our experiment.

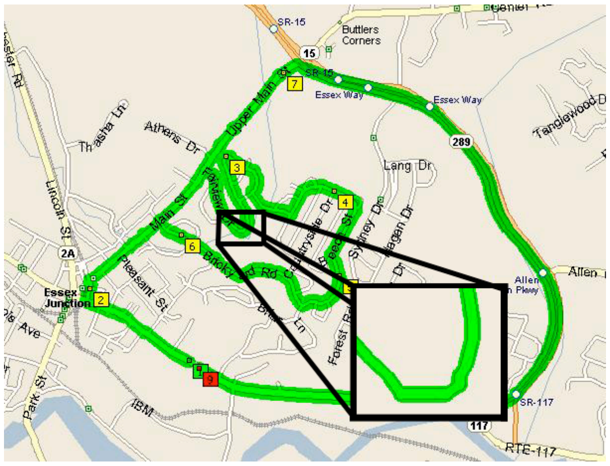


Figure 7: Trajectory recorded in GPS log file, Essex Jct., VT, USA.

The selected road curve is definitely a nice sharp turn that occurs at medium speeds (~60kph). It was felt that this turn would be a good scenario to test the improvements on trajectory estimation.

The code was implemented in Microsoft Visual Basic 6 and Microsoft MapPoint 2004, allowing the software display information in real time on the map as the vehicle moves. Being able to look at the estimated points on an actual map makes it easier to visually inspect and present the system.

1.5.1. Implementation of Kalman Filters

To be able to evaluate, the four KF Models, KF-CL (Kalman Filter Constant Location), KF-CV (Kalman Filter Constant Velocity), KF-CA (Kalman Filter Constant Accelerator), and KF-CJ (Kalman Filter Constant Jerk) had to be coded, tested and tuned individually to get as accurate estimations as are possible. It is a given that one of these models will not be very accurate all the time on a real time GPS log; therefore, in order to calibrate them individually, the GPS log for the full trajectory shown in Figure

7 was used to calculate the measurement noise covariance and also each of the process noise covariance for the four models to exercise only one model at the time. To find the values for the process and measurement noise covariance matrices, the data was smoothed out using a moving average window to remove any outlier. The measurement noise covariance was obtained for each of the filters by calculating the covariance of going frame by frame, and calculating the error of the real data to fit into each of the KF models defined in section 1.2.

Once the filters were tuned, they were individually run through the different scenarios and only the results for the data points in the selected curve were recorded in Table 1.

Running the four filters together showed how, when one was very close to the real value, the other ones were not that accurate. In some instances more than one filter was accurate, probably when speed changes or acceleration changes were very small. In other cases none of the four filters was accurate at all, probably because of an abrupt change in direction or even in speed. The system reads data from the GPS every one second, so it is possible, though not common at higher speeds, to have a big change occur during that one second, especially in curves. For the most part one second will not allow the speed and direction to change by a big amount (except in some lower speed scenarios, such as at intersections when making a sharp turn), allowing the filters to estimate the next location somewhat accurately. The error calculated in Table 1, Table 2 and Table 3 are based off the actual GPS location. It is the distance between the estimated three seconds ahead location and the actual GPS location three seconds later.

Actual GPS errors are not accounted for in this research, so both the estimated future location and the actual GPS location should be similarly affected by the GPS error.

Table 1: Average 3 sec ahead estimation error

	CL	CV	CA	CJ
KF	14.9002	9.8786	7.0812	8.9952

Units are in meters. Used 21 data points for the selected curve.

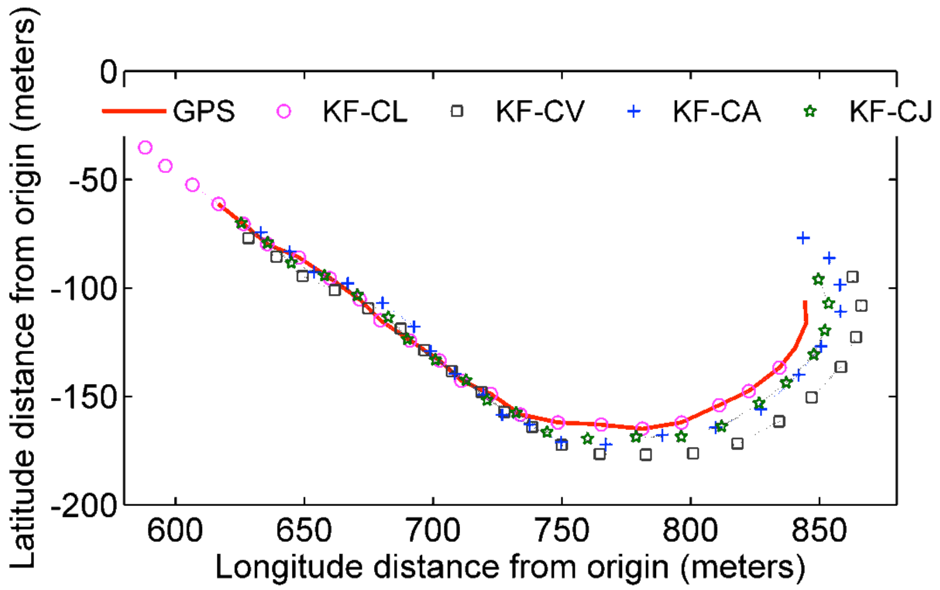


Figure 8: Comparison of estimated 3 sec ahead location and actual GPS reading for all four KF models using 21 data points for the selected curve.

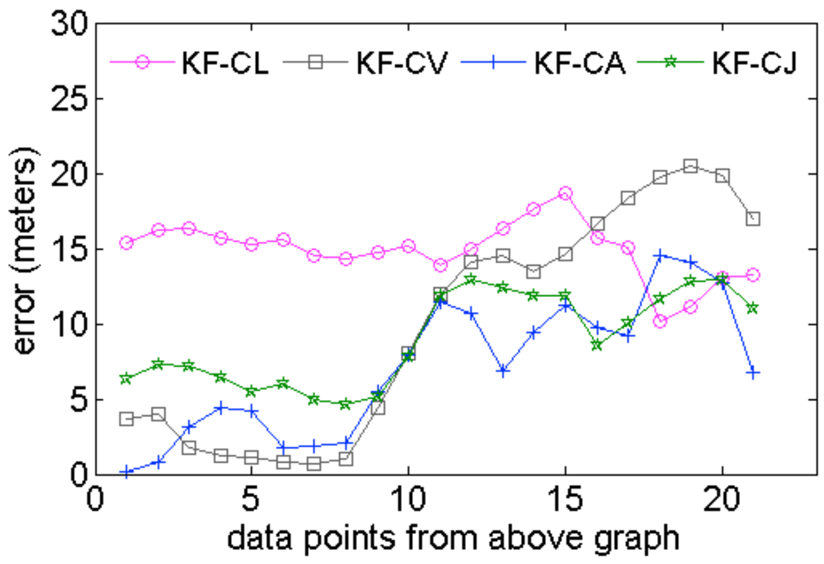


Figure 9: Calculated error for all KF models between 3 sec ahead estimation and actual GPS location 3 sec later using 21 data points for selected curve.

From Figure 8 and Figure 9 we can analyze the results of running the KF by themselves (each KF is predicting the future location 3 seconds later in time). Figure 8 shows the predicted location 3 seconds ahead in time on the spatial trajectory (same curve as shown in Figure 7), while Figure 9 shows the error for each of the predictions of the future vehicle’s location 3 seconds later in time compared to the actual GPS measurement. Both graphs are needed because KF-CL seems to be accurate in Figure 8 because it will always be on a real GPS location since it assumes no movement (Constant Location). Actually the KF-CL shows a lot of error in Figure 9 since the vehicle was always moving through this curve. The estimated future location for this model will be where the current GPS location is (right over the GPS line), but this will not be accurate if the vehicle is moving, and this is where Figure 9 displays this error.

Because the curve selected in Figure 7 was driven at a somewhat constant speed, it can be noted that both the KF-CV and the KF-CA are the most accurate in this case until the curve starts.

1.5.2. Evaluation of Interacting Multiple Models

To set up the IMM it was necessary to calculate the transition probability matrix in equation (11) using the GPS log for the full trajectory shown in Figure 7. From this full GPS log that contained multiple scenarios, it was determined which transition was occurring frame by frame by comparing the actual measurements from the GPS to the smoothed measurements. The smoothing of the data was done with a rolling window using a combination of median smoothing, splitting the sequence, and Hann's sequence, which removed any abrupt changes from the data. Each transition was determined by the type of change, such as no change, a constant change, and so on. Similarly, by calculating the covariance of the differences in the measurements to each other, the measurement noise covariance matrix (R) was obtained. And last, by calculating the covariance of the differences in the measurements compared to their respective *x* and *y* components, the process covariance noise (Q) was obtained for each KF. From this type of information the transition probability matrix below was obtained.

$$p^{ij} = \begin{bmatrix} 0.154 & 0.154 & 0.385 & 0.308 \\ 0.011 & 0.470 & 0.305 & 0.214 \\ 0.014 & 0.259 & 0.458 & 0.269 \\ 0.002 & 0.243 & 0.508 & 0.247 \end{bmatrix} \qquad (22)$$

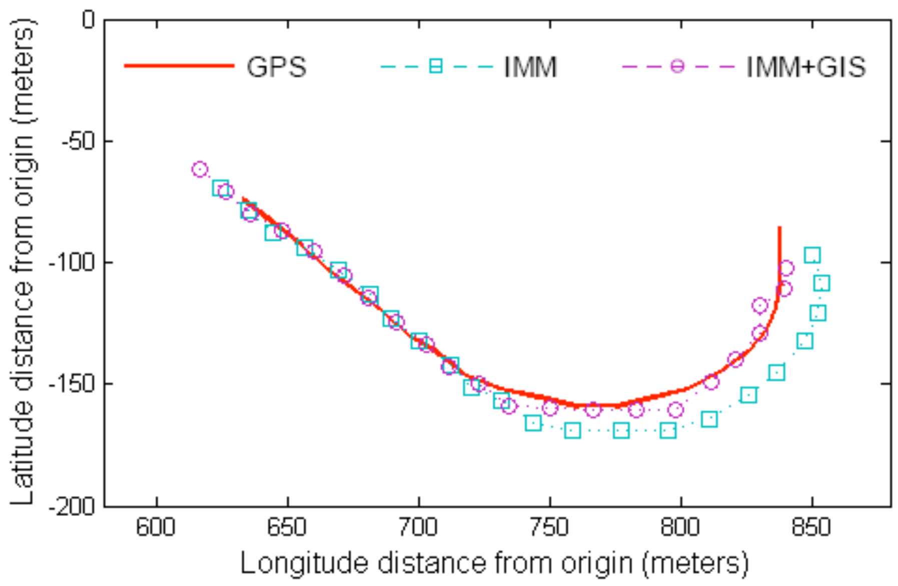


Figure 10: Comparison of 3 sec ahead estimated location between IMM, IMM+GIS and actual GPS locations 3 sec later using 21 data points for selected curve.

Table 2: Average Estimation Error for selected curve

Estimated position	*1 sec ahead*	*3 sec ahead*
IMM	2.9056	8.7880
IMM with GIS	1.7834	4.8244

Units are in meters. Used 21 data points for selected curve.

Table 3: Average Estimation Error for whole trajectory

Estimated position	*1 sec ahead*	*3 sec ahead*
IMM	1.9461	6.5276
IMM with GIS	1.8872	5.1423

Units are in meters. Used 800 data points for whole trajectory in Figure 8.

Looking at equation (22) some scenarios are clearly identified. For example, when in a CL state (first row), it is more probable for it to change to a CA or CJ state than to a CV state, and this is understandable because when a vehicle is at a complete stop, to start moving it will need to accelerate.

Also, only the 3 seconds away estimation results will be looked at as this is the most important one for this study. Looking at a 1 second ahead estimation allows for some very accurate results but this would not be enough warning time for the driver to react, so this research will look at 3 second away estimation and how accurate it can be obtained.

The results obtained from the IMM were not good enough to make this system very reliable by itself. In Table 2 a 45% improvement was identified when using the GIS correction method, but the error is still significant when predicting the vehicle's location 3 seconds ahead of time. Table 3, similarly to Table 2, shows the average errors for the estimated future vehicle's location 1 and 3 seconds ahead in time, but the whole trajectory as shown in Figure 8 was used to test this system. The numbers do not show as great an improvement as in Table 2 because, when the vehicle is traveling in a straight line, the error in the estimated future location is smaller, and therefore adding GIS correction is not as beneficial. Overall, even though GIS does show to be very helpful, especially during curves, it is still not enough to use it by itself, as it was set up for this research. A much needed improvement would be the implementation of more sensors that could run at higher frequencies.

In Figure 10, this study visually compared the estimated 3 second ahead positions with the GPS values. It also shows that the IMM had a lot of error at the beginning of the turn and after a few seconds converged more with the actual data. So this method used as part of a collision avoidance system would produce many false warnings.

1.5.3. Geographical Information System (GIS)

The implementation of GIS data with the IMM estimation process showed very promising results.

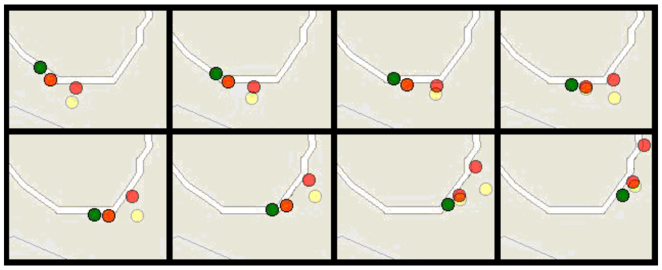


Figure 11: Frame shots of simulation during the selected curve.

In Figure 11, the frame shots of the simulation program can be seen. It shows in light yellow the three positions corresponding to 1 and 3 second away estimations. In red, the images show the corrected predicted location for each of the 1 and 3 second away estimations. It is easy to see how much the GIS correction helps with the actual estimation of future positions of the vehicle. To look at some numbers, Table 2 can be used to confirm this visual conclusion. The table shows the average error for the selected turn and we can see a noticeable difference compared to the method without any GIS correction, especially when looking at the 3 seconds ahead. This method is a lot more reliable and should give a lot less false warnings because the approximate 3 meters of error it has is a little more than a compact vehicle's width and about the same as its length.

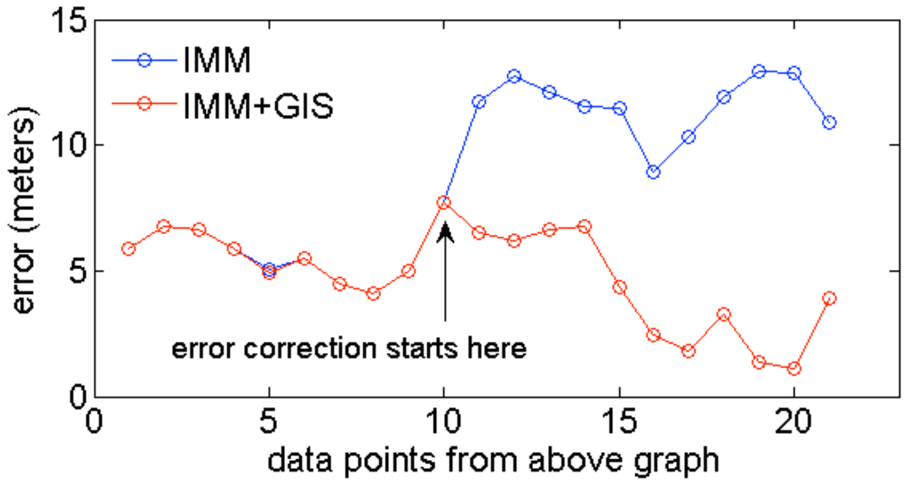


Figure 12: Error measured between the 3 sec ahead IMM estimation (with and without GIS) and the actual GPS readings 3 sec later using 21 data points for selected curve.

Figure 12 is a further visual aid to be able to compare it to the previous two methods and see how much more accurate this is.

The GIS error correction method used in this research is somewhat simple and straightforward. It can possibly be improved with other existing methods, but it was enough to help determine whether using GIS data with the trajectory estimation models was an improvement.

1.6. Conclusions

This chapter implemented four KF to account for the identified possible states an automobile can be found in (constant location, constant velocity, constant acceleration, and constant jerk). These four KF were set up to be part of an IMM system that provided the predicted future location of the automobile up to 3 seconds ahead in time. To improve the prediction error of the IMM setup, this study added an iterated geometrical error detection method based on the GIS system. The assumption made was that the automobile would remain on the road, so predictions of future locations that fall outside of the road were corrected accordingly, making great reduction to prediction error, as shown in the experimental results.

The research observed estimation values at 3 seconds ahead in time to allow for enough reaction time if this setup were to be used in some type of driver's aid system. As shown in this research, a 3 seconds ahead estimation has a lot of error, but, with the help of GIS data, this error can be reduced drastically, especially during turns, which is where research seems to have the most problems with [10].

The idea of merging spatial GPS data with GIS road information, given some assumptions, has proven to improve the accuracy of predicting a vehicle's future. And, in some scenarios, it could be an interesting addition to a collision avoidance system.

Despite the improved predictions shown in this chapter, this system can be further improved. The implemented GIS method in this study was straightforward and could be improved by looking into more detailed GIS data and being able to determine the lane the vehicle is driving in to correct with more accuracy a bad estimated future location. The spatial data used from the GPS can also be complemented by using other types of sensors less error prone and that can run at a frequency higher than 1Hz.

1.7. References

[1] A. Tascillo, R. Miller, "An in-vehicle virtual driving assistant using neural networks", Proceedings of the International Joint Conference on Volume 3, pp. 2418-2423, July 2003.

[2] M. Lee, Y. Kim, "An efficient multitarget tracking algorithm for car applications", Industrial Electronics, IEEE Transactions on Volume 50, Issue 2, pp. 397-399, April 2003.

[3] A. Amditis, E. Bertolazzi, M. Bimpas, F. Biral, P. Bosetti, M. Da Lio, L. Danielsson, A. Gallione, H. Lind, A. Saroldi, Sjo, A. Gren, "A holistic approach to the integration of safety applications: The INSAFES subproject within the European Framework Programme 6 Integrating Project PReVENT", IEEE Trans. Intell. Trans. Syst., Volume 11, no. 3, pp. 554-566, December 2009.

[4] Y. Ikemoto, Y. Hasegawa, T. Fukuda, K. Matsuda, "Zipping, weaving: Control of vehicle group behavior in non-signalized intersection", IEEE Proceedings of International Conference on Robotics and Automation, Volume 5, pp. 4387-4391, May 2004.

[5] S. G. Wu, S. Decker, P. Chang, T. Camus, and J. Eledath, "Collision sensing by stereo vision and radar sensor fusion," IEEE Trans. Intell. Transp. Syst., Volume 10, no. 4, pp. 606-614, December 2009.

[6] S. Pietzsch, T. D. Vu, J. Burlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig, "Results of a precrash application based on laser scanner and short-range radars," IEEE Trans. Intell. Transp. Syst., Volume 10, no. 4, pp. 584-593, December 2009.

[7] M. Chowdhary, “Driver assistance applications based on automotive navigation system infrastructure”, Proceedings from ICCE International Conference in Consumer Electronics, pp. 38- 39, June 2002.

[8] C. Drane, C. Rizos, “Positioning systems in Intelligent Transportation Systems (book style)”, Artech House Inc., 1998.

[9] R. Bishop, “Intelligent vehicle technology and trends”, Artech House Inc., 2005.

[10] J. Ueki, J. Mori, Y. Nakamura, Y. Horii, H. Okada, “Development of vehicular-collision avoidance support system by Inter-Vehicle Communications”, Proceedings of IEEE 59th Vehicular Technology Conference, Volume 5, pp. 2940-2945, May 2004.

[11] R. Miller, H. Qingfeng, “An adaptive peer-to-peer collision warning system”, Proceedings of IEEE 55th Vehicular Technology Conference, Volume 1, pp. 317-321, May 2002.

[12] B. P. Zhang, J. Gu, E. Milios, and P. Huynh, “Navigation with IMU/GPS/digital compass with unscented Kalman filter”, Proceedings of the IEEE International Conference on Mechatronics & Automation, pp. 1497-1502, July 2005.

[13] B. Barshan, H. F. Durrant-Whyte, “Inertial navigation systems for mobile robots”, IEEE International Transactions on Robotics and Automation. Volume II, no. 3, pp. 328-342, June 1995.

[14] R. Toledo, M. A. Zamora, B. Ubeda, A. F. Gomez-Skarmeta, “An integrity navigation system based on GNSS/INS for remote services implementation in terrestrial vehicles”, IEEE Proceedings from the Intelligent Transportation Systems Conference, pp. 477-480, Washington, DC, 2004.

[15] R. Toledo, M. A. Zamora, B. Ubeda, A. F. Gomez, “High integrity IMM-EKF based road vehicle navigation with low cost GPS/INS”, IEEE Transaction on Intelligent Transportation Systems, Volume 8, no. 3, ITISFG, September 2007.

[16] W. Kim, G. Jee, and J. Lee, “Efficient use of digital road map in various positioning for ITS”, IEEE Transaction on Position Localization and Navigation, San Diego, CA, 2000.

[17] X. Zhang, Q. Wang, D. Wan, “Map matching in road crossings of urban canyons based on road traverses and linear heading-change model”, IEEE Transaction on Instrumentation and Measurement, Volume 56, no. 6, pp. 2795-2803, December 2007.

[18] Chang Bok Lee, Dong Doo Lee, Nak Sam Chung, Ik Soo Chang, E. Kawai, F. Takahashi, “Development of a GPS codeless receiver for ionospheric calibration and time transfer”, IEEE Transaction on Instrumentation and Measurement, Volume 42, no. 2, pp. 494-497, April 1993.

[19] M. Matosevic, Z. Salcic, S. Berber, “A comparison of accuracy using a GPS and a low-cost DGPS”, IEEE Transaction on Instrumentation and Measurement, Volume 55, no. 5, pp. 1677-1683, October 2006.

[20] M.E. Najjar, P. Bonnifait, “A roadmap matching method for precise vehicle localization using belief theory and Kalman filtering”, IEEE 11th International Conference in Advanced Robotics, Coimbra, Portugal, 2003.

[21] Y. Cui, S.S. Ge, “Autonomous vehicle positioning with GPS in urban canyon environments”, IEEE Transactions on Robotics and Automation, Volume 19, no.1, pp. 15-25, February 2003.

[22] Holux Technology Inc., Holux GR-213 GPS Specifications, http://www.holux.com/JCore/en/support/DLF.jsp?DLU=http://www.holux.com/JCore /UploadFile/7011686.pdf

[23] A. Lahrech, C. Boucher, and J. C. Noyer, "Fusion of GPS and odometer measurements for map-based vehicle navigation", IEEE Proceedings from the International Conference on Industrial Technology, December 2004.

[24] S. Sukkarieh, Low Cost, High Integrity, "Aided inertial navigation systems for autonomous land vehicles", Ph.D. Thesis from the University of Sydney Australia, 2000.

[25] C. Hide, T. Moore, M. Smith, "Multiple model Kalman filtering for GPS and low-cost INS integration", Proceedings of ION GNSS, 2004.

[26] J. Bohg, "Real-time structure from motion using Kalman filtering", Technische Universitat Dresden. March 2005.

[27] G. Welch, G. Bishop, "An introduction to the Kalman filter", SIGGRAPH 2001, Course notes.

[28] J. Bohg, "Real-times structure from motion using Kalman filtering", Ph.D. Dissertation, 2005.

[29] C. Hu, W. Chen, Y. Chen, D. Liu, "Adaptive Kalman filtering for vehicle navigation", Journal of Global Positioning Systems, Volume 2, no. 1, pp. 42-47, 2003.

[30] Y. Zhang, H. Hu, H. Zhou, "Study on adaptive Kalman filtering and algorithms in human movement tracking", Proceedings of the IEEE International Conference on Information Acquisition, 2005.

[31] L. C. Yang, J. H. Yang, E. M. Feron, “Multiple model estimation for improving conflict detection algorithms”, IEEE Proceedings from the Conference on Systems, Man and Cybernetecis, Volume 1, pp. 242-249, October 2004.

[32] Wang, Xuezhi, “Maneuvering target tracking and classification using multiple model estimation theory”, Ph.D. Dissertation, University of Melbourne, 2001.

[33] C. Hilde, T. Moore, M. Smith, “Multiple model Kalman filtering for GPS and low-cost INS integration”, Institute of Engineering, Surveying and Space Geodesy, University of Nottingham, 2004.

[34] E. Derbez, B. Remillard, “The IMM CA CV performance”, unpublished.

[35] Y. Bar-Shalom, X. R. Li, T. Kirubarajan, “Estimation with applications to tracking and navigation”, Wiley and Sons, 2001.

[36] L. A. Johnson, V. Krishnamurthy, “An improvement to the Interactive Multiple Model (IMM) Algorithm”, IEEE Transaction on Signal Processing, Volume 49, no.12, December 2001.

[37] M. Green, “How long does it take to stop? Methodological analysis of driver perception-brake times”, Transportation Human Factors, Volume 2, no. 3, pp. 195-216, September 2000.

[38] H. Himberg, Y. Motai, “Head orientation prediction: Delta quaternions versus quaternions”, IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, in press, 2009.

[39] Bar-Shalom, Yaakov, Li, Xiao-Rong, “Estimation and tracking: Principles, techniques and software”, YBS Publishing (1993 by Artec House, Inc.) 1998.

CHAPTER 2: Asynchronous Heterogeneous Sensor Fusion using Dead Reckoning and Kalman Filters

2.1. Introduction

Sensor fusion and tracking techniques have potential applications for the vehicle and the infrastructure as introduced in [1], something we can appreciate from the Intelligent Transport Systems (ITS) area [2]. The overall function of ITS is to improve decision making, often in real time, improving the operation of the entire transport system. This can range from systems with intelligent route planning implemented to avoid some specific type of traffic from certain areas [3], to registering the position of vehicle-borne sensors for infrastructure assessment [4], to systems designed to prevent collisions between the users [5]. This research could fall under the latter category.

A collision avoidance system is as good as its accuracy in warning the driver – either human or automated. An accurate system will minimize the number of false warning so the driver takes them seriously. Designing the architecture of this type of system involves using many sensors, and finding the right balance between the number of sensors implemented, type and their overall contributions to the system.

There are mainly two types of designs for a collision avoidance system: self-sufficient and interactive systems. Self-sufficient systems are those that can obtain enough information by themselves such as those in [6-8] where they placed sensors around the vehicle to maintain a safe following distance or to detect vehicles in the surroundings. Interactive systems are those that interact with the infrastructure or other vehicles to detect a dangerous scenario, such as researched in [9-11] where their systems send spatial information to nearby vehicles to judge the possibility of a collision in the future. While self-sufficient systems are limited to line of sight detection, the interactive systems account for scenarios farther ahead or even around corners or intersections by predicting and communicating the future estimated trajectories. With its benefits also comes its complexity. Estimating the future trajectory of a vehicle requires multiple sensors that need to be merged together and put through a set of prediction models.

Multi Sensor Data Fusion (MSDF) techniques are used in many diverse fields, although most of the literature addresses the fields of military target tracking or autonomous robotics [12]. MSDF is required to combine and process data. This has been traditionally performed by some form of Kalman [13] or Bayesian filters as shown in the examples above; however, in recent years, there has been a trend toward the use of soft techniques such as fuzzy logic and artificial neural networks [14]. Furthermore, there can be two ways of setting up a MSDF system: centralized or decentralized. While a centralized approach suffices for most common scenarios where the sensors are synchronous, a decentralized approach is more convenient when the sensors should be treated independently [15-19], as with asynchronous sensors.

In [20], the authors discuss one solution they have developed: the Optimal Asynchronous Track Fusion Algorithm (OATFA), which evolved from their earlier research on an Asynchronous/Synchronous Track Fusion (ASTF) [21]. They base their technique in the Interacting Multiple Model algorithm (IMM), but replaced the conventional Kalman filters with their OATFA (which contains several Kalman filters of its own). The OATFA treats each sensor separately, passing the output from each to a dedicated Kalman filter, departing from the idea that the best way to fuse data is to deliver it all to a central fusion engine. The chapter's IMM-OATFA results tend to show position estimation errors that are about half of those that the conventional IMM produces. However, as pointed out by the same authors in [22], all measurement data must be processed before the fusion algorithm is executed. With a similar approach as the technique described above, the authors of [23] create asynchronous holds, where, from a sequence of inputs sampled at a slow sampling rate, it generates a continuous signal that may be discretized at a high sampling rate. Despite the benefits of making the asynchronous system into a synchronous one by using these methods, restrictions arise where, if for some reason a sensor is delayed in providing its data or is off-line for a few cycles. The whole system needs to wait, as it is designed to work with certain data at specific rates.

In [24-26], the authors also look into problems of getting measurements from multiple sensors, but they focus on measurements being out-of-sequence and not on missing measurements. Therefore, while this is a very important topic on some scenarios, for the system that was used in this study, having all the sensors being processed locally, it will be assumed that all measurements are in the correct sequence, and there should not be a reason for some of them getting out-of-sequence.

Another method to fuse asynchronous sensors is discussed in [27]. In this chapter the authors synchronize the output of the sensors by estimating the data of the slower sensors for the time stamps where no data was available from them. Even though the method used to estimate the unavailable readings is very rudimentary (based only on the previous reading), this idea allows the system to run at the fastest frequency of its sensors. This difference, compared to the previously referenced papers, allows the system to make any corrections as soon as data are available, making its estimations more accurate in some scenarios.

The goal of this study is to develop a new method that allows the system to handle asynchronous sensors but run at the frequency of its fastest sensor, while also being able to handle homogeneous and heterogeneous data. A system that can handle all of these scenarios should be able to yield superior trajectory predictions than more conventional systems that have to run at the frequency of its slowest sensor.

The contribution of this chapter is a dead-reckoning (DR) system that runs at the frequency of its fastest sensor to update its prediction as soon as a change is detected. The difference from other DR implementations, subject to cumulative errors, is that our DRWDE continually updates the noise covariance matrices when any sensor remains offline. This constant modification of the truth weight of each measurements helps counteract the cumulative error of the DR when the measurements are estimated and not real. We then use the IMM to merge the predictions of the vehicle's future position. We will describe, later in this chapter, how we setup our systems, and how it compares to a more conventional implementation of a MSDF using KF and IMM.

This chapter is organized as follows. First there is a section that provides a quick overview of the KF and the IMM framework. Then it goes into describing the architecture of the system, the models defined, and how the dynamic noise covariance matrices are constructed. Then proceed to define a method to evaluate the results of this new system. Lastly a look at the experimental results and end with a conclusion.

2.2. Position Estimation Techniques

Position and attribute estimation are the process of taking the associated measurements and calculating the state of the target (vehicle). It is necessary to perform Target Motion Analysis (TMA) to calculate an estimate of the range and velocity. In this section, we review the methods for position/attribute estimation chosen for this research.

The Kalman Filter (KF) [28] was first proposed in the 1960s and it has been the most commonly used technique in target tracking and robot navigation ever since. The basic KF has been shown to be a form of Bayesian filter [29], which is an optimal estimator for linear Gaussian systems. From a series of noisy measurements, the KF is capable of estimating the state of the system in a two-step process: correct and then predict [30-32].

The elements of the state vector (x) are: the position, velocity, and acceleration of the vehicle; all available from the different sensors. Keep in mind that the position (x_v) and velocity (v_v) components of the state estimate have an x and y component to them (east-west and north-south directions), and the acceleration (a_v) has an n and t component to it (normal and tangential acceleration). So, the state vector matrix will be $X=(x_x, x_y, v_x, v_y, a_n, a_t)$.

The estimated error covariance (P) for the state estimate is based on the relationships between each of the elements to the others. The error covariance matrix is a dataset that specifies the estimated accuracy in the observation errors between all possible pairs of vertical levels.

The estimated error covariance (*P*) is used together with the Jacobian matrix of the measurement model (*H*) and the measurement noise covariance (*R*), together with the measurement noise (σ_m), to calculate the Kalman Gain (*K*). Once the Kalman Gain is calculated, the system looks at the measured data (*Z*) to identify the error of the predicted position and uses it to adjust the estimated error covariance (*P*).

The KF has a long history of accurately predicting future states of a moving object and has been applied to many different fields [33-36], including transportation, which is why it was chosen for this research.

Static (non-switching) algorithms have been around since the 1960s, though practical algorithms have only been available more recently, favored by the accessibility to more CPU speeds. The Interacting Multiple Models (IMM) framework was used in this system. It can calculate the probability of success of each model at every filter execution, providing combined solution for the vehicle behavior [37-39]. These probabilities are calculated according to a Markov model for the transition between maneuver states, as detailed in [40]. To implement the Markov model, it is assumed that at each execution time there is a probability p^{ij} that the vehicle will make a transition from model state i to state j.

In Johnson and Krishnamurthy's paper [41], they describe the IMM as a recursive suboptimal algorithm that consists of four core steps interacting with the KF, as illustrated in Figure 13.

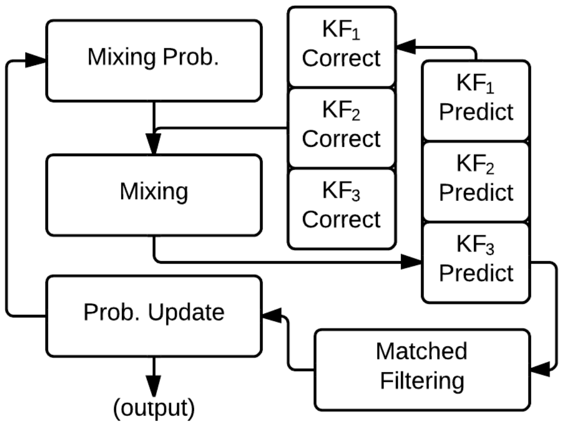


Figure 13: Flow-chart for the three KF in an IMM framework.

The four step IMM process starts with the calculation of the mixing probabilities, which uses the transition matrix and the previous iteration mode probabilities $\mu_{k-1}(i)$ to compute the normalized mixing probabilities $\mu_k(i|j)$. The mixing probabilities are re-computed each time the filter iterates before the mixing step.

The second step uses the mixing probabilities, which are used to compute new initial conditions for each of the *n* filters. The initial state vectors are formed as the weighted average of all of the filter state vectors from the previous iteration x_{k-1}^{oj}. The error covariance corresponding to each of the new state vectors is computed as the weighted average of the previous iteration error covariance's conditioned with the spread of the means $\left(P_{k-1}^{oj}\right)$.

The third step calculates mode matched filtering, using the $\hat{x}_{k-1}^{0j}$ and P_{k-1}^{0j}, the bank of *n* Kalman filters produce outputs $\hat{x}_k^j$, the covariance matrix P_k^j, and the probability density function $f_n(z_k)$ for each filter (*n*).

The fourth set of calculations begins once the new initial conditions are computed; the filtering step generates a new state vector, error covariance and likelihood function for each of the filter models. The probability update step then computes the individual filter probability $\left(\mu_k(j)\right)$ as the normalized product of the likelihood function and the corresponding mixing probability normalization factor.

The estimate and covariance combination is used for output purposes only: $\hat{x}_k = \sum_{j=1}^{n} \mu_k^j \cdot \hat{x}_k^j$; it is not part of the algorithm recursions.

2.3. Dead Reckoning with Dynamic Error (DRWDE) using Kalman Filters

For this system to react to change as soon as it occurs, it will need to run at the frequency of its fastest sensor (10Hz), and will need the flexibility to be able to use the available data from only the online sensors to predict the trajectory of the vehicle.

This system uses three different sensors: a Garmin 16HVS GPS receiver and Fugawi 3 GPS navigation software, an AutoEnginuity OBDII ScanTool (which obtains the velocity from the vehicle's internal system), and a Crossbow 3-axis accelerometer. This set of sensors offer data at different rates (asynchronous) and also at the same rates (synchronous); one measurement from two of the sensors overlap (homogeneous) but most of them do not (heterogeneous). The accelerometer measures normal and tangential acceleration every tenth of a second, the ScanTool measures velocity every 1 second, and the GPS measures position, velocity and heading every 1 second.

A problem with some of the existing research, as mentioned in section 2.1, is that sensors can unexpectedly go offline and not provide data when expected. The system in this study will need to handle this without slowing down the running frequency of the overall system and then wait for the sensor to come back online. This in turn means that the system can run at the frequency of its fastest sensor. If the system can continue to run and handle the missing data, it will allow for a quicker correction of the estimation if a change occurs in the spatial movement of the vehicle. For example, a GPS outage could occur when going through a tunnel, and waiting for the vehicle to exit the tunnel for the system to resume estimating the trajectory of the vehicle would not be a good solution. Or, the program used to interact with a sensor could freeze for a fraction of a second and not provide the measurement to the system. In this setup, because pre-recorded data is being used, the unavailable data also comes from the computer not reading and recording the data of the accelerometer fast enough, so even though the accelerometer works at 10Hz, the data recorded does not always exist for each tenth of a second.

2.3.1. System Architecture

For the overall trajectory estimation of a vehicle, modified KF will be used in running an Interacting Multiple Models (IMM) framework. The KF and IMM are commonly used in the trajectory estimation.

In this setup, the GPS sensor provides the location (s_x, s_y), the velocity (v) and the angle of direction (β) using north as the zero. Then the ScanTool sensor provides the velocity (v), and the accelerometer provides normal acceleration (a_n) and tangential acceleration (a_t).

When all the sensors are online, the general state equations can be defined as:

$$s(k) = s(k-1) + v(k-1)\cdot\Delta k + \frac{1}{2}a(k-1)\cdot\left(\Delta k\right)^2 + \frac{1}{6}j(k-1)\cdot\left(\Delta k\right)^3$$

$$v(k) = v(k-1) + a(k-1)\cdot\Delta k + \frac{1}{2}j(k-1)\cdot\left(\Delta k\right)^2$$

$$a(k) = a(k-1) + j(k-1)\cdot\Delta k$$

As shown in the above equations, the jerk j (acceleration change) in this study's equations are included as the factor responsible for the noise in the measurements; therefore, the jerk term is represented as the prediction noise(σ_p). Also, in the equations, Δk represents the time difference between the current iteration (k) and the previous iteration $(k-1)$ of the system.

From the above equations the different models used in this setup are defined as:

Constant Location Model (CL)

$$\begin{aligned} s(k) &= s(k-1) + \sigma_{p_s} \\ v(k) &= 0 \\ a(k) &= 0 \end{aligned} \qquad (1)$$

Constant Velocity Model (CV)

$$\begin{aligned} s(k) &= s(k-1) + v(k-1)\cdot\Delta k + \sigma_{p_s} \\ v(k) &= v(k-1) + \sigma_{p_v} \\ a(k) &= 0 \end{aligned} \qquad (2)$$

Constant Acceleration Model (CA)

$$
\begin{aligned}
s(k) &= s(k-1) + v(k-1)\cdot\Delta k + \frac{1}{2}a(k-1)\cdot\left(\Delta k\right)^2 + \sigma_{p_s} \\
v(k) &= v(k-1) + a(k-1)\cdot\Delta k + \sigma_{p_v} \\
a(k) &= a(k-1) + \sigma_{p_a}
\end{aligned}
\qquad (3)
$$

In the flow of this setup (Figure 14), when a sensor goes offline and the data needed for the models are not present, for example velocity, the missing data are derived from the data obtained by the remaining online sensors, making this estimation more accurate than only using the offline previous reading of the sensor to estimate what would be its current value. This is insufficient, however, as the longer a sensor remains offline the more noise is accumulated in the estimation of its value, which in turn affects the overall prediction of the future spatial location of the vehicle. To handle this properly, we have to dynamically modify the noise covariance matrices.

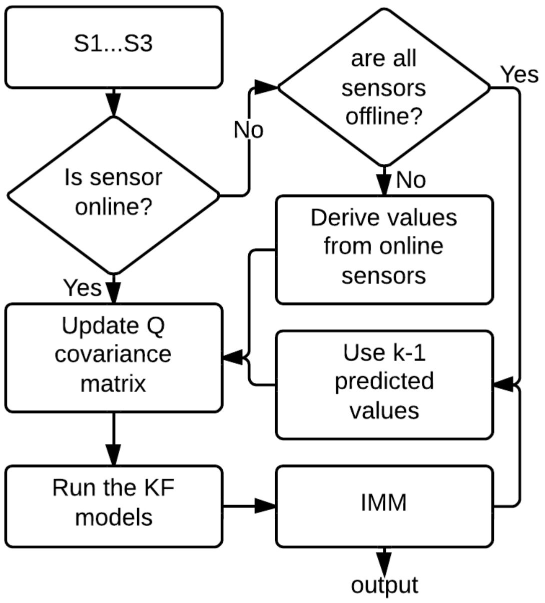


Figure 14: Flow-chart of our DRWDE system.

2.3.2. The Q Matrix in the Kalman Filter

The process noise covariance (Q) of the KF is defined based on the estimated prediction noise$^{(\sigma_p)}$. A simple approach to estimate this is by using an extensive dataset of common scenarios. For this system, because this research wanted to be able to handle sensors going offline at any given time and for any given period of time, an innovative method was devised that makes the Q matrix dynamic, allowing the noise to vary depending on the number of iterations the different variables go through without getting an actual measurement from the corresponding sensor.

But, before going into the details of the dynamic process noise covariance matrix, it is useful to understand theoretically the errors introduced into the system when one or more sensors are offline and how to improve the estimation.

2.3.2.1. Mathematical Framework for Improvement

A discrete and dynamic lineal system can be generally expressed as shown below, where k is the current instance and $k+1$ is the future instant for which the data are being estimated.

$$\begin{aligned} x_{k+1} &= \phi_k \cdot x_k + \psi_k \cdot u_k + w_k \\ w'_{k+1} &= H_{k+1} \cdot x_{k+1} + v_{k+1} \end{aligned}$$

Given the intermediate data for the instant t_k between the instances where all sensors are online (t_i and t_{i+1}), it is possible to make a prediction for the instant t_{k+1} posteriori to t_k, which will also be posteriori to t_i, which will most probably result in a better prediction than if using the instant t_i for an estimation farther ahead in time. Two approaches handled the missing data when sensors are offline:

The first option is to fill in the missing measurements with those of $\hat{x}_{k+1}$, which is the prediction to the intermediate instant. The risk for the minimal quadratic error is $\left(\hat{x}-E(x)\right)^t \cdot M\left(\hat{x}-E(x)\right)+trace\left[MP\right]$ where *M* is the defined positive matrix of the quadratic error, and $P=E\left\langle\left(\hat{x}-E(x)\right)\cdot\left(\hat{x}-E(x)\right)^t\right\rangle$ [43], with the corresponding reduction of the actual measurements when sensors are online.

The second option is to calculate, with the current data obtained from the online sensors, the noise errors for the given small time interval, and obtain a better approximation of the missing measurements, with the goal of obtaining a better *Q* covariance matrix.

In the first option, the error will generally be greater, the greater the interval $\hat{x}$ and $E(x)$. In the second option there may not always be a relationship that will yield a good estimation, but experimental runs can help evaluate this approach to determine if the estimation is indeed better.

As proven in A.3, a smaller trace of the *Q* matrix would suppose a general improvement of the covariance matrix of the process, and, therefore, the resulting estimation. However, if a sufficiently general condition is required, then there is a need to study the matrix $E\left[m_{k+1}m_{k+1}^t\right]$ for each specific case.

To approximate the unknown magnitudes, if $x=\left(x_0x_1...x_n\right)^t$ verifies that $x_{l+1}=\dot{x}_l$ $\forall l=0,1,...,n-1$, and x_n is the function we have for known measurements in the intermediate instances, it is possible to approximate any x_p for $p=0,1,...,n-1$ through a Taylor polynomial as shown in A.4.

With the new data obtained from the online sensors, the process can be repeated for the next intermediate instances $t_{k+1}, t_{k+2}, \ldots$; which, in general, the error will continue to increase as the time gap increases. The exact value of the errors will be unknown in general, so this research will have to be bounded through statistical estimations; even though, in reality, the actual implementation, and not the theoretical validation of the formulas, will be the one to determine if there is an improvement in the estimations. For this, it must be taken into account that, due to the cumulative error accumulated with each iteration, the excessive number of iterations will be counterproductive, and will make the estimations worse.

In the case that the function of the more frequent known measurements x_q is not x_0 or x_n, it will suffice to consider on one hand $(x_0 \ldots x_q)$, and on the other hand $(y_0 \ldots y_m) = (x_0 \ldots x_m)$, and proceed with each group accordingly. If there were more functions with known measurement data, in general, the remaining would be estimated using the closest one, or one of the closest ones.

2.3.2.2. Dynamic Process Noise Covariance (Q)

In the case when all the sensors are available, the formulas for the *CA* models will depend on the location, velocity and acceleration measurements in a given instant, and also will depend on the prediction noise σ_p. In this case σ_p is based on the jerk (j), which will have a variable and unknown value. Based on the *Lagrange form of the remainder* of Taylor's formula, there is a value for j which will yield the exact measurements. Therefore, to set an upper bound of the expected value (E), it suffices to identify an upper bound for j, and calculate the corresponding integrals to obtain each

E. But, because in a real time execution of the system all the values of j are not known ahead of time, this research made it a moving range so the system can dynamically tune itself.

In summary, to determine $Q = E[\sigma_p \sigma_p^T]$, this research starts by defining j_k (acceleration change) as the least upper bound (supremum) of the dataset collected so far, i.e. $\max\{|j_{t_k}|,|j_{n_k}|,|j_{t_{k-1}}|,|j_{n_{k-1}}|\cdots|j_{t_{k_0}}|,|j_{n_{k_0}}|\}$.

If the state vector defined in section 2.2 and the Kalman models defined in section 2.3.1 is used, and if j_n is to the right of j_t, and for the *CA* model (3), $x(k) = F(k)\cdot x(k-1) + \sigma_p$ has:

$$x(k) = \begin{bmatrix} 1 & \Delta k & \frac{1}{2}(\Delta k)^2 \\ 0 & 1 & \Delta k \\ 0 & 0 & 1 \end{bmatrix} \cdot x(k-1) + \begin{bmatrix} \frac{1}{6} j(\Delta k)^3 \\ \frac{1}{2} j(\Delta k)^2 \\ j(\Delta k) \end{bmatrix} \qquad (4)$$

Furthermore, in this system it will also take into account the error in the estimations for location, velocity and acceleration when the sensor providing the corresponding value is offline, and consider for how long it has been offline.

Now, given $M_{k'}(x)$ as the total measurement error of a variable x such that in the step when all sensors are online $m = 0$, and in the following m step(s) only the accelerometer sensor is online. Because sensors can go offline independently of each other, a different m is needed to identify each sensor: m_1 for the GPS sensor, m_2 for the ScanTool sensor, and m_3 for the accelerometer.

Therefore, this research can now define *Q* as shown below.

$$Q_{CA} = \begin{bmatrix} M(s)^2 & M(sv) & M(sa) \\ M(vs) & M(v)^2 & M(va) \\ M(as) & M(av) & M(a)^2 \end{bmatrix} \qquad (5)$$

So it can now derive each of the process error elements in the Q matrix. For the position elements (*x/y*) it is obtained that $E\left[M^2\left(s\right)\right] \le \frac{(\Delta k)^6}{36}\sum_{i=0}^{m_1-1} j_{k-i}^2$, with the details shown in A.5. Then, using a similar approach, it was found that for the velocity elements (*x/y*) $E\left[M^2\left(v\right)\right] \le \frac{(\Delta k)^4}{4}\sum_{i=0}^{m_2-1} j_{k-i}$, and, finally, for the acceleration elements (*n/t*) it was derived $E\left[M^2\left(a\right)\right] \le (\Delta k)^2\sum_{i=0}^{m_3-1} j_{k-i}^2$. Also, for the non-zero elements outside of the diagonal, it was calculated that $E\left[M\left(s\cdot v\right)\right] = E\left[M\left(v\cdot s\right)\right] \le \frac{(\Delta k)^5}{12}\sum_{i=0}^{m_{1\|2}-1} j_{k-i}^2$.

For a given tangential or normal acceleration, the locations and velocities in the axis directions can be any; therefore, the location and velocity variables are independent from the value of the tangential or normal accelerations. And, similarly, the tangential and normal accelerations are independent of each other. Therefore, the expected value of those errors are zero; and the final Q matrix that will dynamically increase the corresponding measurement error proportionally to how long some sensors (m_i) have been offline (Δk) is defined below.

$$Q_{CA} = \begin{bmatrix} \frac{(\Delta k)^6}{36}\sum_{i=0}^{m_1-1} j_{k-i}^2 & \frac{(\Delta k)^5}{12}\sum_{i=0}^{m_{1\|2}-1} j_{k-i}^2 & 0 \\ \frac{(\Delta k)^5}{12}\sum_{i=0}^{m_{1\|2}-1} j_{k-i}^2 & \frac{(\Delta k)^4}{4}\sum_{i=0}^{m_2-1} j_{k-i} & 0 \\ 0 & 0 & (\Delta k)^2\sum_{i=0}^{m_3-1} j_{k-i}^2 \end{bmatrix} \qquad (6)$$

Using a similar approach as shown above, this research can derive the dynamic Q matrix for the CV model as shown below.

$$Q_{CV} = \begin{bmatrix} \frac{(\Delta k)^4}{4}\sum_{i=0}^{m_1-1} a_{k-i}^2 & \frac{(\Delta k)^3}{2}\sum_{i=0}^{m_{1|2}-1} a_{k-i}^2 \\ \frac{(\Delta k)^3}{2}\sum_{i=0}^{m_{1|2}-1} a_{k-i}^2 & (\Delta k)^2 \sum_{i=0}^{m_2-1} a_{k-i}^2 \end{bmatrix} \qquad (7)$$

And for the CL model $Q_{CL} = \left[(\Delta k)^2 \sum_{i=0}^{m_1-1} v_{k-}^2 \right]$

These Q matrices will be used in the KF prediction step to estimate the error covariance for each of the models. And, as shown in the Q matrices above, the moment a sensor comes back online ($m_i = 0$), the corresponding element in the dynamic Q matrix can be reset to its minimum value.

2.4. Evaluation Criteria

To verify the improvements of using the DRWDE we will implement and compare the results of the following setups:

- Synchronous sensors using a common KF+IMM implementation (GPS @1Hz, ScanTool @1Hz, and Accelerometer @1Hz)
- Asynchronous sensors using our dynamic DRWDE implementation (GPS @1Hz, ScanTool @1Hz, and Accelerometer @10Hz)

The first setup is to get the IMM working at 1Hz, which will only run when all sensors are online; therefore, not really using the dynamic part of the Q matrix.

The DRWDE setup is to increase the frequency of the system to 10Hz to try to take advantage of all the readings from the accelerometer, and try to correct the predictions sooner, instead of having to wait for the all sensors to come back online, as in the first setup. This second setup uses the dynamic Q matrix technique described in section 2.3.2 to account for the error in the estimation of the data when some sensors are offline.

Using the above two setups helps to track improvements to the overall trajectory prediction when the frequency of the system increases along with the proper handling of the accumulated error in the predictions. If the DRWDE is flexible enough to handle all the different synchronous and asynchronous, homogeneous and heterogeneous data from the sensors in use, improvements should be seen on the predicted future locations; and the system should be able to detect and correct a spatial change in the vehicle much sooner than when the system is forced to run at the speed of its slowest sensor.

The evaluation criteria will be based on comparing the actual prediction errors for both the DRWDE and IMM 1Hz systems against the true location data obtained from the GPS receiver. Both systems will be run through the same trajectory, and the results looked at in several different ways. First, this research will look at the average prediction error for whole trajectory, but then also separate the trajectory into straight lines, smooth curves and sharp curves, to better evaluate both systems in the different scenarios. This research will also select one specific smooth curve and one specific sharp curve, and it will look at those results in greater detail, calculating Root Mean Square (*RMS*) values using the actual and predicted position *S* of both systems.

$$RMS = \sqrt{\frac{\sum_{k}^{k'} \left(S_k - S_{k-3}\right)^2}{\left(k' - k\right)}}$$

The goal is to determine quantitatively the improvements in prediction error of the DRWDE system in the different types of trajectory.

2.5. Experimental Performance of the DRWDE System

2.5.1. Dataset Characteristics

The dataset consists of measurements from the three sensors while driving a vehicle for over one hour. The trajectory followed is shown in Figure 15, where the vehicle followed the route marked in red.

For this experiment, the GPS sensor takes measurements of the current geographical coordinates in degrees, heading in degrees, and velocity in miles per hour every 1 second. These measurements were converted to meters, radians, and meters per second respectively.

The ScanTool reads the measurements of the velocity determined by sensors coupled to the wheels of the vehicle in miles per hour every 1 second. This measurement is more accurate than the one obtained from the GPS, so it is used instead of the one from the GPS (except when it is not available).

The last sensor used in this experiment is an accelerometer, which takes measurements of the normal and tangential accelerations in volts every 0.1 seconds. Using a calibration formula provided by the manufacturer of the sensor, the conversion is units to meters per second squared.

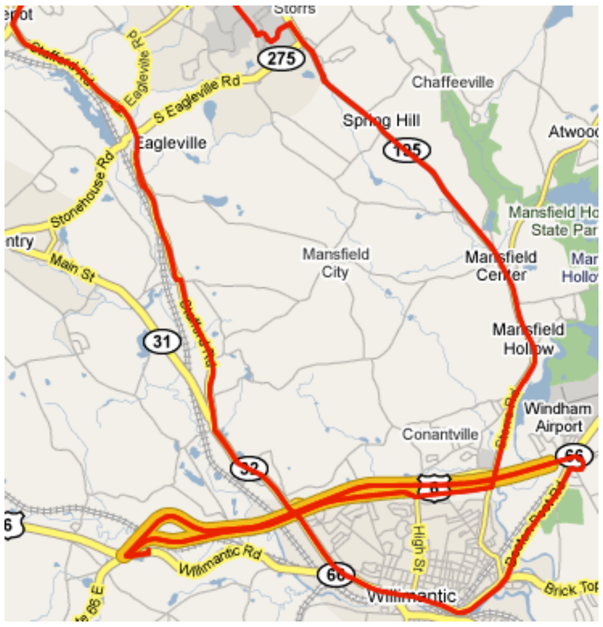


Figure 15: Map of whole trajectory in Mansfield City, CT (Google maps).

To be able to create a useful dataset of the data recorded from the trajectory shown in Figure 15, this research had to create scripts to map the values from the log file of each sensor to each other, using the timestamp as the common reference between them. In the end a dataset was created with all the desired measurements in columns, with all available readings in a row for each timestamp. Because only the accelerometer works at 10Hz, many of the rows only contain acceleration measurements, and this is where the system comes into action and takes advantage of these extra measurements. Table 4 shows the average and standard deviations of the data used, to take a general look at the characteristics of the dataset worked with.

For this experiment the focus was on predicting a trajectory when the vehicle is going through curves, which are the more problematic areas. To be able to evaluate this better, the dataset of the whole trajectory was classified into straight lines, smooth curves, and sharp curves. To determine whether a set of consecutive points in the trajectory was a curve or a straight line the change in the heading after a period of 2 seconds was observed; if more than 2°**,** then it was defined as a curve. And, to determine if the curve was a sharp one, the change had to be greater than 10°, otherwise it was defined as a smooth curve.

Table 4: Representative Data Set

	Distance (m)	*Velocity* (m/s)	*Acc.norm* (m/s^2)	*Acc.tang* (m/s^2)
Whole	16.34	15.20	-0.44	0.69
Trajectory	±6.97	±5.86	±1.10	±0.58
Smooth	19.04	17.66	-1.69	0.04
Curves	±5.58	±4.97	±1.72	±0.53
Sharp	10.59	9.36	0.37	1.19
Curves	±6.84	±5.78	±1.34	±0.58

Values represent median ± standard deviation of all data points used.

Looking at Table 4 it can be seen that the dataset used for this experiment agrees with how a vehicle would be driven under normal conditions. For example, the standard deviations are not very different from each other for the distance and velocity measured by the sensors, which is expected, as the values do not change much from one point to the next for an average vehicle driving on normal roads. The average for distance and velocity are smaller for the smooth curves than for the sharp curves, which means that the vehicle's speed is more constant through the smooth curves than the sharp curves. The change in movement for sharp curves agrees with how a vehicle would behave in such a scenario, as it will usually have to slow down considerably while turning and then accelerate again as the drivers get a handle on the curve.

Since the main problem with trajectory estimation is during curves based on research reviewed in section 2.1, this research selected a specific curved scenario from Figure 15 and use that dataset to evaluate the DRWDE system and its performance.

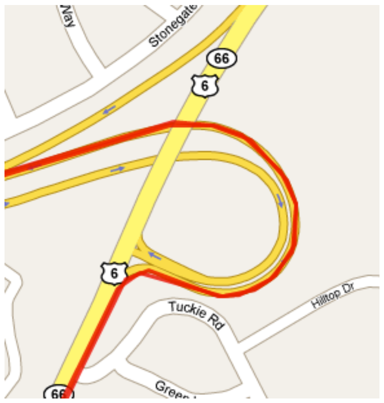


Figure 16: Map of selected curve for testing (Google maps).

The section of the trajectory shown in Figure 16 was selected because it has a sharp curve and then a smooth constant curve, which should be a good scenario to test if the system can correct its prediction when the vehicle enters the curve, and maintain it through the whole curve. Sharper curves allow our dynamic system to be tested properly as the curve ends up being very short and does not allow a slower system to estimate a trajectory during the actual turn if it only lasts a few of seconds. The "Selected Smooth Curve" refers to the longer curve in Figure 16 (~30 seconds of data), and the "Selected Sharp Curve" represents the small curve (bottom left) shown in Figure 16 as well (~10 seconds of data).

The DRWDE setup for this experiment, as explained in section 2.3, runs at the frequency of its fastest sensor (10Hz), and uses the dynamic matrices accounting for the accumulated noise of the missing measurements. Also, as mentioned in section 2.4, this data will be running through a common IMM implementation (Synchronous Sensors) to be able to compare results to the DRWDE setup.

Since the common IMM can only run at the frequency of its slowest sensor, this research defined Δk to be 1 second (1Hz), and, because all sensors are available during each iteration of the system, this setup does not utilize the dynamic portion of the Q matrix defined in section 2.3.2.

Also, to properly compare this run to the 10Hz run, it cannot be assumed the vehicle would move in a straight line between each second, so a 10 intermediate points between each second based on the dynamics of the vehicle was defined. This allows to more accurately compare both runs visually.

2.5.2. Evaluation of the Prediction Error

Following the evaluation criteria defined in section 2.4, the data recorded from the trajectory shown in Figure 15 was executed through both systems. The results for the Overall Trajectory, All Smooth and Sharp Curves, and the Selected Smooth and Sharp Curves were then recorded in Table 5.

Table 5: Average Prediction Error (3 s ahead)

	DRWDE	*IMM 1Hz*
Whole Trajectory	2.719±2.030	3.044±1.800
All Smooth Curves	2.811±1.925	2.972±1.737

All Sharp Curves	3.236±2.844	4.456±3.307
Selected Smooth Curve	3.051±1.173	3.212±1.205
Selected Sharp Curve	2.277±2.388	4.090±2.241

Values represent median prediction error in meters ± standard deviation of all data points used.

Table 5 shows the average prediction errors for both the DRWDE and the IMM 1Hz run for broader scenarios as well as for our selected curves. If the results for the Whole Trajectory were observed, only a negligible improvement was seen, as expected, since the number of sharp curves in the whole trajectory is very small. Similarly, there is almost no improvement if All Smooth Curves were observed in the trajectory when compared to the IMM 1Hz. But, since the DRWDE was created to react quickly to changes, it was observed that when taken into account All Sharp Curves, improvements to the 3 second ahead estimation were seen that are considerably greater for the DRWDE system (3.2m vs. 4.5m).

If now the focus is on the Selected Smooth and Sharp Curves, the result of the IMM run at 1Hz is shown in Figure 17. The red dotted line shows the predicted location every second (red dots) and the intermediate points derived in between each second (dotted line) to simply show visually what may be happening in between each second.

Now, for the DRWDE run, Δk was defined to be 0.1 seconds, which is the period of its fastest sensor (10Hz). Since only the accelerometer runs at 10Hz, there will be many system iterations where the other sensors will be offline, and this is where the dynamic Q variance introduced in section 2.3.2 come into play. The result of the DRWDE run at 10Hz is also shown in Figure 17, as the blue solid line.

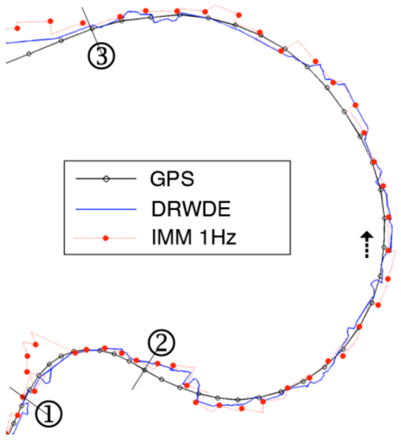


Figure 17: Comparison between actual path (GPS) and predicted paths by both systems (DRWDE 10Hz and IMM 1Hz) for the selected curves. Sharp curve between (1) and (2), Smooth curve between (2) and (3). Direction of movement shown by arrow.

Figure 17 displays the actual trajectory of the vehicle represented by the GPS line, and then the predicted locations 3 seconds earlier in time by both the IMM 1Hz run and the DRWDE 10Hz run (prediction performance is shown later in Figure 19). It can be observed that both the 1Hz and the 10Hz runs behave somewhat similarly during the smooth curve; this is also represented quantitatively in Table 5.

The average error in the predicted locations during the Selected Smooth Curve is only slightly better for the DRWDE (3.0m vs. 3.2m). The benefits are clearly seen in the Selected Sharp Curve, where the average error is much lower for the DRWDE (2.3m vs. 4.1m). Looking at Figure 17, it can be seen that, as the vehicle enters the sharp curve (bottom left), the slower system (red dotted line) is estimating its location to be in more of a straight line, as the vehicle is travelling in a straight line before taking the exit ramp (see Figure 16). It can even be seen that there are three red dots (each dot represents 1 second) before the system realizes that the vehicle is turning and can adjust its 3-second ahead prediction accordingly. Looking at the blue line representing the DRWDE run, it can be seen that its line is a lot closer to where the vehicle actually moves through 3 seconds later in time. The DRWDE 10Hz system is able to react and correct its future prediction much quicker, using its dynamic covariance matrices to take into account how long a measurement has not been corrected by an actual sensor. As shown in Table 5, in the Selected Sharp Curve a difference of over 1.5 meters in accuracy between the two systems can be seen, which is a significant improvement.

2.5.3. RMS Error Distribution

A simple visualization of the error distribution for the "Whole Trajectory", "All Smooth Curves" and "All Sharp Curves" prediction errors is shown in Figure 18. The charts have the individual prediction errors categorized into groups, where group "0-1" in the x axis contains all the prediction errors that fall between 0 and 1 meter, and the y axis shows how frequently the errors fall in each of the groups.

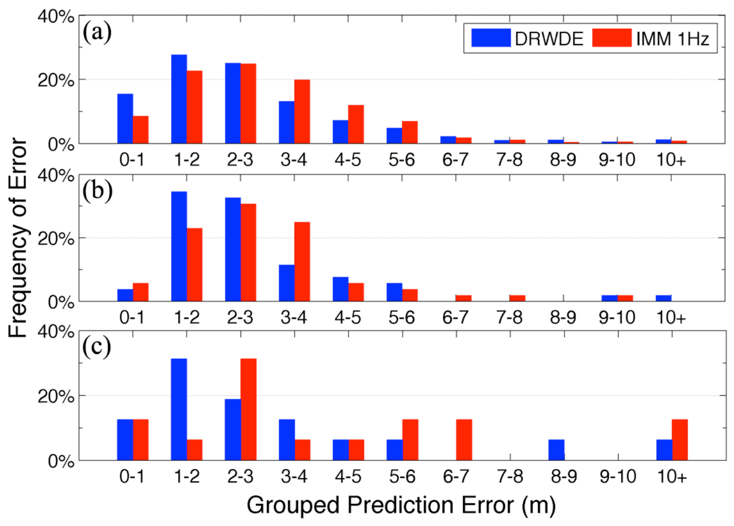


Figure 18: Frequency of each system's 3 s ahead prediction error. (a) uses data from the whole trajectory; (b) represents data for all smooth curves; (c) represents data for all sharp curves.

Looking at the histograms in Figure 18 it can be observed how the DRWDE system tends to be more often in the first groups, which represent less prediction error. The taller the bars on a given group means that more often the error falls in that error group; therefore, the taller the blue bars on the smaller groups, the more accurate the system is.

In Figure 18-a it can only be seen that the DRWDE outperform the IMM 1Hz by a small amount when looking at the Overall Trajectory, and a larger difference when looking at the results for All Smooth Curves in Figure 18-b. However, when All Sharp Curves in Figure 18-c is observed, a more distinct difference in the prediction accuracy between the DRWDE and the IMM 1Hz can be seen. To analyze the results for Selected Smooth and Sharp curves specifically, as shown in Figure 17, Figure 19 was created.

Figure 19-a represents the error between the estimated future distance the vehicle will travel in the following 3seconds, and the actual distance travelled as recorded by the GPS sensor for the Selected Smooth and Sharp Curves. Time zero in the figure is set a few seconds before the vehicle enters the sharp curve shown in Figure 16. Right at the beginning of the sharp curve, the error in the estimation is quite large for both systems, and that is due because the vehicle is moving in a somewhat straight path, so the estimated future position assumes the vehicle will continue to move in the same direction. As soon as the vehicle enters the sharp curve, the first system to detect this change in direction is the DRWDE 10Hz, as expected, as it can detect this change using the accelerometer, while the GPS is still offline. Once the GPS sensor is back online, the 1Hz system can also detect this change and can correct its prediction. The upward trend of the lines in Figure 19 simply indicates that the vehicle is slowly increasing its velocity and is covering more distance in the same period of time (3-seconds). Only dots at each full second are shown to be able to compare between the two systems.

For a different view of the kind of errors the DRWDE 10Hz system has, Figure 19-b was created, which shows the root mean squared (RMS) error between the estimated future location (3-seconds later) and the actual location measured by the GPS.

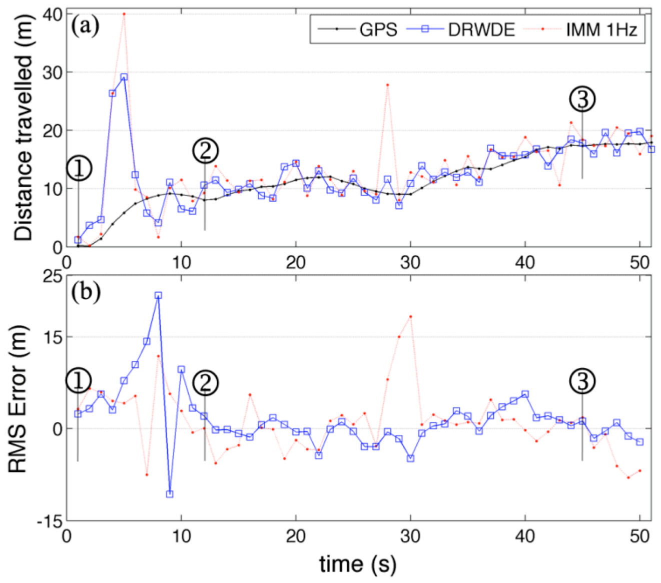


Figure 19: Position error for 3 s ahead predictions during our selected curves as shown in Figure 17. (a) displays actual vs. predicted distances travelled per s, and (b) displays RMS error in each prediction.

Looking at Figure 18 and Table 5, it can be concluded that the DRWDE setup really stands out when abrupt changes occur in the movement of the vehicle, and, only then, the fast reaction time shows substantial improvements in the prediction. These abrupt changes could be a sharp curve, as illustrated in this experiment, but they could also be, for example, a quick maneuver of the vehicle to correct its direction to avoid a collision.

2.5.4. Computational Complexity

For completeness, it was also looked into how much of an extra load it is to run the DRWDE system with the dynamic noise matrices compared to the simpler approach of the 1Hz IMM system. Because the dataset had already been recorded, only the processing time of the system itself was measured. If taking into account the processing time of the sensors, especially the accelerometer, the CPU times would be even larger.

Table 6: Computational Complexity

	tic/toc (s)	*cputime (s)*	*Data points*	*Avg. load*
DRWDE	389.72	382.88	17,525	1.31
IMM 1Hz	48.53	47.29	2,187	1.30

Measurements taken on system running through the whole trajectory.

Table 6 shows different Matlab commands used to measure CPU times for each of the systems. All two commands (tic/toc and cputime) measure actual CPU time used by the Matlab code, but this research is showing both to get a better idea on the accuracy of the measurements. The column tic/toc represents actual start/stop time of execution, while cputime displays the actual CPU time in that was spent executing the code. The system was run on a machine with a dual core 2.0 GHz CPU.

As expected, Table 6 shows that the DRWDE 10Hz system requires a lot more processing power than the simpler IMM 1Hz system. This is as expected, since the DRWDE system has to handle close to 10 times more data points, and, therefore, yields much longer CPU times. On the same token, if looking at the last column, it can be observed that the average load times for every record processed is almost the same for both systems, which shows that the extra computational requirements of the DRWDE's dynamic error processing and measurement noise matrices are not significant at all.

Despite the large amount of CPU time the DEWDE system takes, consider that it is processing about half an hour's worth of data in a little over six minutes. A problem is not foreseen with the same computer handling the DRWDE system plus the extra load added by the operating system and also by the sensors themselves as data is being recorded during a live run.

2.6. Conclusions

The key contribution of this research's DRWDE system is the introduction of dynamic noise covariance matrix merged together by an Interacting Multiple Models (IMM). The longer a sensor remains offline, the less accurate the overall prediction is, so the dynamic Q presented in section 2.3.2 tells the system how true the value being used is.

This DRWDE setup only had three sensors, of which only one of them was running at 10Hz. The accelerometer is very sensitive to changes in the road, including road bumps; so, relying on this sensor to estimate the values of the other sensors when they were offline had its challenges. However, looking at section 5, it can be concluded that by properly handling the accumulating error for missing measurements, running the system at a higher frequency can yield better predictions, especially when abrupt changes occur. The key here was to be able to accurately account for the accumulating error when sensors go offline and remain offline for an unknown amount of time.

An improvement to this system could be to add more sensors running at high frequencies, for redundancy and to minimize the times sensors are offline. Also, this system could be combined with the previous research [42], where the predicted location is compared against geographical information system (GIS) to reduce false predictions.

2.7. References

[1] J. C. Miles, A. J. Walker, "The potential application of artificial intelligence in transport," *Transactions on Intelligent Transport Systems, IEEE*, vol. 153, no. 3, pp. 183-198, September 2006.

[2] F. Wang, A. Broggi, C. C. White, "The road to transactions on intelligent transportation systems: A eecade's success [Transactions on ITS]," *Transactions on Intelligent Transportation Systems Magazine, IEEE*, vol. 1, no. 4, pp. 29-32, Winter 2009.

[3] V. Di Lecce, A. Amato, "Route planning and user interface for an advanced intelligent transport system," *Intelligent Transport Systems, IET* , vol. 5, no. 3, pp. 149-158, September 2011.

[4] X. Xu, T. Xia, A. Venkatachalam, D. Huston "The development of a high speed ultrawideband ground penetrating radar for rebar detection," *Journal of Engineering Mechanics*, vol. 139, no. 3, pp. 272-285, March 2013.

[5] T. Taleb, A. Benslimane, K. Ben Letaief, "Toward an effective risk-conscious and collaborative vehicular collision avoidance system," *Transactions on Vehicular Technology*, IEEE, vol. 59, no. 3, pp.1474-1486, March 2010.

[6] F. Jimenez, J. Eugenio Naranjo, "Improving the obstacle detection and identification algorithms of a laserscanner-based collision avoidance system," *Transportation Research Part C-Emerging Technologies*, vol. 19, no. 4, pp. 658-672, August 2011.

[7] R. Toledo-Moreo, M.A. Zamora-Izquierdo, "Collision avoidance support in roads with lateral and longitudinal maneuver prediction by fusing GPS/IMU and digital

maps," *Transportation Research Part C-Emerging Technologies*, vol. 18, no. 4, pp. 611-625, August 2010.

[8] C. Hakyoung; L. Ojeda; J. Borenstein, "Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope," *Transactions on Robotics and Automation, IEEE*, vol.17, no.1, pp.80-84, February 2001.

[9] R. Toledo-Moreo, M. Pinzolas-Prado, J. Manuel Cano-Izquierdo, "Maneuver prediction for road vehicles based on a neuro-fuzzy architecture with a low-cost navigation unit," *Transactions on Intelligent Transportation Systems, IEEE*, vol. 11, no. 2, pp. 498-504, June 2010.

[10] J. Ueki, J.M., Y. Nakamura, Y. Horii, H. Okada, "Development of vehicular-collision avoidance support system by inter-vehicle communications," *59th Vehicular Technology Conference, IEEE*, vol. 5, pp. 2940-2945, May 2004.

[11] H. D. Weerasinghe, R. Tackett, H. Fu, "Verifying position and velocity for vehicular ad-hoc networks," *Security and Communication Networks*, vol. 4, no. 7, pp. 785-791, July 2011.

[12] R. C. Luo, C. C. Y., K. L. Su, "Multisensor fusion and integration – Approaches, applications, and future research directions," *Sensors, IEEE*, vol. 2, pp. 107-119, April 2002.

[13] J. B. G. a. C. J. Harris, "Some remarks on Kalman filters for the multisensor fusion," *Elsevier Information Fusion*, vol. 3, pp. 191-201, 2002.

[14] R. R. Murphy, "Sensor fusion," Handbook of Brain Theory and Neural Networks, Bradford Book, 2003.

[15] S. C. Felter, “An overview of decentralized Kalman filter techniques,” *Proceedings of the 1990 Southern Tier Technical Conference, IEEE*, pp. 79-87, April 1990.

[16] M. Hua, T. Bailey, P. Thompson et al., “Decentralised solutions to the cooperative multi-platform navigation problem,” *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 47, no. 2, April 2011.

[17] E. M. Nebot, M. Bozorg, H. F. Durrant-Whyte, “Decentralized architecture for asynchronous sensors,” *Autonomous Robots*, vol. 6, no. 2, pp. 147-164, April 1999.

[18] S. I. Roumeliotis, G. A. Bekey, “Distributed multirobot localization,” *Transactions on Robotics and Automation, IEEE*, vol. 18, no. 5, pp. 781-795, October 2002.

[19] H. M. Wang, Q. Yin, X. Xia, “Fast Kalman equalization for time-frequency asynchronous cooperative relay networks with distributed space-time codes,” *Transactions on Vehicular Technology, IEEE*, vol. 59, no. 9, pp. 4651-4658, November 2010.

[20] G. A. Watson, T. R. R., A. T. Alouani, “An IMM architecture for track fusion,” *Signal Processing, Sensor Fusion, and Target Recognition (Proceedings of SPIE 4052)*, vol. IX, pp. 2-13, 2000.

[21] A. T. Alouani, T. R. R., “On optimal asynchronous track fusion,” *Proceedings of 1st Australian Symposium on Data Fusion, IEEE*, November 1996.

[22] G. A. Watson, T. R. R., A. T. Alouani, “Optimal track fusion with feedback for multiple asynchronous measurements,” *Proceedings of SPIE Acquisition, Tracking and Pointing XIV*, April 2000.

[23] L. Armesto, G. Ippoliti, S. Longhi et al., "Probabilistic self-localization and mapping - An asynchronous multirate approach," *Transactions on Robotics & Automation Magazine, IEEE*, vol. 15, no. 2, pp. 77-88, June 2008.

[24] Y. Bar-Shalom, H. M. Chen, "IMM estimator with out-of-sequence measurements," *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 41, no. 1, pp. 90-98, January 2005.

[25] X. Shen, Y. Zhu, E. Song et al., "Optimal centralized update with multiple local out-of-sequence measurements," *Transactions on Signal Processing, IEEE*, vol. 57, no. 4, pp. 1551-1562, April 2009.

[26] X. Shen, E. Song, Y. Zhu et al., "Globally optimal distributed Kalman fusion with local out-of-sequence-measurement updates," *Transactions on Automatic Control, IEEE*, vol. 54, no. 8, pp. 1928-1934, August 2009.

[27] W. Jiangxin, S. Y. Chao, A. M. Agogiono, "Validation and fusion of longitudinal positioning sensors in AVCS," *Proceedings of the 1999 American Control Conference*, vol. 3, pp. 2178-2182, 1999.

[28] R. Kalman, "A new approach to linear filtering and prediction problems," *Transaction ASME*, vol. 82, pp. 34-45, 1960.

[29] Y. Ho, R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *Transactions on Automatic Control, IEEE*, vol. 9, no. 4, pp. 333-339, October 1964.

[30] G. Welch, G.B., "An introduction to the Kalman filter," SIGGRAPH Course notes, 2001.

[31] Y. Bar-Shalom, X. R. L., T. Kirubarajan, “Estimation with applications to tracking and navigation,” Wiley and Sons, 2001.

[32] Y. Bar-Shalom, X. R. L., “Estimation and tracking: Principles, techniques and software,” Artech House, 1993.

[33] B. Mokaberi; A. A. G. Requicha, “Drift compensation for automatic nanomanipulation with scanning probe microscopes,” *Transactions on Automation Science and Engineering, IEEE*, vol. 3, no. 3, pp. 199-207, July 2006.

[34] P. N. Pathirana, A. V. Savkin, S. Jha, “Location estimation and trajectory prediction for cellular networks with mobile base stations,” *Transactions on Vehicular Technology, IEEE*, vol. 53, no. 6, pp. 1903-1913, November 2004.

[35] C. F. Graetzel, B. J. Nelson, S. N. Fry, “A dynamic region-of-interest vision tracking system applied to the real-time wing kinematic analysis of tethered drosophila,” *Transactions on Automation Science and Engineering, IEEE*, vol. 7, no. 3, pp. 463-473, July 2010.

[36] S. M. Bhandarkar, L. Xingzhi, R. F. Daniels, E. W. Tollner, “Automated planning and optimization of lumber production using machine vision and computed tomography,” *Transactions on Automation Science and Engineering, IEEE*, vol. 5, no. 4, pp. 677-695, October 2008.

[37] L. Hong, “Multirate interacting multiple model filtering for target tracking using multirate models,” *Transactions on Automatic Control, IEEE*, vol. 44, no. 7, pp. 1326-1340, July 1999.

[38] E. Mazor, "Interacting multiple model methods in target tracking: A survey," *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 34, no. 1, pp. 103-123, January 1998.

[39] B. S. Chen, C. Y. Yang, F. K. Liao, "Mobile location estimator in a rough wireless environment using extended Kalman-based IMM and data fusion," *Transactions on Vehicular Technology, IEEE*, vol. 58, no. 3, pp. 1157-1169, March 2009.

[40] Y. B.-S.a.H.A.P. Blom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *Transactions on Automatic Control, IEEE*, vol. 33, no. 8, August 1988.

[41] L. A. Johnson, V. K., "An improvement to the Interactive Multiple Model (IMM) algorithm," *Transaction on Signal Processing, IEEE*, vol. 49, no. 12, December 2001.

[42] C. Barrios, Y. Motai, "Improving estimation of vehicle's trajectory using the latest Global Positioning System with Kalman filtering," *Transactions on Instrumentation and Measurement, IEEE*, vol. 60, no. 12, pp. 3747-3755, December 2011.

[43] M. S. Grewal, A. P. Andrews, "Kalman filtering theory and practice using MATLAB", 2nd edition, p. 132, 2001.

CHAPTER 3: Can Smartphones Fill in the V2V/V2I Implementation Gap?

3.1. Introduction

The overall function of an Intelligent Transport Systems (ITS) is to improve decision-making, often in real time, improving the operation of the entire transport system. This can go from systems with intelligent route planning implemented to avoid some specific type of traffic in certain areas [1], to keeping track of the position of the vehicle for infrastructure assessment [2], to systems designed to aid with the prevention of collisions between the vehicles [3]. For this study the research is going to focus on the collision avoidance aspect of ITS, and will evaluate the use of smartphones as an intermediate step to accelerate the implementation of V2V and vehicle to infrastructure (V2I) communications between all vehicles.

There are two main types of collision avoidance systems: self-sufficient and interactive systems. Self-sufficient systems are those that can obtain enough information from their own sensors, such as those in [4-6], where they placed sensors around the vehicle to maintain a safe following distance or to detect vehicles in the surroundings. Interactive systems are those that, as the name implies, interact with the infrastructure and/or other vehicles, such as researched in [7-9], where their systems send spatial information to nearby vehicles to estimate the probability of a future collision. While self-sufficient systems are limited to line of sight detection, the interactive systems account for scenarios farther ahead or even around corners or intersections by predicting and communicating the future estimated trajectories.

The V2V and V2I areas are being well researched these days [10-16], as government is carefully evaluating the implementation of new technologies to make our roads safer. In an article published on February 3, 2014 by the United States Department of Transportation (USDOT) [17], the National Highway Safety Administration announced its decision to begin taking the next steps toward implementing V2V technology in all new cars and trucks, after years of research and unprecedented coordination between industry and across government.

Figure 20: Illustration for V2V and V2I from [17]

When the steps toward implementing V2V technology are defined, car and truck manufacturers will be mandated to enable this in their new vehicles. But, because V2V relies on other vehicles nearby also supporting V2V technology, there will be a gap of many years when the V2V/V2I will not able to show its true potential in improving road safety. In an article published by Forbes on March 14, 2013 [18], they calculated that the age of the average vehicle on the road is at a record high of 10.8 years, which means there are vehicles on the roads that are 20 years old. Keeping this in mind, it is a long time to wait to ensure full V2V/V2I reliability.

The contribution of this research includes the evaluation of using the smartphone as a temporary hook into the V2V/V2I infrastructure for collision avoidance warnings. If the evaluation were successful, it would allow for some companies to make programs for the different smartphones that could send and receive information to the V2V/V2I infrastructure. Allowing drivers of older vehicles the possibility of taking advantage of this new technology would not only benefit them, but it would also benefit the rest of the V2V/V2I enabled vehicles, as the number of vehicles participating in the system would be much greater. Also, since many drivers may already own a smartphone, there could be no extra cost to them, which encourages even more people to participate.

Another contribution that helps in the reliable implementation of using the smartphone's internal sensors are the stricter laws being passed in many states and countries; drivers are slowly being discouraged from touching their phones while driving, which places the phone on a fixed location in the vehicle, usually the dashboard. With the phone not being moved from its position while driving, one can assume that the internal accelerometer and gyroscope sensor measurements from a smartphone are produced by the dynamics of the vehicle itself.

Since this research wants to evaluate the use of a smartphone's built-in sensors in a setup that could be used in a collision avoidance system, it will focus on the prediction of a vehicle's future trajectory, and compare the results with the use of more robust sensors mounted on a vehicle to predict the same future trajectory. Given that multiple sensors will be used, some type of sensor fusion will be needed to use the different measurements in the prediction.

Multi Sensor Data Fusion (MSDF) techniques are used in many diverse fields, although most of the literature addresses the fields of military target tracking or autonomous robotics [19]. MSDF is required to combine and process data; which has been traditionally performed by some form of Kalman [20] or Bayesian filters. Furthermore, there can be two ways of setting up a MSDF system: centralized or decentralized. While a centralized approach suffices for most common scenarios where the sensors are synchronous, a decentralized approach is more convenient when the sensors should be treated independently [21-25], as with asynchronous sensors.

In [26], the authors discuss one solution they have developed: the Optimal Asynchronous Track Fusion Algorithm (OATFA), which evolved from their earlier research on an Asynchronous/Synchronous Track Fusion (ASTF) [27]. They base their technique in the Interacting Multiple Model (IMM) algorithm, but replaced the conventional Kalman filters with their OATFA (which contains several Kalman filters of its own). The OATFA treats each sensor separately, passing the output from each to a dedicated Kalman filter, departing from the idea that the best way to fuse data is to deliver it all to a central fusion engine. The results from the IMM-OATFA show position estimation errors half of those of what the conventional IMM produces. However, as pointed out by the same authors in [28], all measurement data must be processed before the fusion algorithm is executed. With a similar approach as the technique described above, the authors of [29] create asynchronous holds, where, from a sequence of inputs sampled at a slow sampling rate, it generates a continuous signal that may be discretized at a high sampling rate. Despite the benefits of making the asynchronous system into a synchronous one by using these methods, restrictions arise where, if for some reason a sensor is delayed in providing its data or is off-line for a few cycles. The whole system needs to wait, as it is designed to work with certain data at specific rates.

To evaluate whether smartphones can fill in the V2V/V2I implementation gap, a system to estimate a future position of a vehicle will be set up to determine where the vehicle will be 3 seconds later. A system like this could be used to communicate its future location to other vehicles or to the infrastructure and detect if there could be a collision, and that is why this research is using it to evaluate the built-in sensors of smartphones compared to sensors mounted on a vehicle like a manufacturer will do once the USDOT mandates it for new vehicles. This system presents the trajectory estimation at 3 seconds ahead of time, which is based on the average human reaction time of 1.5 seconds to stop a vehicle [30]. Looking at 3 seconds ahead of time was chosen as a reference point that is double the reaction time of an average human being. In reality, this number will probably vary in relation to the speed and the type of the vehicle since a faster or heavier vehicle will need more time to slow down, but it is taken as a reference point.

The next section will introduce the position estimation framework used in this system, and how it is setup. Target Motion Analysis (TMA) is needed to calculate an estimate of the range and velocity, for which this research opted to use KF, explained in section 3.2. The position and attribute estimation is the process of taking the associated measurements and calculating the current state of the target (vehicle), especially when some of the associated measurements are unavailable. A Dead-Reckoning (DR) approach was selected to complement the KF and be able to run the system at the frequency of its fastest sensor and update its prediction as soon as a change is detected. Also used was the IMM to merge the predictions and obtain one future position of the vehicle.

3.2. Position Estimation with Kalman Filters

The KF [31] was first proposed in the 1960s and it has been the most commonly used technique in target tracking and robot navigation since. The basic KF has been presented as a form of Bayesian filter [32], which is an optimal estimator for linear Gaussian systems. From a series of noisy measurements, the KF is capable of estimating the state of the system in a two-step process: correct and then predict [33-35].

The elements of this state vector (x) are: position, velocity, and acceleration of the vehicle. The position (x_v) and velocity (v_v) components of the state estimate have an x and y component to them (east-west and north-south directions), and the acceleration (a_v) has an n and t component to it (normal and tangential acceleration). So, the full state vector matrix will be $X=(x_x, x_y, v_x, v_y, a_n, a_t)$.

The estimated error covariance (P) for the state estimate is based on the relationships between each of the elements to the others. The error covariance matrix is a dataset that specifies the estimated accuracy in the observation errors between all possible pairs of vertical levels.

Together with P, the Jacobian matrix of the measurement model (H), the measurement noise covariance (R), and with the measurement noise (σ_m), are used to calculate the Kalman Gain (*K*). Once the *K* is calculated, the system looks at the measured data (*Z*) to identify the error of the predicted position and uses it to adjust *P*.

The KF has a long history of accurately predicting future states of a moving target, and has been applied to many different fields [36-39], including transportation, which is why it was selected for this research.

This research also opted for the use of IMM, which can calculate the probability of success of each KF model at every filter execution, providing a combined solution for the vehicle behavior [40-42]. These probabilities are calculated according to a Markov model for the transition between maneuver states, as detailed in [43]. To implement the Markov model, it is assumed that at each execution time, there is a probability p^{ij} that the vehicle will make the transition from model state i to state j.

In Johnson and Krishnamurthy's paper [44], they describe the IMM as a recursive suboptimal algorithm that consists of four core steps, interacting with the KF steps as illustrated in Figure 21.

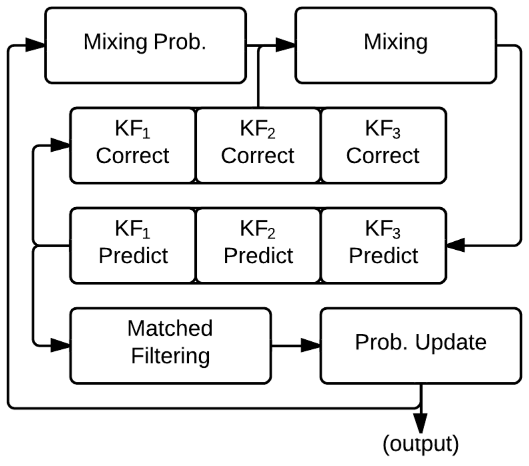


Figure 21: Flow-chart for three KF in an IMM framework.

The four step IMM process starts with the calculation of the mixing probabilities, which uses the transition matrix and the previous iteration mode probabilities $\mu_{k-1}(i)$ to compute the normalized mixing probabilities $\mu_k(i\,|\,j)$. The mixing probabilities are re-computed each time the filter iterates before the mixing step.

The second step uses the mixing probabilities, which are used to compute new initial conditions for each of the *n* filters. The initial state vectors are formed as the weighted average of all of the filter state vectors from the previous iteration x_{k-1}^{oj}. The error covariance corresponding to each of the new state vectors is computed as the weighted average of the previous iteration error covariance's conditioned with the spread of the means $\left(P_{k-1}^{oj}\right)$.

The third step calculates mode matched filtering, using the $\hat{x}_{k-1}^{0j}$ and P_{k-1}^{0j}, the bank of *n* Kalman filters produce outputs $\hat{x}_k^j$, the covariance matrix P_k^j, and the probability density function $f_n(z_k)$ for each filter (*n*).

The fourth set of calculations begins once the new initial conditions are computed; the filtering step generates a new state vector, error covariance and likelihood function for each of the filter models. The probability update step then computes the individual filter probability $\left(\mu_k(j)\right)$ as the normalized product of the likelihood function and the corresponding mixing probability normalization factor.

The estimate and covariance combination is used for output purposes only $\hat{x}_k = \sum_{j=1}^{n} \mu_k^j \cdot \hat{x}_k^j$; it is not part of the algorithm recursions.

3.3. Position Estimation Framework Using GPS and Accelerometer Sensors

For the overall trajectory estimation of a vehicle, modified Kalman Filters (KF) running in an Interacting Multiple Models (IMM) framework will be used. The KF and IMM are commonly used in the trajectory estimation.

In this setup, the GPS sensor provides the location (s_x, s_y), the velocity (v) and the angle of direction (β) using north as the zero. Then the accelerometer provides normal acceleration (a_n) and tangential acceleration (a_t).

When all the sensors are online, the general state equations can be defined as:

$$s(k) = s(k-1) + v(k-1)\cdot\Delta k + \frac{1}{2}a(k-1)\cdot\Delta k^2 + \frac{1}{6}j(k-1)\cdot\Delta k^3$$

$$v(k) = v(k-1) + a(k-1)\cdot\Delta k + \frac{1}{2}j(k-1)\cdot\Delta k^2$$

$$a(k) = a(k-1) + j(k-1)\cdot\Delta k$$

As shown in the above equations, included are the jerk j (acceleration change) in the equations as the factor responsible for the noise in the measurements; therefore, the jerk term can be represented as the prediction noise (σ_p). Also, in the equations, Δk represents the time difference between the current iteration (k) and the previous iteration $(k-1)$ of the system.

From the above general state equations, the different models to be used in this setup are defined, which represents the different spatial states the vehicle can be found in:

Constant Location Model (CL)

$$s(k) = s(k-1) + \sigma_{p_s}$$
$$v(k) = 0$$
$$a(k) = 0 \qquad (1)$$

Constant Velocity Model (CV)

$$s(k) = s(k-1) + v(k-1) \cdot \Delta k + \sigma_{p_s}$$
$$v(k) = v(k-1) + \sigma_{p_v}$$
$$a(k) = 0 \qquad (2)$$

Constant Acceleration Model (CA)

$$s(k) = s(k-1) + v(k-1) \cdot \Delta k + \frac{1}{2} a(k-1) \cdot \Delta k^2 + \sigma_{p_s}$$
$$v(k) = v(k-1) + a(k-1) \cdot \Delta k + \sigma_{p_v}$$
$$a(k) = a(k-1) + \sigma_{p_a} \qquad (3)$$

At any given time a measurement could be missing, either due to the sensor not being able to take the measurement (system running at a faster frequency than the sensor, malfunction or no satellites in view for the GPS) or due to the processing CPU not being able to read/write fast enough. When a measurement is absent and the value is needed for the models, the missing values are calculated from the measurements obtained by the remaining sensors based on previous real measurements, not estimations, when available.

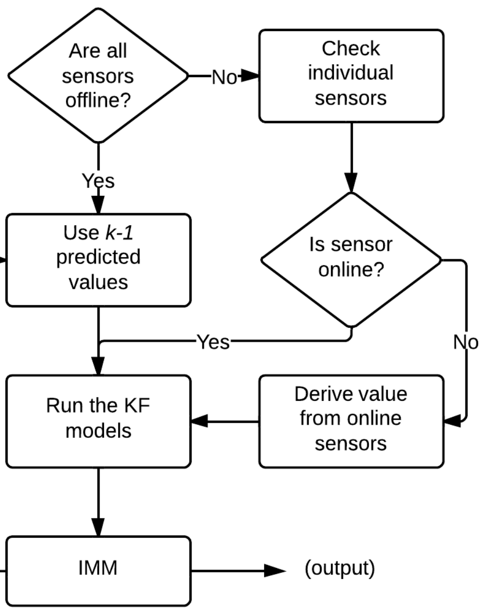


Figure 22: Flow-chart of the position estimation framework used by the vehicle-mounted and smartphone sensors.

Using the DR approach outlined in Figure 22 is more accurate than waiting until all measurements are available again, or predicting these measurements a second time, using only previously estimated values. Only when all measurements are missing, which will be very unlikely, the system will use all the previously estimated values to feed the models and obtain the new position estimation.

3.4. Car and Smartphone Sensors Setup for a V2V/V2I System

As described towards the end of section 3.1, this research will estimate the future position of the vehicle 3 seconds away using the framework described in section 3.3, which is something that could be shared with other vehicles or with the infrastructure as part of a collision avoidance system. The vehicle-mounted (VM) sensors specifically setup for this specific task will be used, like manufacturers will implement in their vehicles, but smartphone (SP) sensors will also need to be evaluated when used for position estimation.

To properly evaluate if smartphones can be used in a V2V/V2I system, this research plans to set a baseline by running the VM measurements through the position estimation framework defined in section 3.3 and calculate the position errors in the estimations by comparing them to the actual GPS data. Once the baseline is established and a determination of what are the amounts of prediction errors obtained, the individual SP measurements will be fed into the same position estimation framework and the error will be calculated in the position estimations, as illustrated in Figure 23. This research can then proceed to compare the results between the different sensors used and evaluate whether the smartphones' built-in sensors yield similar prediction errors or now.

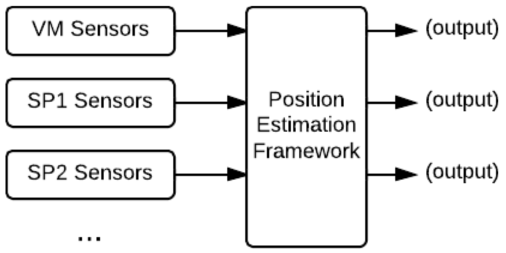


Figure 23: VM and SP GPS and accelerometer sensors using the same framework to predict future positions.

The setup on the VM sensors consists of a Garmin 16HVS GPS receiver running at 1Hz and a Crossbow 3-axis accelerometer running at 10Hz. An

AutoEnginuity OBDII ScanTool (which obtains the velocity from the vehicle's internal system at 1Hz) is also available, but it will not be used in this evaluation because the smartphones this research is using do not have a way of connecting into the ODBII system. The data from the sensors used is post processed from the different log files recorded on an earlier date, and matched based on time stamps. Since these were mounted on a van from the University of Connecticut, it is labeled as UConn data.

For the smartphones used in this evaluation, some were selected from different manufacturers and at different price ranges, to identify if there is some limitation on which ones can be used to fill in the gap in the V2V/V2I implementation. Also smartphones are used with different operating systems as well, to improve the evaluation experiment and take that into account as well. They were mounted securely on the vehicle to ensure the accelerometer readings truly reflect the dynamics of the vehicle. Because several smartphones were running at the same time, they were mounted in the trunk where they would still have a clear view of the sky, as shown in Figure 24, but a more common implementation would be to mount only one of them on the dashboard. Figure 24 shows six smartphones, but one of them did not record any GPS data so it had to be removed from this experiment. The smartphones used in the evaluation of whether they could be used to fill in the V2V/V2I implementation gap are listed in Table 7.

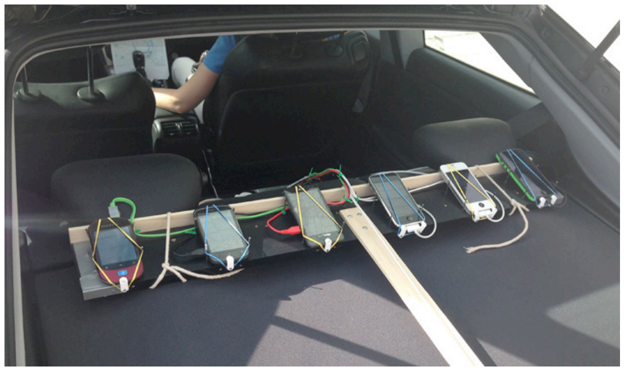


Figure 24: Smartphones securely mounted in the trunk of a hatchback vehicle.

Table 7: Smartphone specs

Manufacturer	*Model*	*OS*	*Rel. Date*	*Base Price*
Alcatel	OneTouch 908F	Android 2.2	6/2011	$130
HTC	Desire C	Android 4.0	6/2012	$150
LG	Lucid VS 840	Android 2.3	4/2012	$300
Apple	iPhone 3GS	iOS 5.1	6/2009	$199*
Apple	iPhone 5	iOS 7.01	9/2012	$650

Details about these smartphones obtained from gsmarena.com.

* Subsidized price; this model could not be purchased without a contract, so real price could be two or three times more.

All smartphones listed above have a built-in accelerometer sensor that can take measurements at 10Hz, but no details were found on their model or sensitivity. These smartphones also have a built-in GPS sensor, and only very basic information was found about them. The iPhone 5 has an A-GPS/GLONASS sensor, while the other four smartphones do not have support for GLONASS (Global Navigation Satellite System by the Russians). Also, both Apple smartphones can take measurements from the GPS sensor at 10Hz, while the other three smartphones can only take measurements at 1Hz.

Some smartphones also have a three-axis gyro sensor and a compass, which could be used as well to better estimate a position of a vehicle; but to match more closely the sensors mounted on the vehicle, and have a more equal comparison, they were not used in this experiment.

The measurements from the internal sensors of the iPhone smartphones are recorded by running the SensorLog v1.4 application written by B. Thomas. The sensors' measurements on the Android smartphones are recorded using the Data Recording v1.0.2.0 application written by T. Wolf. The data used is labeled by smartphone manufacturer, except where more than one device per manufacturer, in which case the data was labeled by model name.

To properly exercise the position estimation framework described in section 3.3, the route shown in Figure 25 for this evaluation was selected, which contains several curves (smooth and sharp) and straight paths, driven at different speeds in the larger and smaller roads. There were also some traffic lights on the way, and even a U-turn, providing also some stop and go scenarios. The route driven, shown in Figure 25, is approximately 44 km long and takes about 45 minutes to drive all of it.

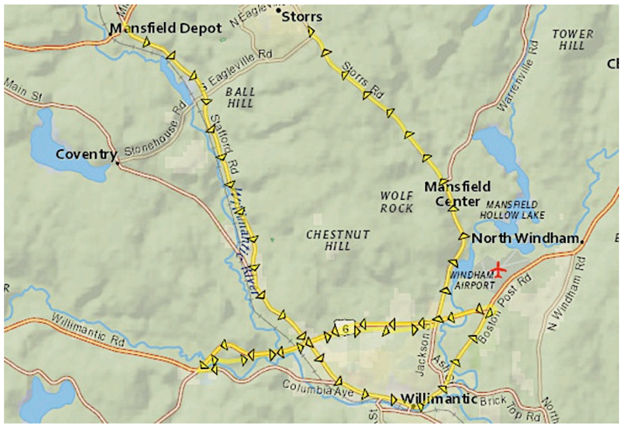


Figure 25: Map of the entire recorded route near the University of Connecticut.

3.5. Evaluation Criteria

To evaluate whether smartphones can properly fill in the V2V/V2I implementation gap, the position estimation error between the VM sensors and the SP sensors was selected using the same KF models and IMM framework. To start, the position estimation error between both setups will be evaluated for the whole trajectory recorded. Also, since position estimation errors tend to increase during non-straight paths, this research will also divide the trajectory recorded into smooth and sharp curves. To determine whether a set of consecutive points in the trajectory is a curve or a straight line, the change in the heading (angle) between the current heading and the heading 2 seconds before was looked at; if more than 5° then it was defined as a curve. And, to determine if the curve is a sharp one, the change has to be greater than 20°, otherwise it was defined as a smooth curve.

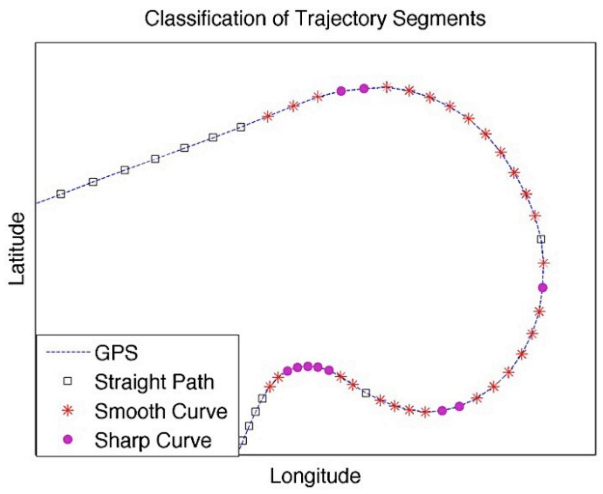


Figure 26: Automated classification of road segments based on heading angle changes every 2 s shown on a small subset of Figure 25.

One singular set of curves extracted from the whole route is shown in Figure 26. In it one can see the different markers indicating which positions were assigned as part of a straight path, smooth curve, or sharp curve. It is not perfect as can be seen sometimes a segment type fluctuates between types for a single or a couple of positions. But, for the most part, the classification shown in Figure 26 is logical, and it determines it is made of 74% straight paths, 18% smooth curves, and 8% sharp curves. This classification is only used to partition the whole dataset and be able to analyze the results in groups, as this research expects position estimation errors to increase as going from a straight path to a sharp curve.

To calculate the position estimation error in each step, this research will compare the estimated position to the actual position measured by the GPS 3 seconds later. This will allow a dataset of calculated errors to be built for the whole trajectory, that then it can be divided into the route sections described in the previous paragraph.

This research will look at average prediction errors and root mean square (RMS) prediction errors to try to evaluate whether the sensors built into smartphones can properly fill in the V2V/V2I implementation gap.

$$RMS = \sqrt{\frac{\sum_{k}^{k'} \left(err_k - err_{k-1}\right)^2}{\left(k' - k\right)}}$$

For this experiment, a tolerance of 10% from the position estimation errors obtained from the VM sensors will be used; therefore, if a smartphone yields more than 10% higher estimation error than the VM sensors, then it will be concluded that such a smartphone cannot be used as a temporary solution to fill in the V2V/V2I implementation gap. Due to the limitations explained at the end of section 3.4 with not being able to record the measurements of the VM and all SP sensors during the same drive, the research cannot try to also compare actual measurement differences between the VM and SP measurements to help with this evaluation, as it is known they will be different because they are recorded on different drives of the same route shown in Figure 25.

3.6. Experimental Evaluation

3.6.1. Dataset Characteristics

The characteristics of the complete recorded dataset is shown in Table 8, where the mean and standard deviation for the position difference between each second, velocity, normal acceleration, and tangential acceleration are displayed.

Table 8: Representative Datasets

	Device	*Distance* (m)	*Velocity* (m/s)	*Acc.norm* (m/s^2)	*Acc.tang* (m/s^2)
Whole Trajectory	UConn	18.14±8.99	17.95±8.45	-0.17±0.58	0.61±0.60
	Alcatel	16.31±8.40	16.30±7.95	-0.12±0.73	0.66±0.95
	HTC	16.21±8.24	16.29±8.01	-0.09±0.56	0.64±0.68
	LG	16.79±9.13	16.45±7.83	-0.18±0.58	0.61±0.64
	iPhone3GS	16.59±9.37	16.21±7.96	-0.22±0.74	0.51±0.88
	iPhone5	16.25±9.09	16.24±8.18	-0.11±0.79	0.62±0.82
Straight Paths	UConn	19.43±8.29	19.02±8.14	-0.16±0.48	0.55±0.57
	Alcatel	18.76±6.92	18.72±6.70	-0.12±0.72	0.64±0.95
	HTC	18.40±6.56	18.38±6.51	-0.10±0.54	0.62±0.57
	LG	18.98±7.74	18.42±6.65	-0.18±0.56	0.59±0.50
	iPhone3GS	16.96±9.04	16.63±7.77	-0.20±0.74	0.50±0.88
	iPhone5	16.23±9.24	16.23±8.29	-0.10±0.78	0.60±0.73
Smooth Curves	UConn	16.19±9.26	15.81±8.03	-0.29±0.89	0.74±0.63
	Alcatel	14.66±7.90	14.58±7.36	-0.11±0.75	0.73±0.98
	HTC	12.76±8.32	12.92±7.99	-0.07±0.64	0.78±0.91
	LG	14.01±9.28	13.53±7.65	-0.21±0.64	0.68±0.82
	iPhone3GS	12.47±11.31	11.99±8.49	-0.34±0.70	0.58±0.87
	iPhone5	17.99±6.64	17.86±5.83	-0.16±0.89	0.82±1.29
S	UConn	8.52±9.17	9.91±7.65	-0.10±0.89	1.01±0.68
	Alcatel	7.13±9.60	7.49±8.35	-0.16±0.79	0.63±0.88

HTC	5.29±9.05	5.97±8.68	-0.07±0.64	0.59±0.94
LG	5.48±8.86	7.09±8.01	-0.16±0.66	0.59±1.03
iPhone3GS	16.30±12.13	13.50±9.49	-0.15±1.05	0.70±0.89
iPhone5	12.00±8.54	12.18±8.54	-0.50±1.04	0.62±1.58

Values represent median ± standard deviation of all sensor measurements collected by each device (~25000 data points).

Looking at Table 8, it is quickly noticed that the values between the Distance and Velocity columns are very similar, as expected, because this research is measuring the position change every 1 second. Also, as mentioned in section 3.4, the UConn data was obtained on an earlier date, so it can be seen that, overall, the University of Connecticut van was driven a little bit faster than the vehicle with the smartphones. Also, because all five smartphones were in the same vehicle, their sensor measurements should have been very similar, which is not the case in several places. For example, for sharp curves, the two iPhones seemed to be moving at a much faster speed than the other three devices, while during straight paths they seemed to be moving a little slower than the rest. The tangential acceleration for all devices seems to be fairly consistent across all devices, while the normal acceleration is not as consistent, especially when smooth and sharp curves were observed, which could imply that some sensors are more sensitive than others.

The accelerations shown in Table 8, both normal and tangential, have large deviation values compared to their calculated mean values, which is not expected in straight lines. This could be caused by bumps on the road or uneven pavement, where the sensitive accelerometers record a gravity pull in some direction, but quickly returns back to the "normal" state, not being any real heading change in the vehicle's trajectory.

3.6.2. Evaluation of Position Estimation Error

To set up the IMM it is necessary to calculate the transition probability matrix, so the GPS position measurements for the whole trajectory shown in Figure 25 was used. From this full GPS log that contained multiple scenarios, it was determined which transition occurs frame by frame by comparing the actual measurements from the GPS to the smoothed measurements. The smoothing of the data was done with a rolling window using a combination of median smoothing, split the sequence, and Hann's sequence, which removed any abrupt changes from the data. The type of spatial change, such as no change, a constant change, and so on, determines each transition. Similarly, by calculating the covariance of the differences in the measurements to each other, the measurement noise covariance matrix (R) was obtained. And last, by calculating the covariance of the differences in the measurements compared to their respective *x* and *y* components, the process covariance noise (Q) for each KF was obtained. From this type of information, calculating the frequency the vehicle changes from one state to another, the transition probability matrix below is derived.

$$p^{ij} = \begin{bmatrix} 0.177 & 0.656 & 0.167 \\ 0.027 & 0.576 & 0.397 \\ 0.023 & 0.501 & 0.478 \end{bmatrix}$$

Next each of the devices were run through the same IMM system using the KF models described in section 3.3, and for each new measurement obtained from any of the sensors, the system predicts the position of where the vehicle is going to be 3 seconds later in time.

Table 9: RMS prediction error (3 s ahead)

	UConn	*Alcatel*	*HTC*	*LG*	*iPhone3GS*	*iPhone5*
Whole Trajectory	1.21 ±2.64	1.88 ±2.62	1.00 ±1.03	1.13 ±1.25	2.34 ±3.00	1.41 ±2.08
Straight Paths	0.92 ±1.65	1.77 ±2.56	0.89 ±1.10	0.98 ±1.13	2.21 ±2.81	1.35 ±2.05
Smooth Curves	2.28 ±4.87	1.89 ±2.14	0.97 ±0.97	1.07 ±1.23	3.49 ±4.02	1.86 ±2.40
Sharp Curves	2.51 ±4.36	2.11 ±3.83	1.21 ±1.23	1.43 ±1.59	4.78 ±4.70	2.13 ±2.05
Whole Trajectory	0%	+55%	-17%	-7%	+93%	+17%
Straight Paths	0%	+92%	-3%	+7%	+140%	+47%
Smooth Curves	0%	-17%	-57%	-53%	+53%	-18%
Sharp Curves	0%	-16%	-52%	-77%	+90%	-15%

The upper half contains values representing median prediction error in meters ± standard deviation by all devices for trajectory shown in Figure 25 (~25000 data points). The lower half contains percentage deviation compared to UConn errors.

Table 9 displays the RMS distance between the predicted and actual positions. This prediction error can only be calculated when the time stamps between the predicted position and GPS reading match. It is assumed that the GPS reading is correct and it calculates the distance vector to the predicted position. Then the mean and standard deviation was calculated of all the calculated RMS error vectors for the whole trajectory and also for the classified by segment types.

As expected, the prediction error was less during straight paths, and it increased during curves. Based on the values recorded in Table 9, the prediction errors can double during curves. Also, the prediction errors for smooth curves were better than during the sharp curves, which makes sense because, in a smooth curve, the vehicle is changing its heading less abruptly than in a sharp curve, allowing the system more time to recalculate and correct its next prediction.

This research also observed that the prediction error was not the same between all devices, and sometimes a device that has a small prediction error in one segment type may not be as good on a different segment type, making it hard to draw conclusions from Table 9. In spite of these results, if one looks at the percent deviation of prediction errors compared to the UConn results, it can be narrowed down to the HTC and LG cellphones having the smaller prediction errors overall and meeting the tolerance of no more than 10% more error than obtained with the UConn sensors.

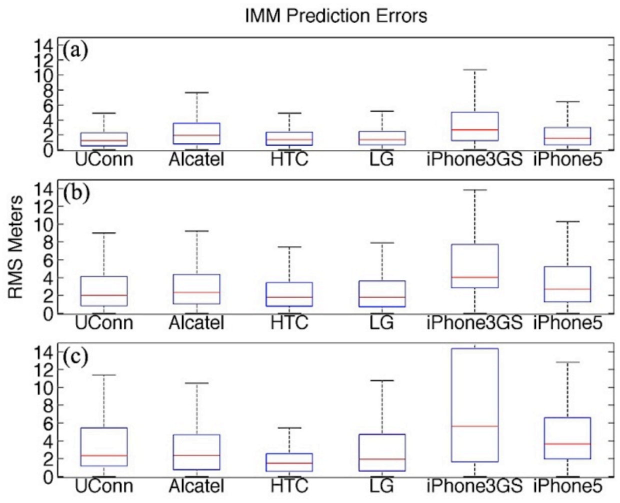


Figure 27: Prediction errors during (a) straight paths, (b) smooth curves, and (c) sharp curves.

Figure 27 is another way of representing the prediction errors for each of the devices in the different segment types. The boxplots display the median value as the solid line dividing the box in two, and then the upper and lower half of the boxes represent the inter-quartiles, which together represent 50% of the calculated prediction errors. The upper whisker indicates that 75% of the errors fall below it, and the lower whisker indicates the 25% marker. With this in mind it can be seen that for the straight paths, except for the iPhone3GS, the boxes are very short, which means that the prediction errors have a high level of agreement. One can also see small boxes in the smooth and sharp curves for the HTC and LG, so it can be observed that their predictions are fairly consistent most of the time, unlike the boxplot for the iPhone3GS where it is a very large box indicating a very low level of agreement between the predictions. Also, the lower the boxes to the x-axis, the smaller the prediction errors, so a small box close to the x-axis, like the HTC in sharp curves, or the UConn in straight lines, represents a very accurate prediction system. Again, looking at the boxplots for the five smartphones, one can visually pick the HTC and LG to be fairly good, then maybe Alcatel and iPhone5, though it looks like the iPhone5 is not as reliable as Alcatel during sharp curves.

When looking at the iPhone3GS results, both in Table 9 and Figure 27, it can be observed that this device has prediction errors much larger than the other devices. Especially when this research looks at the boxplot, it can be seen that the inter-quartiles cover a very large range of error values, making this device very unreliable. It seems this device has a problem, so this research looked at its GPS readings and it seems that it is losing its signal quite often, introducing more errors to what was assumed to be the "true" position (see Figure 28). This could be due to a defective GPS unit on the device, but no other iPhone3GS was available to confirm if this was the case or if this model has a hard time locking onto a GPS signal when trees/buildings are blocking it.

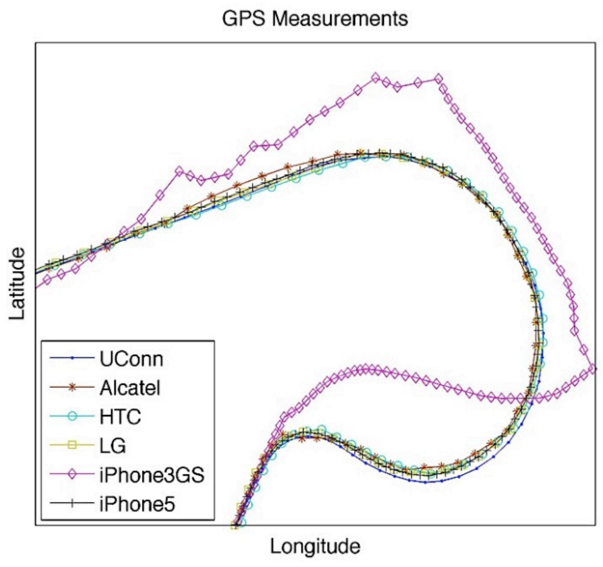


Figure 28: GPS recorded positions by each device on a small subset of Figure 25. Trajectory is in a counter-clockwise direction.

In Figure 28 one can see that most devices recorded very similar position measurements from their GPS sensor in this small subset set of curves from the whole route, except for the iPhone3GS, which seems to be very different than the others. As mentioned earlier, it will be assumed that it had a defective GPS sensor, but this research will leave the results in this experiment just in case older smartphones had GPS units that cannot easily lock satellite signals and the results from this old device are valid.

To look at a subset of the whole route shown in Figure 25, the couple of curves shown in Figure 28 was selected, and represented the results in a similar way, but only for this small subset of the dataset. This selected segment of the route represents only 0.8 km (36% straight path, 44% smooth curve, and 20% sharp curve as displayed in Figure 26), which takes around 10 seconds to go through.

When looking at Table 10, the first difference that might be observed when comparing it to the results for the whole route shown in Table 9 is that the average prediction error for whole trajectory of the selected subset is different. In this case, straight paths are a small percentage of the whole selected subset while smooth curves are the most abundant. For this very specific set of curves, the UConn data is better than any of the smartphones in all trajectory types. The LG device yields the smallest prediction errors of all the smartphones, and still within the selected 10% tolerance when comparing to the UConn results. The next best devices seem to be the HTC and iPhone5 smartphones where, despite having prediction errors over the 10% tolerance, their prediction errors are around 20% worse than the UConn results.

Table 10: RMS prediction error (3 s ahead)

	UConn	*Alcatel*	*HTC*	*LG*	*iPhone3GS*	*iPhone5*
Whole	1.46	3.22	1.81	1.56	6.34	1.79
Trajectory	±0.97	±3.64	±1.87	±1.65	±6.95	±2.18
Straight	0.69	2.96	1.42	0.77	4.50	0.95
Paths	±0.53	±4.60	±2.11	±0.58	±5.76	±0.74
Smooth	1.69	3.13	1.75	1.84	11.57	1.57
Curves	±0.54	±2.18	±1.43	±2.13	±6792	±0.85
Sharp	1.98	3.52	2.67	2.19	14.73	2.69
Curves	±0.97	±3.81	±2.03	±1.26	±6.24	±3.12

Values represent median prediction error in RMS meters ± standard deviation by all devices for the selected curve shown in Figure 28 (~550 data points).

Another difference one can observe in Table 10 is that, unlike the results in Table 9, the HTC device did not seem to perform as well in this selected set of curves than when evaluated over the whole route. Even when looking at the results for Smooth and Sharp Curves, the HTC results were always worse than the UConn prediction errors, which is not the case when looking at the data in Table 9. In Table 10, it seems that most smartphones performed worse than the UConn setup in this set of curves. Since this is consistent across all smartphones, it can be concluded that there was something on the curves that affected the prediction, like bumps or maybe unleveled pavement, especially over the sharp curve section.

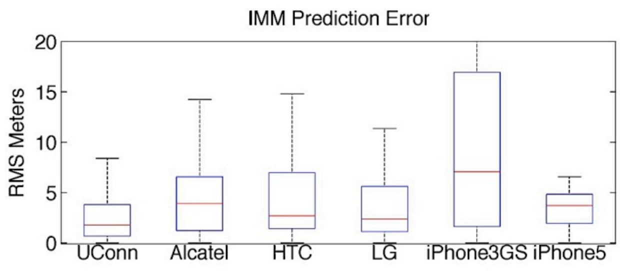


Figure 29: Prediction errors for the selected route subset shown in Figure 28.

The boxplots shown in Figure 29 also show that the UConn data yields better position predictions than the other devices, and one can also see that the LG and iPhone5 are the next two devices with the centerline and the inter-quartiles closest to the x-axis. For some reason, in this specific set of curves, the HTC did not perform as well, with a larger box telling that there was not as much agreement in the prediction errors as with the LG and iPhone5 devices which have a smaller box. Again one can see that in Figure 29, the iPhone3GS is extremely unreliable, which, as stated earlier, could be caused by the poor GPS measurements obtained from this device.

A visual representation of the prediction errors during the small subset set of curves previously mentioned is shown in Figure 30, where each predicted trajectory is compared to the actual GPS position measured 3 seconds later. One can observe that for some devices there is a smooth trajectory of predicted positions, like for the UConn and LG, closely followed by the HTC; but, it can also be observed that some other devices are constantly correcting its predicted position drastically, causing all those spikes during the curves. One positive thing of looking at the predicted position errors as shown in Figure 30 is that, despite conclusions obtained from Table 10 and Figure 29, that the iPhone5 was predicting a future position better than several of the other devices, this research would not think this is a reliable device after looking at Figure 30. Therefore, even though Figure 30 cannot be used by itself to draw some final conclusions, it is a very useful addition to Table 10 and Figure 29.

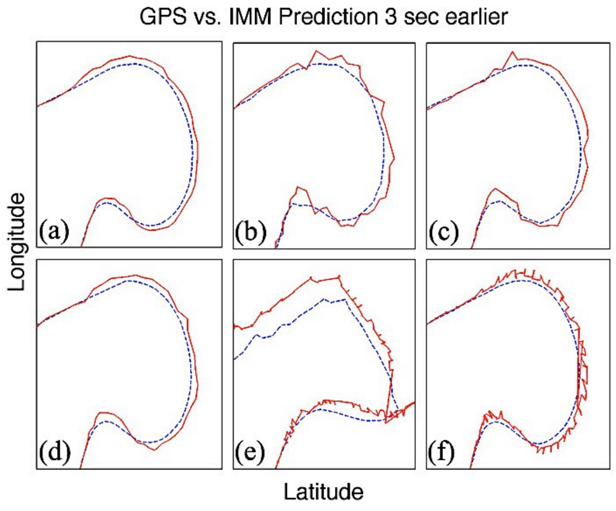


Figure 30: Dotted lines represent GPS measurements while solid lines represent IMM predicted positions 3 s earlier for (a) UConn, (b) Alcatel, (c) HTC, (d) LG, (e) iPhone3GS, and (f) iPhone5.

3.6.3. Computational Complexity

For completeness, this research also looked into how much of a load it is to run this IMM system with the different devices explored in this chapter. Because the dataset had already been recorded, only the processing time of the system itself was measured. If taking into account the processing time of the reading and recording data from the sensors, the CPU times may be larger.

Table 11: Computational complexity

	tic/toc (s)	*cputime (s)*	*Data points*	*Avg. load*
UConn	992.53	992.60	20081	71.83
Alcatel	2179.15	2179.23	27141	109.11
HTC	2079.86	2079.98	27104	104.62
LG	2135.83	2135.90	27060	108.62
iPhone3GS	5010.49	5010.61	26976	153.83
iPhone5	5211.41	5211.47	26878	148.47

Measurements taken on IMM system running through the whole trajectory.

Table 11 shows different Matlab commands used to measure CPU times for each of the devices. All two commands (tic/toc and cputime) measure actual CPU time used by the Matlab code, but this research is showing both to get a better idea on the accuracy of the measurements. The column tic/toc represents actual start/stop time of execution, while cputime displays the actual CPU time in that was spent executing the code. The system was run on a machine with a dual core 2.0 GHz CPU.

As expected, the values for the different devices are very similar in Table 11, as most of them had a very similar number of records in their dataset. One can observe that the UConn dataset had a smaller number of records, due to the machine reading and recording the sensors mounted on the van was not fast enough and kept skipping some of the 10Hz measurements, and hence a shorter processing time. Another difference that is easily observed in the processing time columns is that it took more than twice as long to process the iPhones' datasets. The reason behind this is that the iPhones recorded the GPS positions at 10Hz, and the Matlab code had to convert each latitude/longitude coordinate degrees to spatial distances in meters, and this had to be done 10 times more often than for the other devices.

3.7. Conclusions

Based on the results shown in Table 9 and Table 10, it seems possible to use the built-in sensors of some smartphones to predict a future position of a vehicle. It is not as clear, however, how to determine the criteria that identifies which smartphone will perform well before testing it. For example, though price seemed to play a small role, the HTC smartphone is one of the cheaper ones used in this experiment, and it performed quite well in some scenarios. The more expensive LG device yielded more reliable results in more scenarios, so price could be a factor; but then the iPhone 5, being the most expensive one, did not contribute well to this factor.

This research also learned that the use of the accelerometer sensor from the smartphones might not be the most accurate sensor to detect a spatial change in the vehicle's movement. They are meant to measure the tilting of the device and are sometimes very sensitive, making the system used to believe there are abrupt changes when maybe it was a bump on the road. Also, because they are measuring the gravitational pull, when the device is at an angle, it records that gravitational pull on that axis, which may not be related to a centrifugal force caused by a curve.

Future research will include the use of other sensors also found in smartphones. A couple of the smartphones used have gyroscope sensors, but this as well could give false calculations due to road bumps. A few of the smartphones also have a compass sensor which, with reliable magnetic north measurements, could be a more accurate sensor to detect as soon as the vehicle is changing heading at the start of a curve.

3.8. References

[44] V. Di Lecce, A. Amato, “Route planning and user interface for an advanced intelligent transport system,” *Intelligent Transport Systems, IET* , vol. 5, no. 3, pp. 149-158, September 2011.

[45] X. Xu, T. Xia, A. Venkatachalam, D. Huston, “The development of a high speed ultrawideband ground penetrating radar for rebar detection,” *Journal of Engineering Mechanics*, vol. 139, no. 3, pp. 272-285, March 2013.

[46] T. Taleb, A. Benslimane, K. Ben Letaief, “Toward an effective risk-conscious and collaborative vehicular collision avoidance system,” *Transactions on Vehicular Technology*, IEEE, vol. 59, no. 3, pp. 1474-1486, March 2010.

[47] F. Jimenez, J. Eugenio Naranjo, “Improving the obstacle detection and identification algorithms of a laserscanner-based collision avoidance system,” *Transportation Research Part C-Emerging Technologies*, vol. 19, no. 4, pp. 658-672, August 2011.

[48] R. Toledo-Moreo, M. A. Zamora-Izquierdo, “Collision avoidance support in roads with lateral and longitudinal maneuver prediction by fusing GPS/IMU and digital maps,” *Transportation Research Part C-Emerging Technologies*, vol. 18, no. 4, pp. 611-625, August 2010.

[49] C. Hakyoung, L. Ojeda, J. Borenstein, “Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope,” *Transactions on Robotics and Automation, IEEE*, vol. 17, no. 1, pp. 80-84, February 2001.

[50] R. Toledo-Moreo, M. Pinzolas-Prado, J. Manuel Cano-Izquierdo, "Maneuver prediction for road vehicles based on a neuro-fuzzy architecture with a low-cost navigation unit," *Transactions on Intelligent Transportation Systems, IEEE*, vol. 11, no. 2, pp. 498-504, June 2010.

[51] J. Ueki, J. M., Y. Nakamura, Y. Horii, H. Okada, "Development of vehicular-collision avoidance support system by inter-vehicle communications," *59th Vehicular Technology Conference, IEEE*, vol. 5, pp. 2940-2945, May 2004.

[52] H. D. Weerasinghe, R. Tackett, H. Fu, "Verifying position and velocity for vehicular ad-hoc networks," *Security and Communication Networks*, vol. 4, no. 7, pp. 785-791, July 2011.

[53] J. J. Blum, A. Eskandarian, "A reliable link-layer protocol for robust and scalable intervehicle communications," *Intelligent Transportation Systems, IEEE Transactions on* , vol. 8, no. 1, pp. 4,13, March 2007.

[54] Jin Wen-Long, W. Recker, "An analytical model of multihop connectivity of inter-vehicle communication systems," *Wireless Communications, IEEE Transactions on* , vol. 9, no. 1, pp. 106,112, January 2010.

[55] B. Dalla Chiara, F. Deflorio, S. Diwan, "Assessing the effects of inter-vehicle communication systems on road safety," *Intelligent Transport Systems, IET* , vol. 3, no. 2, pp. 225,235, June 2009.

[56] S. Sohaib, D.K.C. So, "Asynchronous cooperative relaying for vehicle-to-vehicle communications," *Communications, IEEE Transactions on* , vol. 61, no. 5, pp. 1732,1738, May 2013.

[57] M. Fogue, P. Garrido, F. J. Martinez, J. C. Cano, C. T. Calafate, P. Manzoni, "Automatic accident detection: Assistance through communication technologies and vehicles," *Vehicular Technology Magazine, IEEE* , vol. 7, no. 3, pp. 90,100, September 2012.

[58] Liu Chunhua, K. T. Chau, Wu Diyun, Gao Shuang, "Opportunities and challenges of vehicle-to-home, vehicle-to-vehicle, and vehicle-to-grid technologies," *Proceedings of the IEEE* , vol. 101, no. 11, pp. 2409,2427, November 2013.

[59] T. Abbas, L. Bernado, A. Thiel, C. Mecklenbrauker, F. Tufvesson, "Radio channel properties for vehicular communication: merging lanes versus urban intersections," *Vehicular Technology Magazine, IEEE* , vol. 8, no. 4, pp. 27,34, December 2013.

[60] D. Freidman, (2014, Feb. 3). V2V: Cars communicating to prevent crashes, deaths, injuries. US DOT. [Online]. Available: *http://www.dot.gov/fastlane/v2v-cars-communicating-prevent-crashes-deaths-injuries*

[61] J. Gorselanv, (2013, Mar. 14). Cars that can last for 250,000 miles (or more). Forbes Corp. [Online]. Available: *http://www.forbes.com/sites/jimgorzelany/2013/03/14/cars-that-can-last-for-250000-miles/*

[62] R. C. Luo, C. C. Y., K. L. Su, "Multisensor fusion and integration – Approaches, applications, and future research directions," *Sensors, IEEE*, vol. 2, pp. 107-119, April 2002.

[63] J.B.G.a.C.J. Harris, "Some remarks on Kalman filters for the multisensor fusion," *Elsevier Information Fusion*, vol. 3, pp. 191-201, 2002.

[64] S. C. Felter, “An overview of decentralized Kalman filter techniques,” *Proceedings of the 1990 Southern Tier Technical Conference, IEEE*, pp. 79-87, April 1990.

[65] M. Hua, T. Bailey, P. Thompson et al., “Decentralised solutions to the cooperative multi-platform navigation problem,” *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 47, no. 2, April 2011.

[66] E. M. Nebot, M. Bozorg, H. F. Durrant-Whyte, “Decentralized architecture for asynchronous sensors,” *Autonomous Robots*, vol. 6, no. 2, pp. 147-164, April 1999.

[67] S. I. Roumeliotis, G. A. Bekey, “Distributed multirobot localization,” *Transactions on Robotics and Automation, IEEE*, vol. 18, no. 5, pp. 781-795, October 2002.

[68] H. M. Wang, Q. Yin, X. Xia, “Fast Kalman equalization for time-frequency asynchronous cooperative relay networks with distributed space-time codes,” *Transactions on Vehicular Technology, IEEE*, vol. 59, no. 9, pp. 4651-4658, November 2010.

[69] G. A. Watson, T. R. R., A. T. Alouani, “An IMM architecture for track fusion,” *Signal Processing, Sensor Fusion, and Target Recognition (Proceedings of SPIE 4052)*, vol. IX, pp. 2-13, 2000.

[70] A. T. Alouani, T. R. R., “On optimal asynchronous track fusion,” *Proceedings of 1st Australian Symposium on Data Fusion, IEEE*, November 1996.

[71] G. A. Watson, T. R. R., A. T. Alouani, “Optimal track fusion with feedback for multiple asynchronous measurements,” *Proceedings of SPIE Acquisition, Tracking and Pointing XIV*, April 2000.

[72] L. Armesto, G. Ippoliti, S. Longhi et al., "Probabilistic self-localization and mapping - An asynchronous multirate approach," *Transactions on Robotics & Automation Magazine, IEEE*, vol. 15, no. 2, pp. 77-88, June 2008.

[73] M. Green, "How long does it take to stop? Methodological analysis of driver perception-brake times," *Transportation Human Factors*, vol. 2, no. 3, pp. 195-216, September 2000.

[74] R. Kalman, "A new approach to linear filtering and prediction problems," *Transaction ASME*, vol. 82, pp. 34-45, 1960.

[75] Y. Ho, R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *Transactions on Automatic Control, IEEE*, vol. 9, no. 4, pp. 333-339, October 1964.

[76] G. Welch, G. B., "An introduction to the Kalman filter," SIGGRAPH Course notes, 2001.

[77] Y. Bar-Shalom, X. R. L., T. Kirubarajan, "Estimation with applications to tracking and navigation," Wiley and Sons, 2001.

[78] Y. Bar-Shalom, X. R. L., "Estimation and tracking: Principles, techniques and software," Artech House, 1993.

[79] B. Mokaberi; A. A. G. Requicha, "Drift compensation for automatic nanomanipulation with scanning probe microscopes," *Transactions on Automation Science and Engineering, IEEE*, vol. 3, no. 3, pp. 199-207, July 2006.

[80] P. N. Pathirana, A. V. Savkin, S. Jha, "Location estimation and trajectory prediction for cellular networks with mobile base stations," *Transactions on Vehicular Technology, IEEE*, vol. 53, no. 6, pp. 1903-1913, November 2004.

[81] C. F. Graetzel, B. J. Nelson; S. N. Fry, “A dynamic region-of-interest vision tracking system applied to the real-time wing kinematic analysis of tethered drosophila,” *Transactions on Automation Science and Engineering, IEEE*, vol. 7, no. 3, pp. 463-473, July 2010.

[82] S. M. Bhandarkar, L. Xingzhi, R. F. Daniels, E. W. Tollner, “Automated planning and optimization of lumber production using machine vision and computed tomography,” *Transactions on Automation Science and Engineering, IEEE*, vol. 5, no. 4, pp. 677-695, October 2008.

[83] L. Hong, “Multirate interacting multiple model filtering for target tracking using multirate models,” *Transactions on Automatic Control, IEEE*, vol. 44, no. 7, pp. 1326-1340, July 1999.

[84] E. Mazor, “Interacting multiple model methods in target tracking: A survey,” *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 34, no. 1, pp. 103-123, January 1998.

[85] B. S. Chen, C. Y. Yang, F. K. Liao, “Mobile location estimator in a rough wireless environment using extended Kalman-based IMM and data fusion,” *Transactions on Vehicular Technology, IEEE*, vol. 58, no. 3, pp. 1157-1169, March 2009.

[86] Y. B.-S.a.H.A.P. Blom, “The interacting multiple model algorithm for systems with Markovian switching coefficients,” *Transactions on Automatic Control, IEEE*, vol. 33, no. 8, August 1988.

[87] L. A. Johnson, V. K., “An improvement to the Interactive Multiple Model (IMM) algorithm,” *Transaction on Signal Processing, IEEE*, vol. 49, no. 12, December 2001.

CHAPTER 4: Conclusions

For this research different KF was implemented to account for the identified possible states *a vehicle can be found in, and they* were set up to be part of an IMM system that predicted future locations of the vehicle 3 seconds ahead.

To improve the prediction error of the IMM setup, in Chapter an iterated geometrical error detection method was added based on the GIS system. The assumption made was that the automobile would remain on the road, so predictions of future locations that fell outside of the road were corrected accordingly, making great reduction to prediction error during curves, as shown in the experimental results.

In Chapter 2 this research looked into running the IMM setup at the rate of its fastest sensor to improve the prediction error. The key contribution of the DRWDE system is the introduction of dynamic noise covariance matrix (*Q*). The longer a sensor remains offline, the less accurate the predicted value is, so the dynamic *Q* tells the KF how true the value being used really is, by accurately accounting for the accumulating error the longer a sensor remains offline. The experiments showed improvements in predicted positions between 25%-50%.

Chapter 3 was not about improving the prediction error, but more of an evaluation whether common smartphones could also be used to predict the future locations of a vehicle 3 seconds ahead. Based on the results obtained from the experiment, it seems possible to use the built-in sensors of some of the smartphones to predict a future position, though the use of the accelerometer sensor is easily biased by bumps or inclines on the road.

In summary, this research demonstrated two methods that yield more accurate predictions of future locations of a vehicle:

1. Correct predicted positions using GIS data to ensure that they always fall on a road.
2. Run system at the rate of its fastest sensor while correctly accounting for accumulated error caused by slower sensors.

This research also proposed a possible temporary solution to the new V2V and V2I collision avoidance systems, so that older vehicles can also contribute, making it a more useful system from earlier on, without having to wait many years until most cars on the road support this since they were manufactured.

Comprehensive Bibliography

- T. Abbas, L. Bernado, A. Thiel, C. Mecklenbrauker, F. Tufvesson, "Radio channel properties for vehicular communication: merging lanes versus urban intersections," *Vehicular Technology Magazine, IEEE* , vol. 8, no. 4, pp. 27, 34, December 2013.
- A. T. Alouani, T. R. R., "On optimal asynchronous track fusion," *Proceedings of 1st Australian Symposium on Data Fusion, IEEE*, November 1996.
- A. Amditis, E. Bertolazzi, M. Bimpas, F. Biral, P. Bosetti, M. Da Lio, L. Danielsson, A. Gallione, H. Lind, A. Saroldi, Sjo, A. Gren, "A holistic approach to the integration of safety applications: The INSAFES subproject within the European Framework Programme 6 Integrating Project PReVENT", IEEE Trans. Intell. Trans. Syst., Volume 11, no. 3, pp. 554-566, December 2009.
- L. Armesto, G. Ippoliti, S. Longhi et al., "Probabilistic self-localization and mapping - An asynchronous multirate approach," *Transactions on Robotics & Automation Magazine, IEEE*, vol. 15, no. 2, pp. 77-88, June 2008.
- Y. Bar-Shalom, X. R. L., T. Kirubarajan, "Estimation with applications to tracking and navigation," Wiley and Sons, 2001.
- Bar-Shalom, Yaakov, Li, Xiao-Rong, "Estimation and tracking: Principles, techniques and software", YBS Publishing (1993 by Artec House, Inc.), 1998.
- Y. Bar-Shalom, H. M. Chen, "IMM estimator with out-of-sequence measurements," *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 41, no. 1, pp. 90-98, January 2005.

- Y. Bar-Shalom, X. R. Li, T. Kirubarajan, “Estimation with applications to tracking and navigation”, Wiley and Sons, 2001.
- C. Barrios, Y. Motai, “Improving estimation of vehicle’s trajectory using the latest Global Positioning System with Kalman filtering,” *Transactions on Instrumentation and Measurement, IEEE*, vol. 60, no. 12, pp. 3747-3755, December 2011.
- B. Barshan, H. F. Durrant-Whyte, “Inertial navigation systems for mobile robots”, IEEE International Transactions on Robotics and Automation. Volume II, no. 3, pp. 328-342, June 1995.
- S. M. Bhandarkar, L. Xingzhi, R. F. Daniels, E. W. Tollner, “Automated planning and optimization of lumber production using machine vision and computed tomography,” *Transactions on Automation Science and Engineering, IEEE*, vol. 5, no. 4, pp. 677-695, October 2008.
- R. Bishop, “Intelligent vehicle technology and trends”, Artech House Inc., 2005.
- Y. B.-S.a.H.A.P. Blom, “The interacting multiple model algorithm for systems with Markovian switching coefficients,” *Transactions on Automatic Control, IEEE*, vol. 33, no. 8, August 1988.
- J. J. Blum, A. Eskandarian, “A reliable link-layer protocol for robust and scalable intervehicle communications,” *Intelligent Transportation Systems, IEEE Transactions on* , vol. 8, no. 1, pp. 4,13, March 2007.
- J. Bohg, “Real-time structure from motion using Kalman filtering”, Technische Universitat Dresden. March 2005.
- Chang Bok Lee, Dong Doo Lee, Nak Sam Chung, Ik Soo Chang, E. Kawai, F. Takahashi, “Development of a GPS codeless receiver for ionospheric calibration and

time transfer“, IEEE Transaction on Instrumentation and Measurement, Volume 42, no. 2, pp. 494-497, April 1993.

- B. S. Chen, C. Y. Yang, F. K. Liao, “Mobile location estimator in a rough wireless environment using extended Kalman-based IMM and data fusion,” *Transactions on Vehicular Technology, IEEE*, vol. 58, no. 3, pp. 1157-1169, March 2009.
- M. Chowdhary, “Driver assistance applications based on automotive navigation system infrastructure”, Proceedings from ICCE International Conference in Consumer Electronics, pp. 38-39, June 2002.
- Liu Chunhua, K. T. Chau, Wu Diyun, Gao Shuang, “Opportunities and challenges of vehicle-to-home, vehicle-to-vehicle, and vehicle-to-grid technologies,” *Proceedings of the IEEE* , vol. 101, no. 11, pp. 2409,2427, November 2013.
- Y. Cui, S.S. Ge, “Autonomous vehicle positioning with GPS in urban canyon environments”, IEEE Transactions on Robotics and Automation, Volume 19, no.1, pp. 15-25, February 2003.
- B. Dalla Chiara, F. Deflorio, S. Diwan, “Assessing the effects of inter-vehicle communication systems on road safety,” *Intelligent Transport Systems, IET* , vol. 3, no. 2, pp. 225,235, June 2009.
- E. Derbez, B. Remillard, “The IMM CA CV performance”, unpublished.
- V. Di Lecce, A. Amato, “Route planning and user interface for an advanced intelligent transport system,” *Intelligent Transport Systems, IET* , vol. 5, no. 3, pp. 149-158, September 2011.
- C. Drane, C. Rizos, “Positioning systems in intelligent transportation systems (book style)”, Artech House Inc., 1998.

- S. C. Felter, "An overview of decentralized Kalman filter techniques," *Proceedings of the 1990 Southern Tier Technical Conference, IEEE*, pp. 79-87, April 1990.
- M. Fogue, P. Garrido, F. J. Martinez, J. C. Cano, C. T. Calafate, P. Manzoni, "Automatic accident detection: Assistance through communication technologies and vehicles," *Vehicular Technology Magazine, IEEE* , vol. 7, no. 3, pp. 90,100, September 2012.
- D. Freidman, (2014, Feb. 3). V2V: Cars communicating to prevent crashes, deaths, injuries. US DOT. [Online]. Available: *http://www.dot.gov/fastlane/v2v-cars-communicating-prevent-crashes-deaths-injuries*
- J. Gorselanv, (2013, Mar. 14). Cars that can last for 250,000 miles (or more). Forbes Corp. [Online]. Available: http://www.forbes.com/sites/jimgorzelany/2013/03/14/cars-that-can-last-for-250000-miles/
- C. F. Graetzel, B. J. Nelson, S. N. Fry, "A dynamic region-of-interest vision tracking system applied to the real-time wing kinematic analysis of tethered drosophila," *Transactions on Automation Science and Engineering, IEEE*, vol. 7, no. 3, pp. 463-473, July 2010.
- M. Green, "How long does it take to stop? Methodological analysis of driver perception-brake times", *Transportation Human Factors*, vol. 2, no. 3, pp. 195-216, September 2000.
- M. S. Grewal, A. P. Andrews, "Kalman filtering theory and practice using MATLAB", 2nd edition, p. 132, 2001.

- C. Hakyoung, L. Ojeda, J. Borenstein, "Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope," *Transactions on Robotics and Automation, IEEE*, vol. 17, no. 1, pp. 80-84, February 2001.
- J. B. G. a. C. J. Harris, "Some remarks on Kalman filters for the multisensor fusion," *Elsevier Information Fusion*, vol. 3, pp. 191-201, 2002.
- C. Hilde, T. Moore, M. Smith, "Multiple model Kalman filtering for GPS and low-cost INS integration", Institute of Engineering, Surveying and Space Geodesy, University of Nottingham, 2004.
- H. Himberg, Y. Motai, "Head orientation prediction: Delta quaternions versus quaternions", IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, in press, 2009.
- Y. Ho, R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *Transactions on Automatic Control, IEEE*, vol. 9, no. 4, pp. 333-339, October 1964.
- Holux technology Inc., Holux GR-213 GPS Specifications, http://www.holux.com/JCore/en/support/DLF.jsp?DLU=http://www.holux.com/JCore/UploadFile/7011686.pdf
- L. Hong, "Multirate interacting multiple model filtering for target tracking using multirate models," *Transactions on Automatic Control, IEEE*, vol. 44, no. 7, pp. 1326-1340, July 1999.
- C. Hu, W. Chen, Y. Chen, D. Liu, "Adaptive Kalman filtering for vehicle navigation", Journal of Global Positioning Systems, Volume 2, no. 1, pp. 42-47, 2003.

- M. Hua, T. Bailey, P. Thompson et al., "Decentralised solutions to the cooperative multi-platform navigation problem," *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 47, no. 2, April 2011.
- Y. Ikemoto, Y. Hasegawa, T. Fukuda, K. Matsuda, "Zipping, weaving: Control of vehicle group behavior in non-signalized intersection", IEEE Proceedings of International Conference on Robotics and Automation, Volume 5, pp. 4387-4391, May 2004.
- W. Jiangxin, S. Y. Chao, A. M. Agogiono, "Validation and fusion of longitudinal positioning sensors in AVCS," *Proceedings of the 1999 American Control Conference*, vol. 3, pp. 2178-2182, 1999.
- Jimenez, J. Eugenio Naranjo, "Improving the obstacle detection and identification algorithms of a laserscanner-based collision avoidance system," *Transportation Research Part C-Emerging Technologies,* vol. 19, no. 4, pp. 658-672, August 2011.
- L. A. Johnson, V. Krishnamurthy, "An improvement to the Interactive Multiple Model (IMM) algorithm", IEEE Transaction on Signal Processing, Volume 49, no. 12, December 2001.
- L. A. Johnson, V. K., "An improvement to the Interactive Multiple Model (IMM) algorithm," *Transaction on Signal Processing, IEEE*, vol. 49, no. 12, December 2001.
- R. Kalman, "A new approach to linear filtering and prediction problems," *Transaction ASME*, vol. 82, pp. 34-45, 1960.
- W. Kim, G. Jee, J. Lee, "Efficient use of digital road map in various positioning for ITS", IEEE Transaction on Position Localization and Navigation, San Diego, CA, 2000.

- Lahrech and C. Boucher, J.C. Noyer, "Fusion of GPS and odometer measurements for map-based vehicle navigation.", IEEE Proceedings from the International Conference on Industrial Technology, December 2004.
- M. Lee, Y. Kim, "An efficient multitarget tracking algorithm for car applications", Industrial Electronics, IEEE Transactions on Volume 50, Issue 2, pp. 397-399, April 2003.
- R. C. Luo, C. C. Y., K. L. Su, "Multisensor fusion and integration – Approaches, applications, and future research directions," *Sensors, IEEE*, vol. 2, pp. 107-119, April 2002.
- M. Matosevic, Z. Salcic, S. Berber, "A comparison of accuracy using a GPS and a low-cost DGPS", IEEE Transaction on Instrumentation and Measurement, Volume 55, no. 5, pp. 1677-1683, October 2006.
- E. Mazor, "Interacting multiple model methods in target tracking: A survey," *Transactions on Aerospace and Electronic Systems, IEEE*, vol. 34, no. 1, pp. 103-123, January 1998.
- J. C. Miles, A. J. Walker, "The potential application of artificial intelligence in transport," *Transactions on Intelligent Transport Systems, IEEE*, vol. 153, no. 3, pp. 183-198, September 2006.
- R. Miller, H. Qingfeng, "An adaptive peer-to-peer collision warning system", Proceedings of IEEE 55th Vehicular Technology Conference, Volume 1, pp. 317-321, May 2002.

- B. Mokaberi; A. A. G. Requicha, “Drift compensation for automatic nanomanipulation with scanning probe microscopes,” *Transactions on Automation Science and Engineering, IEEE*, vol. 3, no. 3, pp. 199-207, July 2006.
- R. R. Murphy, “Sensor fusion,” Handbook of Brain Theory and Neural Networks, Bradford Book, 2003.
- M.E. Najjar, P. Bonnifait, “A roadmap matching method for precise vehicle localization using belief theory and Kalman filtering”, IEEE 11th International Conference in Advanced Robotics, Coimbra, Portugal, 2003.
- E. M. Nebot, M. Bozorg, H. F. Durrant-Whyte, “Decentralized architecture for asynchronous sensors,” Autonomous Robots, vol. 6, no. 2, pp. 147-164, April 1999.
- P. N. Pathirana, A. V. Savkin, S. Jha, “Location estimation and trajectory prediction for cellular networks with mobile base stations,” *Transactions on Vehicular Technology, IEEE*, vol. 53, no. 6, pp. 1903-1913, November 2004.
- S. Pietzsch, T. D. Vu, J. Burlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, B. Radig, “Results of a precrash application based on laser scanner and short-range radars,” IEEE Trans. Intell. Transp. Syst., vol. 10, no. 4, pp. 584-593, December 2009.
- S. I. Roumeliotis, G. A. Bekey, “Distributed multirobot localization,” *Transactions on Robotics and Automation, IEEE*, vol. 18, no. 5, pp. 781-795, October 2002.
- X. Shen, Y. Zhu, E. Song et al., “Optimal centralized update with multiple local out-of-sequence measurements,” *Transactions on Signal Processing, IEEE*, vol. 57, no. 4, pp. 1551-1562, April 2009.

- X. Shen, E. Song, Y. Zhu et al., “Globally optimal distributed Kalman fusion with local out-of-sequence-measurement updates,” *Transactions on Automatic Control, IEEE*, vol. 54, no. 8, pp. 1928-1934, August 2009.
- S. Sohaib, D.K.C. So, “Asynchronous cooperative relaying for vehicle-to-vehicle communications,” *Communications, IEEE Transactions on* , vol. 61, no. 5, pp. 1732,1738, May 2013.
- S. Sukkarieh, Low Cost, High Integrity, “Aided inertial navigation systems for autonomous land vehicles”, Ph.D. Thesis from the University of Sydney Australia, 2000.
- T. Taleb, A. Benslimane, K. Ben Letaief, “Toward an effective risk-conscious and collaborative vehicular collision avoidance system,” *Transactions on Vehicular Technology*, IEEE, vol. 59, no. 3, pp.1474-1486, March 2010.
- A. Tascillo, R. Miller, “An in-vehicle virtual driving assistant using neural networks”, Proceedings of the International Joint Conference on Volume 3, pp. 2418-2423, July 2003.
- R. Toledo, M. A. Zamora, B. Ubeda, A. F. Gomez-Skarmeta, “An integrity navigation system based on GNSS/INS for remote services implementation in terrestrial vehicles”, IEEE Proceedings from the Intelligent Transportation Systems Conference, pp. 477-480, Washington, DC, 2004.
- R. Toledo, M. A. Zamora, B. Ubeda, A. F. Gomez, “High integrity IMM-EKF based road vehicle navigation with low cost GPS/INS”, IEEE Transaction on Intelligent Transportation Systems, Volume 8, no. 3, ITISFG, September 2007.

- R. Toledo-Moreo, M. A. Zamora-Izquierdo, “Collision avoidance support in roads with lateral and longitudinal maneuver prediction by fusing GPS/IMU and digital maps,” *Transportation Research Part C-Emerging Technologies*, vol. 18, no. 4, pp. 611-625, August 2010.
- R. Toledo-Moreo, M. Pinzolas-Prado, J. Manuel Cano-Izquierdo, “Maneuver prediction for road vehicles based on a neuro-fuzzy architecture with a low-cost navigation unit,” *Transactions on Intelligent Transportation Systems, IEEE*, vol. 11, no. 2, pp. 498-504, June 2010.
- J. Ueki, J. Mori, Y. Nakamura, Y. Horii, H. Okada, “Development of vehicular-collision avoidance support system by Inter-Vehicle Communications”, Proceedings of IEEE 59th Vehicular Technology Conference, Volume 5, pp. 2940-2945, May 2004.
- Wang, Xuezhi, “Maneuvering target tracking and classification using multiple model estimation theory”, Ph.D. Dissertation, University of Melbourne, 2001.
- F. Wang, A. Broggi, C. C. White, “The road to transactions on intelligent transportation systems: A decade’s duccess [Transactions on ITS],” *Transactions on Intelligent Transportation Systems Magazine, IEEE*, vol. 1, no. 4, pp. 29-32, Winter 2009.
- H. M. Wang, Q. Yin, X. Xia, “Fast Kalman equalization for time-frequency asynchronous cooperative relay networks with distributed space-time codes,” *Transactions on Vehicular Technology, IEEE*, vol. 59, no. 9, pp. 4651-4658, November 2010.

- G. A. Watson, T. R. R., A. T. Alouani, “An IMM architecture for track fusion,” *Signal Processing, Sensor Fusion, and Target Recognition (Proceedings of SPIE 4052)*, vol. IX, pp. 2-13, 2000.
- G. A. Watson, T. R. R., A. T. Alouani, “Optimal track fusion with feedback for multiple asynchronous measurements,” *Proceedings of SPIE Acquisition, Tracking and Pointing XIV*, April 2000.
- H. D. Weerasinghe, R. Tackett, H. Fu, “Verifying position and velocity for vehicular ad-hoc networks,” Security and Communication Networks, vol. 4, no. 7, pp. 785-791, July 2011.
- Welch, Gary Bishop, “An introduction to the Kalman filter”, SIGGRAPH 2001, Course notes.
- Jin Wen-Long, W. Recker, “An analytical model of multihop connectivity of inter-vehicle communication systems,” *Wireless Communications, IEEE Transactions on* , vol. 9, no. 1, pp. 106,112, January 2010.
- S. G. Wu, S. Decker, P. Chang, T. Camus, J. Eledath, “Collision sensing by stereo vision and radar sensor fusion,” IEEE Trans. Intell. Transp. Syst., vol. 10, no. 4, pp. 606-614, Dec. 2009.
- X. Xu, T. Xia, A. Venkatachalam, D. Huston “The development of a high speed ultrawideband ground penetrating radar for rebar detection,” *Journal of Engineering Mechanics*, vol. 139, no. 3, pp. 272-285, March 2013.
- L. C. Yang, J. H. Yang, E. M. Feron, “Multiple model estimation for improving conflict detection algorithms”, IEEE Proceedings from the Conference on Systems, Man and Cybernetecis, Volume 1, pp. 242-249, October 2004.

- P. Zhang, J. Gu, E. Milios, P. Huynh, "Navigation with IMU/GPS/digital compass with unscented Kalman filter", Proceedings of the IEEE International Conference on Mechatronics & Automation, pp. 1497-1502, July 2005.
- X. Zhang, Q. Wang, D. Wan, "Map matching in road crossings of urban canyons based on road traverses and linear heading-change model", IEEE Transaction on Instrumentation and Measurement, Volume 56, no. 6, pp. 2795-2803, December 2007.
- Y. Zhang, H. Hu, H. Zhou, "Study on adaptive Kalman filtering and algorithms in human movement tracking", Proceedings of the IEEE International Conference on Information Acquisition, 2005.

Appendices

A.0. Preface

This section contains information supporting or clarifying the research described in the previous chapters. It contains acronym and symbol definitions, mathematical derivations for formulas introduced, and some representative code used in this research.

Below is a list of the items included in this appendix:

A.1. Acronym Definitions

ASTF	Asynchronous/Synchronous Track Fusion
CA	Constant Acceleration
CJ	Constant Jerk
CL	Constant Location
CPU	Central Processing Unit

CV	Constant Velocity
USDOT	United States Department of Transportation
DR	Dead-Reckoning
DRWDE	Dead-Reckoning with Dynamic Error
FAA	Federal Aviation Administration
GIS	Geographic Information System
GPS	Global Positioning System
IMM	Interacting Multiple Model
ITS	Intelligent Transportation System
KF	Kalman Filter
MSDF	Multi Sensor Data Fusion
OATFA	Optimal Asynchronous Track Fusion Algorithm
RMS	Root Mean Square
SP	Smartphone
TMA	Target Motion Analysis
V2I	Vehicle to infrastructure
V2V	Vehicle to vehicle
VM	Vehicle-Mounted
WAAS	Wide Area Augmentation System

A.2. Symbol Definitions

P	Estimated error covariance
H	Jacobian of the measurement model
A	Jacobian of the system model with respect to state
K	Kalman Gain
Z	Measurement data
σ_m	Measurement noise
R	Measurement noise covariance
N	Number of filters
σ_p	Prediction noise
λ	Probability
Q	Process noise covariance
x	State estimate
Dk	Time interval
p^{ij}	Transition probability matrix

A.3. Mathematical limitation for improved estimations:

Because $x_{i+1} = \left[I + F_{k+1}(t_{i+1} - t_{k+1})\right] + w'_{k+1}(t_{i+1} - t_{k+1})$

$x_{i+1} = \left[I + F_{k+1}(t_{i+1} - t_{k+1})\right]x_{k+1} + w'_{k+1}(t_{i+1} - t_{k+1}) - \left[I + F_{k+1}(t_{i+1} - t_{k+1})\right]u_{k+1}$

the corresponding process covariance matrix will be $m = w'_{k+1}(t_{i+1} - t_{k+1}) - \left[I + F_{k+1}(t_{i+1} - t_{k+1})\right]u_{k+1}$. If φ_l is the vector formed by the elements of row l from F_{k+1}, $\left|\varphi_l \cdot u_{k+1}\right| \le \left\|\varphi_l\right\| \cdot \left\|u_{k+1}\right\|$, and if we operate at $E\left[m_{k+1}^t m_{k+1}\right] \le E\left[\left\|w'_i\right\|^2\right](t_{i+1} - t_{k+1})^2 + E\left[\left\|u_{k+1}\right\|^2\right]\left[1 + \left(\sum \left\|u_{k+1}\right\|\right)(t_{i+1} - t_{k+1})\right]^2$ we can define the trace of the covariance matrix of the process as $E\left[w'^t_i w'_i\right](t_{i+1} - t_{k+1})^2 = E\left[\left\|w'_i\right\|^2\right](t_{i+1} - t_{k+1})^2$, which will show an improvement when:

$$E\left[\left\|w'_i\right\|^2\right](t_{i+1} - t_{k+1})^2 + E\left[\left\|u_{k+1}\right\|^2\right]\left[1 + \left(\sum \left\|u_{k+1}\right\|\right)(t_{i+1} - t_{k+1})\right]^2 < E\left[\left\|w'_i\right\|^2\right](t_{i+1} - t_i)^2$$

which can be rewritten as:

$$E\left[\left\|u_{k+1}\right\|^2\right] < trace(Q_i) \cdot \frac{1 - \left(\dfrac{t_{i+1} - t_{k+1}}{t_{i+1} - t_i}\right)^2}{\left[1 + \left(\sum \left\|\varphi_l\right\|\right)(t_{i+1} - t_{k+1})\right]^2}$$

A.4. Taylor polynomial representation with its respective error:

$$x_p(t) = x_p(t_i) + \frac{x_{p+1}(t_i)}{1!}\Delta t_k + \dots + \frac{x_n(t_i)}{(n-p)!}\left(\Delta t_k\right)^{n-p} + \int_{t_i}^{t} \frac{1}{(n-p)!}\dot{x}_n(t_i - y)^{n-p}\, dy$$

The measurements of the variables will have an error. Given $\overline{x}_l$ the obtained measurement of x_l, the corresponding error $\varepsilon_l = \overline{x}_l - x_l$, in this first step, is due to $w\left(t_i\right)\Delta t_k$. Then we can accordingly modify the Taylor polynomial as shown below:

$$x_p(t_j) = x_p(t_i) + \frac{x_{p+1}(t_i)}{1!}\Delta t_k + \dots + \frac{x_n(t_i)}{(n-p)!}\left(\Delta t_k\right)^{n-p} - \sum_{m=p}^{n} \frac{\varepsilon_m(t_i)}{(m-p)!} + \int_{t_i}^{t} \frac{1}{(n-p)!}\dot{x}_n(t_i - y)^{n-p}\, dy$$

With this procedure, the measurement $\overline{x}_p(t_i)$ of $x_p(t_i)$ will have an error of:

$$\varepsilon_j = \sum_{m=p}^{n} \frac{\varepsilon_m(t_i)}{(m-p)!} + \int_{t_i}^{t} \frac{1}{(n-p)!}\dot{x}_n(t_i - y)^{n-p}\, dy$$

If the function of which we have known measurements in $t_j \in (t_i, t_{i+1})$ is x_0, then:

$$\begin{aligned} x_{p+1}(c) = \dot{x}_p(c) = \frac{\overline{x}_p(t_k) - \overline{x}_p(t_i)}{\Delta t_k} = \\ \overline{x}_{p+1}(t_i) + \frac{\overline{x}_{p+2}(t_i)}{2!}(\Delta t_k)^1 + \dots + \frac{\overline{x}_n(t_i)}{(n-p)!}(\Delta t_k)^{n-p-1} - \\ \frac{1}{\Delta t_k}\left[\varepsilon_p(t_k) - \sum_{m=p}^{n} \frac{\varepsilon_m(t_i)}{(m-p)!} + \int_{t_i}^{t} \frac{1}{(n-p)!}\dot{x}_n(t_i - y)^{n-p}\, dy\right] \end{aligned}$$

For a given $c \in (t_i, t_k)$, we can approximate $x_{p+1}(t_k)$ as:

$$x_{p+1}(t_j) = \overline{x}_{p+1}(t_i) + \frac{\overline{x}_{p+2}(t_i)}{2!}(\Delta t_k)^1 + \dots + \frac{\overline{x}_n(t_i)}{(n-p)!}(\Delta t_k)^{n-p-1}$$

with an error of:

$$\varepsilon_{p+1}(t_j) = x_{p+1}(c) - x_{p+1}(t_k) + \frac{1}{\Delta t_k}\left[\varepsilon_p(t_k) - \sum_{m=p}^{n}\frac{\varepsilon_m(t_i)}{(m-p)!} + \int_{t_i}^{t}\frac{1}{(n-p)!}\dot{x}_n(t_i - y)^{n-p}\,dy\right]$$

where the difference $x_{p+1}(c) - x_{p+1}(t_j)$, which depends on the stability of $x_{p+1}(t)$, is expected to be lower as Δt_k is small.

A.5. Proof of the expected value calculations for each prediction noise (σ_p) element in the process noise covariance (*Q*) matrix to show how to arrive at (6) starting from (5).

Derivation for $E\left[M^2(s)\right]$:

$$E\left[M_k^2(s)\right] = E\left[\left(M_{k-1}(s) + \frac{1}{6}j_k(\Delta k)^3\right)^2\right] =$$

$$E\left[M_{k-1}^2(s) + 2M_{k-1}(s)\frac{1}{6}j_k(\Delta k)^3 + \left(\frac{1}{6}j_k(\Delta k)^3\right)^2\right] =$$

$$E\left[M_{k-1}^2(s)\right] + 2\cdot E\left[M_{k-1}(s)\right]\cdot E\left[\frac{1}{6}j_k(\Delta k)^3\right] + E\left[\frac{1}{36}j_k^2(\Delta k)^6\right] =$$

$$E\left[M_{k-1}^2(s)\right] + E\left[\frac{1}{36}j_k^2(\Delta k)^6\right]$$

And,

$$E\left[\sigma_{p_1}\sigma_{p_1}\right] = E\left[\sigma^2_{p_1}\right] =$$

$$E\left[\left(\frac{1}{6}(j_t \sin\beta + j_n \cos\beta)(\Delta k)^3\right)^2\right] =$$

$$\frac{(\Delta k)^6}{36}\cdot\left(E\left[(j_t^2 \sin^2\beta + j_n^2 \cos^2\beta)\right] + 2E\left[j_t\right]E\left[j_n\right]E\left[\sin\beta + \cos\beta\right]\right) \le$$

$$\frac{(\Delta k)^6}{36}\cdot\left(E\left[(j^2 \sin^2\beta + j^2 \cos^2\beta)\right] + 2\cdot 0\cdot 0\cdot E\left[\sin\beta + \cos\beta\right]\right) =$$

$$\frac{j^2(\Delta k)^6}{36}\cdot E\left[(j_t^2 \sin^2\beta + j_n^2 \cos^2\beta)\right] = \frac{j^2(\Delta k)^6}{36}$$

Therefore,

$$E\left[M_k^2(s)\right] \le \frac{1}{36} j_k (\Delta k)^2 + \frac{1}{36} j_{k+1} (\Delta k)^2 + E\left[M_k^2(s)\right] =$$

$$\frac{1}{36} j_k^2 (\Delta k)^6 + \ldots + \frac{1}{36} j_{k+1}^2 (\Delta k)^6 + E[\sigma_{p_1}^2] \le \frac{(\Delta k)^6}{36}\sum_{i=0}^{m-1} j_{k-i}^2$$

A.6. Representative Visual Basic code

```
Sub Update_Current_Location()

   Locx_prev1 = Locx
   Locy_prev1 = Locy
   Locy = Kalman_Filters.Convert_Deg2Rad(curlat)
   Locx = Kalman_Filters.Convert_Deg2Rad(curlong)
   Vx_prev1 = Vx
   Vy_prev1 = Vy
   Vx = Locx - Locx_prev1
   Vy = Locy - Locy_prev1
   Ax_prev1 = Ax
   Ay_prev1 = Ay
   Ax = Vx - Vx_prev1
   Ay = Vy - Vy_prev1
   Jx_prev1 = Jx
   Jy_prev1 = Jy
   Jx = Ax - Ax_prev1
   Jy = Ay - Ay_prev1

   'Loading Z matrix with measured data
   Z(0, 0) = Locx: Z(1, 0) = Locy: Z(2, 0) = Vx: Z(3, 0) = Vy
   Z(4, 0) = Ax:   Z(5, 0) = Ay:   Z(6, 0) = Jx: Z(7, 0) = Jy
```

```
If (loop_cnt = PERIOD) Then
   B_cnt = 1  'seconds ahead to estimate

   CL_X_(0, 0) = Z(0, 0):   CL_X_(1, 0) = Z(1, 0):   CL_X_(2, 0) =
      Z(2, 0):     CL_X_(3, 0) = Z(3, 0)
   CL_X_(4, 0) = Z(4, 0):   CL_X_(5, 0) = Z(5, 0):   CL_X_(6, 0) =
      Z(6, 0):     CL_X_(7, 0) = Z(7, 0)
   CV_X_(0, 0) = Z(0, 0):   CV_X_(1, 0) = Z(1, 0):   CV_X_(2, 0) =
      Z(2, 0):     CV_X_(3, 0) = Z(3, 0)
   CV_X_(4, 0) = Z(4, 0):   CV_X_(5, 0) = Z(5, 0):   CV_X_(6, 0) =
      Z(6, 0):     CV_X_(7, 0) = Z(7, 0)
   CA_X_(0, 0) = Z(0, 0):   CA_X_(1, 0) = Z(1, 0):   CA_X_(2, 0) =
      Z(2, 0):     CA_X_(3, 0) = Z(3, 0)
   CA_X_(4, 0) = Z(4, 0):   CA_X_(5, 0) = Z(5, 0):   CA_X_(6, 0) =
      Z(6, 0):     CA_X_(7, 0) = Z(7, 0)
   CJ_X_(0, 0) = Z(0, 0):   CJ_X_(1, 0) = Z(1, 0):   CJ_X_(2, 0) =
      Z(2, 0):     CJ_X_(3, 0) = Z(3, 0)
   CJ_X_(4, 0) = Z(4, 0):   CJ_X_(5, 0) = Z(5, 0):   CJ_X_(6, 0) =
      Z(6, 0):     CJ_X_(7, 0) = Z(7, 0)

   'Predict next position with updated filter
   Call CL_model_predict
   Call CV_model_predict
   Call CA_model_predict
   Call CJ_model_predict

ElseIf (loop_cnt > PERIOD) Then

   'Display Current Location
   Set objPin = objMap.FindPushpin("Current Location")
   objPin.Delete
   Set objLoc = objMap.GetLocation(curlat, curlong)
   objMap.AddPushpin objLoc, "Current Location"
   Set objCurLoc = objMap.FindPushpin("Current Location")
   objCurLoc.Symbol = 30 '82
   objCurLoc.Location.GoTo

   '----- EKF Correct Step -----
   Call CL_model_correct   'Correct previous prediction with
      obtained data
   Call CV_model_correct   'Correct previous prediction with
      obtained data
   Call CA_model_correct   'Correct previous prediction with
      obtained data
   Call CJ_model_correct   'Correct previous prediction with
      obtained data

   '----- IMM Steps 1,2,3 -----
   Call MM_filter_part1

   '----- EKF Predict Step -----
   Call CL_model_predict   'Predict next position with updated
      filter
      Call CV_model_predict   'Predict next position with updated
         filter
      Call CA_model_predict   'Predict next position with updated
         filter
      Call CJ_model_predict   'Predict next position with updated
         filter

      '----- IMM Steps 4,5 -----
      Call MM_filter_part2

      'go into this extended loop of MMAE
      If (loop_cnt > PERIOD * 5) Then
      '----------------------------------------
         For B = 2 To 3
               B_cnt = B
               delta_k_loop = 1

               '----- EKF Correct Step -----
               'Correct previous prediction with obtained data
               Call CL_model_correct_loop
               Call CV_model_correct_loop
               Call CA_model_correct_loop
               Call CJ_model_correct_loop

               '----- IMM Steps 1,2,3 -----
               Call MM_filter_part1_loop

               '----- EKF Predict Step -----
               'Predict next position with updated filter
               Call CL_model_predict_loop
               Call CV_model_predict_loop
               Call CA_model_predict_loop
               Call CJ_model_predict_loop

               '----- IMM Steps 4,5 -----
               Call MM_filter_part2_loop

         Next
      '----------------------------------------
      End If

   End If

   loop_cnt = loop_cnt + 1
End Sub


Sub CL_model_correct()

   CL_HP2HT = Mat.Multiply(Mat.Multiply(H, CL_P2), HT)
   CL_VRVT = Mat.Multiply(Mat.Multiply(V, CL_R), VT)
   CL_S = Mat.Add(CL_HP2HT, CL_VRVT)
   CL_Sinv = Mat.Inv(CL_S)

   CL_P2HT = Mat.Multiply(CL_P2, HT)
```

```
    CL_k = Mat.Multiply(CL_P2HT, CL_Sinv)

    CL_X_tmp = Mat.Add(CL_h_, Mat.Multiply(CL_k, Mat.Subtract(Z,
    CL_h_)))
    CL_P = Mat.Multiply(Mat.Subtract(i, Mat.Multiply(CL_k, H)), CL_P2)

    CL_X(0, 0) = CL_X_tmp(0, 0):    CL_X(1, 0) = CL_X_tmp(1, 0):
    CL_X(2, 0) = CL_X_tmp(2, 0):    CL_X(3, 0) = CL_X_tmp(3, 0)
    CL_X(4, 0) = CL_X_tmp(4, 0):    CL_X(5, 0) = CL_X_tmp(5, 0):
    CL_X(6, 0) = CL_X_tmp(6, 0):    CL_X(7, 0) = CL_X_tmp(7, 0)
    CL_X_(0, 0) = CL_X(0, 0):       CL_X_(1, 0) = CL_X(1, 0):
    CL_X_(2, 0) = CL_X(2, 0):       CL_X_(3, 0) = CL_X(3, 0)
    CL_X_(4, 0) = CL_X(4, 0):       CL_X_(5, 0) = CL_X(5, 0):
    CL_X_(6, 0) = CL_X(6, 0):       CL_X_(7, 0) = CL_X(7, 0)

End Sub


Sub CL_model_predict()

    CL_APAT = Mat.Multiply(Mat.Multiply(CL_A, CL_P), CL_AT)
    CL_WQWT = Mat.Multiply(Mat.Multiply(W, CL_Q), WT)
    CL_P2 = Mat.Add(CL_APAT, CL_WQWT)

    CL_h_(0, 0) = CL_X(0, 0): CL_h_(1, 0) = CL_X(1, 0): CL_h_(2, 0) =
    CL_X(2, 0): CL_h_(3, 0) = CL_X(3, 0)
    CL_h_(4, 0) = CL_X(4, 0): CL_h_(5, 0) = CL_X(5, 0): CL_h_(6, 0) =
    CL_X(6, 0): CL_h_(7, 0) = CL_X(7, 0)

End Sub


Sub CV_model_correct()

    CV_HP2HT = Mat.Multiply(Mat.Multiply(H, CV_P2), HT)
    CV_VRVT = Mat.Multiply(Mat.Multiply(V, CV_R), VT)
    CV_S = Mat.Add(CV_HP2HT, CV_VRVT)
    CV_Sinv = Mat.Inv(CV_S)

    CV_P2HT = Mat.Multiply(CV_P2, HT)
    CV_k = Mat.Multiply(CV_P2HT, CV_Sinv)

    CV_X_tmp = Mat.Add(CV_h_, Mat.Multiply(CV_k, Mat.Subtract(Z,
    CV_h_)))
    CV_P = Mat.Multiply(Mat.Subtract(i, Mat.Multiply(CV_k, H)), CV_P2)

    CV_X(0, 0) = CV_X_tmp(0, 0):    CV_X(1, 0) = CV_X_tmp(1, 0):
    CV_X(2, 0) = CV_X_tmp(2, 0):    CV_X(3, 0) = CV_X_tmp(3, 0)
    CV_X(4, 0) = CV_X_tmp(4, 0):    CV_X(5, 0) = CV_X_tmp(5, 0):
    CV_X(6, 0) = CV_X_tmp(6, 0):    CV_X(7, 0) = CV_X_tmp(7, 0)
    CV_X_(0, 0) = CV_X(0, 0):       CV_X_(1, 0) = CV_X(1, 0):
    CV_X_(2, 0) = CV_X(2, 0):       CV_X_(3, 0) = CV_X(3, 0)
    CV_X_(4, 0) = CV_X(4, 0):       CV_X_(5, 0) = CV_X(5, 0):
    CV_X_(6, 0) = CV_X(6, 0):       CV_X_(7, 0) = CV_X(7, 0)
```

```
End Sub


Sub CV_model_predict()

   CV_X(0, 0) = CV_X_(0, 0) + CV_X_(2, 0) * delta_k
   CV_X(1, 0) = CV_X_(1, 0) + CV_X_(3, 0) * delta_k
   CV_X(2, 0) = CV_X_(2, 0)
   CV_X(3, 0) = CV_X_(3, 0)
   CV_X(4, 0) = 0 'CV_X_(4, 0)
   CV_X(5, 0) = 0 'CV_X_(5, 0)
   CV_X(6, 0) = 0 'CV_X_(6, 0)
   CV_X(7, 0) = 0 'CV_X_(7, 0)

   CV_APAT = Mat.Multiply(Mat.Multiply(CV_A, CV_P), CV_AT)
   CV_WQWT = Mat.Multiply(Mat.Multiply(W, CV_Q), WT)
   CV_P2 = Mat.Add(CV_APAT, CV_WQWT)

   CV_h_(0, 0) = CV_X(0, 0): CV_h_(1, 0) = CV_X(1, 0): CV_h_(2, 0) =
   CV_X(2, 0): CV_h_(3, 0) = CV_X(3, 0)
   CV_h_(4, 0) = CV_X(4, 0): CV_h_(5, 0) = CV_X(5, 0): CV_h_(6, 0) =
   CV_X(6, 0): CV_h_(7, 0) = CV_X(7, 0)

End Sub


Sub CA_model_correct()

   CA_HP2HT = Mat.Multiply(Mat.Multiply(H, CA_P2), HT)
   CA_VRVT = Mat.Multiply(Mat.Multiply(V, CA_R), VT)
   CA_S = Mat.Add(CA_HP2HT, CA_VRVT)
   CA_Sinv = Mat.Inv(CA_S)

   CA_P2HT = Mat.Multiply(CA_P2, HT)
   CA_k = Mat.Multiply(CA_P2HT, CA_Sinv)

   CA_X_tmp = Mat.Add(CA_h_, Mat.Multiply(CA_k, Mat.Subtract(Z,
   CA_h_)))
   CA_P = Mat.Multiply(Mat.Subtract(i, Mat.Multiply(CA_k, H)), CA_P2)
   CA_X(0, 0) = CA_X_tmp(0, 0):    CA_X(1, 0) = CA_X_tmp(1, 0):
   CA_X(2, 0) = CA_X_tmp(2, 0):    CA_X(3, 0) = CA_X_tmp(3, 0)
   CA_X(4, 0) = CA_X_tmp(4, 0):    CA_X(5, 0) = CA_X_tmp(5, 0):
   CA_X(6, 0) = CA_X_tmp(6, 0):    CA_X(7, 0) = CA_X_tmp(7, 0)
   CA_X_(0, 0) = CA_X(0, 0):       CA_X_(1, 0) = CA_X(1, 0):
   CA_X_(2, 0) = CA_X(2, 0):       CA_X_(3, 0) = CA_X(3, 0)
   CA_X_(4, 0) = CA_X(4, 0):       CA_X_(5, 0) = CA_X(5, 0):
   CA_X_(6, 0) = CA_X(6, 0):       CA_X_(7, 0) = CA_X(7, 0)

End Sub


Sub CA_model_predict()

   CA_X(0, 0) = CA_X_(0, 0) + CA_X_(2, 0) * delta_k + (1 / 2) *
   CA_X_(4, 0) * delta_k * delta_k
```

```
    CA_X(1, 0) = CA_X_(1, 0) + CA_X_(3, 0) * delta_k + (1 / 2) *
    CA_X_(5, 0) * delta_k * delta_k
    CA_X(2, 0) = CA_X_(2, 0) + CA_X_(4, 0) * delta_k
    CA_X(3, 0) = CA_X_(3, 0) + CA_X_(5, 0) * delta_k
    CA_X(4, 0) = CA_X_(4, 0)
    CA_X(5, 0) = CA_X_(5, 0)
    CA_X(6, 0) = 0 'CA_X_(6, 0)
    CA_X(7, 0) = 0 'CA_X_(7, 0)


    CA_APAT = Mat.Multiply(Mat.Multiply(CA_A, CA_P), CA_AT)
    CA_WQWT = Mat.Multiply(Mat.Multiply(W, CA_Q), WT)
    CA_P2 = Mat.Add(CA_APAT, CA_WQWT)

    CA_h_(0, 0) = CA_X(0, 0): CA_h_(1, 0) = CA_X(1, 0): CA_h_(2, 0) =
    CA_X(2, 0): CA_h_(3, 0) = CA_X(3, 0)
    CA_h_(4, 0) = CA_X(4, 0): CA_h_(5, 0) = CA_X(5, 0): CA_h_(6, 0) =
    CA_X(6, 0): CA_h_(7, 0) = CA_X(7, 0)

End Sub


Sub CJ_model_correct()

    CJ_HP2HT = Mat.Multiply(Mat.Multiply(H, CJ_P2), HT)
    CJ_VRVT = Mat.Multiply(Mat.Multiply(V, CJ_R), VT)
    CJ_S = Mat.Add(CJ_HP2HT, CJ_VRVT)
    CJ_Sinv = Mat.Inv(CJ_S)

    CJ_P2HT = Mat.Multiply(CJ_P2, HT)
    CJ_k = Mat.Multiply(CJ_P2HT, CJ_Sinv)

    CJ_X_tmp = Mat.Add(CJ_h_, Mat.Multiply(CJ_k, Mat.Subtract(Z,
    CJ_h_)))
    CJ_P = Mat.Multiply(Mat.Subtract(i, Mat.Multiply(CJ_k, H)), CJ_P2)

    CJ_X(0, 0) = CJ_X_tmp(0, 0):    CJ_X(1, 0) = CJ_X_tmp(1, 0):
    CJ_X(2, 0) = CJ_X_tmp(2, 0):    CJ_X(3, 0) = CJ_X_tmp(3, 0)
    CJ_X(4, 0) = CJ_X_tmp(4, 0):    CJ_X(5, 0) = CJ_X_tmp(5, 0):
    CJ_X(6, 0) = CJ_X_tmp(6, 0):    CJ_X(7, 0) = CJ_X_tmp(7, 0)
    CJ_X_(0, 0) = CJ_X(0, 0):       CJ_X_(1, 0) = CJ_X(1, 0):
    CJ_X_(2, 0) = CJ_X(2, 0):       CJ_X_(3, 0) = CJ_X(3, 0)
    CJ_X_(4, 0) = CJ_X(4, 0):       CJ_X_(5, 0) = CJ_X(5, 0):
    CJ_X_(6, 0) = CJ_X(6, 0):       CJ_X_(7, 0) = CJ_X(7, 0)

End Sub


Sub CJ_model_predict()

    CJ_X(0, 0) = CJ_X_(0, 0) + CJ_X_(2, 0) * delta_k + (1 / 2) *
    CJ_X_(4, 0) * delta_k * delta_k + (1 / 6) * CJ_X_(6, 0) * delta_k *
    delta_k * delta_k
    CJ_X(1, 0) = CJ_X_(1, 0) + CJ_X_(3, 0) * delta_k + (1 / 2) *
    CJ_X_(5, 0) * delta_k * delta_k + (1 / 6) * CJ_X_(7, 0) * delta_k *
    delta_k * delta_k
    CJ_X(2, 0) = CJ_X_(2, 0) + CJ_X_(4, 0) * delta_k + (1 / 2) *
    CJ_X_(6, 0) * delta_k * delta_k
    CJ_X(3, 0) = CJ_X_(3, 0) + CJ_X_(5, 0) * delta_k + (1 / 2) *
    CJ_X_(7, 0) * delta_k * delta_k
    CJ_X(4, 0) = CJ_X_(4, 0) + CJ_X_(6, 0) * delta_k
    CJ_X(5, 0) = CJ_X_(5, 0) + CJ_X_(7, 0) * delta_k
    CJ_X(6, 0) = CJ_X_(6, 0)
    CJ_X(7, 0) = CJ_X_(7, 0)

    CJ_APAT = Mat.Multiply(Mat.Multiply(CJ_A, CJ_P), CJ_AT)
    CJ_WQWT = Mat.Multiply(Mat.Multiply(W, CJ_Q), WT)
    CJ_P2 = Mat.Add(CJ_APAT, CJ_WQWT)

    CJ_h_(0, 0) = CJ_X(0, 0): CJ_h_(1, 0) = CJ_X(1, 0): CJ_h_(2, 0) =
    CJ_X(2, 0): CJ_h_(3, 0) = CJ_X(3, 0)
    CJ_h_(4, 0) = CJ_X(4, 0): CJ_h_(5, 0) = CJ_X(5, 0): CJ_h_(6, 0) =
    CJ_X(6, 0): CJ_h_(7, 0) = CJ_X(7, 0)

End Sub


Sub MM_filter_part1()

    '--- IMM Step 1 --- Calculation of the mixing probabilities
    For col = 0 To 3
    cb(col) = 0
    For row = 0 To 3
    cb(col) = cb(col) + BT(row, col) * U1(row)
    Next row
    For row = 0 To 3
    U(row, col) = (1 / cb(col)) * BT(row, col) * U1(row)
    Next row
    Next col

    '--- IMM Step 2 --- Mixing
    For r = 0 To 7
    CL_X0(r, 0) = CL_X_(r, 0) * U(0, 0) + CV_X_(r, 0) * U(1, 0) +
    CA_X_(r, 0) * U(2, 0) + CJ_X_(r, 0) * U(3, 0)
    CV_X0(r, 0) = CL_X_(r, 0) * U(0, 1) + CV_X_(r, 0) * U(1, 1) +
    CA_X_(r, 0) * U(2, 1) + CJ_X_(r, 0) * U(3, 1)
    CA_X0(r, 0) = CL_X_(r, 0) * U(0, 2) + CV_X_(r, 0) * U(1, 2) +
    CA_X_(r, 0) * U(2, 2) + CJ_X_(r, 0) * U(3, 2)
    CJ_X0(r, 0) = CL_X_(r, 0) * U(0, 3) + CV_X_(r, 0) * U(1, 3) +
    CA_X_(r, 0) * U(2, 3) + CJ_X_(r, 0) * U(3, 3)

    CL_errj0_(r, 0) = (CL_X_(r, 0) - CL_X0(r, 0))
    CV_errj0_(r, 0) = (CV_X_(r, 0) - CV_X0(r, 0))
    CA_errj0_(r, 0) = (CA_X_(r, 0) - CA_X0(r, 0))
    CJ_errj0_(r, 0) = (CJ_X_(r, 0) - CJ_X0(r, 0))
    Next r

    CL_errj0 = Mat.Multiply(CL_errj0_, Mat.Transpose(CL_errj0_))
```

```
   CV_errj0 = Mat.Multiply(CV_errj0_, Mat.Transpose(CV_errj0_))
   CA_errj0 = Mat.Multiply(CA_errj0_, Mat.Transpose(CA_errj0_))
   CJ_errj0 = Mat.Multiply(CJ_errj0_, Mat.Transpose(CJ_errj0_))

   For r = 0 To 7
   For col = 0 To 7

   CL_P0(r, col) = (CL_P(r, col) + CL_errj0(r, col)) * U(0, 0) +
   (CV_P(r, col) + CV_errj0(r, col)) * U(1, 0) + (CA_P(r, col) +
   CA_errj0(r, col)) * U(2, 0) + (CJ_P(r, col) + CJ_errj0(r, col)) *
   U(3, 0)
   CV_P0(r, col) = (CL_P(r, col) + CL_errj0(r, col)) * U(0, 1) +
   (CV_P(r, col) + CV_errj0(r, col)) * U(1, 1) + (CA_P(r, col) +
   CA_errj0(r, col)) * U(2, 1) + (CJ_P(r, col) + CJ_errj0(r, col)) *
   U(3, 1)
   CA_P0(r, col) = (CL_P(r, col) + CL_errj0(r, col)) * U(0, 2) +
   (CV_P(r, col) + CV_errj0(r, col)) * U(1, 2) + (CA_P(r, col) +
   CA_errj0(r, col)) * U(2, 2) + (CJ_P(r, col) + CJ_errj0(r, col)) *
   U(3, 2)
   CJ_P0(r, col) = (CL_P(r, col) + CL_errj0(r, col)) * U(0, 3) +
   (CV_P(r, col) + CV_errj0(r, col)) * U(1, 3) + (CA_P(r, col) +
   CA_errj0(r, col)) * U(2, 3) + (CJ_P(r, col) + CJ_errj0(r, col)) *
   U(3, 3)

   Next col
   Next r

   'Updating value to KF parameters calculated in Correct Step to be
   used in Predict Step
   For r = 0 To 7
   CL_X_(r, 0) = CL_X0(r, 0)
   CV_X_(r, 0) = CV_X0(r, 0)
   CA_X_(r, 0) = CA_X0(r, 0)
   CJ_X_(r, 0) = CJ_X0(r, 0)

   For col = 0 To 7
   CL_P(r, col) = CL_P0(r, col)
   CV_P(r, col) = CV_P0(r, col)
   CA_P(r, col) = CA_P0(r, col)
   CJ_P(r, col) = CJ_P0(r, col)
   Next col
   Next r

End Sub


Sub MM_filter_part2()

   '--- IMM Step 3 --- Mode matched filtering
   'Likelihood funcion for each of the EKF
   MM0_V = Mat.Subtract(Z, CL_h_)
   MM0_VT = Mat.Transpose(MM0_V)
   MM0_S = Mat.Add(CL_HP2HT, CL_R)
   MM0_IS = Mat.Inv(MM0_S)
   MM0_S2 = Math.Sqr(Mat.Det(MM0_S))
```

```
MM0_X = Mat.Multiply(Mat.Multiply(MM0_VT, MM0_IS), MM0_V)
MM0_X2 = MM0_X(0, 0)
MM0_m = filters 'number of filters
MM0_f = (1 / (((2 * 3.14) ^ (MM0_m / 2)) * MM0_S2)) ^ ((-1 / 2) *
MM0_X2)

MM1_V = Mat.Subtract(Z, CV_h_)
MM1_VT = Mat.Transpose(MM1_V)
MM1_S = Mat.Add(CV_HP2HT, CV_R)
MM1_IS = Mat.Inv(MM1_S)
MM1_S2 = Math.Sqr(Mat.Det(MM1_S))
MM1_X = Mat.Multiply(Mat.Multiply(MM1_VT, MM1_IS), MM1_V)
MM1_X2 = MM1_X(0, 0)
MM1_m = filters 'number of filters
MM1_f = (1 / (((2 * 3.14) ^ (MM1_m / 2)) * MM1_S2)) ^ ((-1 / 2) *
MM1_X2)

MM2_V = Mat.Subtract(Z, CA_h_)
MM2_VT = Mat.Transpose(MM2_V)
MM2_S = Mat.Add(CA_HP2HT, CA_R)
MM2_IS = Mat.Inv(MM2_S)
MM2_S2 = Math.Sqr(Mat.Det(MM2_S))
MM2_X = Mat.Multiply(Mat.Multiply(MM2_VT, MM2_IS), MM2_V)
MM2_X2 = MM2_X(0, 0)
MM2_m = filters 'number of filters
MM2_f = (1 / (((2 * 3.14) ^ (MM2_m / 2)) * MM2_S2)) ^ ((-1 / 2) *
MM2_X2)

MM3_V = Mat.Subtract(Z, CJ_h_)
MM3_VT = Mat.Transpose(MM3_V)
MM3_S = Mat.Add(CJ_HP2HT, CJ_R)
MM3_IS = Mat.Inv(MM3_S)
MM3_S2 = Math.Sqr(Mat.Det(MM3_S))
MM3_X = Mat.Multiply(Mat.Multiply(MM3_VT, MM3_IS), MM3_V)
MM3_X2 = MM3_X(0, 0)
MM3_m = filters 'number of filters
MM3_f = (1 / (((2 * 3.14) ^ (MM3_m / 2)) * MM3_S2)) ^ ((-1 / 2) *
MM3_X2)

'--- IMM Step 4 --- Mode probability update
c = MM0_f * cb(0) + MM1_f * cb(1) + MM2_f * cb(2) + MM3_f * cb(3)
U1(0) = (1 / c) * MM0_f * cb(0)
U1(1) = (1 / c) * MM1_f * cb(1)
U1(2) = (1 / c) * MM2_f * cb(2)
U1(3) = (1 / c) * MM3_f * cb(3)

'--- IMM Step 5 --- For OUTPUT purposes only (not part of algorithm
recursions)

For r = 0 To 7
C_X(r, 0) = CL_h_(r, 0) * U1(0) + CV_h_(r, 0) * U1(1) + CA_h_(r, 0)
* U1(2) + CJ_h_(r, 0) * U1(3)
CL_errj_(r, 0) = CL_h_(r, 0) - C_X(r, 0)
CV_errj_(r, 0) = CV_h_(r, 0) - C_X(r, 0)
CA_errj_(r, 0) = CA_h_(r, 0) - C_X(r, 0)
```

```
    CJ_errj_(r, 0) = CJ_h_(r, 0) - C_X(r, 0)
    Next r

    CL_errj = Mat.Multiply(CL_errj_, Mat.Transpose(CL_errj_))
    CV_errj = Mat.Multiply(CV_errj_, Mat.Transpose(CV_errj_))
    CA_errj = Mat.Multiply(CA_errj_, Mat.Transpose(CA_errj_))
    CJ_errj = Mat.Multiply(CJ_errj_, Mat.Transpose(CJ_errj_))

    For r = 0 To 7
    For col = 0 To 7
    C_P(r, col) = (CL_P2(r, col) + CL_errj(r, col)) * U1(0) + (CV_P2(r,
    col) + CV_errj(r, col)) * U1(1) + (CA_P2(r, col) + CA_errj(r, col))
    * U1(2) + (CJ_P2(r, col) + CJ_errj(r, col)) * U1(3)
    Next col
    Next r

    If (loop_cnt > PERIOD * 5) Then
    If SNAP2ROAD Then
    Call Snap_to_Road_2.Start(Kalman_Filters.Convert_Rad2Deg(C_X(0, 0) -
    offset_lat), Kalman_Filters.Convert_Rad2Deg(C_X(1, 0) - offset_lon),
    Kalman_Filters.Convert_Rad2Deg(Z(0, 0)),
    Kalman_Filters.Convert_Rad2Deg(Z(1, 0)))
    C_X(0, 0) = Kalman_Filters.Convert_Deg2Rad(curlong) + offset_lon
    C_X(1, 0) = Kalman_Filters.Convert_Deg2Rad(curlat) + offset_lat
    '---------------------------------------------------
    Set objPin = objMap.FindPushpin("Estimated Location b")
    objPin.Delete
    estlong = Convert_Rad2Deg(C_X(0, 0))
    estlat = Convert_Rad2Deg(C_X(1, 0))
    Set objLoc = objMap.GetLocation(estlat, estlong)
    objMap.AddPushpin objLoc, "Estimated Location b"
    Set objCurLoc = objMap.FindPushpin("Estimated Location b")
    objCurLoc.Symbol = 25
    '---------------------------------------------------
    Else
    '---------------------------------------------------
    Set objPin = objMap.FindPushpin("Estimated Location")
    objPin.Delete
    estlong = Convert_Rad2Deg(C_X(0, 0))
    estlat = Convert_Rad2Deg(C_X(1, 0))
    Set objLoc = objMap.GetLocation(estlat, estlong)
    objMap.AddPushpin objLoc, "Estimated Location"
    Set objCurLoc = objMap.FindPushpin("Estimated Location")
    objCurLoc.Symbol = 26
    '---------------------------------------------------
    End If
    End If

    XM(0) = C_X(0, 0)   'estimated x location
    XM(1) = C_X(1, 0)   'estimated y location

End Sub
```

A.7. Representative Matlab code

```
function [OUT_val,OUT_err,OUT_data]=sf_main(data)
%  s* indicates available systems: s1 (sensor1=GPS), s2
(sensor2=ScanTool), s3 (sensor3=Accelerometer),
%  s*_ekf are lists of EKF filters (each number represents an EKF id
supported in that system)
%  s*_P{#}, s*_W{#}, s*_Q{#}, s*_A{#}, s*_X{#} are arrays of matrices
where "#" indicates EKF id values the matrix is for, and s* indicates
for what system
%  s*_H, s*_V, s*_I, s*_R, s*_U1, s*_BT, s*_Z are matrices shared
between all the EKFs in each system (no # needed)

data_name = inputname(1); %save name of data array passed

%-----------------------------------------------------
%VARIABLES SETTING SECTION
%-----------
Dk_orig = 0.1;           %in seconds (0.1 for 10Hz) NOTE: anything less
than 1 requires IMM to be running
sensors = [1 2 3];       %define which sensors to use in the system
options=1,2,3
ekfs = [1 2 3];          %number of KFs in use (can NOT change this
without affecting BT)
use_ekf = 1;             %set to 0 for estimation of Z only, or set to
1 to run system
use_imm = 1;             %set to 0 for EKF only run (no IMM), or set to
1 for IMM run as well
est_sec_ahead = 3;       %set to far estimation location 3 seconds
ahead (must also set use_imm=1), OTHERWISE set to 0
est_sec_toGPS = 0;       %0=always estimate $est_sec_ahead; 1=adjust
estimation to always match record with GPS value
if Dk_orig == 1
   est_mid_points = 0;      %1=estimate intermediate points between
   est_sec_ahead and est_sec_ahead+0.9
end

use_Q_calc_vars = 0;     %1=uses dynamic Q variable; 0=uses simple Q
variable
calc_missing_values = 1; %1=calc missing values based on online
sensors;  0=use IMM estimated values
gps_difORtot = 1;        %1=diff between starting point and current
value, 2=full value (for the location units)
%-----------

[rows,cols] = size(data);%Get size of data matrix
tot_recs=rows-50;        %Define total number of records to process;
default is all minus last 50
tot_loops=rows;          %Define max number of loops allowed (set to
large number if not used)
s_loop_start = 100;      %number of loop_count (rows of data) to start
system on [sensors section] (must be >3)
selected_curve = 0;      %1=run selected curve only, 0=run whole
trajectory
if strcmpi(data_name,'data_UConn')
   rec=17;               %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=206;
   if selected_curve == 1
      rec=8901;
      tot_loops=1000;
   end
elseif strcmpi(data_name,'data_Alcatel')
   rec=20;              %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=3500;
   if selected_curve == 1
      rec=15800;
      tot_loops=1000;
   end
elseif strcmpi(data_name,'data_HTC')
   rec=20;              %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=3500;
   if selected_curve == 1
      rec=15800;
      tot_loops=1000;
   end
elseif strcmpi(data_name,'data_LG')
   rec=20;              %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=3500;
   if selected_curve == 1
      rec=15800;
      tot_loops=1000;
   end
elseif strcmpi(data_name,'data_iPhone3GS')
   rec=18;              %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=3498;
   if selected_curve == 1
      rec=15798;
      tot_loops=1000;
   end
elseif strcmpi(data_name,'data_iPhone5')
   rec=18;              %Define starting record (minus 1) to read (can
   not have a zero for GPS data)
   rec=3498;
   if selected_curve == 1
      rec=15808;
      tot_loops=1000;
   end
else
   disp(['data array name not supported: ' data_name]);
   return;
end

%-----------------------------------------------------
%Initializing variables that will hold the data from the sensors for
the different seconds
Dks1=0;%Dks1 will contain the gap between each set of data for s1
Dks2=0;%Dks2 will contain the gap between each set of data for s2
```

```
Dks3=0;%Dks3 will contain the gap between each set of data for s3
%EKF initialize step (defines all corresponding variables for all EKFs
per sensor in use)
[H,I,A,P,BT,U1,U]=ekf_initialize(sensors,ekfs,Dk_orig);
%----------------------------------------------------

%Loop through records in data array while rec < tot_recs

Dk_prev=Dk;
loop_count = loop_count+1;

while loop_count <= tot_loops & rec <= tot_recs

   %Loading new data into variables
   if strcmpi(data_name,'data_UConn')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_UConn(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_lat
      _orig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s_l
      oop_start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_Alcatel')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_Alcatel(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_l
      at_orig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s
      _loop_start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_HTC')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_HTC(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_lat_o
      rig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s_loo
      p_start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_LG')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_LG(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_lat_or
      ig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s_loop
      _start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_iPhone3GS')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_iPhone3GS(Dk,Dk_orig,sensors,data,rec,tot_recs,gps
      _lat_orig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count
      ,s_loop_start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_iPhone4')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_iPhone4(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_l
      at_orig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s
      _loop_start,gps_dir,selected_curve);
   elseif strcmpi(data_name,'data_iPhone5')
      [sensor_status,gps_lat2,gps_lon2,gps_dir2,gps_vel2,sct_vel2,acc_a
      vx2,acc_avy2,rec,Dk,Dk_prev,cur_sec,prev_sec,next_sec,cur_min,cur
      _hr]=load_data_iPhone5(Dk,Dk_orig,sensors,data,rec,tot_recs,gps_l
      at_orig,gps_lon_orig,gps_difORtot,use_rolling_window,loop_count,s
      _loop_start,gps_dir,selected_curve);
   end

   %Update Dk for each of the sensors to keep track of time since last
   time it was online
   if sensor_status(1,1) == 0
      Dks1 = Dks1 + Dk;
   else
      Dks1 = 0;
   end
   if sensor_status(2,1) == 0
      Dks2 = Dks2 + Dk;
   else
      Dks2 = 0;
   end
   if sensor_status(3,1) == 0
      Dks3 = Dks3 + Dk;
   else
      Dks3 = 0;
   end

   %LOAD Z matrices for each sensor
   %assumes each EKF for the same sensor will have the same matrix size
   [6x1]

   %--- acceleration --------------
   if sensor_status(3,1)==1 %if acc is online use measured data
      Z(5,1) = acc_avx(1);
      Z(6,1) = acc_avy(1);
      ax2 = Z_prev(5,1);
      ay2 = Z_prev(6,1);
      %determining vectors based on acceleration and previous direction
      Anx = ax2*cos(gps_dir(2));
      Atx = ay2*sin(gps_dir(2));
      Any = ax2*sin(gps_dir(2));
      Aty = ay2*cos(gps_dir(2));
      if ay2 > 0
         Ax  = Atx + Anx;
         Ay  = Aty - Any;
      else
         Ax  = Atx - Anx;
         Ay  = Aty + Any;
      end
      %Calculate new velocities based on new accelerations
      Vx  = Z_prev(3,1) + Ax *Dk_prev;
      Vy  = Z_prev(4,1) + Ay *Dk_prev;
      %Calculate new positions based on new accelerations
      Sx = Z_prev(1,1) + ( Z_prev(3,1) )*Dk_prev + (1/2)*( Ax
      )*Dk_prev^2;
      Sy = Z_prev(2,1) + ( Z_prev(4,1) )*Dk_prev + (1/2)*( Ay
      )*Dk_prev^2;
      %Recalculate angle of direction based on changes in location
      dx=Sx-Z_prev(1,1);
      dy=Sy-Z_prev(2,1);
      if dy == 0
         if dx >0
            gps_dir(1) = pi/2;
         else
            gps_dir(1) = 3*pi/2;
         end
      else
         gps_dir(1) = abs( atan( abs(dx) / abs(dy) ) );
         if dx >0 && dy >0  % 1st quadrant
            gps_dir(1) = gps_dir(1);
         elseif dx >0 && dy <0  % 2nd quadrant
            gps_dir(1) = (pi) - gps_dir(1);
         elseif dx <0 && dy <0  % 3rd quadrant
            gps_dir(1) = (pi) + gps_dir(1);
         else
            gps_dir(1) = (2*pi) - gps_dir(1);
         end
      end
   else
      if use_imm == 1
         Z(5,1) = X_imm(5,1);  %using previously estimated value (we
         don't want to derive it from location for now)
         Z(6,1) = X_imm(6,1);  %using previously estimated value (we
         don't want to derive it from location for now)
      else
         Z(5,1) = Z_prev(5,1);  %using previous value
         Z(6,1) = Z_prev(6,1);  %using previous value
      end
      gps_dir(1) = gps_dir(2);
   end

   %--- GPS is online -------
   if sensor_status(1,1)==1  %If GPS is online use measured data
      %location
      Z(1,1)=gps_lon(1); %x
      Z(2,1)=gps_lat(1); %y
      gps_dir(1) = double(gps_dir2); %use actual data if sensor is
      online and ignore angle calculated when s3 is on
   else                      %If GPS is offline then use previously
   estimated data to assume current location
      %location
      if calc_missing_values == 1 && sensor_status(3,1) == 1
         Z(1,1)=Sx; %x
         Z(2,1)=Sy; %y
      else
         if use_imm == 1
            Z(1,1)=X_imm(1,1); %x
            Z(2,1)=X_imm(2,1); %y
         else
            Z(1,1)=Z_prev(1,1); %x
            Z(2,1)=Z_prev(2,1); %y
         end
      end
   end
```

```
%--- Velocity ------------
if sensor_status(1,1)==1  || sensor_status(2,1)==1 %if GPS is online
use measured data
   if sensor_status(2,1)==1 & ( sct_vel(1) > 0 | sct_vel(1) < 0 )
   %If ST is online use measured data (ST measurement preferred over
   GPS)
      Z(3,1) = sct_vel(1)*sin(gps_dir(1));
      Z(4,1) = sct_vel(1)*cos(gps_dir(1));
   else
      Z(3,1) = Z_prev(3,1);
      Z(4,1) = Z_prev(4,1);
   end
else
   Z(3,1) = Vx;
   Z(4,1) = Vy;
end

if loop_count < s_loop_start

   %----- Initialization stage for the system --------
   %KF prediction step for sensors

   [R,Q]=ekf_update(sensor_status,ekfs,Dk,Dks1,Dks2,Dks3,use_Q_calc_
   vars,use_R_calc_vars); %updating R and Q matrices to use the
   current Dk
   %Loop through each KF defined
   [rows,cols] = size(ekfs);
   n=cols;
   for f=1:n
      if use_ekf == 1
         [ekf_P] = ekf_predict(sensor_status,A{f},P{f},Q{f});
         P{f}=double(ekf_P);
      end
      [ekf_X] =
      ekf_models(rec,1,ekfs(f),sensor_status,Dk,Dk_orig,Z,Z_prev,gps
      _dir,acc_avx,acc_avy,time);
      X{f}=double(ekf_X);
   end


else

   %----- Running stage for the system --------------
   %KF correct step for sensors
   %Loop through each KF defined
   [rows,cols] = size(ekfs);
   n = cols;
   for f=1:n
      if use_ekf == 1
         [ekf_X,ekf_P]=ekf_correct(ekfs(f),H,P{f},R{f},I,Z,X{f});
         P{f}=double(ekf_P);
         X{f}=double(ekf_X);
      end
   end
```

```
%IMM_part1
[X,P,cb,U]=imm_part1(sensor_status,sensors,ekfs,X,P,BT,U1,U,cb);

%KF prediction step for sensors
[R,Q]=ekf_update(sensor_status,ekfs,Dk,Dks1,Dks2,Dks3,use_Q_calc_
vars,use_R_calc_vars); %updating R and Q matrices to use the
current Dk
%Loop through each KF defined
[rows,cols] = size(ekfs);
n = cols;
for f=1:n
   if use_ekf == 1
      [ekf_P] = ekf_predict(sensor_status,A{f},P{f},Q{f});
      P{f}=double(ekf_P);
   end
   [ekf_X] =
   ekf_models(rec,1,ekfs(f),sensor_status,Dk,Dk_orig,Z,Z_prev,gps
   _dir,acc_avx,acc_avy,time);
   X{f}=double(ekf_X);
end

%IMM_part2
[U1,X_imm,mm_f] =
imm_part2(sensor_status,sensors,ekfs,X,P,H,R,cb,mm_f);

%Estimating position 3 seconds ahead.
if est_sec_ahead > 0
   Dk2 = est_sec_ahead;
   %Run KF again but this time using a larger Dk
   [rows,cols] = size(ekfs);
   n = cols;
   for f=1:n
      [ekf_X_ahead] =
      ekf_models(rec,1,ekfs(f),sensor_status,Dk2,Dk_orig,Z,Z_prev
      ,gps_dir,acc_avx,acc_avy,time);
      if ekfs(f)==3 && use_geom_method == 1
         [ekf_X_ahead,Ap,Cp] =
         correction(ss0,ekf_X_ahead,est0,Ap,Cp,ang1,ang2,Z,Z0,rec
         ,cur_hr,cur_min,cur_sec);
         est0 = ekf_X_ahead;
      end
      X_ahead{f}=ekf_X_ahead;
   end

   if use_imm == 1
      %Run IMM_part2 to merge the results from the KF for this
      3sec ahead estimation
      [U1_ahead,X_imm_ahead,mm_f_ahead] =
      imm_part2(sensor_status,sensors,ekfs,X_ahead,P,H,R,cb,mm_f)
      ;
   else
      X_ahead = X;
      X_imm_ahead = X_imm;
   end
```

```
         loop_count = loop_count+1;

      end


   end

loop_count = loop_count+1;
end


function [P2]=ekf_predict(status, A, Pf, Qf)
   APAT = A*Pf*A';
   P2   = APAT+Qf;
   return;
end


function [X2, P2]=ekf_correct(ekf, H, P0, R, I, Z, X0)
   HPHT = H*P0*H';
   S = HPHT + R;  %for KF only
   %Determine how many elements in diagonal are important for this KF
   if ekf == 1
      d = 2;
   elseif ekf == 2
      d = 4;
   else
      d = 6;
   end
   %--removing zeros from the diagonal to be able to do the inverse
   [rows,cols] = size(S);
   for r=d+1:rows
      for c=d+1:cols
         if r == c
            if S(r,c) > -0.0001 & S(r,c) < 0.0001
               S(r,c) = 1;
            end
         end
      end
   end
   k2 = inv(S);
   %--adding zeros back to the diagonal to maintain matrix properties
   [rows,cols] = size(k2);
   for r=d+1:rows
      for c=d+1:cols
         if r == c & k2(r,c) == 1
            k2(r,c) = 0;
         end
      end
   end

   k = k1*k2;
   X2 = X0+(k*(Z-X0));
   P2 = (I-(k*H)) * P0;
   return;
end


function [X2]=ekf_models(rec,sensor, ekf, status, Dk, Dk_orig, Z,
Z_prev, gps_dir,acc_avx,acc_avy,time)
   %---EKF1 - const_location-------------------
   if ekf == 1
      X2(1,1) = Z(1,1);
      X2(2,1) = Z(2,1);
      X2(3,1) = 0;
      X2(4,1) = 0;
      X2(5,1) = 0;
      X2(6,1) = 0;
   end

   %---EKF2 - const_speed----------------------
   if ekf == 2
      X2(1,1) = Z(1,1) + Z(3,1)*Dk;
      X2(2,1) = Z(2,1) + Z(4,1)*Dk;
      X2(3,1) = Z(3,1); %constant velocity
      X2(4,1) = Z(4,1); %constant velocity
      X2(5,1) = 0;      %no acceleration
      X2(6,1) = 0;      %no acceleration
   end

   %---EKF3 - const_acc------------------
   if ekf == 3
      An=Z(5,1); %Saving An as we will assume it does not change
      through the next 3 seconds
      At=Z(6,1); %Saving At as we will assume it does not change
      through the next 3 seconds
      if sqrt( Z(3,1)^2 + Z(4,1)^2 ) == 0
         Ax = 0;
         Ay = 0;
      else
         if At>0
            Ax = ( At*Z(3,1) + An*Z(4,1) ) / sqrt( Z(3,1)^2 + Z(4,1)^2
            );
            Ay = ( At*Z(4,1) - An*Z(3,1) ) / sqrt( Z(3,1)^2 + Z(4,1)^2
            );
         else
            Ax = ( At*Z(3,1) - An*Z(4,1) ) / sqrt( Z(3,1)^2 + Z(4,1)^2
            );
            Ay = ( At*Z(4,1) + An*Z(3,1) ) / sqrt( Z(3,1)^2 + Z(4,1)^2
            );
         end
      end
      Vx = Z(3,1) + Ax *Dk;
      Vy = Z(4,1) + Ay *Dk;

      X2(1,1) = Z(1,1) + ( Z(3,1) )*Dk + (1/2)*( Ax )*Dk^2;
      X2(2,1) = Z(2,1) + ( Z(4,1) )*Dk + (1/2)*( Ay )*Dk^2;
      X2(3,1) = Vx;
      X2(4,1) = Vy;
      X2(5,1) = Z(5,1); %constant acceleration sideways
      X2(6,1) = Z(6,1); %constant acceleration forwards
   end

   return;
end


function [X0,P0,cb0,U0]=imm_part1(ekf_status, sensors, ekfs, X, P, BT, 
U1, U, cb)
   %Get total number of EKFs defined
   [rows,cols] = size(ekfs);
   total_ekfs=cols;

   %--- IMM step 1 --- Calculation of the mixing probabilities
   for c=1:total_ekfs
      cb0(c,1)=0;
      for r=1:total_ekfs
         ttt=cb0(c,1);
         cb0(c,1)=cb0(c,1)+BT(r,c)*U1(r,1);
      end
      if(cb0(c,1) <= 0)
         cb0(c,1)=0.0001;
      end
      for r=1:total_ekfs
         U0(r,c)=(1/cb0(c,1))*BT(r,c)*U1(r,1);
         if(U0(r,c) <= 0)
            U0(r,c)=0.0001;
         end
      end
   end

   %--- IMM step 2 --- Mixing
   for f=1:total_ekfs
      X_ekf  = X{f};
      for r=1:6   %total rows in Z
         X0_ekf(r,1)=0;
         for c=1:total_ekfs
            X0_ekf(r,1)= X0_ekf(r,1)+ (X_ekf(r,1)*U0(c,f));
         end
         errj0_(r,1)= X_ekf(r,1)-X0_ekf(r,1);
      end
      errj0{f} = errj0_*errj0_';
      X0{f} = X0_ekf;
   end

   for f=1:total_ekfs
      for r=1:6   %total rows in Z
         for c=1:6   %total rows in Z
            P0_ekf(r,c)=0;
            for j=1:total_ekfs
               P_ekf = P{j};
               errj0_ekf = errj0{j};
               P0_ekf(r,c) = P0_ekf(r,c) + 
               (P_ekf(r,c)+errj0_ekf(r,c))*U0(j,f);
            end
         end
      end
      P0{f} = P0_ekf;
   end

   return;
end


function [U1,X_imm,mm_f1]=imm_part2(ekf_status, sensors, ekfs, X, P, H,
R, cb, mm_f)
   %Get total number of KFs in use
   [rows,cols] = size(ekfs);
   total_ekfs=cols;

   %--- IMM step 3 --- Mode matched filtering
   for f=1:total_ekfs
      X_ekf = X{f};
      P_ekf = P{f};
      R_ekf = R{f};
      mm_s2   = abs(det(H*P_ekf*H'+R_ekf));
      HPHT = H*P_ekf*H';

      %---Determine how many elements in diagonal are important for
      this KF
      if ekfs == 1
         d = 2;
      elseif ekfs == 2
         d = 4;
      else
         d = 6;
      end

      %--removing zeros from the digonal to be able to do the inverse
      [rows,cols] = size(HPHT);
      for r=d+1:rows
         for c=d+1:cols
            if r == c
               if HPHT(r,c) > -0.000001 & HPHT(r,c) < 0.000001
                  HPHT(r,c) = 1;
               end
            end
         end
      end

      %--doing the inverse of HPHT
      HPHT_inv = inv(HPHT);

      %--adding zeros back to the diagonal to maintain matrix
      properties
      [rows,cols] = size(HPHT_inv);
      for r=d+1:rows
         for c=d+1:cols
            if r == c & HPHT_inv(r,c) == 1
```

```
                 HPHT_inv(r,c) = 0;
              end
           end
        end
        %--------
        mm_x2   = det( HPHT_inv ) ;
        mm_f_ekf = (1/sqrt( ((2*pi)^(total_ekfs/2) )*mm_s2 )) *exp((-
        1/2)*mm_x2);
        mm_f1(f,1) = mm_f_ekf;

     end

     %--- IMM step 4 --- Mode probability update
     c = 0;
     for f=1:total_ekfs
        ttt = c;
        c = c + mm_f1(f,1)*cb(f,1);
     end

     for f=1:total_ekfs
        U1(f,1) = (1/c) * mm_f1(f,1) * cb(f,1);
        if(U1(f,1) <= 0)
           U1(f,1)=0.0001;
        end
     end

     return;
  end
```