Estimation of forward accelerometer parameter using lateral acceleration and Kalman filtering

Shayne Michael Schiltz
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
https://lib.dr.iastate.edu/rtd/20890

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Estimation of forward accelerometer parameter using lateral acceleration and Kalman filtering

by

Shayne Michael Schiltz

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Greg Luecke, Major Professor
Jim Bernard
Loren W. Zachary

Iowa State University
Ames, Iowa
2005
This is to certify that the master's thesis of
Shayne Michael Schiltz
has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>TECHNICAL BACKGROUND</td>
<td>3</td>
</tr>
<tr>
<td>SIMULATION DESCRIPTIONS AND APPLICATION</td>
<td>21</td>
</tr>
<tr>
<td>EXPERIMENTAL SETUP AND RESULTS</td>
<td>33</td>
</tr>
<tr>
<td>SIMULATION RESULTS</td>
<td>39</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>50</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>54</td>
</tr>
</tbody>
</table>
INTRODUCTION

An autonomous vehicle is a robot that is able to navigate and operate in its environment without any outside influence. In the case of the Roomba Robotic FloorVac, it automatically cleans the floors in your house. In another application, autonomous cars could be developed that drive themselves down the highway and park themselves in the garage. Small autonomous vehicles could be developed to function in environments unsuitable for people, such as a collapsed building or sensitive areas of a nuclear power plant.

In order to successfully implement most autonomous vehicle applications, however, several issues need to be addressed. First, the autonomous vehicle must be able to recognize objects, both stationary and moving, in its environment. This is typically accomplished through the use of sensors; some applications have made use of radar sensing, while other possibilities include camera vision and laser range finding.

In addition to knowing where the obstacles are, the autonomous vehicle must also be able to have a good sense of its own location. In some cases, its location can be determined relative to the location of the stationary obstacles in the environment. Sensors are also alternatives for this; for example, GPS units can be a useful way to determine a vehicle’s position and orientation when the satellite signals are available. Other sensors such as rate gyros, accelerometers, and shaft encoders can provide measurements to aid with dead reckoning, navigation based only on knowledge of the vehicle dynamics and its sensed motion.

In this work, a model for estimating the vehicle’s forward velocity based only off of a dual-axis accelerometer is developed. In this process, the signal offset for the longitudinal accelerometer can be
determined and possibly tracked as it changes over time. The lateral acceleration signal from the sensor is sent through a Kalman filter to refine the velocity and offset estimations; when this filtered acceleration signal is combined with reliable knowledge of the turning radius (based on the vehicle’s front steering angle), the vehicle’s velocity can be determined with a fair amount of accuracy.

To develop this, a simple model for the vehicle dynamics is first introduced for use through the rest of the discussion. From this model, the relationship between the vehicle’s front steer angle and turn radius is determined. The equations for the discrete Kalman filter are introduced, with some key issues highlighted for later determination. After a brief discussion on accelerometers in general, the discrete acceleration system model is developed.

A visual simulation of the vehicle dynamics was created to help understand and analyze the system; this program was then used to verify that the relationship between steering angle and turn radius. A scalar example of the discrete Kalman filter used for the actual analysis is also presented. A third simulation used to track vehicle position using several methods is then discussed. At this point, some accelerometer data is collected and analyzed to determine signal and noise parameters. With these parameters in place, the position simulation is executed for four differing cases, and the filter’s ability to determine the forward velocity and sensor offset is examined.
TECHNICAL BACKGROUND

In today’s work environment, it is common to see several different engineering fields coming together to work on a variety of problems; the work on autonomous vehicles is no exception. Some engineers will bring a working knowledge of sensors and electric circuits to the project, while other engineers may be well versed in vehicle dynamics and controls. Software programmers will work with the hardware designers to improve functionality, and experts in other fields may bring pertinent information to the table as well. With the wide range of topics in the field of autonomous vehicles, it should not be surprising to see a varied number of introductory materials covered before getting started.

In the next several sections, a range of topics will be covered. A simplified vehicle dynamics model will be developed and discussed, with some particular attention given to the relationship between a vehicle’s steering angle and the resulting turn radius. Next, some general comments are made on accelerometers before an effective model for measuring lateral acceleration is developed. At this point, a general discussion of the discrete Kalman filter is presented and an issue in application is identified, which leads into the application of the Kalman filter with the lateral acceleration model.

Vehicle Dynamics

A vehicle in motion can be a very complicated system to model and understand mathematically. Several issues can affect the way that a car moves; for example, a car’s suspension can introduce a roll steer effect, and the car’s tires can add additional steering effects when they slip on the surface. A lot of these effects can be fairly involved, but as more
and more of them are considered and modeled, the complexity of the system increases as well.

An autonomous vehicle must drive and navigate quickly, so whatever mathematical model it uses to assist with dead reckoning must be both highly accurate as well as quickly executable. Therefore, it is clear that some balance between accuracy and speed must be met. The most accurate model would be full of complex equations, which would add greatly to the computation time; on the other hand, a very simplistic model would lead to faster calculations, but the omission of some portions of the dynamics could result in deviations between the vehicle model and its actual position and behavior.

For this project, a fairly simplistic vehicle dynamics model was chosen. As the first step of the navigational system for the autonomous vehicle being developed, a simplified model will leave enough computational cycles available for other aspects desired during the development (like image processing from on-board cameras). This fairly basic mathematical model will be developed based off of the vehicle diagram shown in Figure 1, below.
As shown in Figure 1, the two front tires are approximated by a single tire that is in line with the car's center of gravity; similarly, the two rear tires are considered to be a single tire also in line with the center of gravity. These assumptions do not greatly affect the accuracy of this experiment. On the hardware used, one servomotor is used to control the front steering, and that single command does match up well with this front tire setup. The rear wheels on the experimental setup provide the driving force but do not have the ability to steer like the front tires.

The parameters shown in Figure 1 may not be entirely clear to the reader, especially if they follow a different convention from what is
typically encountered. These parameters are described in Table 1, below; the values for these parameters for the experimental hardware are given later in this report.

Table 1: Vehicle Parameter Descriptions for Figure 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>distance between front and rear axle</td>
</tr>
<tr>
<td>b</td>
<td>distance between rear axle and center of gravity</td>
</tr>
<tr>
<td>d_{cg}</td>
<td>distance from center of gravity to center of rotation</td>
</tr>
<tr>
<td>d_f</td>
<td>distance from virtual front wheel to center of rotation</td>
</tr>
<tr>
<td>d_r</td>
<td>distance from virtual rear wheel to center of rotation</td>
</tr>
<tr>
<td>\psi</td>
<td>orientation in global frame</td>
</tr>
<tr>
<td>\delta</td>
<td>front steer angle</td>
</tr>
<tr>
<td>u_r</td>
<td>drive velocity on virtual rear wheel</td>
</tr>
<tr>
<td>u_{cg}</td>
<td>velocity at center of gravity</td>
</tr>
<tr>
<td>\beta</td>
<td>direction of velocity at center of gravity</td>
</tr>
</tbody>
</table>

Two coordinate frames are used to aid in the development of this model. The global frame is used to define the vehicle's location in the navigation space, and for the most part, it can be chosen to be convenient to the situation. The global frame also serves other uses; it could also be used to identify the locations of obstacles and other areas of interest. The second coordinate frame exists on the vehicle itself and is often called the vehicle frame. Not surprisingly, it is tied to the car's center of gravity, with the x-axis pointing off of the front of the car, the z-axis pointing down into the surface, and the y-axis chosen to follow the right-hand rule.

One additional point of interest is the point of intersection located off to the side of the vehicle. This is the point that the vehicle appears to be rotating around at the given instant, and it is interesting to note that all of the velocities present on the vehicle are perpendicular to this point of rotation. In the case that there is no
steering present on the vehicle, some caution will be required when
performing calculations; the effective turn radium is infinite in this
case, but the corresponding division by zero could be disastrous to a
computer simulation.

Now that the vehicle diagram has been defined, some equations based
on the geometry of the model can be determined. These first few equations
are based on simple triangle properties and can be observed from Figure 1.

\[ d_f = \frac{L}{\sin(\delta)} \]  \hspace{1cm} (1)

\[ d_r = \frac{L}{\tan(\delta)} \]  \hspace{1cm} (2)

\[ d_{cg} = \frac{d_r}{\cos(\beta)} = \frac{L}{\tan(\delta)\cos(\beta)} \]  \hspace{1cm} (3)

With these geometric relationships in place, some equations relating
to the position and motion of the vehicle can be defined. The vehicle’s
position is given in the global frame in terms of \( x_{cg} \) and \( y_{cg} \), the
coordinates of the vehicle’s center of gravity. The vehicle’s orientation
also needs to be known, and this is represented by \( \Psi \). These three
parameters are enough to define the transformation matrix between the
global and the vehicle frames, and therefore, these three values are
sufficient to denote the vehicle’s position.

The vehicle’s orientation is fairly easy to see based on knowledge
of angular motion. When the front wheels are turned, the vehicle’s
orientation changes with time; as noted earlier, only the rear wheels are
used to provide the driving velocity of the vehicle. Therefore, the
change in vehicle orientation can be found as Equation (4). In a similar
fashion, the velocity at the vehicle’s center of gravity can be determined
and is shown as Equation (5).
\[ \dot{\psi} = \frac{u_r}{d_r} = \frac{u_r}{L} \tan(\delta) \]  

\[ \dot{\psi} = \frac{u_{cg}}{d_{cg}} \]  

Not surprisingly, the vehicle's position will change depending on the velocity of its center of gravity, \( u_{cg} \). When steering is present, the rear wheel velocity is not equivalent to the velocity at the center of gravity. This relationship between these velocities, as well as the direction of the center of gravity's motion, can be found by the geometry present in Figure 1. The results are shown below as Equations (6) and (7).

\[ u_{cg} = \frac{u_r}{\cos(\beta)} \]  

\[ \tan(\beta) = \frac{b}{d_r} = \frac{b}{L} \tan(\delta) \]  

Now that the velocity of the center of gravity is adequately described in magnitude and direction, the motion of the vehicle's center of gravity can be described with respect to the global frame.

\[ \dot{x}_{cg} = u_{cg} \cos(\psi + \beta) \]  

\[ \dot{y}_{cg} = u_{cg} \sin(\psi + \beta) \]  

From the discussion so far, it should be evident that having a nonzero steer angle on the front wheels is fairly crucial in these equations of motion. Without a front steer angle, a majority of the equations given here become invalid; in particular, Equations (1) through (5) become unnecessary.

The remaining equations, however, are still valid. Equation (6) combined with Equation (7) that a steering angle of zero ultimately leads.
to velocity at the center of gravity to be equal in magnitude and
direction to the velocity at the rear drive wheel. Furthermore, the
change in global of the position becomes a function simply of drive
velocity and vehicle orientation.

Relationship between steer angle and turn radius

With a nonzero front steering angle on the autonomous vehicle,
Equations (1) through (9) are all valid and ready to be used in analysis.
The distances from various points of the vehicle to the instantaneous
center of rotation become well-defined when steering is present, so it
would follow that a relationship between the vehicle’s steering angle and
its result turn radius would exist. In this section, that relationship is
derived mathematically for future use.

In this analysis, the turn radius to be considered will be \( d_{cg} \), the
distance from the instantaneous center of rotation to the vehicle’s center
of gravity. The distances from the front and rear tires to this
instantaneous center may be useful in the future, but their relationships
are quickly obtained from Equations (1) and (2), respectively.

Turning back to the geometry of the model, a relationship between
the turn radius and the angle \( \beta \) can be observed; it is shown below as
Equation (10).

\[
d_{cg} = \frac{b}{\sin(\beta)}
\]  

(10)

At this point, the important trick is finding a relationship between
\( \beta \) and the steering angle \( \delta \); fortunately, it has already been used in the
development of the vehicle’s equations of motion. With some algebraic
manipulation, Equation (7) can be solved for \( \beta \), with the results shown in
Equation (11) below.
\[
\beta = \tan^{-1}\left(\frac{b}{L} \tan(\delta)\right)
\]  

(11)

Using Equation (11), \(\sin(\beta)\) can be evaluated; knowledge of trigonometry and right triangles produces the intermediate step is given as Equation (12), with the final result shown as Equation (13).

\[
\sin(\beta) = \sin(\tan^{-1}\left(\frac{b \sin(\delta)}{L \cos(\delta)}\right))
\]  

(12)

\[
\sin(\beta) = \frac{b \sin(\delta)}{\sqrt{b^2 \sin^2(\delta) + L^2 \cos^2(\delta)}}
\]  

(13)

As with any inverse trigonometry function, some care must be used when determining the sign of the result. Since \(\beta\) is both smaller than and is limited by the front steer angle, a positive sign for \(\tan^{-1}(\cdot)\) was chosen to keep the result in the first quadrant.

Substituting Equation (13) into Equation (10) provides the derived relationship between the front steer angle and instantaneous turn radius. Equation (14) is well defined for any nonzero steering angle the autonomous vehicle may consider, but just like the equations of motion, this result is not valid in cases where there the front steer angle is zero.

\[
d_{cg} = \frac{\sqrt{b^2 \sin^2(\delta) + L^2 \cos^2(\delta)}}{\sin(\delta)}
\]  

(14)

In order to avoid the issue with the sine term in the denominator, both sides of this equation are inverted to avoid the possibility of the division by zero. This equivalent relationship is given as Equation (15).

\[
F = \frac{1}{d_{cg}} = \frac{\sin(\delta)}{\sqrt{b^2 \sin^2(\delta) + L^2 \cos^2(\delta)}}
\]  

(15)
This relationship between the vehicle's front steer angle and turn radius will be useful later on in this development. The details of this relationship will be presented later, along with the relevant simulation.

**Discrete Kalman Filter**

The discrete Kalman filter was a breakthrough in control system theory back in the 1960s, and it is used to this very day. It was designed to track a time-varying state vector associated with noise-related equations of motion using a least-squares approach. The state model of the random process is shown below in Equation (16), and the observation model (also known as the measurement model) is shown in Equation (17).

\[
x_{k+1} = \Phi_k x_k + w_k
\]  
(16)

\[
z_k = H_k x_k + v_k
\]  
(17)

The only other symbols that are used later on in this development are \(Q_k\) and \(R_k\). By definition, \(Q_k\) is a component of the covariance matrix for the state noise \(w_k\); similarly, \(R_k\) is a component of the covariance matrix for the measurement noise \(v_k\).

This Kalman filter works well in the digital setting, as it was originally developed as a recursive solution to the discrete data linear filtering problem. It is important to note at this point that the same algorithm works for both scalar- and matrix-valued problems.

The Kalman filter needs a few things as inputs, namely an initial estimate for \(x\), the initial error covariance associated with the initial state \(P_k^{-}\), the error covariance matrices \(Q_k\) and \(R_k\), and the measured data sequence \(\{x_k\}\). The recursive algorithm for the Kalman filter is described below in Equations (18)-(22). In the first equation, the Kalman gain \(K_k\) is
calculated. Next, the difference between the estimated and measured signal is adjusted by the Kalman gain in order to obtain a better estimate of the state. From there, the last three equations get matrices prepared for the next iteration, with the "hat" indicating the output of the Kalman filter.

\[
K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + R_k)^{-1} \tag{18}
\]

\[
\hat{x}_k = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-) \tag{19}
\]

\[
P_k = (I - K_k H_k) P_k^- \tag{20}
\]

\[
\hat{x}_{k+1} = \Phi_k \hat{x}_k \tag{21}
\]

\[
P_{k+1}^- = \Phi_k P_k^- \Phi_k^T + Q_k \tag{22}
\]

As the Kalman filter equations work with the system data, the computed Kalman gains work toward creating a better estimate of the state variables, given that there is a system output and a state variable that are related. This fact is shown in Equation (19), where the measurement signal \( z \) is subtracted from the output from the state equations.

A recursive algorithm is well suited to conversion into a computer simulation, and this algorithm is no exception. With the initial value of the inputs known, a Kalman filter can be implemented with a simple programming loop.

With any computer algorithm, some attention must be given to its computational intensity. Due to the inverse operation, Equation (18) should raise an eyebrow when discussing computational issues. In a scalar Kalman filtering problem, the inverse operation is not too costly in terms of computer cycles; on the other hand, when working with a matrix-valued Kalman filtering problem, the calculation of a matrix inverse becomes more and more expensive as the matrix size increases.
On an autonomous vehicle, the additional processor time required to handle these matrix inversions are unwanted, as they may interfere with other operations; therefore, another alternative must be examined. In this project, the matrices used are no larger than $2 \times 2$; while this does add some overhead when compared to the scalar problem, these small matrix operations should be able to be tolerated.

**Accelerometer Discussion**

In order to give the autonomous vehicle some sense about its position, electronic sensors are often used to measure various aspects of its motion. Rate gyros can provide a measure of a vehicle's rotation rate, accelerometers can provide some measure of the vehicle's linear motion, and GPS units can query satellites to provide some information on the vehicle's position and possible orientation. The focus of this project involves the use of an accelerometer, although some of the discussion and analysis provided here can be applied to other types of sensors (rate gyros, for example).

As with any sensor, an accelerometer provides an output voltage that is related to the acceleration experienced by the sensor hardware. A sample of a sensor's accelerometer input-output relationship with voltage is shown below in Figure 2.
From the previous figure, it should be clear that there are two parameters that directly affect the accelerometer's measurements: the signal offset $\varepsilon$ and the scale factor $m$. Following the linear relationship shown in Figure 2, the relationship between the input and the output of the accelerometer is given as Equation (23).

\[ \text{volt} = m \times \text{accel.} + \varepsilon \]  

(23)

Most of the time, the goal of using an accelerometer is to get some measure of the acceleration. This relationship, shown below as Equation (24), is obtained by solving Equation (23) for acceleration.

\[ \text{accel.} = \frac{1}{m} (\text{volt} - \varepsilon) \]  

(24)

In general, the signal offset and scale factor parameters could vary over time, as various effects such as operating temperature and operating
time are taken into consideration. In most cases, any changes experienced by these parameters are virtually negligible over a small period of time.

**Acceleration Model Development**

When using an accelerometer in any vehicle application, both the longitudinal and lateral accelerations are reasonable signals to measure. Both signals can be integrated in order to help verify the vehicle’s velocity in both directions; in addition, this knowledge of the vehicle’s velocities could be used to verify the vehicle’s rate of rotation.

In this work, the lateral acceleration of the autonomous vehicle will be considered and measured. Whenever an object is rotating around a point at a given forward speed $v_x$ and radius $R$, the object’s lateral acceleration $a_y$ can be calculated by the relationship given in Equation (25). When written in normal-tangential coordinates, the lateral acceleration relationship changes to become Equation (26).

$$ a_y = \frac{v_x^2}{R} \tag{25} $$

$$ a_y = \frac{s^2}{R} \tag{26} $$

The lateral can also be measured using a single-axis accelerometer, which would follow the relationship given in Equation (24).

Both of these acceleration relationships (Equation (24) and Equation (26)) can be combined together to form an initial state-space model for the system. The accelerometer relationship can be used to define the state equation for the model, and the lateral acceleration relationship is used to define the measurement model (also called the output model).

From the previous discussion of an accelerometer, there are three parameters that are expected to change over time; aside from the expected...
changes in velocity (to provide the accelerations), the signal offset and scale factor parameters are subject to changes over time as well. These three parameters would therefore make good state variables for the system, and the rest of this development is handled with these in mind.

First, when written in normal-tangential notation, Equation (24) becomes Equation (27), given below. Note that \( V \) will denote the voltage value of the accelerometer for the remainder of the discussion.

\[
\dot{s} = \frac{1}{m} (V - \varepsilon) \tag{27}
\]

In order to see the effect of the three parameter changes on the lateral acceleration, Equation (27) is expanded to show the change in acceleration through the relationship shown in Equation (28). The final result of this operation can be found as Equation (29), where it is important to note that the zero subscript indicates that the current value of the parameter should be used for the evaluation.

\[
\Delta \ddot{s} = \ddot{s} - \ddot{s}_0 = \frac{\partial}{\partial \varepsilon} \Delta \varepsilon + \frac{\partial}{\partial m} \Delta m + \frac{\partial}{\partial V} \Delta V \tag{28}
\]

\[
\Delta \ddot{s} = \ddot{s} - \ddot{s}_0 = -\frac{1}{m_0} \Delta \varepsilon + \frac{1}{m_0^2} (\varepsilon_0 - V_0) \Delta m + \frac{1}{m_0} \Delta V \tag{29}
\]

From here, this result will be used to develop the state equations for the system model. At this point, Equation (29) contains the three state parameters that were identified earlier (changes in velocity, signal offset, and scale factor) as well as one input signal (voltage measured by the accelerometer). In an analysis of an accelerometer sensor system, it would be reasonable that the sensor voltage would be a signal of interest to the analysis. Therefore, this equation can be manipulated into an initial state-space model for the autonomous vehicle accelerometer system, which is given below.
\[
\begin{bmatrix}
\Delta \ddot{s} \\
\Delta \dot{m} \\
\Delta \dot{e}
\end{bmatrix} =
\begin{bmatrix}
0 & \frac{1}{m_0} (e_0 - V_0) & -\frac{1}{m_0} \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta \dot{s} \\
\Delta m \\
\Delta e
\end{bmatrix} +
\begin{bmatrix}
\frac{1}{m_0} \\
0 \\
0
\end{bmatrix} \Delta V
\] (30)

Taking a closer look at this state-space realization, it becomes apparent that this may not provide enough information to be useful. The output model for this system would probably have a single output, which would be related to the forward velocity or lateral acceleration; an output consisting of the scale factor or signal offset parameters would have no real ties to the physical system of the vehicle. This only system output will provide some insight on when the acceleration changes, but with no real dynamics given for the scale factor and signal offset, it would be impossible to determine which of these two parameters actually affected the acceleration in each instance.

In order to accommodate for this issue, a minor change in the assumptions must be made. Originally, it was assumed that both the scale factor and signal offset change with respect to time. Instead, if one of the parameters was considered to be a fixed constant, changes in the other parameter could then be properly noticed in the output equation. For the rest of the work presented here, the scale factor is assumed to be constant with respect to time; in fact, the rest of the discussion in this report would also be easily applied for the constant signal offset case.

Implementing this discussion changes the state equation model developed up to this point. The modified model, under the assumption that the signal scale factor is constant, is shown below in Equation (31).

\[
\begin{bmatrix}
\Delta \ddot{s} \\
\Delta \dot{e}
\end{bmatrix} =
\begin{bmatrix}
0 & -\frac{1}{m_0} \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta \dot{s} \\
\Delta e
\end{bmatrix} +
\begin{bmatrix}
\frac{1}{m_0} \\
0
\end{bmatrix} \Delta V
\] (31)
Development of the Dual Accelerometer System

Equation (31) is a continuous time representation for the acceleration model, but a discrete time model is needed for use with the discrete Kalman filter. This conversion is typically done by finding the fundamental matrix for the continuous system, with the fundamental matrix defined below in Equation (32).

\[
\Phi(t) = I + Ft + \frac{F^2 t^2}{2!} + \frac{F^3 t^3}{3!} + ... \tag{32}
\]

For the purposes of this project, the first two terms of the expansion are useful in determining the discrete state equations; this calculation and result are shown below in Equation (33) and (34) respectively, with the discrete time step \( T \) introduced here as well. The higher-order terms might help to improve the accuracy of the fundamental matrix, but each term successively adds more computational time when the Kalman filter is applied to the discrete model. As a result, the higher-order terms are neglected in this work.

\[
\Phi(T) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & -\frac{1}{m} \\ 0 & 0 \end{bmatrix} T \tag{33}
\]

\[
\Phi(T) = \begin{bmatrix} 1 & -\frac{T}{m} \\ 0 & 1 \end{bmatrix} \tag{34}
\]

This fundamental matrix can then be used to develop the discrete state model for the acceleration system. Notice that the ones on the diagonal were removed by switching the state variables over to the change in the variable rather than the actual value of the variable itself.

\[
\begin{bmatrix} \Delta\xi_{k+1} \\ \Delta\varepsilon_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{T}{m} \\ 0 & m \end{bmatrix} \begin{bmatrix} \Delta\xi_k \\ \Delta\varepsilon_k \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix} [\Delta V] \tag{35}
\]
With a state equation defined, the discrete output model can now be developed for use with the Kalman filter. Going back to Equation (26), the change in the lateral acceleration can be defined as shown in Equation (36), below. The nonlinear terms can be made linear as shown in Equation (37), and then this result can be used to construct the measurement model found in Equation (38).

\[
\Delta a_{y,k} = a_{y,k} - a_{y,k-1} = \frac{s_k^2}{R} - \frac{s_{k-1}^2}{R} \tag{36}
\]

\[
\Delta a_{y,k} = 2 \frac{s_k^2}{R} - 2 \frac{s_{k-1}^2}{R} = \frac{2}{R} \Delta \delta_k \tag{37}
\]

\[
a_{y,k} = Hx_k = \begin{bmatrix} 2/R & 0 \\ 0 & \Delta \delta_k \end{bmatrix} \tag{38}
\]

These two equations, Equations (35) and (38), make up a discrete linear model from the continuous time plant dynamics. At this point, it is interesting to note that all of the states being considered actually deal with the change in the variable between time steps rather than the actual value of the variable. With the equations in this form, the discrete Kalman filter equations can still be used to get proper results; however, this case is often referred to as extended Kalman filtering. In addition, the value of the parameter at a particular instance must be calculated using the change at the current time step as well as the calculated value from the previous time step.

In order for the Kalman filter to be effective, this discrete time model must be observable. For the system to be observable, the observability matrix shown in Equation (39) must have full rank. It is interesting to note that the observability matrix will be square if the system has only one output signal. This property can be seen in Equation
below, which is the observability matrix for the discrete time system that was just developed.

\[
O = \begin{bmatrix}
H \\
H \Phi
\end{bmatrix}
\]  \hspace{1cm} (39)

\[
O_{\text{system}} = \begin{bmatrix}
\frac{2}{R} & 0 \\
0 & -\frac{2T'}{Rm}
\end{bmatrix} \Rightarrow \text{rank} = 2
\]  \hspace{1cm} (40)

Therefore, the discrete time system for the acceleration model is observable, and the Kalman filter can be applied with some confidence.
SIMULATION DESCRIPTIONS AND APPLICATION

To explore some of the issues discussed in the previous section, computer simulations were used to explore and validate some of the derived relationships. In general, computer simulation is a wonderful tool for identifying and exploring various issues before purchasing costly hardware or devoting too much time to an initially flawed project.

Three different programs make up the computer simulations used for the duration of this project. The first simulation was developed using Microsoft Visual C++ (version 6), for its ability to handle visual displays of information. Using the OpenGL Utility Toolkit (GLUT) adapted for use with Windows computers by Nate Robins, this simulation is able to calculate the movement of an autonomous vehicle and visually show its motion and path on the computer screen.

The second simulation was developed using the MATLAB software by Mathworks, Inc. With its ability to handle vector and matrix operations as part of its basic implementation, MATLAB was an ideal candidate to simulate the developed Kalman filter equations in a simple example.

The third simulation used was also created in MATLAB. This application places the vehicle dynamic equations in the same program as the Kalman filter equations, and this program is used to generate the "experimental" results obtained in this work. Again, the built-in vector and matrix operations are a great benefit here, as implementing similar functions in the C++ programming language adds some noticeable overhead to the computations. An analysis of the vehicle's motion is then displayed visually, so the differences between the actual position of the vehicle, the calculated position from filtered data, and the calculated position
from accelerometer information alone is included for the sake of comparison.

The vehicle dynamics, Kalman filter, and position simulations are discussed in more detail in the following sections. Some information on their possible uses is discussed, and some results from the simulations are shown. In addition, the vehicle dynamics simulation is used to verify and better define the relationship between the autonomous vehicle’s front steer angle and effective turn radius.

**Vehicle Dynamics Simulation**

The vehicle dynamics simulation developed using Microsoft Visual C++ and GLUT is based off Equations (1) through (9) and the entire simplified vehicle model discussed earlier in this document. The simulation takes the driving velocity on the rear wheels and the steering angle on the front wheels as its inputs, and it numerically steps through the model equations in order to show how the vehicle’s position and orientation changes with respect to time.

In order to give the user a better understanding of the vehicle, several aspects of the dynamic system are reflected in the display. The simulation illustrates the steering angles on the front tires as well as the slight offset introduced by the velocity’s direction at the center of gravity. In addition, as the vehicle moves, its position over time is recorded. Using this information, the path the vehicle followed is reflected in the simulation’s display. Figure 3 shows an example of the simulation output; in it, the steering angle and the path taken can be seen clearly.
The vehicle dynamics simulation also contains several useful input models for both current and future analysis. The starting front steer angle and rear drive velocity are defined as constants at the beginning of the program, so they can be altered quickly and easily for examining a specific situation. In addition, several methods exist for altering these two input parameters while the simulation is running. With the touch of a key, the steering can be incremented or decremented by a fixed amount of two degrees; as an additional feature, the front steering angle can be randomly changed by as much as five degrees in either direction by pressing a different key. In a similar way, the rear wheel velocity can be altered by a fixed amount in either direction. In order to protect the stability of the simulation, maximum and minimum limits have been placed on both of these input parameters.
As mentioned above, the vehicle simulation records the position of the vehicle over time. Rather than keeping track of every individual position calculated, which would result in extremely large memory requirements as well as additional computational effort to display the path, the vehicle simulation saves every tenth data point. In this case, the missing data points are not visibly noticeable in the simulation display; as a result, the improved performance of the simulation and the lower space requirement justify their omission. The data points that are recorded are also saved in a space-delimited text file, so the steering angle, drive velocity, and vehicle position information for a given simulation run is available for other applications as well.

**Investigation of steer angle and turn radius by simulation**

With the vehicle dynamics simulation available, the relationship between the front steering angle and the effective turn radius developed earlier as Equation (15) can be validated. The function relating these two parameters will also be refined for use in the future.

The vehicle simulation was used to generate data files for constant front steer angles and rear drive velocities. To make these simulations more applicable with the actual hardware to be used, several vehicle parameters were included in the vehicle simulation (these values will be properly presented later when discussing the experimental setup). These data files were read into memory using MATLAB and analyzed, in order to obtain data for both steering angle and turning radius. Once collected, these data points were plotted on top of the theoretical curve given in Equation (15); the resulting plot is shown below as Figure 4.
From this plot, it is clear that the theoretical equation for the relationship developed in the earlier section of this report agrees very closely with the data from the vehicle simulation.

From this simulation result, a little more information is needed for the lateral acceleration model developed earlier. The theoretical relationship between steer angle and turn radius is rather cumbersome, with square root and trigonometric functions used a total of four times. In order to simplify the calculations to be performed, a best-fit equation will be used to approximate the relationship and save on some computational cycles. The linear and quadratic best-fit curves and the corresponding equations are shown below in Figure 5.
The linear fit shows some small deviations between itself and the simulation data points; on the other hand, the quadratic fit shows a striking resemblance to the simulated data. As a result, the quadratic fit equation will be used in the lateral acceleration system model. In summary, this quadratic relationship between $F$ and the steering angle $\delta$ (given in degrees) is shown below as Equation (41).

$$F = (4.6 \times 10^{-6})\delta^2 + (0.00052)\delta + (0.00052)$$  \hspace{1cm} (41)$$

Kalman filter simulation

As mentioned previously, the discrete Kalman filter works well as a computer algorithm; as a result, a Kalman filter was implemented in MATLAB for later use with experimental data. In order to demonstrate working knowledge of the Kalman filter before applying it to accelerometer data, a simple example for the filter is created and presented here.
In this initial example, a scalar state and measurement model will be used for simplicity, although it should be noted that in MATLAB, as long as the matrix dimensions match up properly, the exact same simulation will also work for n-dimensional problems as well. Both the scalar and measurement models are combined with noise, and without any special knowledge of the signals needed, the Kalman filter will sort through the noise on the measurement signal and provide a good estimate of the related state.

For the example, both the state and measurement vectors were created and corrupted with white noise with known covariance values. In simulations using white noise generation in MATLAB, some care needs to be taken when considering the initial data points. Occasionally, those data points will contain an unexpected form of dynamics that can alter the analysis. For this analysis, all of the data points of the simulation will be considered accurate, since the omission of the initial data points could also prevent the preliminary effects of the Kalman filter from being observed.

With the measurement together with its associate noise ready for use, the Kalman filter can be initialized. Since no knowledge of the state signal is assumed, some choices will have to be made for some of the initial pieces of information. The initial guess at the state was arbitrarily set at zero, with the expected error in the initial guess considered to be arbitrarily large. It would also be appropriate to use any knowledge of the error covariance for this initial error as well, although for this example, no such knowledge is assumed.

At this point, Equations (18) to (22) are implemented within a loop structure in the MATLAB simulation. At each discrete time step, the new Kalman gain is calculated and then used to minimize the error between the
measurement and the state. The expected error in the state is recalculated, the results are saved in arrays for later use, and the next iteration is started.

The simulation run was executed with 2000 discrete time steps, with the Kalman filter applied to the noisy measurement signal as discussed above. The results of the simulation, shown below in Figure 6, demonstrate the effects of the Kalman filter.

![Kalman filtering example](image)

**Figure 6: Simulation results from a scalar Kalman filter.**

From this figure, it is a fair assessment that the Kalman filter is doing its job properly; that is, the filter is using the calculated Kalman gains to minimize the error between the state and the measurement. As the time increases, the Kalman filter’s output remains more or less consistent
with the value of the state, although the rapidly-changing nature of white
noise appear to cause some minor discrepancies over time.

The error from this Kalman filter example can also be examined to
look at the effectiveness of the simulation. Figure 7 shows the errors as
calculated by the Kalman filter. Keep in mind that the initial guess on
the amount of error present was arbitrarily large, but within ten time
steps, the Kalman filter has reduced the error down to a small value.
Ideally, the Kalman filter should eventually drive the error to zero;
however, due to the use of white noise for this simulation, some error
will remain present throughout the simulation. By definition, white noise
is unpredictable by nature, so it is reasonable to see a minor amount of
error still present in the simulation.

![Graph](image)

**Figure 7:** Kalman filter simulation error.

These previous two figures seem to indicate that the Kalman filter
used in this simulation is valid. As expected, the error between the
state and measured values is minimized relatively quickly, and the Kalman filter simulation used here does a respectable job of handling unpredictable white noise. As a result, the Kalman filter simulation will be used as the basis for the experimental analysis.

**Position Simulation**

This simulation combines virtually everything discussed earlier in the report to create a close approximation to the actual experimental system. The vehicle dynamics equations are used to determine the actual position of the autonomous vehicle, and the accelerometer information is used to generate appropriate signals for both the longitudinal and lateral acceleration sensors. The discrete system model is used to generate system states, and the discrete Kalman filter is used to reduce the error in these states.

As mentioned previously, the program keeps track of three different vehicles positions: the actual position, the position obtained from the filter, and the position obtained from just the sensors alone; these three positions are indicated by the dark blue star, the pink star, and the light blue dot, respectively. The simulation time of the program is also displayed for reference. A sample of the program during execution is shown below in Figure 8.
This example demonstrates one of the difficulties in position determination using filters. At the initial stages, it is evident that there can be a relatively significant difference between the actual and filtered position in the initial stages. This can be due to several factors, ranging from a cautious initial guess of the states supplied to the filter to unreliable sensor information while the sensor takes a few instances to power up. Of course, the Kalman filter removes the error in the state variables as time progresses, but it cannot much against this initial error.

To account for this, a global positioning unit (GPS) will be used to verify the actual position of the vehicle determined by orbiting
satellites. The addition of the GPS position signal provides a way to correct for this initial error; the Kalman filter has time to work to reduce the error between the filtered and actual system states, and then the GPS position is used on a fixed interval to remove the position drift from that stage of operation. The Kalman filter is reinitialized at this step, using a better guess of the states that it calculated previously, and the vehicle's position as calculated by the filtered information starts tracking the actual position better as this process continues.
EXPERIMENTAL SETUP AND RESULTS

In this section, the hardware used as the basis for the experimental analysis will be identified and discussed. Some parameters from the autonomous vehicle will be provided, since they have been used in various areas of this work.

In addition to the hardware discussion, the data acquisition system intended for use on the autonomous vehicle is introduced. The data acquisition hardware was purchased as a part of a kit that also contained a wide variety of sensors, including a dual-axis analog accelerometer that was perfect for this application.

At this point, the Kalman filter will be applied to a simulated lateral acceleration signal, since the vehicle software was not fully implemented for experimental analysis. Much like a real-world example, no prior knowledge of the simulated measurement signal will be used in the application of the Kalman filter and the resulting analysis.

Hardware identification

The autonomous vehicle hardware to be considered for this work was developed by the Robotics Club at Iowa State University. A picture of the vehicle is shown below as Figure 9.
The vehicle was developed from a purchased radio-controlled truck. The plastic body was removed, but the sturdy plastic frame and rugged wheels from the original setup remain. The motor used to control the steering on the front wheels has been replaced by a stronger motor to allow for reliable repeatability in the steering. The battery for controlling the two motors is strapped on the underside of the frame.

The most notable feature on the autonomous vehicle is the white box mounted in the middle of the frame. That box contains a computer built off of a miniature ATX motherboard, and just like any other computer, the autonomous vehicle has its own processor, hard drive, RAM, USB ports, and other components. The data acquisition hardware (discussed below), the servo motor controller, and a wireless Ethernet card are hooked to this portable computer system through some of its USB ports. A global
positioning (GPS) unit can also be hooked up to the autonomous vehicle system through one of these USB ports.

This hardware setup did provide some information used in both the simulations as well as the experimental cases. Measurements were taken to determine the vehicle’s wheelbase, and the center of gravity of the vehicle was determined by carefully balancing the vehicle hardware on a small width of metal. All measurements were taken in inches and then converted to centimeters. These physical measurements can be found in Table 2 for ease of reference.

Table 2: Measured Vehicle Hardware Parameters.

<table>
<thead>
<tr>
<th>L</th>
<th>27.94 cm</th>
<th>distance between front and rear axle</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>17.145 cm</td>
<td>distance between rear axle and center of gravity</td>
</tr>
<tr>
<td>a</td>
<td>10.795 cm</td>
<td>distance between front axle and center of gravity</td>
</tr>
</tbody>
</table>

At the current time, the vehicle itself is not fully autonomous. It is currently given commands by means of a wireless network socket, with the motor control interface implemented in Java. At one time, cameras were mounted on the hardware, to aid with the remote operation of the vehicle.

**Data acquisition discussion**

The data acquisition system selected for this autonomous vehicle project is the Phidget Interface Kit. As mentioned above, the data acquisition system connects to a computer through the USB port. The kit contained the servo motor controller, a wide variety of sensors (ranging from the accelerometer used in this work to force and touch sensors), and a few other components.
In addition, the Phidget Interface Kit has its benefits with respect to software. With the required libraries and files made available by the manufacturers, the data acquisition card can be addressed using three different programming languages: Basic, Java, and C. Sample programs in all three of these computer languages, including functional utility programs for monitoring the data acquisition system’s status and data logging software, to name just two.

A dependence on the manufacturer for software libraries can occasionally be a limiting factor, and unfortunately, it is currently having an effect here as well. The Java libraries are well defined for the event-driven nature of commanding the motors for driving and steering the vehicle; however, at this time, the analog inputs have not yet been implemented in Java. As a result, experimental data from the accelerometer was not taken using the vehicle hardware; rather, acceleration data was taken using the provided data logging software and analyzed to get a better estimate for the noise and sensor measurements for use with the position simulation.

The data logging software recorded all of the data points and converted them to an integer value between zero and 4095, indicating that ten bits of precision were used in the conversion. Data points were collected, and these recorded values were converted back to volts, since it was known that the accelerometers used had a voltage range from 0 to +3 volts (per the manufacturer’s specifications). The voltages were then plotted versus their corresponding accelerations, and the linear fit through the points was obtained. The resulting plot is shown in Figure 10, below, and the values used for \( m \) and \( \varepsilon \) in the position simulation are shown in Table 3.
With some sensor measurements, the noise on those signals could also be examined. Using the same data values, the maximum and the minimum voltages were determined, and this information was used to determine the magnitude of the white noise used to interfere with the ideal signal. These values are shown below in Table 4.

Table 3: Accelerometer Parameters.

<table>
<thead>
<tr>
<th></th>
<th>m</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.470</td>
<td>1.455</td>
</tr>
</tbody>
</table>

Table 4: Values from Noise Analysis

<table>
<thead>
<tr>
<th>Acceleration</th>
<th>Min</th>
<th>Max</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 g</td>
<td>1.46</td>
<td>1.45</td>
<td>0.01</td>
</tr>
<tr>
<td>1 g</td>
<td>1.96</td>
<td>1.92</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Therefore, the difference between the maximum and minimum measured voltages is taken to be a good indication of the noise on the accelerometer signals. Just to be safe in the position simulation, the white noise magnitude was chosen to be larger (0.0625) to account for the discrete time steps used by the data logger, unexpected noise spikes, and similar factors.

With the white noise magnitude known, the error covariance values used by the Kalman filter \( Q_k \) and \( R_k \) can be estimated. For noise magnitudes less than one, the error covariance of a signal is approximately the square of the standard deviation of the signal. To verify this, the MATLAB function \( \text{randn()} \) was used to generate a large amount of white noise; after multiplying this white noise vector by 0.0625, the standard deviation and covariance of the vector was determined and compared. The numerical results can be found below in Table 5.

<table>
<thead>
<tr>
<th>standard deviation</th>
<th>0.0636</th>
</tr>
</thead>
<tbody>
<tr>
<td>covariance</td>
<td>0.004</td>
</tr>
</tbody>
</table>

Table 5: Noise standard deviation and covariance values.

These values for the accelerometer signal and noise characteristics were added to the position simulation to generate more valid results. The simulation cases and results are now presented in the next section.
SIMULATION RESULTS

To analyze the effectiveness of the extended Kalman filter, four different cases involving both steering and velocity were developed, with each parameter either held constant or varying in a prescribed, sinusoidal pattern. In the variable steering cases, attention was given to always provide a positive steering angle. In each case, the sensor offset was held constant in order to determine the functionality of the Kalman filter in determining the offset value over time. Furthermore, the GPS unit provides position updates to the Kalman filter system once every second.

In the following discussion, each case will be clearly identified, and the simulation results will be presented.

Case 1: Constant steer, constant velocity

In this case, the front steering angle was held constant at 20 degrees, and the driving velocity was set at 0.2 meters per second. By examining the Kalman gains shown in Figure 11, it becomes evident that the filter is working successfully since the Kalman gains are converging toward zero.
With the constant parameters, the Kalman filter performed quite well; it was able to closely determine the value for the signal offset and improve upon the estimate of the vehicle's position. The signal offset value produced by the Kalman filter is shown approaching the actual value in Figure 12, and the distance that the Kalman filter's position differed from the GPS data is shown approaching zero in Figure 13.
Figure 12: Signal Offset Determination for Case 1.

Figure 13: Reset Distances for Case 1.
Case 2: Variable steer, constant velocity

In this case, the drive velocity was held constant while the front steering angle was allowed to vary smoothly between 10 and 30 degrees. The sinusoidal steering angle is presented below as Figure 14 for reference.

![Variable Front Steering](image)

Figure 14: Variable Front Steering used in Simulation.

The position simulation returned some favorable results for this case as well. The Kalman gains are converging to zero once again, meaning that the extended Kalman filter is working sufficiently. The Kalman gains for this case are presented in Figure 15.
In addition, the filtered values for the signal offset still approach the actual values used in the simulation. This can be seen below in Figure 16. Also, the distances reset by the GPS position update are getting smaller over time, as shown in Figure 17.
Figure 16: Signal Offset Determination for Case 2.

Figure 17: Reset Distances for Case 2.
Case 3: Constant steer, variable velocity

In this case, the drive velocity is allowed to slowly change in a sinusoidal pattern. As shown in Figure 18, this variation has a fairly low frequency, as half of an oscillation is not completed within 20 seconds. It is interesting to note that the velocity is relatively low at the beginning of the simulation, but then increases fairly dramatically as time goes on.

![Variable Drive Velocity](image)

*Figure 18: Variable Drive Velocity Used in Simulation.*

At first, the Kalman gains shown in Figure 19 seem to approach zero, but as time goes on and the velocity increases, the Kalman gains seem to indicate that the filter does a poorer job at correcting the states, in particular the changes the velocity.
The changes in the Kalman gains also affect the signal offset tracking, as shown in Figure 20. Not surprisingly, with these difficulties, the Kalman filter has some difficulty honing in on the vehicle position accurately, as shown in Figure 21.
Figure 20: Signal Offset Determination for Case 3.

Figure 21: Reset Distances for Case 3.
Case 4: Variable steer, variable velocity

This final case combines the variable front steer angle shown in Figure 14 with the variable drive velocity shown in Figure 18 to see which variable effect has the most influence on the performance of the Kalman filter. Just as before, a look at the Kalman gains given in Figure 22 indicate that the filter has some difficulty with this case as well.

![Kalman Gains](image)

Figure 22: Kalman Gains for Case 4.

In a similar fashion to Case 3, the Kalman filter has some difficulties determining the proper signal offset value, and therefore, has some difficulties properly tracking vehicle position as well. These two results are shown in Figures 23 and 24, respectively.
Figure 23: Signal Offset Determination for Case 4.

Figure 24: Reset Distances for Case 4.
CONCLUSION

From these simulation results, several conclusions can be drawn that could prove useful in an actual implementation on an autonomous vehicle. Some of these observations are presented, with some discussion on the validity of the observations given as well. Some thoughts on future work along these lines are also made.

First, when the vehicle’s velocity is held relatively constant, this discrete Kalman filter does an excellent job in tracking the sensor offset parameter while, at the same time, correcting velocity and position estimates in a relatively fast manner. The results from Cases 1 and 2 show that the filter successfully determines the signal offset parameter fairly accurately within ten seconds, five seconds in the constant velocity case. This observation seems reasonable, since the steering comes into play only in the measurement model of the discrete acceleration system; once the filter has a good estimate of the states based on the lateral accelerometer, those good estimate of the states ultimately help the filter track the vehicle’s motion that much better.

Second, large changes in velocity make it difficult for this Kalman filter to reduce the error in the states properly, as shown in Cases 3 and 4. After the velocity started changing fairly rapidly, this extended Kalman filter had some difficulty keeping up with the changes. One major difference might attribute to this; the constant velocity cases had a parameter change in the measurement model, but the changing velocity cases caused a parameter change in one of the very states that the filter is trying to correct.

Looking at the two variable velocity cases in another way, the Kalman filter does appear to handle small changes in velocity fairly well.
In fact, the Kalman filter does appear to converge during the first few seconds of those two simulations fairly quickly. During this timeframe, the extended Kalman filter appears to have similar results to the constant velocity cases.

From these observations, any application of this accelerometer setup and discrete Kalman filter would be most effective using only small changes in drive velocity since the front steering angle changes do not seem to have that much effect on the results. Therefore, the magnitude of the velocity would seem not to be a significant factor. In most cases, this Kalman filter setup would initially be used during the initial movement of the autonomous vehicle, so small changes in velocity would initially be limited to a range of low velocities.

Driving at low velocity has some additional benefits as well, which could make this result even more useful. When driving at low velocity, a vehicle’s steering angle is less affected by tire slip; in most cases, this translates to better knowledge of the front steering angle. In addition, at low velocities the problem of obstacle detection and avoidance becomes somewhat less daunting. While the problem remains difficult, the slower speed gives more time to accurately detect the obstacles and estimate their positions with other sensors.

Knowledge of the vehicle’s speed would provide some indication on the reliability of the Kalman filter results. A shaft encoder on the drive axle, for example, could detect a change in velocity significant enough to cause difficulties for the Kalman filter application. In such an instant, the vehicle would switch over to some other dead reckoning system for a while; once the velocity changes drop below that threshold, the Kalman filter could be brought back online and provide useful results quickly when operating in the proper mode.
Another advantage of the work presented here is the ability for the discrete Kalman filter to adapt to almost any accelerometer attached to the system. With some knowledge of the scale factor parameter, which can be obtained from manufacturer’s specifications and/or determined experimentally, the Kalman filter is able to determine and track the signal offset parameter under the conditions discussed above. Kalman filters have been used to track various parameters in several different applications, so it would seem reasonable that any changes in the offset parameter (due to temperature or voltage changes) could be detected and followed reasonably well with a Kalman filter.

The next logical step in verifying this work is to place accelerometers on an actual vehicle and gather some data while the vehicle is turning. The data obtained from the sensors could then be sent into this discrete Kalman filter setup to see if comparable results are obtained.

In order to implement this on actual autonomous vehicle hardware, the matrix inversion in the discrete Kalman filter algorithm would need to be bypassed. The relatively small size of this system had some role in minimizing noticeable effects on the computational time, but even a small amount of processing time in a real-time application can have serious, even disastrous, results. Some examination into the continuous time Kalman filter, which does not have a matrix inversion, could prove to be useful in this instance.

Most of the analysis in this work focused on nonzero steering angles, but several equations in the vehicle dynamics model fall apart if no steering was present. When driving straight, no lateral acceleration is experienced by the vehicle, so the lateral acceleration model used here would not be useful. In an actual implementation on an autonomous vehicle...
vehicle, this issue would need to be addressed in some manner; even if the vehicle is not driving straight, the steer angle must pass through zero steering when transitioning from a positive to a negative steer angle. An alternative does exist to handle this issue with the current hardware. In most cases, the longitudinal accelerometer can be used in the implementation of the sensor system, and when no steering angle is present, the accelerometer on the longitudinal axis could be used to estimate the vehicle’s forward velocity by simply continuing to use the knowledge obtained by the Kalman filter up to that point.


