Abstract—This paper looks into various practical methods of improving existing collision avoidance systems that lose accuracy in non-straight paths. Different frameworks of models are studied to identify their accuracy in predicting a vehicle’s future location in non-straight path. From a simple mathematical approach to more complex frameworks of multiple Kalman Filters are evaluated to obtain more accurate collision warnings in situations that current systems can not handle correctly. To make any improvements cost effective, only a Global Positioning System (GPS) receiver was used to obtain all spatial information related to the vehicle. This paper goes through the implementation of several Extended Kalman Filters (EKF) to achieve full coverage of the possible vehicle movements. It then goes on to create multiple model frameworks that combine the different EKF to provide even more accurate estimations of a vehicle’s trajectory. The two frameworks studied in this paper are the Multiple Models Adaptive Estimation System (MMAE) and the Interacting Multiple Models Estimation (IMME). Both the MMAE and the IMME have proven very reliable in the robotics field, so they should also work in the field of automobiles. The last improvement this paper investigates is the integration of the Geographic Information System (GIS) into this system. Road information is available in the GIS data, and spatial information is available through the GPS receiver, therefore, this paper investigates a way to verify that the predicted vehicle’s location is on actual roads. This innovative method reduces the error by more than half when used as part of the system.

I. INTRODUCTION

A vehicle avoidance system by using sensors around the car is one of many ideas behind collision avoidance systems. Engineers have been chipping away at the staggering number of fatalities for a long time by designing air bags and seat belts, stronger frames and special interior design to increase the safety of a car. However the only way to save far more lives is to keep cars from colliding into each other in the first place [8]. Previous research have experimented by placing sensors in the front of a vehicle to have the car’s computer maintain a safe distance from the car in front; sensors in the back to be activated only when in reverse to estimate space behind the car; and sensors by the side mirrors to detect objects in the blind spots and prevent collisions when lanes merge or cars change lanes [8].

Another study investigated a method to calculate the suggested safe distance to follow a car. It took into account safety zones around the car in case of maneuvering to avoid a collision. Results showed that the safe distance depended on the car and driver’s awareness, and illustrated how hard it is to get a good system to work well and not give warnings too often [2].

The next step for many of these warning systems is to implement automatic braking capabilities so that they do not need to rely on driver’s capabilities [3]-[11]. In [7], researchers experimented with a dynamic model for brake control using a solenoid-valve-controlled hydraulic brake actuator system. They came up with a proposed brake control law that can provide the collision warning and collision avoidance vehicles with an optimized compromise between safety and comfort [7].

Systems like the ones described above are limited to line of sight for the sensors to detect other vehicles. Their accuracy is also inconsistent as speed and direction varies. The best way to prevent vehicle collisions is to know where vehicles are at all times, where they are heading, and where they will be in the future. Having this knowledge would allow systems to calculate if vehicles’ paths might intersect in the near future and warn a driver of a possible collision if it were a passive system, or apply the brakes automatically in case of an active system.

These types of complex and dynamic collision avoidance systems take into consideration the location of other vehicles nearby, even if not in line of sight. Researchers like the one at the Kansai University of Japan [1] or the one by Miller and Huang [6] investigate the option of implementing inter-vehicle communication to be able to, through some judgment algorithm, identify if the trajectory of the vehicles will intersect and possibly collide using Global Positioning System (GPS) data collected from the different vehicles. The methods used to estimate the intersection of the paths are somewhat simple and do not give accurate results in scenarios like curves where the estimated future position of the vehicles will not be a straight path.

It is clear that to have better collision avoidance systems we need a more accurate way to estimate the trajectory of the vehicles in all different scenarios. This is where the Kalman Filter (KF) comes into play. The KF has a long
Similar systems designed to estimate 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 cheap but accurate GPS receiver to do a similar task and also add the benefits of a location based system as already implemented in some areas [13]-[15], [19], [20], [22].

To evaluate the need for the extensive mathematical computations a Kalman Filter (KF) framework requires, some Simple Estimations (SE) will be define which do not take into account any previous data. These estimation models will not require much processor load, which would be excellent for a real-time system. But, even though the SE might be fast they might not be as accurate as using a Kalman Filter, and considering we are already using a GPS device which has some error, we do not want to loose more accuracy in the other steps.

The conventional Kalman filtering algorithm requires the definition of a dynamic and stochastic model. The dynamic model describes how the errors that are modeled develop over time, whereas the stochastic model describes the noise of the new measurements and the stochastic properties of the process being modeled [21]-[37].

Because a vehicle can move in very different ways, to be able to estimate or predict its trajectory we need to define different models. Each model will be 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 a vehicle can be found in. Three models have been identified that seem to cover all vehicles’ behaviors: a vehicle traveling at constant velocity, or with constant acceleration, or with constant jerk (change in acceleration). These models, whether by themselves or a combination of them, should be able to cover a vehicle’s movement accurately. The different models provide a mathematical set of equations that can be used to estimate the vehicle’s future location after a set amount of time. But, even though three models have been identified, there are many ways they can be implemented. Two obvious ways these models can be implemented is without taking into account any previous data, which we will refer to as Simple Estimations (SE), and including previous data in the models to be able to obtain more accurate estimations, as in the case of Kalman Filters (KF).

**Simple Estimations (SE)**

The Simple Estimation models are defined to compare against the Kalman Filters and evaluate how much accuracy is lost when using less CPU processing power. Because a collision avoidance system would need to run through these models many times, and also evaluate data from vehicles nearby, CPU processing power should be considered. These Simple Estimation models are mathematically simple and require very little CPU processing power to run through them.

Three models were defined to account for the possible scenarios a vehicle could be in: constant velocity movement, constant acceleration movement, and constant jerk (acceleration change) movement. The variable “k”
represents the time, which due to the limitation of the GPS, \( \Delta k \) is 1 second.

**Constant Velocity Model (CV)**

\[
x_{k+1} = x_k + (x_k - x_{k-1}) \\
y_{k+1} = y_k + (y_k - y_{k-1})
\]

(1)

**Constant Acceleration Model (CA)**

\[
x_{k+1} = (Vx_k - Vx_{k-1}) \cdot \Delta k + x_k \\
y_{k+1} = (Vy_k - Vy_{k-1}) \cdot \Delta k + y_k
\]

(2)

**Constant Jerk Model (CJ)**

\[
x_{k+1} = (ax_k - ax_{k-1}) \cdot \Delta k^2 + Vx_k \cdot \Delta k + x_k \\
y_{k+1} = (ay_k - ay_{k-1}) \cdot \Delta k^2 + Vy_k \cdot \Delta k + y_k
\]

(3)

**Extended Kalman Filters (EKF)**

The Kalman Filter (KF) is a two-step probabilistic estimation process that is very popular in the robotics world as a tool to predict the next position of the robot in a linear system. Kalman filters are based on linear algebra and the hidden Markov model. The underlying dynamical system is modeled as a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the visible outputs from the hidden state.

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.

The Extended Kalman Filter (EKF) is similar to the KF but it can be used in non-linear systems because it linearizes the transformations via the Taylor Expansions. In the EKF the state transition and observation models need not be linear functions of the state but may instead be (differentiable) functions.

The EFK is a two step process: correct and predict:

a) **Correction (using measurement data)**
   - Compute a gain factor (Kalman Gain) that minimizes the error covariance.
   - Correct state estimation by adding the product of the Kalman gain and the prediction error to the prediction.

b) **Prediction (from the state variables)**
   - Predict the next state from the current state using the system model. Assume the model is perfect (no process noise)

- Predict the error covariance of the next state prediction.
- Correct the error covariance estimation using the Kalman gain.

**Correct Step**

(a) Calculate the Kalman Gain

\[
S = HP^{-1} + V^T R^{-1}
\]

\[
K_k = P_k^{-1} H^T S^{-1}
\]

(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

\[
P_k = (I - K_k H) P_k^{-1}
\]

**Prediction Step**

(a) Predict the state

\[
x_{k+1} = f(x_k, 0)
\]

(b) Predict the error covariance matrix

\[
P^-_{k+1} = AP_{k+1} A^T + Q^T W Q^T
\]

Fig. 3. Extended Kalman Filter

Notation:
- \( x \): state estimate
- \( z \): measurement data
- \( A \): Jacobian of the system model with respect to state
- \( W \): Jacobian of the system model with respect to process noise
- \( V \): Jacobian of measurement model with respect to measurement noise
- \( H \): Jacobian of the measurement model
- \( Q \): process noise covariance
- \( R \): measurement noise covariance
- \( K \): Kalman Gain
- \( P \): estimated error covariance
- \( \sigma_r \): prediction noise
- \( \sigma_m \): measurement noise

For our system the state vector for this system consists of four parameters, each one with an x and y component. Even though not all four of them are used in the models identified in this research such as: constant velocity, constant acceleration, constant jerk, all four parameters will be present in the models for an easier implementation.

\[
x = \begin{bmatrix} x v a j \end{bmatrix} = \begin{bmatrix} \text{Position of vehicle} \\
\text{Velocity of vehicle} \\
\text{Acceleration of vehicle} \\
\text{Jerk of vehicle} \end{bmatrix}
\]

(4)

The estimated \( P \) is used together with the Jacobian matrix \( H \) and the measurement noise covariance (\( R \)) together with the Jacobian matrix \( V \) to calculate the Kalman Gain.
\[
P = \begin{bmatrix} x, x_v, x_v^2, x, a, v, a_i, v_x, j_x \\ v, v_x, v_x^2, v_v, a_x, v, j_v, x_v \\ a, a_x, a_x^2, a, a_i, a_v, j_a, x_a \\ j_x, j_x^2, j_x^3, j, j_v, j, j a, x \\ j_v, j_v^2, j_v^3, j, j_a, j_v, j, j a \\ j_a, j_a^2, j_a^3, j, j_v, j, j_a, j \\ j, j^2, j^3, j, j_v, j, j_a, j \\
\end{bmatrix}
\]
\[
h(x, v) = \begin{bmatrix} x + v \\ v + v \\ \end{bmatrix}
\]
\[
H = \left[ \frac{\partial h(x,0)}{\partial x} \right]_{x=v(k-1)}
\]
\[
R = \sigma^2_n \begin{bmatrix} I & 0 \\ 0 & I \\ \end{bmatrix}
\]
\[
V = \left[ \frac{\partial h(x,0)}{\partial v} \right]_{x=v(k-1)}
\]

Once the Kalman Gain (K) is calculated the system brings in the measured data (Z) to correct the predicted position and also the covariance error. Since this system can only measure location and speed from the GPS the other two parameters are set to zero for the system to calculate from prior data.

\[
Z_k = \begin{bmatrix} x \\ v \\ a \\ j_i \\ \end{bmatrix}
\]

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 and the Jacobian matrix W together with the Process noise covariance (Q) as follows.

\[
A = \left[ \frac{\partial f(x, w)}{\partial x} \right]_{x=v(k-1)}
\]
\[
W = \left[ \frac{\partial f(x, w)}{\partial w} \right]_{x=v(k-1)}
\]
\[
Q = \sigma^2_p \cdot I
\]

III. MULTIPLE MODELS

There are several algorithms that exist to iteratively update the stochastic information on-line. These are termed adaptive Kalman filtering algorithms due to their ability to automatically adapt the filter in real time to correspond to the temporal variation of the errors involved.

One such algorithm is termed Multiple Models Adaptive Estimation (MMAE). The MMAE algorithm runs several Kalman filters in parallel, each operating using different dynamic or stochastic models. The MMAE algorithm is used to select either a single ‘best’ Kalman filter solution, or the algorithm can be used to combine the output from all the Kalman filters in a single solution. A possible limitation of such an approach would be the large computational burden imposed by running multiple Kalman filters. However, with improved processor technology, such an approach can now be considered even for real-time applications [22].

Another such algorithm is the Interacting Multiple Model Estimation (IMME) which, even though it works in a similar manner as the MMAE by running multiple Kalman filters in parallel, it is more mathematically involved and takes into account the probability of the next KF selection, making it more accurate than the MMAE in many scenarios.

As the dynamic state of vehicles is highly variable over time, the model selected has to meet the conditions of very different situations. However, a solution based on the implementation of a unique model that fulfills the consistency requirements of scenarios with high dynamic changes, provokes unrealistic noise considerations when mild maneuvers are performed, diminishing the filter efficiency and impoverishing the final solution. Therefore, two different interactive multi-model filters have been developed and implemented to identify which one is better for this type of scenario.

The Multiple Model Adaptive Estimation (MMAE) algorithm is used to select either a single ‘best’ Kalman filter solution, or the algorithm can be used to combine the output from all the Kalman filters in a single solution. It uses only the previous evaluation of the individual filters used to identify which one should be used in the calculation of the next estimated location.

The IMME algorithm calculates the probability of occurrence for each of the individual filters and uses that

\[
\text{Constant Acceleration Model (CA)}
\]
\[
x_v(k) = x_v(k-1) + v_v(k) \Delta k + w_v
\]
\[
v_v(k) = v_v(k-1) + a_v(k) \Delta k + w_a
\]
\[
a_v(k) = a_v(k-1) + w_a
\]
\[
n_i(k) = 0
\]
\[
\text{Constant Jerk Model (CJ)}
\]
\[
x_j(k) = x_j(k-1) + v_j(k) \Delta k + w_v
\]
\[
v_j(k) = v_j(k-1) + a_j(k) \Delta k + w_a
\]
\[
a_j(k) = a_j(k-1) + w_a
\]
\[
n_i(k) = (1/2)(a_j(k-1) - a_i(k-2)) + w,
\]
information to identify which of the filters will be more predominant. This algorithm continues calculating the probability for each of the steps throughout the whole run; therefore, the IMME should be more accurate than the MMAE.

1. Multiple Models Adaptive Estimation (MMAE)

The classic MMAE uses a bank of $m$ Kalman filters running simultaneously, each tuned to a different data set. The principle of the MMAE algorithm is described by Figure 4 which shows that the new measurements, $z_k$, are used in a bank of $N$ Kalman filters. Each filter is configured to use either different stochastic matrices, or different mathematical models. The updated state estimates, $x_k$, for the $N$ Kalman filters are computed using the extended Kalman filter algorithm. The states from each filter are then combined by computing weight factors, and summing the weighted outputs. There are many different ways in which the weight factors can be computed. The one chosen for this system was the Dynamic Multiple Model method since not one filter will be the correct one at all times. This algorithm is described next.

![Figure 4. Multiple Model Adaptive Estimation.](image)

The weight factors are computed using the recursive formula in (17), for $N$ Kalman filters, where $p_n(k)$ is the probability that the $n$th model is correct. The probability density function, $f_n(z_k)$, is computed for each filter based on $v^T \cdot S^{-1} \cdot v$, and its corresponding covariance, $S_k$, using the formula in (19).

$$p_n(k) = \frac{f_n(z_k) \cdot p_n(k-1)}{\sum_{j=1}^{N} f_j(z_k) \cdot p_j(k-1)}$$ (17)

$$S_k = H \cdot P \cdot H^T$$ (18)

$$f_n(z_k) = \frac{1}{\sqrt{(2\pi)^t |S_k|}} \cdot e^{-\frac{1}{2} v^T \cdot S^{-1} \cdot v}$$ (19)

The expression for the covariance in (18) reflects the filter’s estimate of the measurement residuals, not the actual residuals. This becomes clear when one examines the update expressions for “P” in the Kalman filter: “P” does not depend on the measurement residual. The effect of this is that the expression may indicate some small residual variance, when in fact at particular points in time the variance is relatively large. This is indeed exactly the case when one is simultaneously considering multiple models for a process—one of the models, or some combination of them, is “right” and actually has small residuals, while others are “wrong” and will suffer from large residuals. Thus when one is computing the likelihood of a residual for the purpose of comparing model performance, one must consider the likelihood of the actual measurement at each time step, given the expected performance of each model (17). This likelihood and probability variables allow the MMEA to determine which one of the filters defined should be used in the estimation of the next location, providing an accurate estimation.

2. Interacting Multiple Models Estimation (IMME)

The basic idea of IMME is to simultaneously use several filters and mix their outputs to obtain a better estimation as it is shown in Figure 5. This method allows coping with the uncertainty on the target motion by running a set of possible displacement modes at the same time. Even if the target is supposed to possibly be in each displacement mode, the probability that it is in each of them is considered and updated during execution of the IMME.

The IMME calculates the probability of success of each model at every execution, supplying a realistic combined solution for the vehicle’s behavior. These probabilities are calculated according to a Markov model, which assumes that at each scan time there is a probability $P_{ij}$ that the vehicle will make a transition from model state $i$ to state $j$. These probabilities are assumed to be known a priori and can be expressed in a probability transition matrix. The values chosen for this experiment are shown in (20).

$$P_{ij} = \begin{bmatrix} 0.98 & 0.01 & 0.01 \\ 0.01 & 0.98 & 0.01 \\ 0.01 & 0.01 & 0.98 \end{bmatrix}$$ (20)

![Figure 5. Interacting Multiple Model Estimation.](image)
In Johnson and Krishnamurthy’s paper on “An Improvement to the Interactive Multiple Model (IMM) Algorithm” [37] they describes the IMME as a recursive suboptimal algorithm that consists of five core steps:

• **Step 1** Calculation of the mixing probabilities
  The probability mixing calculation uses the transition matrix (20) and the previous iteration model probabilities (23) to compute the normalized mixing probabilities (21). The mixing probabilities are re-computed each time the filter iterates before the mixing step.

\[
\lambda_k(i \mid j) = \frac{p_{ij} \lambda_{k-1}(i)}{\sum_i p_{ij} \lambda_{k-1}(i)}
\]

(21)

• **Step 2** Mixing
  The mixing probabilities are used to compute new initial conditions for each of the \(N\) filters. The initial state vectors are formed as the weighted average all the filter state vectors from the previous iteration (22). 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 (23).

\[
x_{k-1}^{0j} = \sum_{i=1}^{s} \lambda_{k-1}(i \mid j) \hat{x}_{k-1}^i
\]

(22)

\[
P_{k-1}^{0j} = \sum_{i=1}^{s} \lambda_{k-1}(i \mid j) \cdot \left[ P_{k-1}^i + \left[ \hat{x}_{k-1}^i - \hat{x}_{k-1}^{0j} \right] \left[ \hat{x}_{k-1}^i - \hat{x}_{k-1}^{0j} \right]^T \right]
\]

(23)

• **Step 3** Mode matched filtering
  Using the calculated \(\hat{x}_{k-1}^{0j}\) and \(P_{k-1}^{0j}\) the bank of \(s\) Kalman filters produce outputs \(\hat{x}_{k}^{j}\), the covariance matrix \(P_{k}^{j}\) and the likelihood function (19) where \(j=1\) to \(s\) according to the equations for the EKF in section B, part 2.

• **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 (24).

\[
\dot{\lambda}_k(j) = \frac{f_n(z_k) \sum_{i=1}^{s} p_{ij} \lambda_{k-1}(i)}{\sum_{i=1}^{s} f_n(z_k) \lambda_{k-1}(i)}
\]

(24)

• **Step 5** Estimate and covariance combination
  This step is used for output purposes only; it is not part of the algorithm recursions.

\[
\hat{x}_k = \sum_{j=1}^{s} \lambda_k^{j} \cdot \hat{x}_k^j
\]

(25)

\[
P_k = \sum_{j=1}^{s} \lambda_k^{j} \cdot \left[ P_k^j + \left[ \hat{x}_k^j - \hat{x}_k \right] \left[ \hat{x}_k^j - \hat{x}_k \right]^T \right]
\]

(26)

3. Geographical Information System (GIS)

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, 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 IMME 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 are 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. Correcting an estimated future location with GIS assumes that the vehicle will always remain on the road. This correction will work against detecting real scenarios where the vehicle is going off the road. It is assumed the driver is handling the vehicle properly and awake for this GIS correction to be practical. Also, the GIS correction only occurs when the vehicle’s estimated future location is outside of a road. In any other scenario the GIS correction does not occur so the system is not altered and allows for detection of vehicle’s switching lanes or going through intersections without altering the estimated location. Therefore, if a GIS correction is made, the system assumes the vehicle will remain on the same lane as it is currently on.

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.

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 a lot better, but it was not available in MapPoint.

![Fig. 6. Displaying parameters used in the method to overcome the limitation of MapPoint.](image)

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 “2” shown in Figure 6 are calculated. The angle “2” 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.

$$\text{count} = \frac{\text{circumference}}{\text{arc}}$$ (27)

$$\alpha = \frac{360\text{deg}}{\text{count}}$$ (28)

The variable “arc” used in (27) is the pre-defined distance between points in the circumference, which is used to calculate the count of “arcs” within the circumference. The smaller this value the smaller the increments between check points in the circumference 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 2 meters. This value was selected because the smallest road, even if only one way lane, can not be less than 2 meters wide. If we used a value bigger we could have the possibility of missing a road between check points. The angle “alpha” calculated in (28) is the actual angle increment needed to match the pre-defined “arc” distance on the circumference.

With the angle “2” shown in Figure 6 and the angle “alpha” calculated in (28) we can start running through the checkpoints of the circumference. The estimated location is found at angle “2” and since this estimated location can not be too far from the actual road, we start checking from this angle “2”. The system will check both clockwise and counterclockwise increments of “alpha” until a point is found on the road. Figure 7 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.

![Fig. 7. Geometry used map estimated future location outside the road to a location inside the road.](image)

In Figure 8 we can see in MapPoint the current location in a green dot, the predicted future location in a yellow dot, and the GIS corrected data in the red dot. The small red dots are the clockwise and counterclockwise increments described earlier. Visually, in Figure 8, the estimated future location is probably incorrect as there is no road in that location. Using GIS data to locate the road, we can adjust the predicted location 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 accurate predicted future location. When correcting predictions using GIS data we are actually making some assumptions such as the driver always staying on the road. This method seems to
work well during curves, but its use might have to be complemented by some other collision avoidance systems to be able to be more reliable in other scenarios.

IV. EXPERIMENTAL RESULTS

The experimental setting for testing the different models described in section II needs a log file of GPS data that contains different scenarios, specially those currently causing problems in existing systems. Figure 9 shows the trajectory recorded. It has many turns and contains various changes in speed and direction. This data is perfect to use, and even though it will probably be very hard for the system to give accurate estimations in some sections, it will provide a real life situation where accidents could happen. There is also a short highway section to test the system at higher speeds with less turns also.

The trajectory shown in the overall map Figure 9 was divided into different scenarios: scenario 1 (back streets), scenario 2 (roads), scenario 3 (highways). Scenario 1 consists mainly of slow speeds but curvy streets, Scenario 2 consists in medium speeds with only some turns, and Scenario 3 is mostly high speeds on a highway. These scenarios were defined so it would be easier to evaluate, with more detail, the functionality of the estimation systems. Because we are trying to improve the trajectory estimation we will take a specific curve and run all our tests on it.

In Figure 9 we find the selected road curve out of the whole recorded trajectory. It is definitely a nice sharp turn that occurs at medium speeds (~30-40mph). We felt that this turn would be a good scenario to test our improvements in curves over previous research on trajectory estimation.

The code used for the reading of the GPS logs and implementation of the different filters was written in Visual Basic. This language was chosen because there was source code available to get data directly from the GPS, instead of only log files, to be able to test the system in real-time. It was also chosen because the navigation software used in Figure 9, MapPoint 2004, can be imbedded in Visual Basic, allowing the software to display the estimated future location on the map also. Being able to look at the estimated points on an actual map makes it easier to visually inspect and present the system.

A. Analysis of the Estimation Filters

Because the Kalman Filters are so mathematically involved, requiring a lot of processing power, it is good to measure if their results are better than the SE and by how much. In any system it is always good to evaluate all of its parts to make sure they are all worth it, especially if the idea is to be commercially available.

1. Simple Estimations (SE)

To evaluate the Simple Estimations each of the models, SECV (Simple Estimation Constant Velocity Model), SECA (Simple Estimation Constant Acceleration Model), and SECJ (Simple Estimation Constant Jerk) described in Section II, had to be coded. Running the SE on the same GPS log files recorded for this research provided some results where the error between the estimated location and the actual location were recorded.

Figures 10 and 11 are two graphical representations of the inaccuracy of the SE models for a three second ahead prediction of a vehicle’s position. Figure 10 is compared to the actual reading of the GPS after 3 seconds and the most accurate of the three models is the SECV as the curve was taken at an approximately constant velocity. The other two models’ estimations have a lot of error because they are assuming the vehicle is moving at constant acceleration and at constant jerk. Figure 11 displays the error of each of the models, but showing the number of feet away the three seconds ahead estimated future location is from the actual readings for all three SE models using 21 data points for the above specific turn.

Figures 10 and 11 are two graphical representations of the inaccuracy of the SE models for a three second ahead prediction of a vehicle’s position. Figure 10 is compared to the actual reading of the GPS after 3 seconds and the most accurate of the three models is the SECV as the curve was taken at an approximately constant velocity. The other two models’ estimations have a lot of error because they are assuming the vehicle is moving at constant acceleration and at constant jerk. Figure 11 displays the error of each of the models, but showing the number of feet away the three seconds ahead estimated future location is from the actual location were recorded.
location three seconds later. Again the SECV is more accurate than the others in this scenario, but is still very inaccurate for a reliable collision avoidance system (see Table I for actual values).

2. **Extended Kalman Filters (EKF)**

To be able to evaluate the three different Kalman Filter Models, KFCV (Kalman Filter Constant Velocity), KFCA (Kalman Filter Constant Accelerator), and KFCJ (Kalman Filter Constant Jerk), had to be coded, tested and tuned individually to get as accurate estimations as possible. It is given that one model will not be very accurate all the time on a real time GPS log, so one GPS log was manually created for each of the three models to exercise only one model at the time. By doing this it is possible to tune each of the filters individually knowing that the estimation should be as close as possible at all times.

Once the filters have been tuned they were individually run through the different scenarios, as it was done with the Simple Estimations, and only the results for the data points in the selected curve were recorded in Table I.

Running the three filters together showed how, when one was very close to the real value, the other two 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 three 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, to have a big change occur during that one second, allowing the filters to estimate the next location somewhat accurately.

Fig. 12. Comparison of estimated 3 sec ahead location and actual GPS readings for all three KF models using 21 data points for the above specific turn.

**TABLE I**

<table>
<thead>
<tr>
<th></th>
<th>SE</th>
<th>CV</th>
<th>CA</th>
<th>CJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>34.8652</td>
<td>52.8042</td>
<td>161.0166</td>
<td>161.0166</td>
</tr>
<tr>
<td>EKF</td>
<td>35.2486</td>
<td>35.2590</td>
<td>35.2783</td>
<td>35.2783</td>
</tr>
</tbody>
</table>

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

From Table I above, we can compare the individual models. The CV model is the most accurate for the SE, which is probably because the curve was driven at a constant speed. The problem for the other two SE models is that they do not handle constant speed well as they were design for constant acceleration (CA) or constant jerk (CJ). On the other hand, the EKF models for CA and CJ are mathematically capable of handling a constant velocity behaving similarly to the CV in the observed curve.

Continuing with the comparison between the SE models against the EKF models we can look at Figure 10 for a 3 second ahead position estimation using the SE models, and Figure 12 for the EKF models. Figure 10 shows the SE-CA and SE-CJ models predicting incorrect locations in the curve. The bottom graphs, Figure 11 for SE and Figure 13 for EKF, show the actual error in feet for each of the estimations and it can easily be observed their inaccuracy.

Based on these results, we have chosen the Kalman filters because they perform better than the Simple Estimations overall, which was expected.

3. **Comparison between 1 and 2**

With the data collected from the Simple models and the Kalman filters by running the same set of points, we are able to compare each model at a time and verify if the EKF models are more accurate, which was expected given their good reputation and extensive mathematical equations.

B. **Evaluation of Multiple Models**

Also, we are going to look only at the three seconds away estimation results as this is the most important one for us. Looking at a one second ahead estimation gives us some very accurate results but this would not be enough warning time to prevent an accident, so we will look at three second away estimation and how accurate we can get that.

1. **Multiple Models Adaptive Estimation (MMAE)**

The MMAE was the simplest to implement and it seemed to work nicely converging all three EKF and giving a prediction of a future location closest to the most accurate of the individual EKF. Table II shows the average error in feet of the three second away estimations compared to the actual GPS reading when the vehicle reached that location three seconds later. We can see that this estimated value has too much error to be useful for any type of collision avoidance system. It would just give too many false warnings.
In Figure 14, looking at the MMAE curve, we can see that the part of the estimated positions that had the most error was exactly where the turn is, especially at the beginning of it as the vehicle was coming from a straight line and all of a sudden started turning sharply. It takes a few seconds for the system to correct all that error and become more accurate, which makes it not very reliable.

![Comparison of 3 second ahead estimated location between MMAE, IMME, IMME+GIS and actual GPS location 3 seconds later using 21 data points for the selected curve.](image)

**TABLE II**

Average Estimation Error

<table>
<thead>
<tr>
<th>Estimated position</th>
<th>1 sec ahead</th>
<th>2 sec ahead</th>
<th>3 sec ahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMAE</td>
<td>4.25</td>
<td>10.79</td>
<td>20.09</td>
</tr>
<tr>
<td>IMM</td>
<td>4.08</td>
<td>10.35</td>
<td>19.22</td>
</tr>
<tr>
<td>IMM with GIS</td>
<td>3.49</td>
<td>6.07</td>
<td>8.60</td>
</tr>
</tbody>
</table>

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

2. **Interacting Multiple Models Estimation (IMME)**

The IMME was a lot more complex to implement. We decided to implement it to see if all the extra mathematical computation was worth it for this type of implementation where a vehicle’s direction does not change very fast and we are only able to track it every one second period due to the GPS’ limitation.

The results obtained from the IMME over the specified turn was a little bit better than the MMAE as it was expected, but not enough to make this system more reliable. We can see the average error obtained in Table II.

In Figure 14 we can visually compare the estimated three second ahead positions with the GPS values. It also shows that the IMME had a similar problem to the MMAE, it had a lot of error at the beginning of the turn and after a few seconds converged more with the actual data. So, similar to the MMAE, this method used as a collision avoidance system would produce many false errors.

3. **Geographical Information System (GIS)**

The implementation of GIS data with the IMME estimation process was also complex to implement, but it showed very promising results.

In Figure 15 we can see frame shots of the simulation program. It shows in light yellow the three positions corresponding to 1, 2 and 3 second away estimations. In red, the images show the corrected predicted location for each of the 1, 2 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 we can use Table II to confirm this visual conclusion. The table shows the average error for the selected turn and we can see a huge difference compared to the other two methods, especially when looking at the three seconds ahead estimation which was very bad in the other methods. This method is a lot more reliable and should give a lot less false warnings because the approximate 8.60 feet error it has is barely a vehicle’s width and about half of its length.

![Error measured between the 3 sec ahead IMME estimation and the actual GPS readings 3 sec later using 21 data points for the selected curve.](image)

![Error measured between the 3 sec ahead IMME estimation with GIS correction and the actual GPS readings 3 sec later using 21 data points for the selected curve.](image)

Figures 16 and 17 show another visual aid to be able to compare it to the previous two methods and see how much more accurate this is.
V. CONCLUSIONS

The Kalman Filters are a good choice for predicting a future vehicle’s positions. They performed well as the experimental results showed, and with the ability of being able to work together through a MMAE or IMME system, they are an excellent choice for a position estimation system compared to other simple systems being used [1].

The MMAE and IMME provided some very accurate estimations if the time gap remained small (1 second). The bigger the time gap the greater the inaccuracy. A gap of 1 second is not useful for a collision avoidance system as warning a driver about a possible collision 1 second before it happens would not allow for enough time to do anything to prevent it. The minimum time gap needed would be a 2 or 3 second time gap. 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 current research has the most problems with [1]. The implemented GIS method was very straight forward and could easily 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.

Devices such as the Crossbow sensor accelerometer together with the AutoEnginuity ScanTool can also be used to rely on more accurate and more frequent measurements of velocity and acceleration instead of extracting that information from location changes from the GPS unit. This research mainly wanted to investigate if the very cost effective implementation of a GPS receiver integrated with the GIS system could be used as part of a more sophisticated collision avoidance system. The experimental results showed that in specific scenarios using the GPS receiver in junction with the GIS data, this system proves to be very helpful. Understanding the limitations of GPS units will help with the integration of this system into a more robust collision avoidance system where, for a very little extra cost, the overall system could be more reliable.

ACKNOWLEDGMENT

The authors would like to thank the University of Transportation Center at the University of Vermont for their support in this study.

REFERENCES


