Multi-Vehicle Rover Testbed Using a New Indoor Positioning Sensor

by

Chris Sae-Hau

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2003

Author .....
Department of Electrical Engineering and Computer Science
August 22, 2003

Certified by .....
Jonathan P. How
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Multi-Vehicle Rover Testbed Using a New Indoor Positioning Sensor

by

Chris Sae-Hau

Submitted to the Department of Electrical Engineering and Computer Science
on August 22, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the design and implementation of an indoor multi-rover testbed using a high-precision indoor positioning sensor system. The system model for the rover is derived and used in a Kalman filter that estimates each rover's velocity given the position measurements. These state estimates are then used in high-level path planning and control algorithms. Single and multi-rover tests are performed using single and multi-waypoint path plans. The results from these tests demonstrate the correctness of the system model, estimator, and control algorithms. This testbed serves as a platform for future multi-vehicle coordinated control experiments.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Acknowledgments

I would like to thank my thesis advisor, Professor Jonathan How, for making this work possible. His insight helped solve many problems along the way. His infinite patience during the project was appreciated greatly.

I would like to thank Arthur Richards and Ian Garcia for their timely intuitions and good humor throughout the course of my research. They made the research more educational and enjoyable. Special thanks to Edmund Pendleton at Arc Second Inc. for his help with the indoor GPS sensor system.

Last, but certainly not least, I would like to thank my parents for their love and support. Their confidence in me has driven me to realize my full potential. I thank all my friends for making me the person I am today and for helping me through stressful times, both at school and at home.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes the design and implementation of a multi-vehicle rover testbed using an indoor Global Positioning System (iGPS). It is often impractical to operate the outdoor testbed because of inclement weather and/or time constraints. An indoor testbed would make it more convenient to test the high-level coordination and control algorithms that are being developed [1, 2]. However, a key hurdle to overcome in the design of this testbed, is the selection and integration of a sensing system that is sufficiently accurate and can give large-scale coverage ($\approx$50 m).

Outdoor GPS requires line-of-sight between the receiver and the satellites which is not be available indoors. Previous experience with indoor GPS showed that the approach is feasible, but is difficult to calibrate and is dedicated to one facility [3]. Without having an appropriate highbay to use for an RF indoor GPS system, ease of implementation, portability, and accuracy are factors that must be considered when choosing a good indoor sensor. The sensor must also be able to support multiple vehicles. This last factor ends up greatly influencing the ease of implementation for different sensor choices. The final testbed uses the iGPS sensor system from Arc Second [5]. This system is precise, portable, and easy to setup. Indoor GPS uses transmitters that are mounted in fixed positions that transmit repeating laser signals to iGPS receivers that can calculate their position based on the signal received from each transmitter. The test area measures approximately 18-by-18 meters.

The biggest challenge in implementing this system is designing an estimator that

can provide precise velocity data at a fixed rate. Since the signals are not always received by the iGPS sensors at a uniform rate, creating an accurate velocity estimate at a regular rate is an issue. To solve this problem, a Kalman filter was used on the incoming position data to create the velocity estimate.

1.1 Indoor Positioning Sensor

1.1.1 Alternative Sensory and Tracking Systems

A variety of indoor positioning sensors was considered for use in the indoor testbed. SICK's LMS 200 laser scanner sweeps a single laser beam in an arc and measures the radius of the dot it creates against an object at each laser angle. This measurement is then processed to determine how far an object is for each laser angle. Multiple laser transmitters can be set up around a room to obtain multiple image maps from multiple angles so that the position of a rover can be calculated. There are several major problems with this design. The first problem is that it is difficult to design a scheme so that different trucks can be differentiated. One idea might be to put flags of different heights on top of the rovers and have the trucks be differentiated based on the flag height when the lasers sweep by. The problem with this is that the lasers only provide one-dimensional information about the terrain making it impossible to distinguish different heights unless the lasers are also moving up and down. Designing an apparatus to move multiple laser transmitters up and down in a coordinated fashion to obtain full three-dimensional information is a difficult task. Not only does the software reading the data need to know the exact elevation of each laser transmitter, it also needs to be able to interpret data from multiple angles to put together a full picture of the testbed. This becomes very computationally intensive and difficult to implement. Another problem is that the computer attached to the lasers have to transmit the position data to the rovers before each rover state can be updated. The transport delay associated with sending position data over ethernet or wireless modem slows down the estimation.

To avoid the problem of creating three-dimensional data from one-dimensional sensors, a multiple video camera setup was also considered. The cameras could be mounted on the ceiling or wall and aimed down on the rovers eliminating the need for the cameras to move. The camera information can then be processed to obtain estimates for velocity and position. With this setup, calibration is difficult because the positions of the camera have to be taken into account as well as the angles at which they view the testbed. Also, the mapping from these video images into position is very nonlinear, which makes the system difficult to calibrate. These factors limit the system's portability and ease of use. Another problem with this setup is that multiple streams of video are very difficult to process in real-time, making the system slow and computationally intensive.

A third alternative was a laser surveying system that can track an object as it moves. This type of system is often used in construction. These devices are sold by Faro and Leica, two leaders in high-end surveying equipment. The main problem with this system is that although it provides extremely precise measurements (0.1mm accuracy) and is not very computationally intensive, these systems can cost $100k–$200k. Also each system can only track one point, so expanding the testbed for use with multiple rovers would be a very costly endeavor.

1.1.2 Indoor Global Positioning System

The Arc Second Constellation 3D-i indoor GPS [5] is used to provide the system with position measurements. The indoor GPS is analogous to outdoor GPS in concept in that it uses transmitters to broadcast laser signals that are tracked by an iGPS receiver. Both systems require line-of-sight between the receivers and their respective transmitters or satellites. Both systems also put the burden of position calculation on the receiver side. Satellites and iGPS transmitters do not need to be aware of how many receivers there are in the system to perform their function; both systems can be scaled to accommodate an unlimited number of receivers. Unlike outdoor GPS, iGPS does not require time synchronization between transmitters. To calculate position, the iGPS uses sets of angles between the transmitters and the receiver; GPS uses a

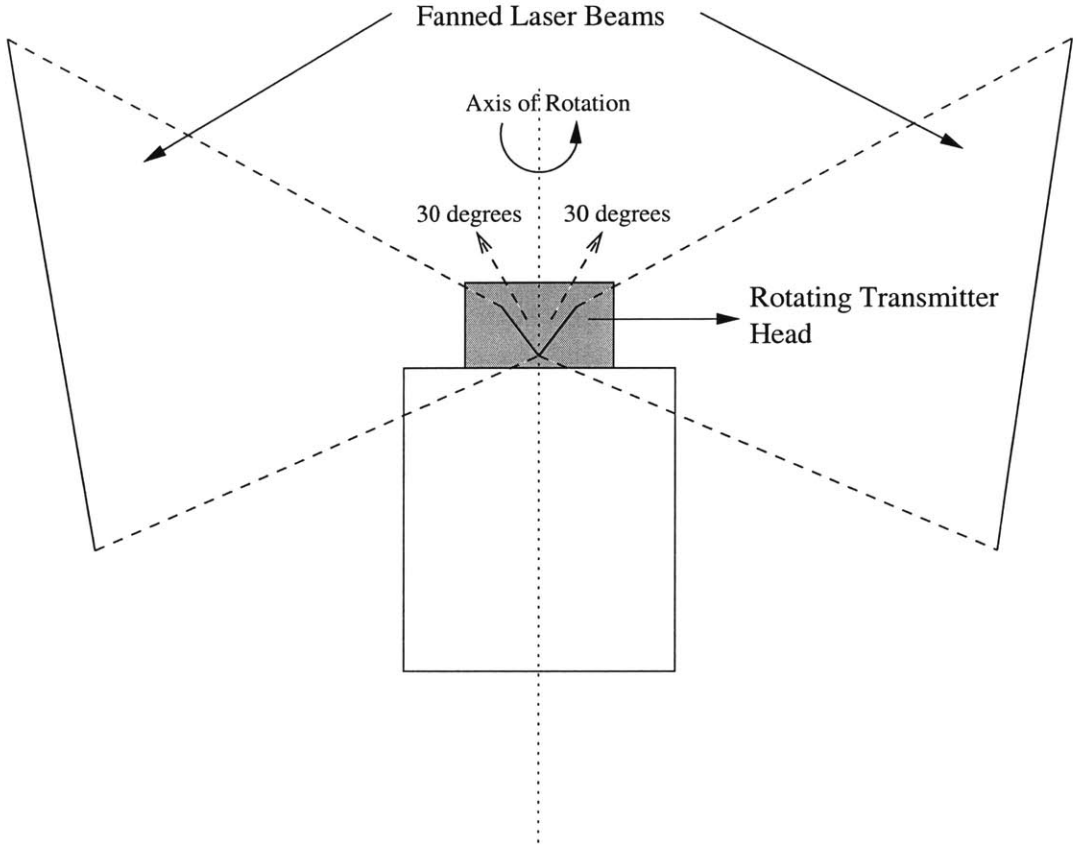

Figure 1-1: An indoor GPS transmitter with its fanned laser beams.

time measurement to determine the pseudorange to each satellite and calculates a position based on several of these pseudoranges.

To determine the position of an iGPS sensor, the azimuth and elevation angles associated with the vector between each transmitter and receiver must be measured. The azimuth angle is defined as the angle in the plane of the rotating transmitter that points towards the receiver. The elevation angle is the angle perpendicular to the transmitter's plane of rotation that points towards the receiver. These two angles, with the position of the transmitter, describe a line that intersects both the transmitter and receiver. Once these two angles are known for all transmitters, the receiver can calculate its position.

Each laser transmitter generates two non-parallel fanned laser beams that rotate on the transmitter's spinning head. The fanned laser beams are angled ± 30 degrees with respect to the axis of the transmitter's rotation (Figure 1-1). The transmitters

rotate at different rates, which are known by the receiver so they can be differentiated. An omni-directional infrared strobe LED flashes at regular intervals. This strobe is used to calculate the azimuth angle by giving the receivers a reference point by which to measure the positions of the sweeping lasers. The elevation angle is determined by measuring the time between laser pulses. The larger the time between laser pulses, the higher the elevation angle.

The calculation for the azimuth angle involves the infrared strobe as well as the two laser beams. When the receiver detects a strobe signal, it measures the time between the stobe and the two laser pulses that happen as the lasers sweep by. Based on this delay and the known rotation rate of each transmitter, an azimuth angle is calculated. To calculate the azimuth angle, a timing measurement is made between the strobe and the laser pulses. Each transmitter's rotation speed is continually tracked by feedback loops in the receivers so the lasers rotate at a constant speed.

Since each receiver obtains two independent angle measurements from each transmitter, a minimum of two transmitters is required to calculate the absolute position of the receiver. The system is calibrated to account for the positions of the transmitters by placing the receiver at six distinct (but not necessarily known) locations and measuring the elevation and azimuth angles associated with those locations for each transmitter. Once that is done, a scale-bar length must be set by selecting two additional points that are a known distance apart. These two measurements define the distance between pairs of receiver measurements so that an absolute coordinate can be associated with different positions.

To calibrate, six parameters are calculated for each laser transmitter. X, Y, Z, R_x, R_y, and R_z. X, Y, and Z, are the coordinates of the transmitter and R_x, R_y, and R_z are the rotation about the z, x, and y axes respectively. The azimuth and elevation angles are measured by each laser transmitter for each of the eight calibration measurements. These measurements create a bundle of rays that intersect at each transmitter. For a solution to converge properly, the rays to each laser from all of the observations must intersect at the same place. This information allows us to determine the relative orientations of the lasers with respect to one another. However,

the system cannot determine the scale of the system with only angle measurements. The scale bar distance simply applies a scale to the solution. A simple analogy is a triangle – the interior angles of a triangle may be the same even if the sides of the triangle are proportionately scaled up or down. In essence, every calibration point produces a series of three-dimensional triangles. The azimuth and elevation angles are measured by the receivers.

To finish the calibration, it is assumed that the first transmitter is at the origin and that the Z-axis is parallel with the spin axis ($R_y = 0$). So the only unknown is R_x, or the rotation about the Z-axis. For the second transmitter, we assume it is on the X-axis and therefore $Y = 0$ and $Z = 0$. With these assumptions, we simply eliminate six of the parameters (X, Y, Z, R_y of the first transmitter and Y and Z of the second transmitter) by forcing them to zero. By constraining the coordinate system, fewer calibration measurements need to be taken. If a different coordinate system is desired, a simple linear transformation can be performed on the position data.

1.1.3 Mathematical Model

The measurement system model is based on the following statements: [5]

- A plane can be completely described by its normal vector;
- The line from the transmitter to the receiver can be described by the vector created by subtracting the transmitter position from the receiver position;
- A line is in a plane when the dot-product between its vector and the plane's normal equals zero.

In this model, the z-axis is the spinning axis of the transmitter and the transmitter spins in the positive direction based on the right hand rule. The following equation mathematically describes when a fan beam from a transmitter intersects with a receiver in terms of fan beam normal vectors and transmitter and receiver locations.

$$N(x,y,z)_{ij}^{1\times 3}\left[R(x,y,z)_{k}^{3\times 1} - T(x,y,z)_{i}^{3\times 1}\right] = 0 \tag{1.1}$$

where:

- i = transmitter index,
- j = fan index(the first or second laser fan of the transmitter),
- k =receiver index,
- $N(x,y,z)_{ij}^{1x3}$ is the normal of the ij fan beam,
- $R(x,y,z)_{k}^{3x1}$ is the position of the kth receiver, and
- $T(x,y,z)_{i}^{3x1}$ is the position of the ith transmitter.

The normal to the fan beams is defined as:

$$N(x,y,z)_{ij}^{1\times3} = Q_{xyz}(rx,ry,rz)_{i}^{3\times3} M_z(\theta)_{ij}^{3\times3} P_{yz}(\phi,\alpha,\theta_{\text{off}})_{ij}^{3\times3} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{1.2}$$

where:

- $Q_{xyz}(rx,ry,rz)_{i}^{3\times3}$ is the i^{th} transmitter rotation matrix,
- rx, ry, rz are the rotation angles about the transmitter's x, y, z-axes respectively.
- $M_z(\theta)_{ij}^{3\times3}$ is the ij measurement,
- θ is the azimuth measurement angle from the timing measurement
- $P_{yz}(\phi,\alpha,\theta_{\text{off}})_{ij}^{3\times3}$ characterizes the plane for the ij fan beam as a function of ϕ, α, and θ_{off}.
- ϕ is the slant angle of the fan beam,
- α describes the conical component of the fan beam,
- θ_{off} describes the placement of the fan beam in the head, and

- $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ gives the normal of the xz reference plane.

1.1.4 Error Budget

Since the iGPS is a high precision system, there are numerous sources of error that can affect measurements. Setup is a significant source of error that presents some unique challenges. When setup points are being sampled, there is an error associated with each measurement due to environmental factors. If local vibrations from the environment disturb the location of the receiver, there will be a small deviation between samples. Also, stray light rays that are at the same frequency as the laser beams might make the measurement noisy. If reflective surfaces are present in the calibration area, receivers may receive the same laser signal multiple times which can cause errors in the calculation. This effect is called multipath. The PCE software handles multipath by throwing out measurements that have too much delay due to reflection. However, this filtering method does not work perfectly and it is best to avoid environments with reflective surfaces present such as mirrors or glass. To help mitigate these effects, many samples should be taken so the averaging algorithm can help remove the noise.

Table 1.1 gives the predicted error values from different sources associated with the indoor testbed setup. This table assumes 5 seconds of averaging (at 20 Hz), or 100 samples, which reduces the measurement error by a factor of 10.[1]

There are four main sources of error on the transmitter side that are not associated with calibration. The beam steering vs. time/temp error occurs because the fan beams in the transmitter head can move slightly with temperature. This exhibits itself as an apparent steering of the optical beam. Arc Second reports that the standard uncertainty was determined experimentally to be 0.25 arcseconds. This

[1]This error reduction is from basic statistics. If one sample has a standard deviation of σ, and 100 samples are taken, then the standard deviation of the mean of those 100 samples is reduced to $\frac{\sigma}{\sqrt{100}}$.

Table 1.1: Error Budget for the iGPS system

Error Source	Linear Error	Azimuth Angle Error (Asec)	Elevation Angle Error (Asec)	Measurement Error
Transmitter Cal.				
Calibration of ϕ and θ		1.00		
Beam Shape - α		2.00		
Other Transmitter Error Sources				
Beam Steering		0.25		
Head Wobble			1.00	
Beam Symmetry		1.00		
Rotation Noise				5.00
Receiver Cal.				
Detector Offset	0.03			
Other Receiver Error Sources				
Detector Symmetry	0.03			
Amplifier		2.50		
Sampling Clock				0.82
Setup				
Mis-Closure of the Bundle	0.03	0.80	0.80	
Total RSS	0.04	2.99	2.37	5.07

error will affect the measurement of the horizontal angle.

The head wobble vs. time/temp error occurs since the head can tilt on the shaft after calibration due to shock and vibration. However the dominant source of head wobble is the precession of the shaft in the bearings. As noted, this effect has a standard uncertainty of 1 arcsecond and influences the measurement of the vertical angle. There is also error associated with beam symmetry. Ideally, the beam should have a Gaussian shape in the cross beam direction. Errors in beam symmetry affect the measurement of horizontal angles. The standard uncertainty for beam symmetry is 1 arcsecond.

Rotation noise also contributes to transmitter error. There is a natural component of noise caused by the fact that the head cannot spin in a completely smooth motion. This noise is due to the windage and imperfections in the shaft to bearing interface. This noise has a uncertainty of 5 arcseconds. The impact of this noise can be reduced through averaging. For example, with 5 seconds of averaging (100 samples), the standard uncertainty would be reduced from 5 arcseconds to 0.5 arcseconds.

There are three main sources of error on the receiver side that are not associated with calibration. Imperfections in the detector cylindrical symmetry will cause some small error since the surface is not perfectly smooth. This slight asymmetry will cause light beams to enter the sensor at slightly different angles. The standard uncertainty is 30 microns.

The amplifier on the PCE can distort the pulses from the detector leading to an angular error. This error is most pronounced when the receiver is close to a transmitter. A standard uncertainty of 2.5 arcseconds is allocated when a receiver is within 10 meters of a transmitter. In all other cases, a standard uncertainty of 0.5 arcseconds is used.

The sampling clock also causes error. The iGPS system is digital and therefore converts the analog optical pulses into digital signals. There is an error associated with the analog to digital conversion. The standard uncertainty for the sampling clock is 0.82 arcseconds. Like transmitter rotation noise, the magnitude of this noise can be reduced through averaging.

The error vales in the table are converted to linear errors using simple geometry. The total error can be calculated by taking the root sum square of the linear error, the angle to linear error, and the linear measurement error. For this testbed, the predicted total system error is 0.12 mm at a sampling rate of 10Hz with four transmitters. However, this testbed was implemented with two transmitters, increasing the error to about 1 mm.

1.1.5 Indoor GPS Performance

To test the performance of the indoor GPS a simple distance measurement test was run. After the system was configured, two fixed points were measured out by hand on the floor and their iGPS coordinates were recorded. The measured distance was then calculated from the iGPS data and compared to the expected distance. This test was performed a total of 9 times, in 3 sets of 3 different lengths. The lengths tested were 20 cm, 50 cm, and 1 m. Each set was done along a line, and each line was done at a different angle relative to the line intersecting both transmitters. Each

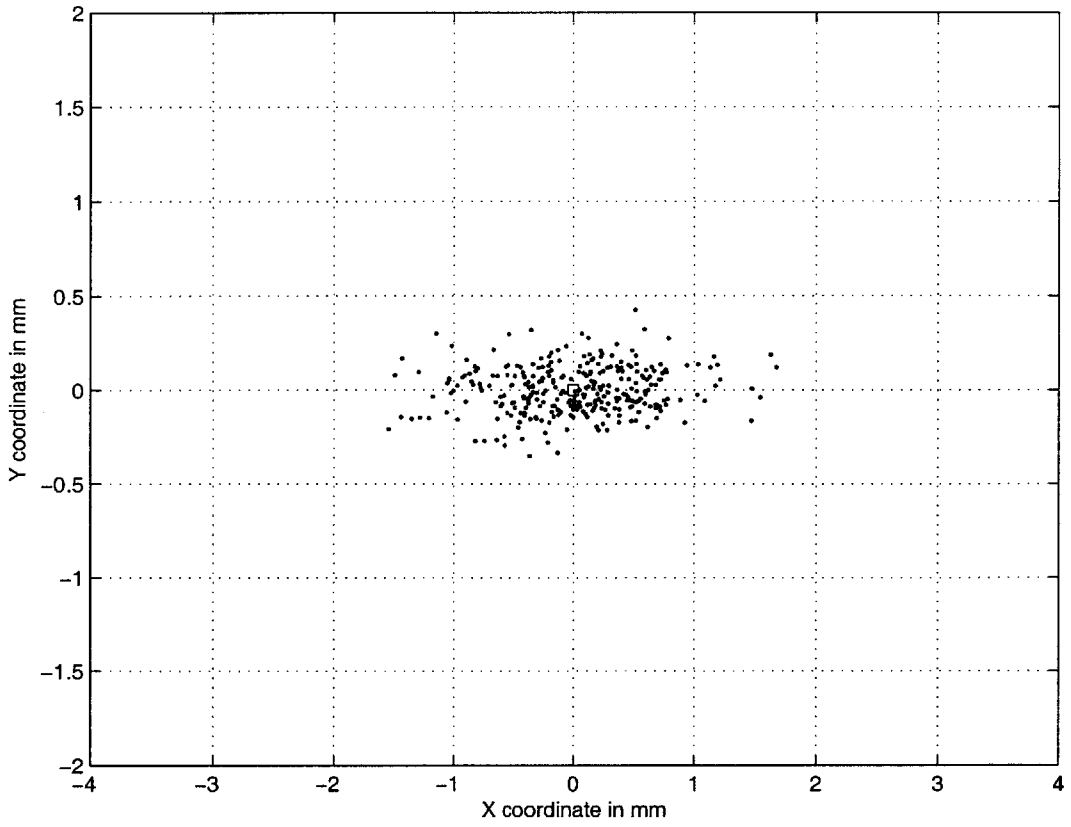

Figure 1-2: Scatter plot of a stationary receiver sampling at 5 Hz for 1 minute.

test yielded similar results. The average error magnitude was 2 mm compared to the expected 1 mm. Figure 1-2 plots the data measured by a receiver sampling at 5 Hz for 1 minute. The square in the plot denotes the average x and y values for the scatter plot. The standard deviation for the x coordinate was 0.5848 mm and the standard deviation for the y coordinate was 0.1254 mm. As the sampling rate was increased, the distribution of the samples widened slightly but never exceeded 3.2 mm. The larger distribution is due to the reduced averaging at higher iGPS sample rates. At 10 Hz the standard deviations increased slightly to 0.565 mm in the x direction and 0.127 mm in the y direction. At 2 Hz the standard deviations decreased slightly to 0.536 mm in the x direction and 0.113 mm in the y direction. The testbed can easily be expanded to use 4 transmitters to reduce the error if desired.

Chapter 2

Estimation

2.1 Estimation Using the Kalman Filter

Velocity data must be calculated from the streaming iGPS position data. Since the system is linear, a Kalman filter can be used to estimate both the position and velocity. Since the x and y coordinates are independent, the following works for either coordinate. The system state vector is defined as:

$$\mathbf{x_n} = \begin{bmatrix} x_{pos} \\ x_{vel} \end{bmatrix}$$

where x_{pos} is the current x position at n, and x_{vel} is the current velocity in the x direction at n. The discrete-time state-space model associated with the evolution of this system is:

$$\mathbf{x}_{n+1} = A\mathbf{x}_n + B_d\mathbf{u}_n \tag{2.1}$$

$$\mathbf{y}_n = C_d\mathbf{x}_n \tag{2.2}$$

$$A_d = \begin{bmatrix} 1 & \delta \\ 0 & 1 \end{bmatrix}, B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix} C_d = \begin{bmatrix} 1 & 0 \end{bmatrix}, \tag{2.3}$$

$\mathbf{y_n}$ is the current position, $\mathbf{u_n}$ is the commanded change in velocity between n and $n+1$, and δ is the sample period. Define the system output error as:

$$\mathbf{e} = \mathbf{y}_n - \bar{\mathbf{y}}_n \tag{2.4}$$

This error measurement, e, can be used to provide feedback to obtain a closed loop prediction estimator:

$$\bar{\mathbf{x}}_{n+1} = A_d\bar{\mathbf{x}}_n + B_d\mathbf{u}_n + L(\mathbf{y}_n - C_d\bar{\mathbf{x}}_n) \tag{2.5}$$

$$\bar{\mathbf{y}}_n = C_d\bar{\mathbf{x}}_n \tag{2.6}$$

where L is the steady-state Kalman gain, which can be chosen depending on what kind of filter characteristics are desired. Calculation of L is described later in this section. Note that in the closed loop estimator the predicted state $\bar{\mathbf{x}}_{n+1}$ is a function of the previous output measurement, $\mathbf{y}_n$. That is, when the estimator propagates to find $\bar{\mathbf{x}}_n$, it does not include information after $n-1$. The key benefit of this is that the control command can be computed during the sample period without having to wait for a measurement update. However, not using the current measurement adds delay to the control loop making it slower to track [7].

This problem can be rectified by altering the estimator slightly to take the current measurement into account when propagating. Assuming the measurement, $\mathbf{y}_n$, is available, the measurement error can be defined in terms of the measurement and the output estimate: $e_n = \mathbf{y}_n - \bar{\mathbf{y}}_n$. Now we update the prediction of the state using this error:

$$\hat{\mathbf{x}}_n = \bar{\mathbf{x}}_n + L_c[\mathbf{y}_n - \bar{\mathbf{y}}_n] \tag{2.7}$$

L_c denotes the Kalman gain associated with the current estimator versus the prediction estimator. Using the current measurement to update the state prediction gives the current measurement estimator: [7]

$$\hat{\mathbf{x}}_n = (I - L_cC_d)\bar{\mathbf{x}}_n + L_cy_n \tag{2.8}$$

The steady state form of the discrete Kalman filter is used in this system. The Kalman gains of the prediction estimators, L and L_c, are calculated by considering the filter loop as $n \to \infty$. L is dependent on the steady state, estimation error covariance matrix P_{ss}^{+}, C_d^T, and the measurement noise covariance matrix, R_k. P_{ss}^{+} is calculated by writing the steady state equations for P_{ss}^{+} and P_{ss}^{-} and solving the following pair of matrix equations: [7]

$$P_{ss}^{-} = A_d P_{ss}^{+} A_d^T + Q_k \tag{2.9}$$

$$P_{ss}^{+} = P_{ss}^{-} - P_{ss}^{-} C_d^T [C_d P_{ss}^{-} C_d^T + R_k]^{-1} C_d P_{ss}^{-} \tag{2.10}$$

where Q_k is the process noise covariance matrix. The Kalman prediction estimator gain is

$$L = P_{ss}^{+} C_d^T R_k^{-1} \tag{2.11}$$

The current Kalman measurement estimator gain, L_c, is simply calculated directly by the relation between the prediction and current Kalman estimator gains:

$$L_c = A_d^{-1} L \tag{2.12}$$

Before L or L_c can be calculated, some assumptions need to be made about the measurement noise covariance (R_k) and the process noise covariance (Q_k). Choices for these variables will affect the bandwidth of the Kalman filter, determined by the ratio of these two noises, $\frac{Q_k}{R_k}$. If a smaller $\frac{Q_k}{R_k}$ is assumed, the Kalman filter will have a lower bandwidth, filtering out high frequency noise. Conversely, a large $\frac{Q_k}{R_k}$ will result in a high bandwidth filter. However, the filter will not be able to track sudden changes in the signal as well as with a large $\frac{Q_k}{R_k}$. A $\frac{Q_k}{R_k}$ of 10 was chosen, with $Q_k = 1$ and $R_k = 0.1$. These values were adjusted experimentally to minimize the velocity estimation noise. Figure 2-1 shows a Bode plot of velocity and position transfer functions (based on an iGPS position input) labelled v_{hat} and p_{hat} respectively. The position estimator is designed to filter out high-frequency position changes due to noise. The velocity estimator is a band limited differentiator that differentiates the

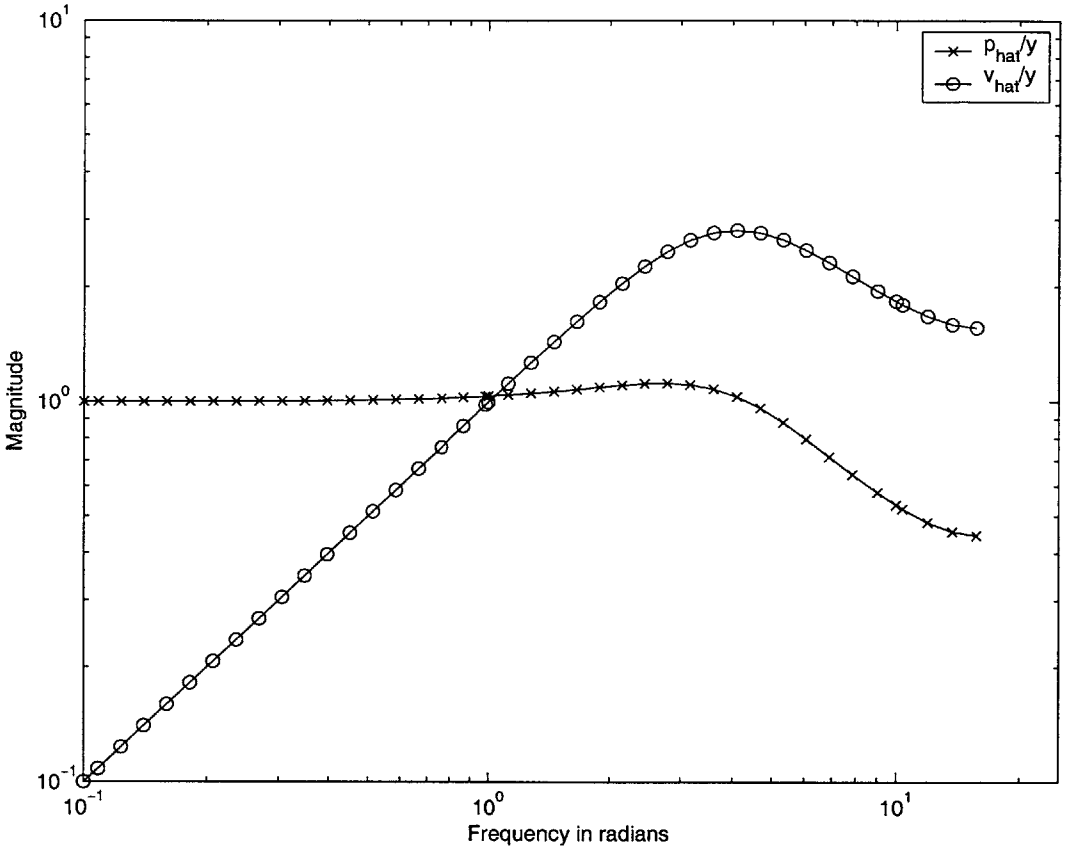

Figure 2-1: Bode plot of the estimator transfer functions for position (p_{hat}) and velocity (v_{hat})).

position data at low frequencies, and filters out high frequency noise. The choice of $\frac{Q_k}{R_k}$ allows the position and velocity of the rover to be tracked accurately given its nominal speed and assumed behavior. That is, we assume the rover will be traveling at low speeds and will not be changing its velocity very erratically.

2.2 Implementation

Since the rover software's update loop runs at an unknown rate that is much faster than 5 Hz, the rate at which the estimator runs must be regulated. The estimator uses the Windows clock to determine if it is time for an estimator update. If 0.2 seconds has elapsed, an update is performed, otherwise the main loop continues to cycle. This scheme guarantees that no updates will ever get called at a rate faster than 5 Hz. The

computers on the rovers all run Microsoft Windows XP Professional, a non-real-time operating system. Due to the nature of the operating system, tasks are often put on hold while the CPU services other system functions. To produce an accurate estimate of a rover's state, it is critical that the streaming iGPS data be handled at a regular rate. If the estimation is being called too slowly and a new measurement is taken each time, the velocity estimate will spike. A late sample will cause this distance to be greater than usual. Since the estimator assumes a constant sample rate, this larger distance will cause a positive velocity spike. If there was no check for over-sampling, negative velocity spikes would also be seen.

A piece of code was implemented to handle the cases where updates are being performed too slowly. This code takes a measurement and propagates the state estimate as long as the time since the last propagation does not exceed a tolerance of 0.025 seconds. This tolerance value was hard-coded into the estimator. If this tolerance is exceeded, then a new measurement is not taken and the state estimates are just propagated. This scheme filters out measurements that are too late so the velocity estimate will not be affected. This tolerance value was determined experimentally to minimize the amount of noise on the velocity estimate.

A plot of the Windows time-stamps was created to see at what rate the estimation was being called and it revealed that, while it was never called faster than 5 Hz, it was being called at a rate slower than 4.8 Hz with a frequency of about 7%. A plot of the typical time periods is shown in Figure 2-2. These results indicate that although Windows XP is not a real-time operating system, the estimator function calls happen within a reasonable tolerance of 5 Hz.

2.2.1 Estimator Performance

Velocity Estimation

To test the performance of the velocity estimator, the rover was driven in a straight line at a constant speed while the estimator output was logged. The estimated velocity was compared to the velocity calculated by differencing the consecutive position

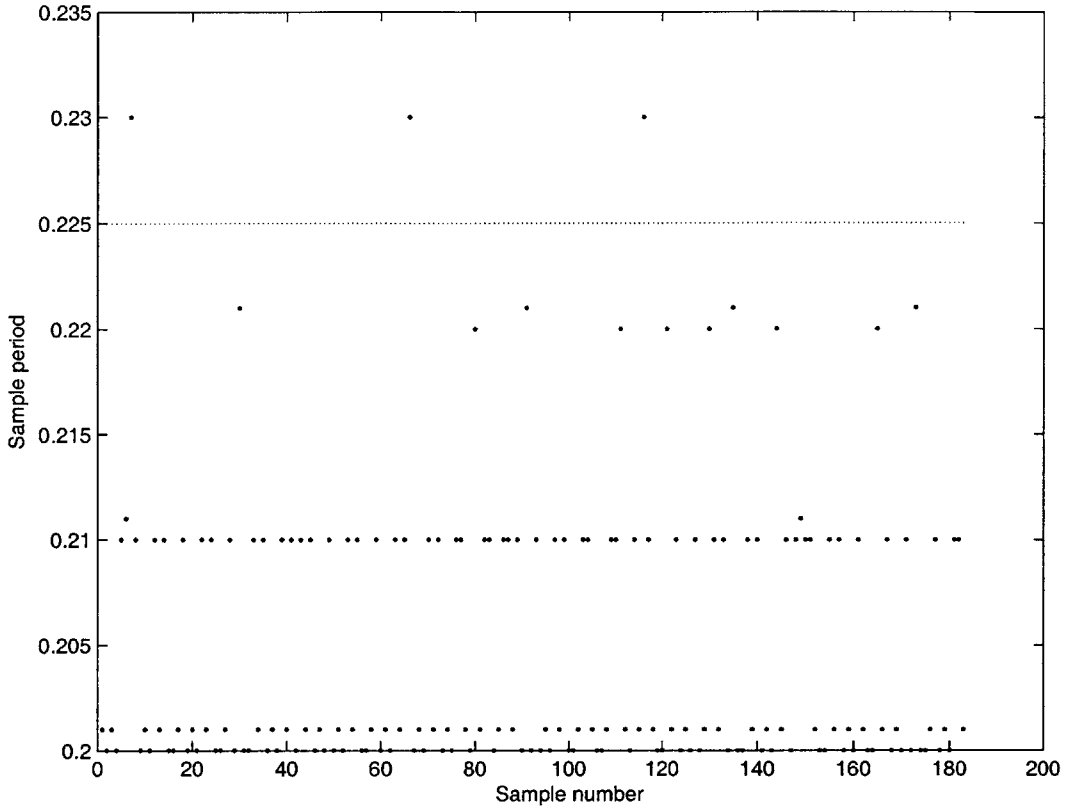

Figure 2-2: Typical time periods between estimator function calls.

measurements, then dividing by the sample period of 0.2 seconds. Figure 2-3 shows the estimated x–y velocities and the calculated x–y velocities. The lines above zero correspond to V_x, and the lines below zero correspond to V_y.

Although the data has a few spikes in it, the estimator does a reasonably good job tracking the velocity of the rover based on the incoming position data. The spikes are due to a synchronization issue between the estimator loop rate and the iGPS sampling rate.

Synchronization

Both the iGPS and the estimator loop were initially set to run at 5 Hz. However, problems arise when the estimator fails to read the iGPS data at the correct rate. The estimator loop software will never allow an iGPS sample to be read at a rate quicker than 5 Hz, however there are times when it will sample the iGPS at a rate slower than 5 Hz. As the estimator runs, its sampling will begin to lag behind the iGPS

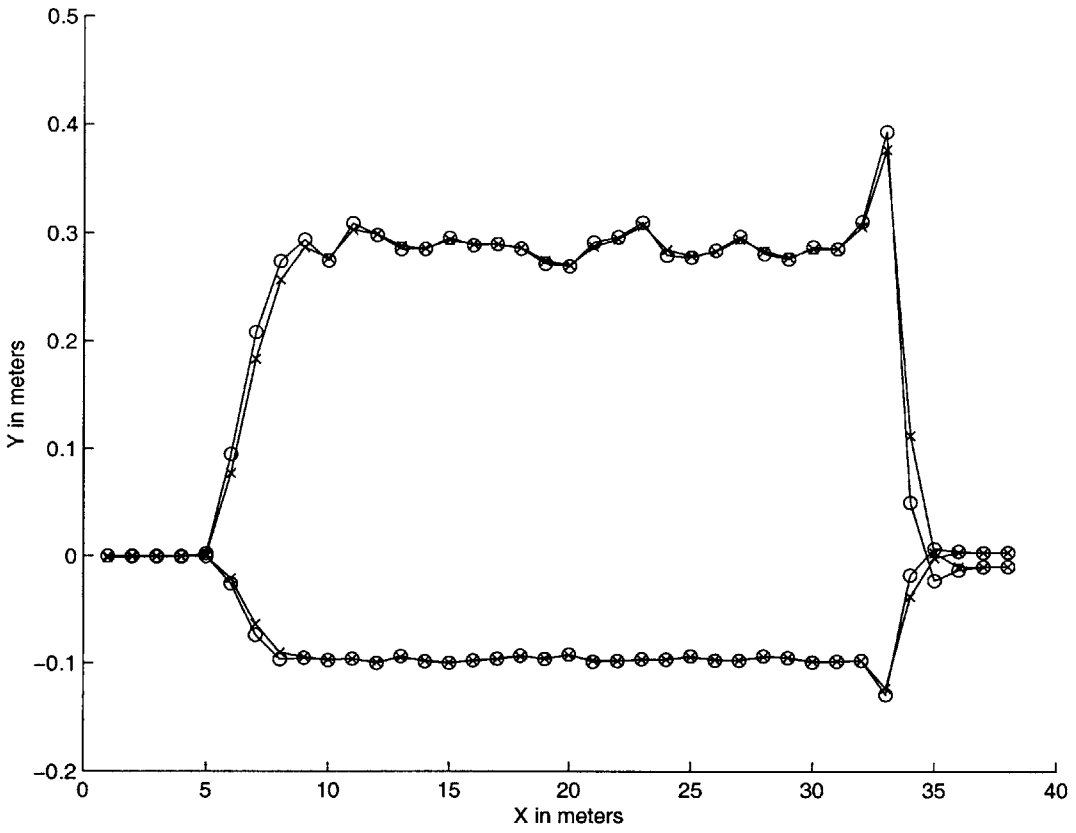

Figure 2-3: Estimated versus calculated velocity for a straight line at constant speed.

output. Eventually, as the lag grows, it is possible that an iGPS measurement can be skipped by the estimator. Figure 2-4 graphically shows this synchronization issue. The iGPS outputs position data at a regular rate while the estimator is sampling with a rate close to, but not equal to, that of the iGPS. As the difference in the two sample periods accumulates over time, a sample will be skipped by the estimator.

Skipped samples are problematic for the velocity estimate. Every time the estimator obtains a new sample from the iGPS, it assumes that the new sample is from 1 period (0.2 seconds at 5 Hz) after the previous one. If a sample is skipped, the sample read will be 2 time periods behind the previous one. Crudely, an average velocity over a time interval can be calculated as the change in position (Δx) divided by the time interval (Δt). If the positions used to calculate Δx are from $2\Delta t$ apart while a time interval of Δt is assumed, the velocity will appear to be doubled (assuming the rover travels at a constant speed). After the skipped sample passes, the velocity

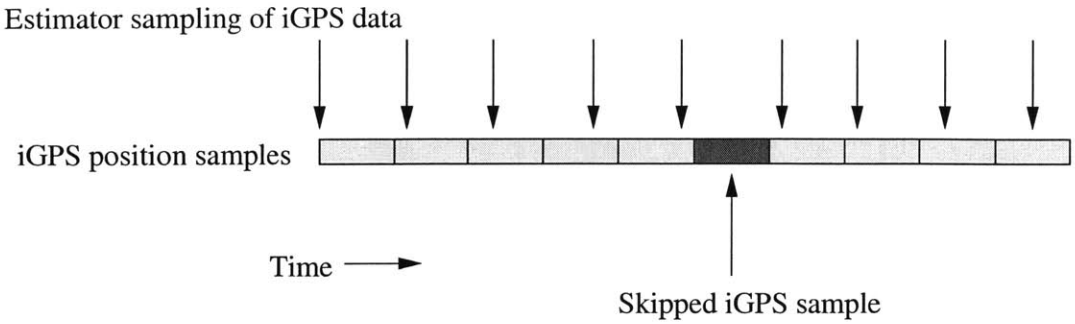

Figure 2-4: The non-constant sample rate of the estimator periodically causes it to miss iGPS samples.

estimate settles down to its true value.

One way to mitigate the problem is to turn up the rate at which the iGPS samples. Consider changing the iGPS sample rate from 5 Hz to 20 Hz with an estimator loop running at 5 Hz. For an iGPS sample rate of 5 Hz, each sample remains valid for 0.2 seconds. Thus, when a sample is taken, the sample is at most 0.2 seconds old. However, if a sample is skipped, a sample will be assumed to be 0.2 seconds old when it is actually 0.4 seconds old, or twice the sample period of the estimator. The following formula is used to calculate the potential spike height factor given an iGPS sample period, Δt_{iGPS}, and an estimator sampling period, Δt_{estim}:

$$100\% + \frac{\Delta t_{iGPS}}{\Delta t_{estim}} \tag{2.13}$$

The time disparity in this case can potentially affect the velocity by a factor of 2, since the estimator assumes a constant sample period of 0.2 seconds.

To compare the two cases, consider a rover that has traveled a fixed distance in one estimator sample period of 0.2 seconds. When a skipped sample occurs, the estimator assumes the sample period is 0.2 seconds, when it is actually 0.2 seconds + 0.2 seconds = 0.4 seconds. This error will cause the velocity to be up to 100% too high. At 20 Hz, each sample remains valid for 0.05 seconds. When a skipped sample occurs, the estimator assumes the sample period is 0.2 seconds, when it is actually 0.2 seconds + 0.05 seconds = 0.25 seconds. This error will cause the velocity to be up to 25% too high, a significant decrease in error from the 5 Hz case.

From the analysis, it is apparent that as the iGPS sample rate increases, skipped samples will have a smaller and smaller effect on the velocity estimate. The tests are conducted using an iGPS sample rate of 20 Hz and an estimator loop rate of 5 Hz.

Estimator Time Delay

The estimator time delay was also measured. If the velocity data is not valid for the current position of the rover, it is useless. For the test, the rover was driven in a circle at a constant speed and the estimator output was logged. The center of the circle was calculated by taking the average of all the points. Then the circle was centered about the origin. A plot of the estimated velocity, superimposed on the position at each coordinate, was made for both x and y. The time difference between the position and velocity reveals how quickly the Kalman filter can adjust to incoming data. Figures 2-5 and 2-6 show the time lags for both dimensions.

When the estimator updates the state, the estimator is always making a prediction about the velocity at the next time step, based on the current measurement; the current position is estimated based on the current position measurement. Thus, the position measurement will always be one cycle behind, or approximately 0.2 seconds. From the zero crossings of the graphs, it appears that there is an estimator phase lag that varies between 0.2 and 0.3 seconds as predicted.

2.2.2 Estimation Conclusions

Modeling a four-wheeled rover was not a simple task since little is known about the friction characteristics between wheels as differential speeds vary. Thus, a two-wheel model was assumed. This model was fine-tuned to match the empirical data collected from the rover. Since there is not sufficient confidence in the model, the estimator process noise covariance was tuned to reflect that uncertainty, relying more on the precision of the measurement sensor.

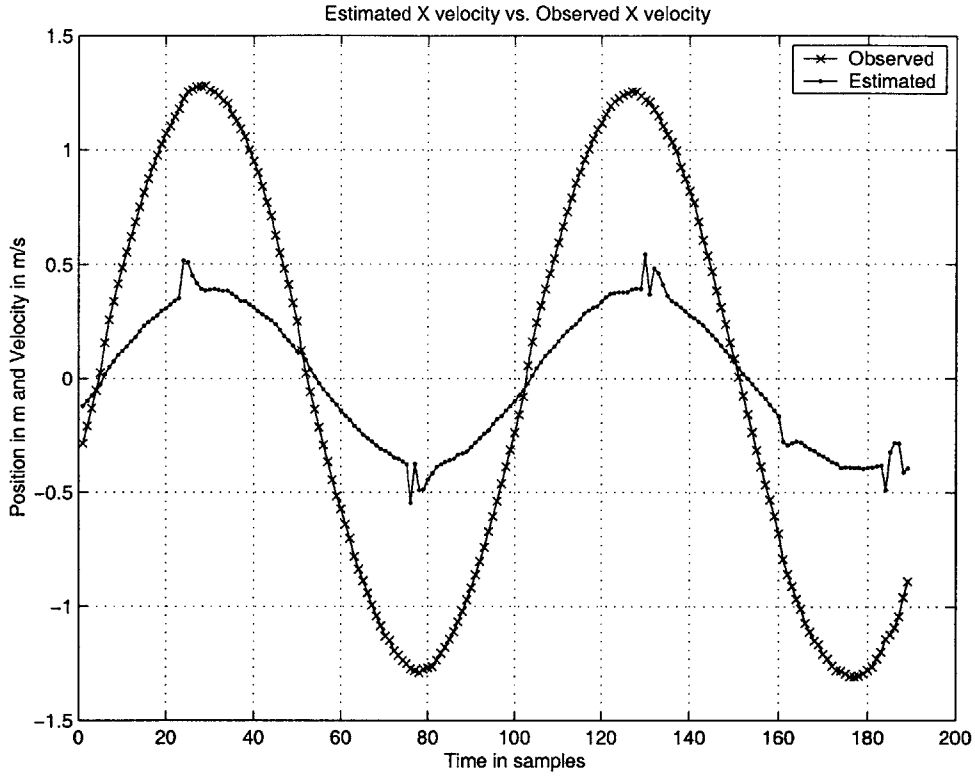

Figure 2-5: Time delay between estimated velocity measurements and estimated position meas urements in the x direction.

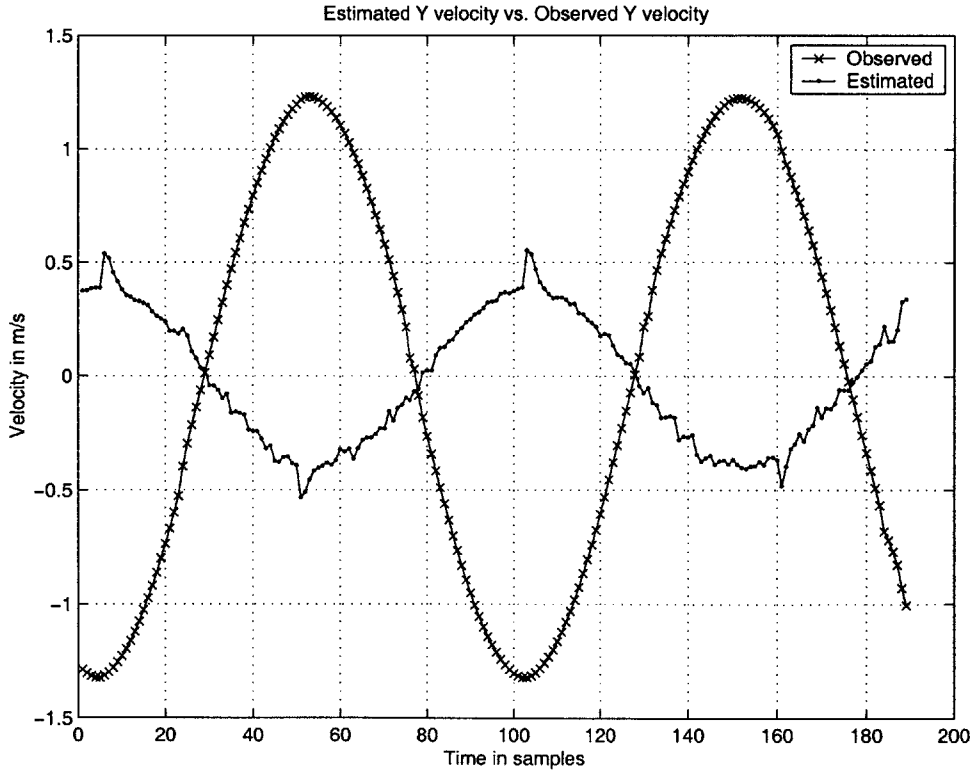

Figure 2-6: Time delay between estimated velocity measurements and estimated position measurements in the y direction.

2.3 Velocity and Heading Control

2.3.1 Velocity Control

There is no closed loop control for velocity. It is assumed that the on-board micro-controller will do a reasonable job of keeping the velocity constant. This was confirmed by setting the rover wheels to different speeds and reading the wheel encoders through the rover software. The wheel encoders showed slight deviations in wheel speed but these fluctuations were never greater than 3 mm/s, which is approximately 1% of the rover's nominal speed. When fluctuations occurred, the truck's micro-controller compensated to bring the error back to zero. The nominal velocity chosen was partially based on Pohlman's outdoor GPS truck system. [4] The nominal speed for his trucks was 0.5 m/s. To perform the outdoor maneuvers inside on the same time scale, the nominal speed needed to be reduced. Pohlman's testbed was approximately 30-by-30 meters while the indoor testbed is about 18-by-18 meters. Therefore, a nominal speed of 0.25 m/s was chosen to keep the time scale of maneuvers approximately equal.

2.3.2 Heading Control

System Modeling

To design a heading controller, the system model must be determined. The rover's software interface does not provide a method for turning with a set radius of curvature so this had to be done in software. The rover software has a method of setting each set of wheels to a different speed. By setting different left-right wheel speed differentials, the rover's turning rate can be controlled.

First, the rover's dynamics must be derived. Since each side of wheels is coupled, we can assume for the derivation that the rover can be modeled as a single wheel axle with a center of mass velocity, V, and left and right wheel speeds of V_1 and V_2, respectively (Figure 2-7). Assume that $V_1 > V_2$. The goal is to derive an expression for the angular velocity ($\dot{\theta}$) of the rover in terms of its separate wheel speeds.

Let R be the turning radius of the rover and let L be its width. $\dot{\theta}$ is the angular

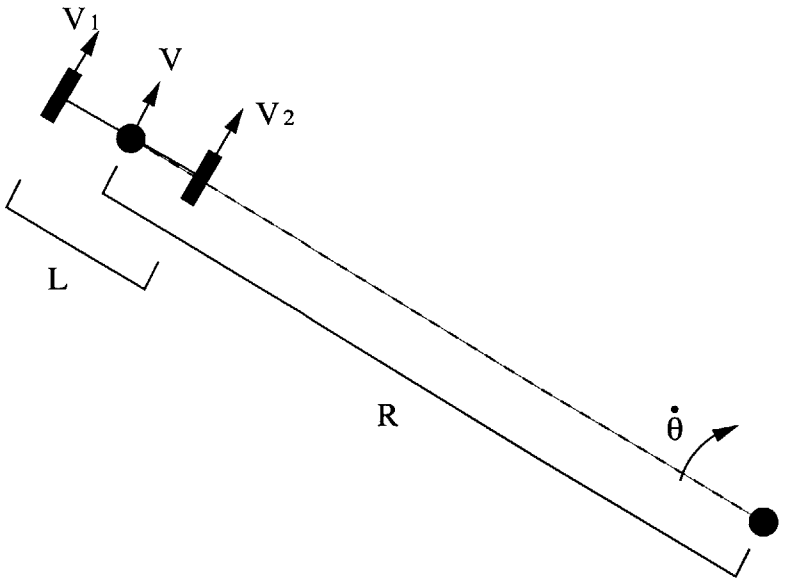

Figure 2-7: Rover model.

velocity of the rover. Let:

$$\begin{cases} V_1 = V + \frac{1}{2}\Delta V \\ V_2 = V - \frac{1}{2}\Delta V \end{cases} \tag{2.14}$$

By simple rotational motion:

$$\dot{\theta} = \frac{V_1}{(R + \frac{1}{2}L)} = \frac{V_1}{(R - \frac{1}{2}L)} \tag{2.15}$$

$$\Rightarrow \frac{V_1}{(R + \frac{1}{2}L)} - \frac{V_1}{(R - \frac{1}{2}L)} = 0 \tag{2.16}$$

$$\Rightarrow V_1(R - \frac{1}{2}) - V_2(R + \frac{1}{2}) = 0 \tag{2.17}$$

$$\Rightarrow R(V_1 - V_2) = \frac{1}{2}L(V_1 + V_2) \tag{2.18}$$

$$\Rightarrow R\Delta V = \frac{1}{2}L(V_1 + V_2) = LV \tag{2.19}$$

$$\Rightarrow \dot{\theta} = \frac{V}{R} = \frac{\Delta V}{L} \tag{2.20}$$

Assuming that the rover's turning angle (θ) is small, we can characterize the change in cross-track error ($\dot{x}$) by the equation, $\dot{x} = V\theta$. Therefore, the rover's dynamics can

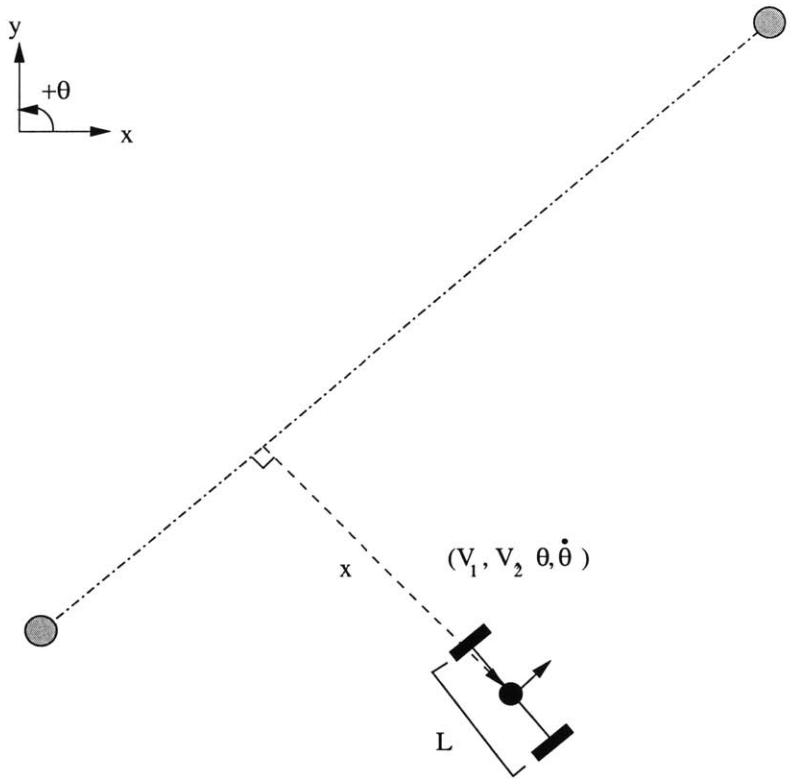

Figure 2-8: Rover heading vectors.

be characterized by the following equations:

$$\begin{cases} \dot{\theta} = \frac{\Delta V}{L} \\ \dot{x} = V\theta \end{cases} \tag{2.21}$$

Figure 2-8 graphically depicts all the quantities necessary for the system model derivation. ΔV is the difference in speeds between the two sides, V is the rover speed, L is the width of the rover, θ is the rover's heading angle, and $\dot{\theta}$ is the speed of rotation of the rover. The cross-track error, x, is the perpendicular distance between the rover's current position and the line joining the last and the next waypoint. $\dot{x}$ is the rate of change of the cross-track error. Depending on the forward velocity desired, ΔV is a speed differential between the wheels on opposite sides to ensure that the rover's center of mass travels with a constant turning radius. Therefore, we can define one wheel velocity as $V_1 = V + \frac{1}{2}\Delta V$ and the other wheel velocity as $V_2 = V - \frac{1}{2}\Delta V$. The velocity of the center of mass is defined as the average between V_1 and V_2, or V.

Combining the two rover dynamic equations in the frequency domain gives

$$sx = \theta V \;\Rightarrow\; \theta = \frac{sx}{V} \;\Rightarrow\; \frac{s^2x}{V} = \frac{\Delta V}{L} \;\Rightarrow\; \frac{x}{\Delta V} = \frac{V}{Ls^2} \tag{2.22}$$

Thus, the plant model for the cross-track controller is $V/(Ls^2)$.

To verify that this system model is correct, the plant's constant $1/L$ must be verified. Although the measured value of L was 0.4m, there can be nonlinearities due to friction that make this value vary for different values of V_1, V_2, and V. To determine this constant experimentally, the rover was set to drive in fixed radius circles with a constant V and a constant ΔV. The radius of the circle and the time it took to make one complete revolution were recorded. These values were then used to calculate the observed speed, V', and the angular velocity, ω, for the fixed values of V and ΔV. This experiment was repeated 3 times for 25 different pairs of V and ΔV. These time and radius measurements were then averaged. Since the rover nominal speed is 0.25 m/s, we chose the ΔV that yields a V' closest to the fixed V.

Figure 2-9 shows the results of the circle tests. Unfortunately, there does not seem to be much of a trend in the data. Possible reasons for this are discussed in the next paragraph. The results indicate that the circle using a $\Delta V = 60$mm/s and a $V = 0.259$m/s yields a $V' = 0.271$m/s, the value closest to the input speed of 0.259 m/s. After this point was chosen, the corresponding value for ω was chosen to calculate the value of L by the dynamics equation, $\dot{\theta} = \Delta V/L$. After this calculation the final system plant becomes:

$$\frac{257}{150}\frac{V_{nom}}{s^2} \tag{2.23}$$

where V_{nom} is the nominal rover speed and ΔV is the wheel input differential. This value for the constant implies a rover width of 0.58 m, higher than the measured 0.4 m.

The disparity between these two values, as well as the suspicious circle data, is partially due to the slipping of the rover's tires against the floor. If there is slipping and the rover is not turning as tightly as it should be, the radius of the circle swept out will be larger, reducing angular velocity. Another reason for the disparity might

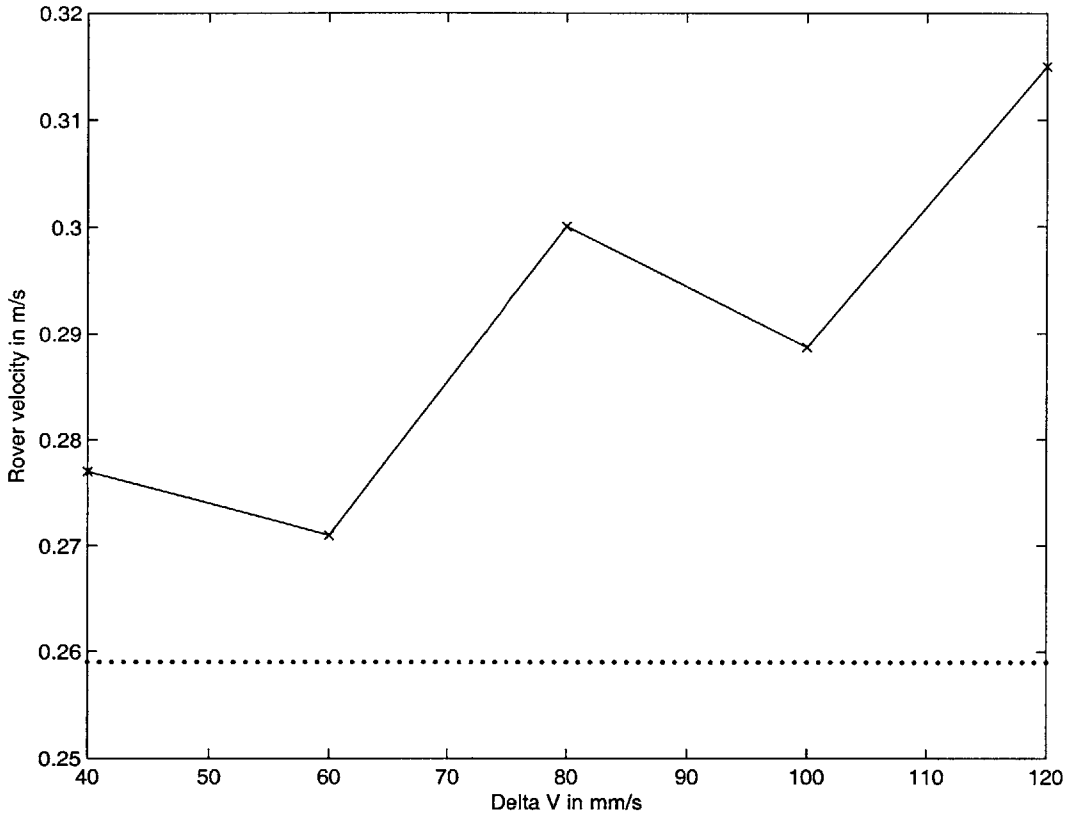

Figure 2-9: Circle test results for $V = 0.259$m/s.

be due to the way the system is modeled. It might not be a fair assumption to model a 4-wheel vehicle as a single axle, even if the sides are coupled. Friction and wheel slippage will play a larger role in the truck's dynamics. If the rover is commanded to make a sharp turn, the back wheels may end up dragging against the floor causing the rover to turn properly, even if the wheel encoders report that the wheels are spinning at the correct velocities. This will result in a slower turning rate, which explains why an L of 0.58 m is more effective than an L of 0.4 m. The friction and slipping associated with the 4 wheeled rover will definitely add nonlinearities in the circle data. Since it is very difficult to predict how the rover's wheels will react given different speed differentials, the current model was kept and calibrated according to the process already described. Changing the constant results in a more accurate angular velocity for an input speed differential since it helps compensate for the model's imperfections. Tests were performed using the two different values of

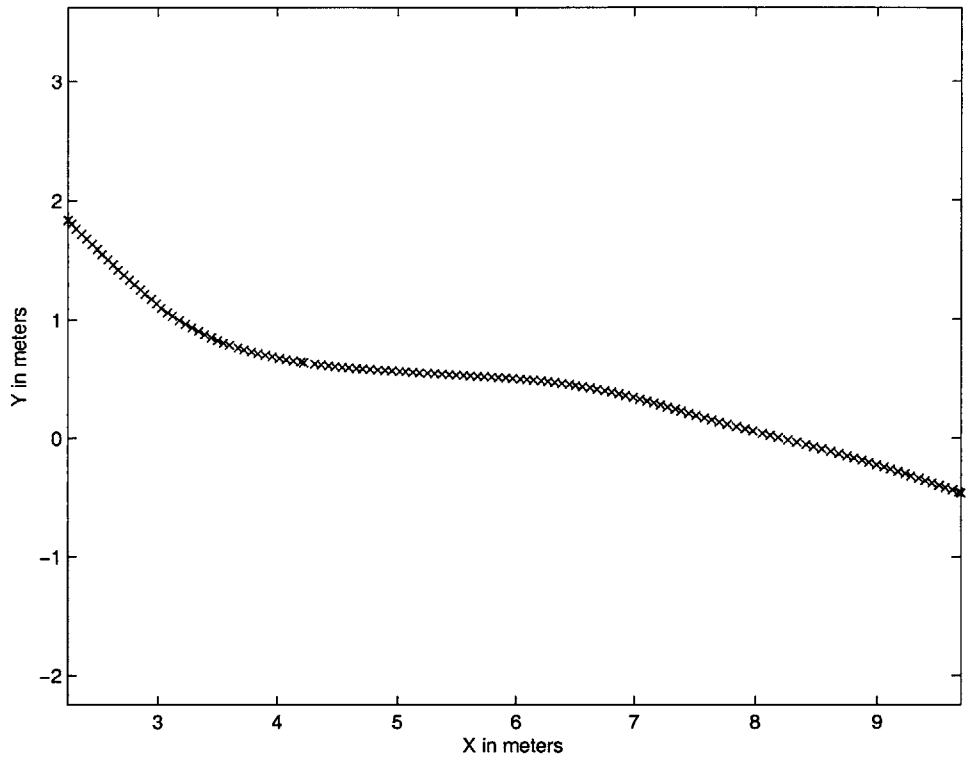

Figure 2-10: x–y coordinate plot of a transient test assuming $L = 0.4$.

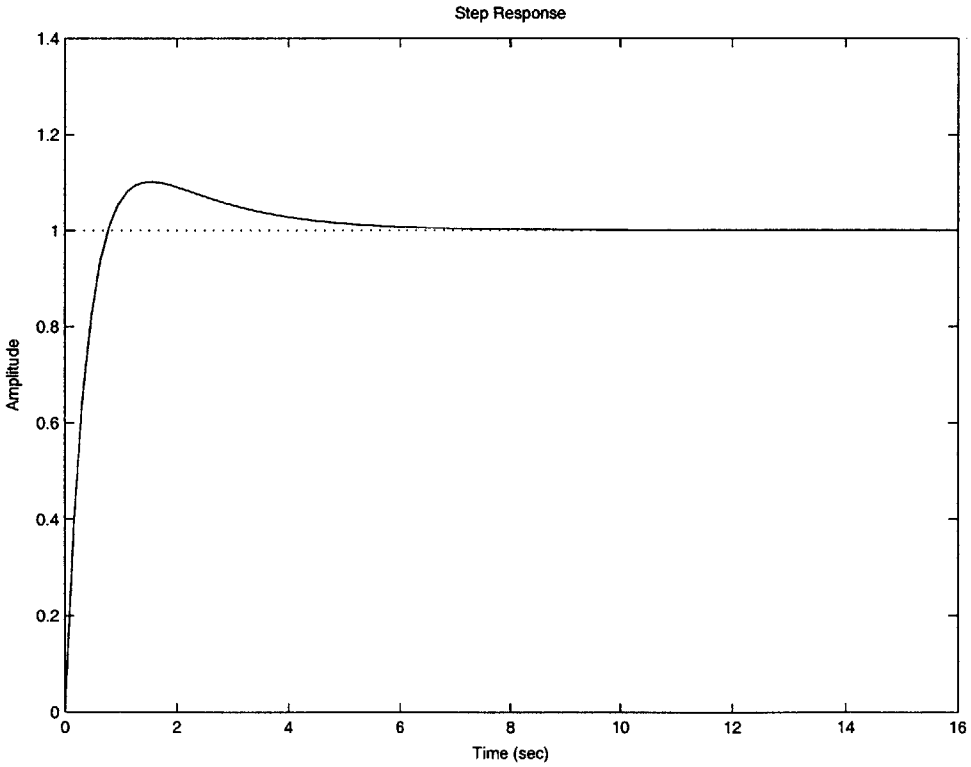

Figure 2-11: Closed loop system response of a cross-track error step, with $L = 0.4$.

L. A rover was driven in a straight line while initially angled 45 degrees away from the straight line path. When $L = 0.58$ the rover exhibited less oscillation and faster settling time than when $L = 0.4$. This result is expected, because with an $L = 0.4$, the cross-track error is overestimated and the rover overcompensates, resulting in a slower settling time. Figure 2-10 is a plot of a run that was done assuming $L = 0.4$. As expected the transient is much slower with a settling time of approximately 6 seconds, compared to the settling time of 4 seconds when using $L = 0.58$. Figure 2-11 shows the step response (assuming the PD controller design described in the next section) with the closed-loop pole frequency at 1 radian/sec, but using the value of $L = 0.4$. The settling time of 6 seconds is consistent with the results. Therefore, an L of 0.58 is a better choice.

Controller Design

The goal is to design a controller that eliminates the cross-track error, x. A proportional plus derivative controller, $H(s)$, is used in a feedback loop to stabilize the system. The block diagram for the control system is shown in Figure 2-12. The parameter $\alpha = K_p/K_v$ gives the location of the open loop zero. The value of α was set to 0.5. The goal for the controller is to minimize settling time without having the rover overshoot.

The root locus diagram of the open loop transfer function, $G(s) = \frac{257}{150}V_{nom}$, with the PD controller, $H(s) = K_v(s + \alpha)$, is plotted in Figure 2-13.

The operating point of the system was chosen where the poles rejoin the real axis because this is where the dominant pole is furthest away from the $j\omega$ axis. This means that the closed loops poles should be at -1 radians/sec. Substituting in this value for the poles in the closed loop equation, the necessary gains can be calculated. This calculation gives $K_v = 4.8$ and $K_p = 2.4$. This calculation was verified by the MATLAB root locus plot. Given that the poles are both operating at -1 radians/sec, the closed-loop system should have a settling time of about 4 seconds. The MATLAB step response plot in Figure 2-14 confirms this prediction. The rover's performance also confirms this prediction. Closer analysis is provided in

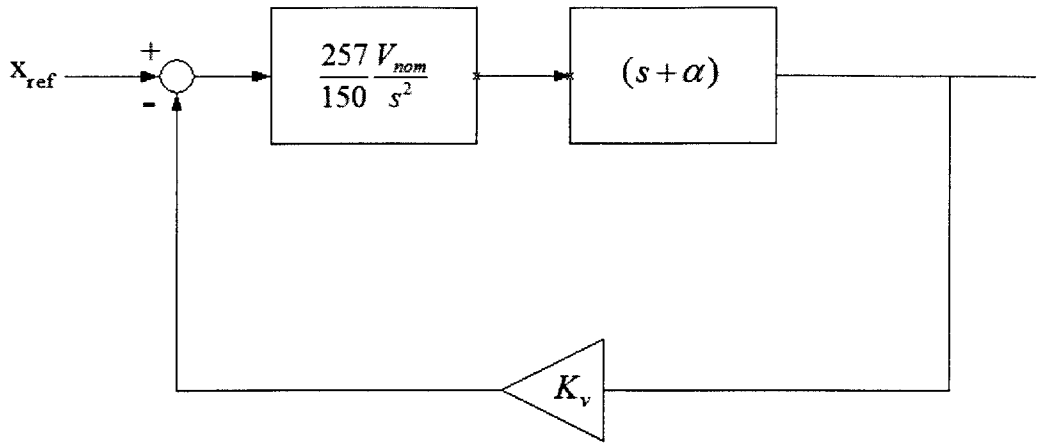

Figure 2-12: Block diagram for heading control system.

the "Experimental Results" section.

2.3.3 Position Control

Once the speed and heading control loops have been established, a position control loop must be designed to drive the rovers towards their assigned waypoints. This position controller assumes that the velocity and heading controllers can respond quickly and precisely to the position controller's commands. The position control algorithm must be able to take the current speed (V) and heading angle (θ) and adjust them appropriately. Since the speed control is being handled by the rover's on-board microcontroller, the speed variable is never changed by the position controller and can be assumed to be a nominal 0.25 m/s. A block diagram of this nested feedback loop configuration is shown in Figure 2-15.

Proportional Control Position Algorithm

One option for a position controller algorithm is the proportional control algorithm. The heading is calculated based on the current position estimate ($\hat{X}$,$\hat{Y}$) and the desired position (X, Y). The heading sent to the heading controller (θ_d) is calculated by the formula:

$$\theta_d = \arctan\left(\frac{Y-\hat{Y}}{X-\hat{X}}\right) \tag{2.24}$$

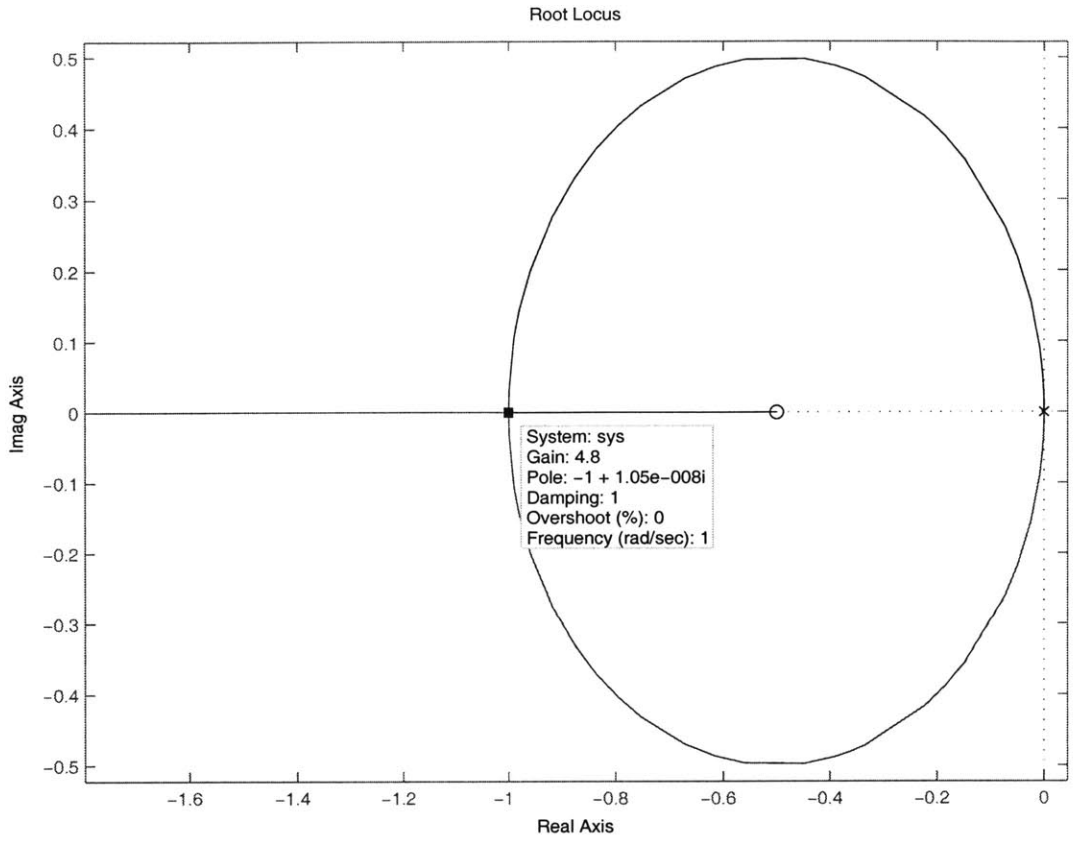

Figure 2-13: Root locus plot of the open loop transfer function $G(s)$ with PD controller, $H(s)$ for V=0.25 m/s.

Figure 2-16 gives a diagram of the rover with estimated and desired coordinates and calculated heading angle.

As noted in previous testbeds, the main advantage of this method is that most of the heading control is set at the beginning of the maneuver. [4] If a truck is already at one of its waypoints, getting the next waypoint is a relatively simple procedure provided that the waypoints are not too close together. If the waypoints are too close together, some problems can arise. If there isn't enough distance to maneuver, the rover will start to turn very sharply. If the rover is instructed to turn more than 90 degrees while it is near a waypoint, the rover's limited turning radius will make it circle endlessly around the waypoint. If the rover does reach the waypoint, it is likely that it will be in an orientation that will make navigation to the next waypoint clumsy and unpredictable.

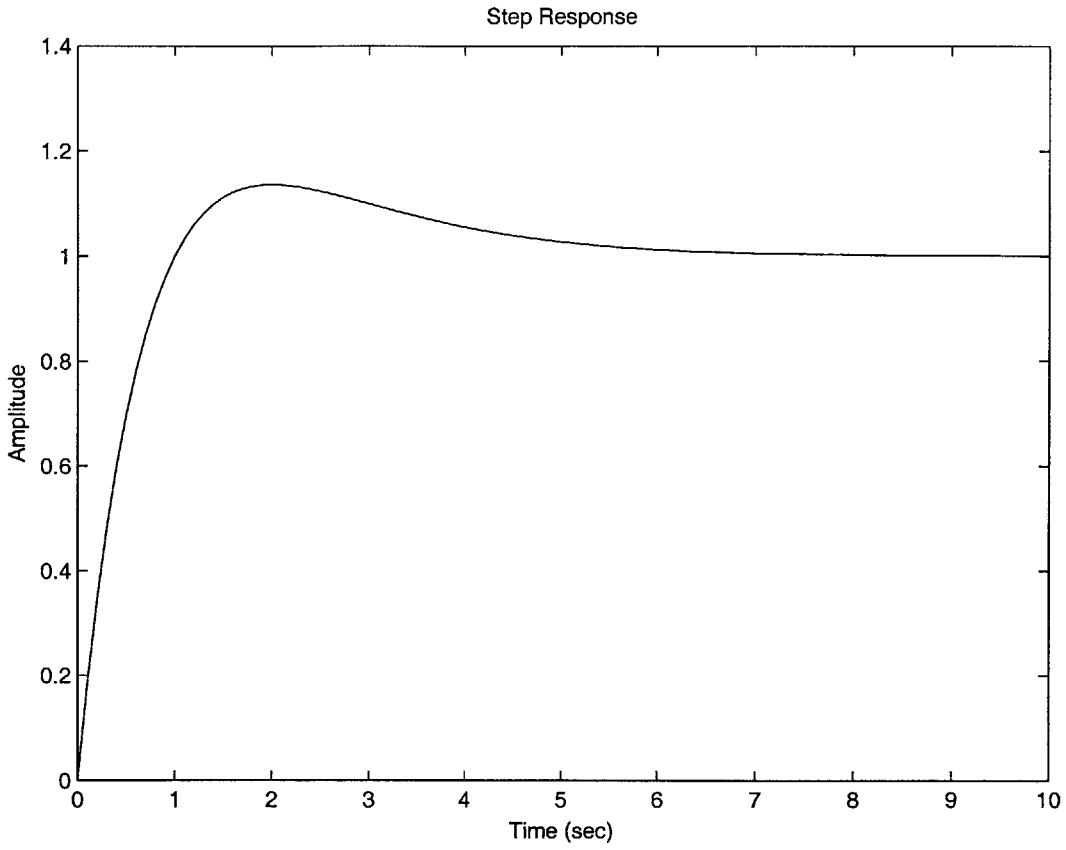

Figure 2-14: Step response of the closed loop system with the PD controller

Experiments on Pohlman's outdoor testbed showed that the proportional control algorithm works very well when waypoints are separated by at least 3 meters. [4] For the indoor testbed, this is equivalent to about 1.5 meters since the speed is scaled down by a factor of 2. While this is not too much of an issue, there is a large disadvantage to this control scheme as a whole.

The path that the rover takes between the way points is not necessarily predictable. Ideally, we would like for the rover to follow the straight line path between waypoints. With this algorithm, there is no guarantee of this since no effort is made to drive along the line connecting waypoints. Therefore, proportional control is not a good choice for this testbed.

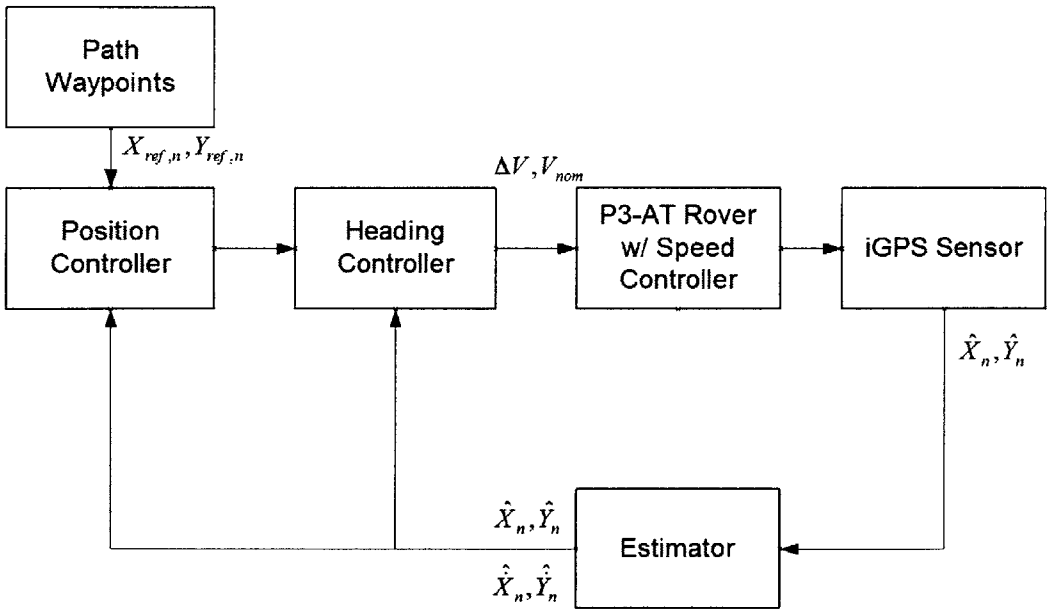

Figure 2-15: Configuration of velocity, heading, and position feedback loops

Heading Autopilot with Ground Track Control

Since being able to predict a rover's behavior between waypoints is important, we chose an algorithm that tries to minimize the error between the rover's position and the line joining two waypoints. [6] This error is called the cross-track error. Figure 2-17 depicts a rover with it's associated state estimate ($\hat{X}$, $\hat{Y}$, $\hat{\dot{X}}$, $\hat{\dot{Y}}$, $\hat{\theta}$) along with the coordinates ((X_n, Y_n) and (X_{n+1}, Y_{n+1})) of two waypoints. The vector joining the waypoints has an angle, θ_{ref}, associated with it. The cross-track error is d and the angle between the local position vector, $\vec{p}$, and the straight path vector, $\vec{r}$, is ϕ.

Given only the rover's state vector and the coordinates of the two waypoints, the desired heading angle, θ_d must be calculated. The local position vector and the straight path vector can be calculated by:

$$\vec{p} = \begin{bmatrix} \hat{X} - X_n \\ \hat{Y} - Y_n \end{bmatrix} \tag{2.25}$$

$$\vec{r} = \begin{bmatrix} X_{n+1} - X_n \\ Y_{n+1} - Y_n \end{bmatrix} \tag{2.26}$$

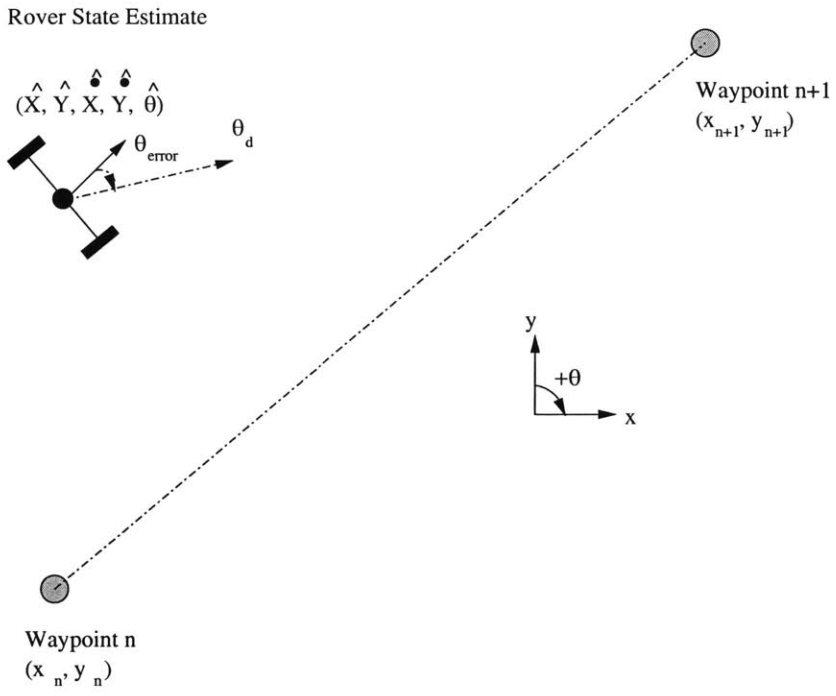

Figure 2-16: Diagram of proportional position control.

Then the angle ϕ, between $\vec{p}$ and $\vec{r}$ is calculated.

$$\phi = \frac{\det[\vec{p}, \vec{r}]}{\| \det[\vec{p}, \vec{r}] \|} \arccos\left(\frac{\vec{p} \cdot \vec{r}}{\|\vec{p}\| \|\vec{r}\|} \right) \tag{2.27}$$

Note that the normalized determinant term of the equation gives ϕ its sign. The cross-track error, d, is calculated by simple trigonometry. By the coordinate system, d is positive when the rover is on the left side of the straight path vector, so

$$d = \|\vec{p}\| \sin(\phi) \tag{2.28}$$

The rate of change of d is a function of the nominal rover speed, v_{ref}, and the difference between the rover heading angle and the reference angle θ_{ref}.

$$\dot{d} = \|v_{nom}\| \sin(\hat{\theta} - \theta_{ref}) \tag{2.29}$$

So when the heading angle and reference angle are equal, $\dot{d}$ is zero, which means that the rover is already on the ideal path between the waypoints. Thus, a heading angle

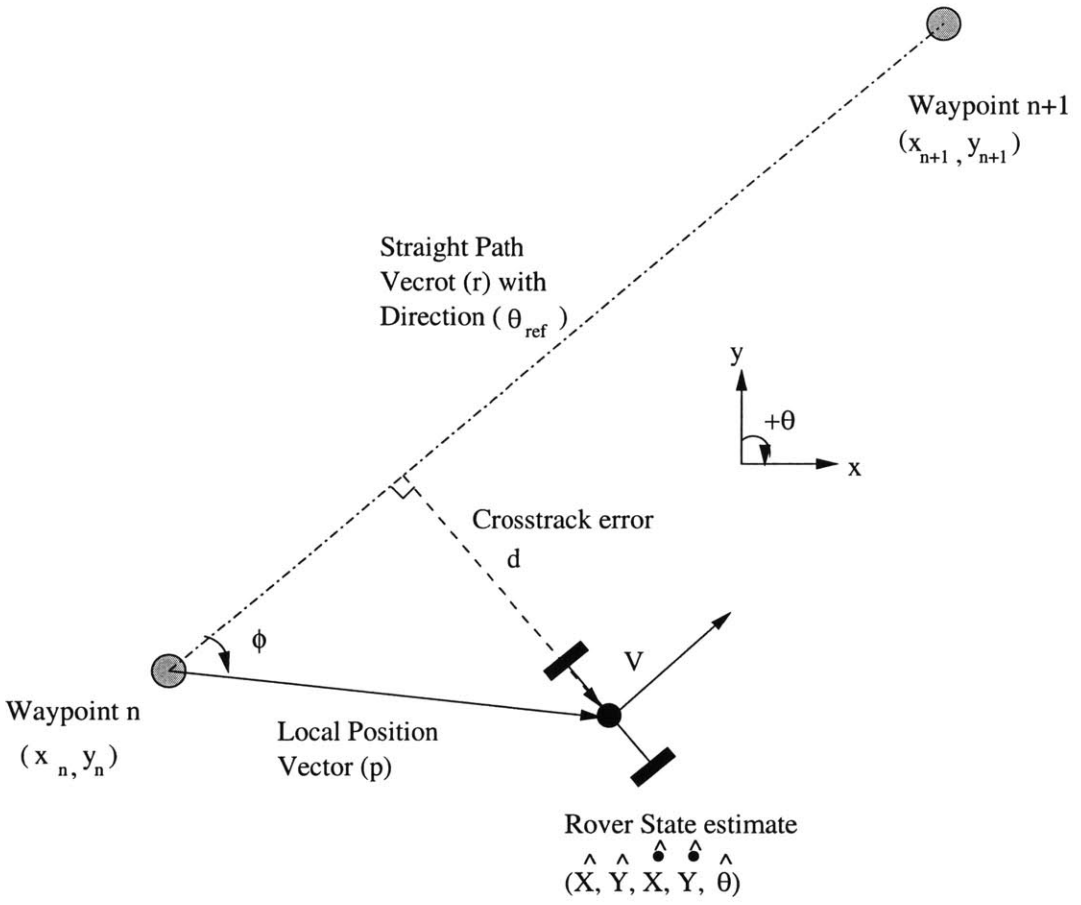

Figure 2-17: Diagram of heading autopilot with ground track control.

θ_d must be calculated based on this relationship to get $\dot{d}$ to equal zero.

Assuming that the difference between the desired heading angle and the reference angle is small, it can be approximated as

$$\sin(\theta_d - \theta_{ref}) \approx \theta_d - \theta_{ref} \tag{2.30}$$

Rearranging the variables in the frequency domain, we can see that the equation for d is a simple integrator. A lag compensator with time constant $\tau\|v_{nom}\|$ is used to derive the equation for θ_d.

$$\theta_d = \theta_{ref} + \frac{d}{\tau\|v_{nom}\|} \tag{2.31}$$

Substituting in the equation for d yields the full equation for the desired heading

angle that is to be sent to the heading controller:

$$\theta_d = \theta_{ref} + \frac{\|\vec{p}\| \sin(\phi)}{\tau \|v_{nom}\|} \tag{2.32}$$

The value of τ ($\approx 1.1 - 2.0$ seconds) was preserved from the previous testbed. [4]

Since the heading autopilot with ground track control does a better job keeping the rover on the vector that joins waypoints by taking into account the rover's current state estimate, it clearly a superior choice.

2.3.4 Control Conclusions

After the rover steering scheme was designed and its response modeled, an effective heading controller was designed. The heading controller was designed for quick response and fast settling times with minimal overshoot. Since the heading controller is fast, the autopilot ground track algorithm can guide the rover efficiently to its waypoints. The proportional control position algorithm does not take advantage of the heading controller performance.

Chapter 3

Experimental Results

3.1 Hardware

3.1.1 Robots

The robots used in testbed are ActivMedia's P3-AT robots. These rugged rovers have four drive motors with gear ratios of 66:1 and 100-tick wheel encoders. The gear ratio for these rovers isn't as high as the gear ratio for the Tamiya Mammoth Dump truck used in Pohlman's thesis. However, a high gear ratio is not required because the P3-ATs have a microcontroller that performs velocity feedback using the wheel encoders. These rovers can be controlled well at slow speeds. On a level floor, the P3-AT can move at speeds of up to 0.7 m/s. Each side of wheels is coupled and turning can only be achieved by running each side of wheels at different speeds. Therefore, the rover can have a turning radius as small as zero. On flat terrain at slow speeds, the P3-AT can carry up to 30 kilograms payload, more than enough to hold the control electronics and sensors. Figure 3-1 depicts one of the rovers. The rovers can carry up to three battery cells which allow the rover to run for 3-6 hours.

The P3-AT robot comes with a software package called Aria that allows developers to write commands to the robot using high level commands. These libraries are written in C++. When Aria functions are called, the commands are sent to the rover through the rover's serial port.

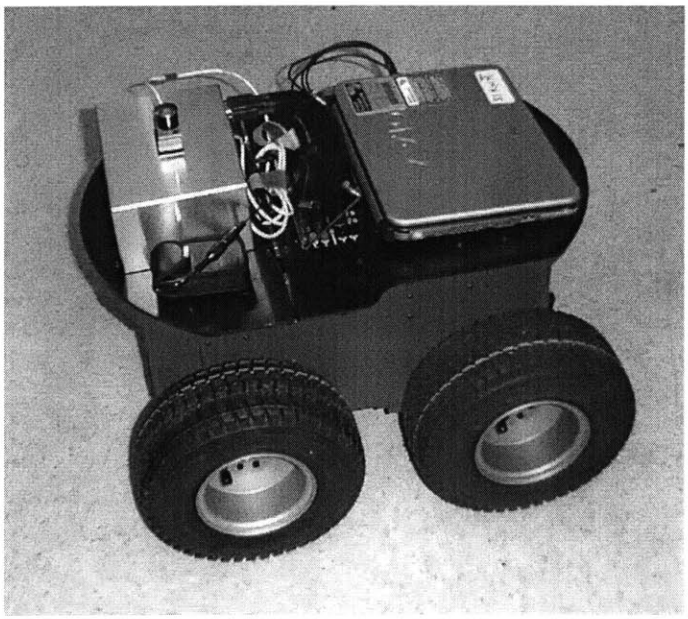

Figure 3-1: An ActivMedia P3-AT Rover.

3.1.2 Control Computer

A 850 MHz Pentium 4 Sony VAIO was mounted onboard the rover to process the incoming sensor position data and control the rover. This computer was running Microsoft Windows XP Professional. This laptop is very light weight and compact so it does not add significant payload burden to the rover. The laptop interfaces to the PCE of the iGPS sensor and the serial port on the rover through a Quatech QSP-100 PCMCIA serial port card. This card enables the Sony VAIO to have up to 4 devices connected at a time on serial ports. An outdoor GPS antenna can be attached if the user wants to use outdoor GPS for navigation.

The Sony VAIO computer also have built-in wireless ethernet capability. The base station and rovers use this wireless connection to communicate using TCP/IP. Each rover is assigned a unique IP address by which it is known to the base station. The block diagram in Figure 3-2 depicts the hardware connections on the rover.

One VAIO is used as a base station, which sends out commands to all rovers. All rovers are not aware of any other rovers in the system. The VAIO base station is the interface between the waypoint planner and the rovers. It sends waypoint coordinates over wireless ethernet to each rover. Each rover receives its waypoint

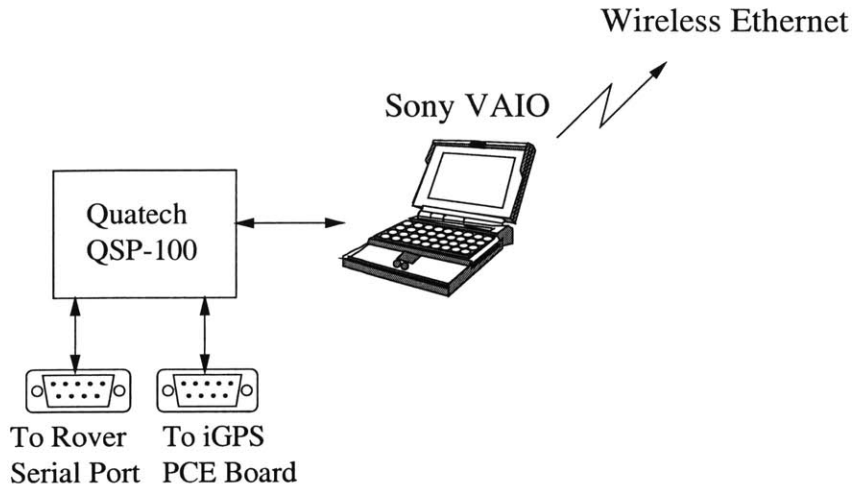

Figure 3-2: Block diagram of the Sony's interface to the rover and the indoor GPS.

command and handles all the low level control. Unlike Pohlman's previous version of this type of testbed, all computation concerning the heading and velocity control is done onboard the robot to maximize responsiveness. Previously, computation would be done on the side of the base station which introduced a time delay due to the radio modem latency. This time delay was significant enough to make the velocity and heading controllers oscillate. To ensure minimal wireless transmission latency, signal amplifiers were often used to make communication more robust.

3.2 Single Rover tests

The fist test was the straight line test. A rover was commanded to follow a straight line while initially angled 30 or 45 degrees with respect to that line. The goal of this test is to see how well the rover can follow the path despite the heading offset. The overshoot, oscillation, and settling time are the factors which determined how well the system functioned.

In the second test, a list of waypoints that mapped out an octagon was given to a rover to follow. This shape was chosen because it requires the vehicle to turn a maximum of 45 degrees at each waypoint. Since the turning radius of the rover is limited, it is unreasonable to expect the rover to make very sharp turns. Commanding it to do so saturates the heading controller and doesn't reveal much about the

performance of the system. The goal of this test was to see if the rover could perform several straight lines successively and trace out a simple closed shape.

3.2.1 Straight Line Tests

For the straight line tests, the rover was positioned at ± 30 degrees and ± 45 degrees to the desired path. Three test runs were performed for each of the angles. The estimator output was logged for each run. The iGPS was set to sample at 20 Hz with no averaging. The estimator runs at 5 Hz. Figure 3-3 shows the estimated position of a straight line transient test with an initial offset of 30 degrees. By counting the number of sample points until the rover has settled and multiplying that by the sampling period, we can calculate the settling time. The settling time is approximately 4.0 seconds as predicted by the model.

There are a few position points that don't follow the regular spacing of the other points exactly. These irregularities are due to the synchronization issues between the estimator loop and the iGPS as discussed in the "Estimator Performance" section. Figure 3-4 shows the velocity estimate for both x and y. From the figure it is clear that the rover is travelling at the nominal 0.25 m/s as expected. There is a bit of noise on the velocity estimate due to the synchronization issues discussed earlier, but since the iGPS sample rate is high compared to the estimator loop rate, the noise is within acceptable margins.

The 45 degree line transient test also showed good position results. Figure 3-5 shows the position data for the 45 degree test. This 45 degree test also had a settling time of about 4.1 seconds as predicted by the model. As before, there are small artifacts in the position data that occur for the same reasons as they occur in the 30 degree test. The corresponding velocity estimation data is depicted in Figure 3-6. The velocity plot has the same deficiency as the one for the 30 degree test. The noise is due to the synchronization issue. The rover's trajectory appears as expected despite the velocity estimate noise.

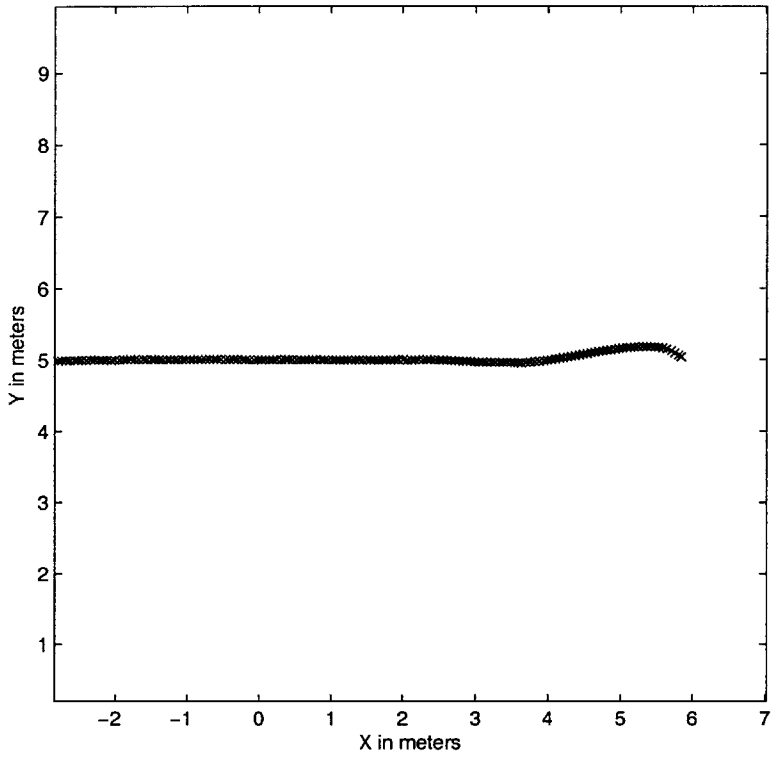

Figure 3-3: Rover line transient positions with 30 degree initial offset.

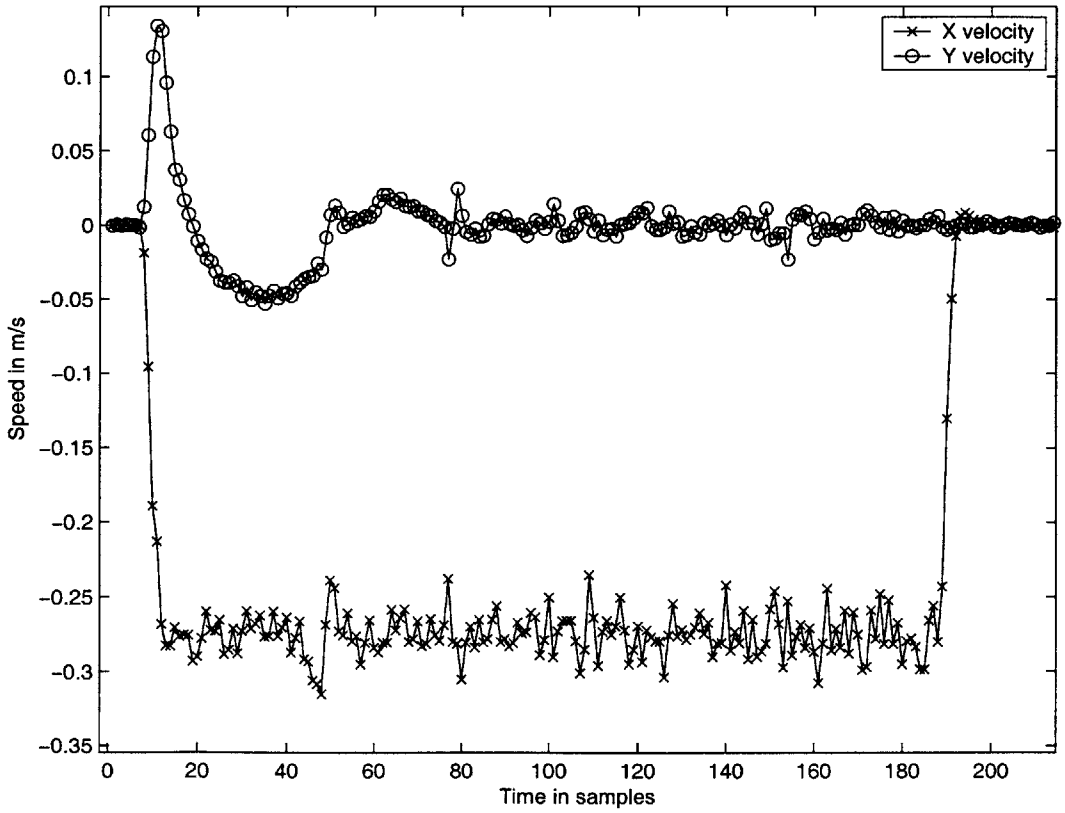

Figure 3-4: Rover line transient velocities for 30 degree initial offset.

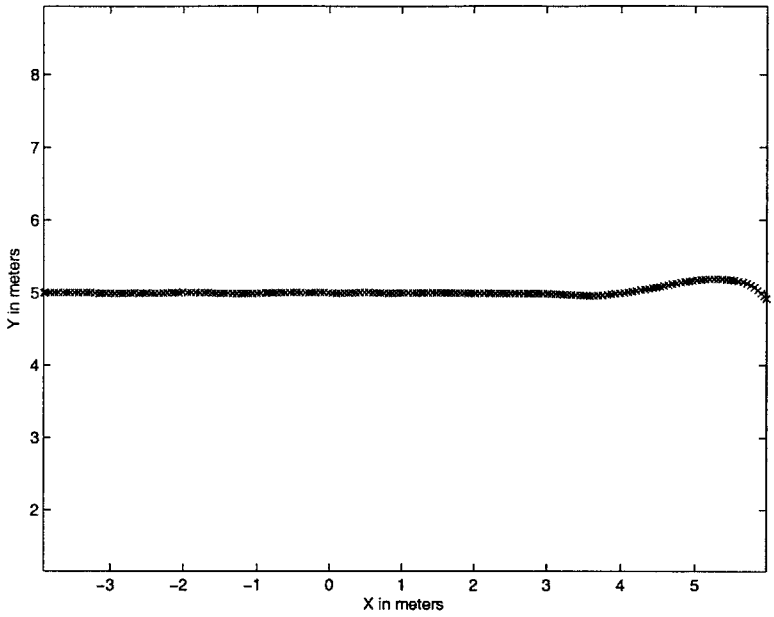

Figure 3-5: Rover line transient positions with 45 degree initial offset.

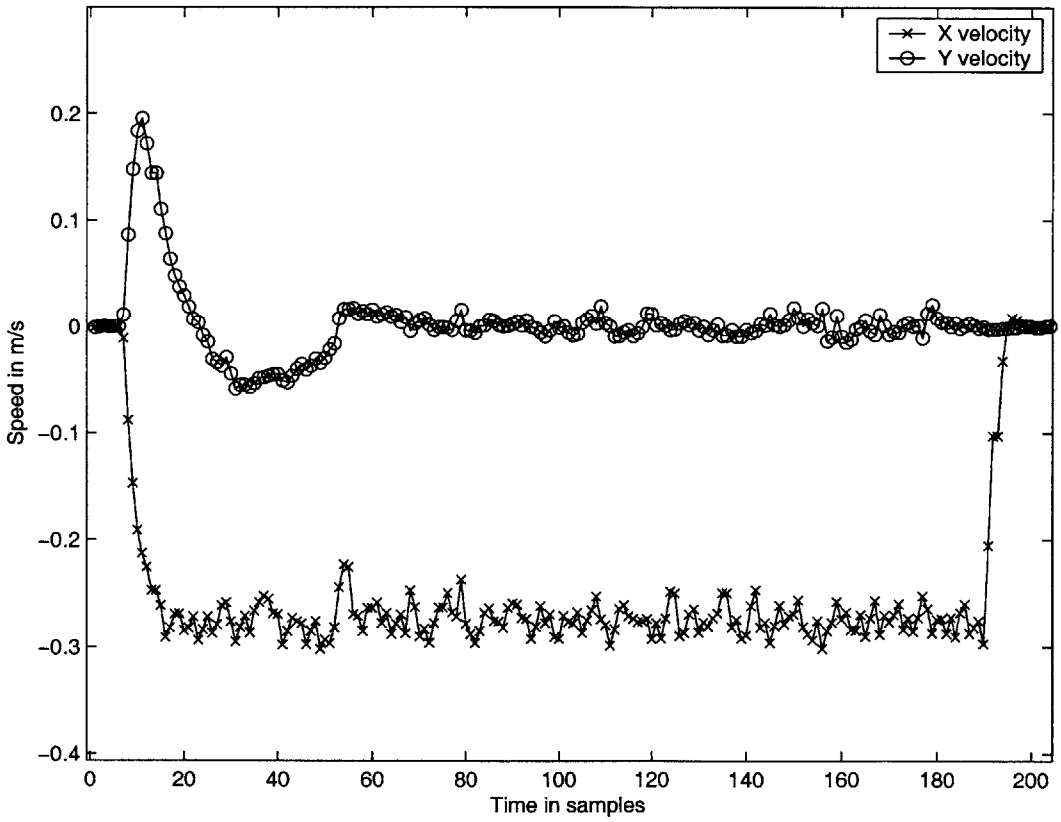

Figure 3-6: Rover line transient velocities for 45 degree initial offset.

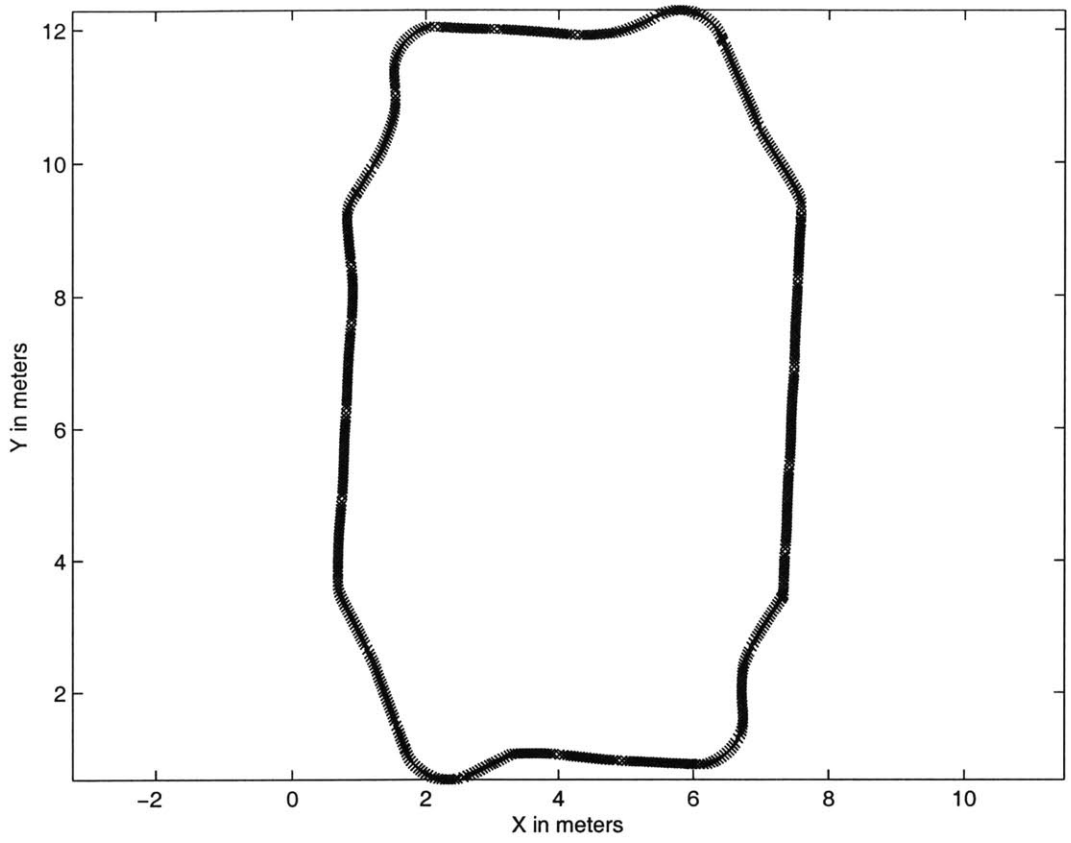

Figure 3-7: Path of rover travelling along an octagonal waypoint path.

3.2.2 Closed Curve Tests

For the closed curve tests, the rover was commanded to drive in an octagon with predetermined iGPS coordinates. The rover was driven around the octagon ending up in the same place as is started. These tests were repeated multiple times with close to identical results. Figure 3-7 shows the rover's path according to the estimator. The waypoints are denoted by circles. The rover originated at (6.005, 927) and ended it's plan there. As in the straight line transient tests, the settling time is about 4 seconds for each turn (45 degrees for an octagon). The rover did a very good job of tracing the points. The good performance is apparent by the way the rover crosses over the center of each waypoint and ends up exactly at its origin. This implies that there is not much problem with the iGPS in terms of error drift as there was in the outdoor GPS testbed [4]. Much like the line transients, there are some small irregularities in the estimated position due to the synchronization issues.

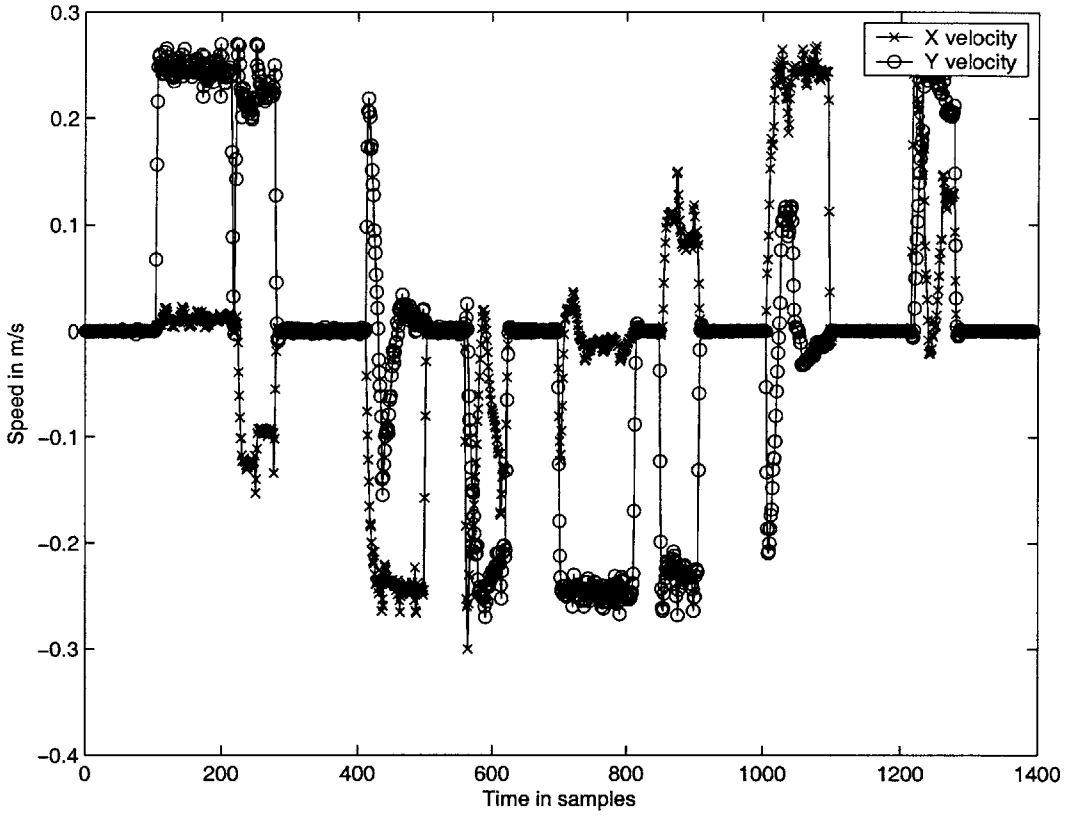

Figure 3-8: Velocity of rover travelling along an octagonal waypoint path.

The velocity plot for the octagon test is in Figure 3-8. As in the line transient case the estimated velocity is plotted along with the velocity calculated through the estimated position. The seven turns that the rover makes are distinctly seen in the velocity plot. The nominal velocity was also verified to be approximately 0.25 m/s. Much like the other velocity plots, there is spiking due to the synchronization issues. Fortunately, they do not affect the path of the rover.

3.3 Multiple Rover tests

3.3.1 Rendezvous Tests

The rendezvous maneuver involves multiple trucks originating at different points meeting in a common area. A two truck rendezvous was performed as one of the multi-vehicle tests. The trucks were placed in separate parts of the testbed and

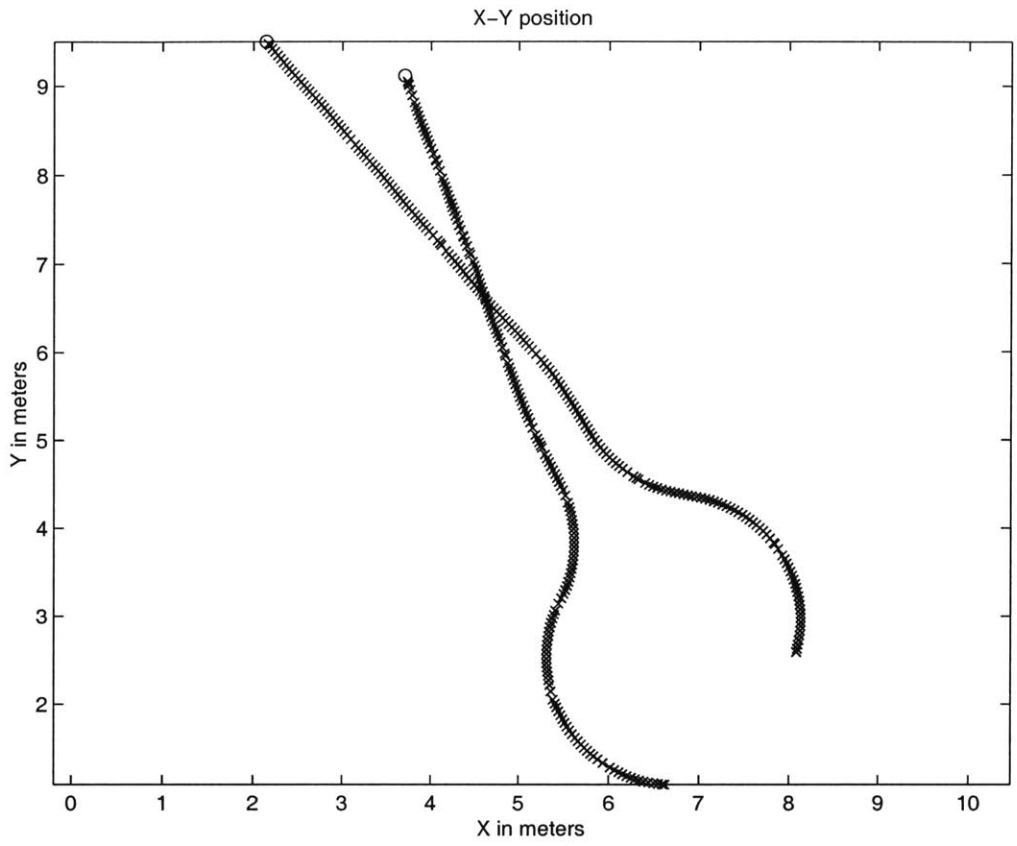

Figure 3-9: Rover position estimates for two-rover rendezvous.

were told to drive to coordinates near each other at approximately the same time. Figure 3-9 shows the paths of the two rovers path according to their estimators.

3.3.2 Closed Curve Tests

The same octagon waypoints were used for the multi-rover closed curve tests as for the single-rover closed curve tests. In this test, one rover was placed one waypoint behind the other and they were simultaneously told to drive to successive waypoints on the octagon. The result is one rover follows the other for a full lap around the octagon. The performance between the multi-rover octagon tests was indistinguishable from that of the single-rover test. Figure 3-10 shows the paths of the two rovers path according to their estimators. The leading rover is in blue and the following rover is in green. Much like the single rover test, both rovers ended their paths exactly where they started off. One rover started at (6.005,.927), the other rover started at

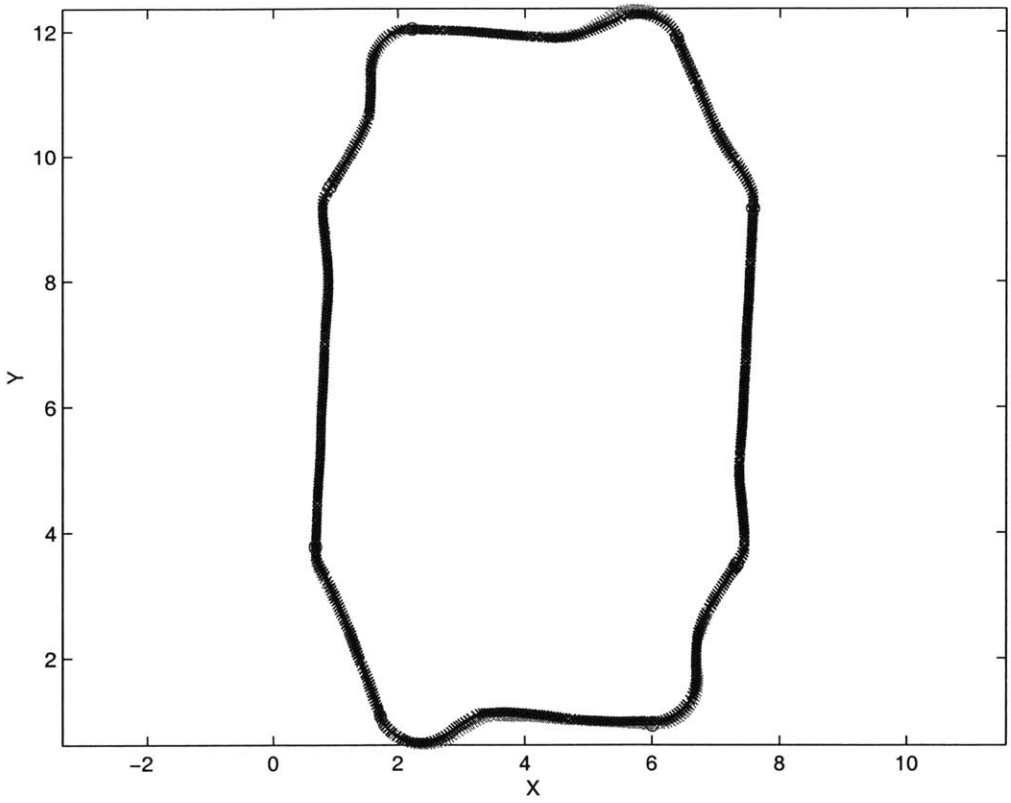

Figure 3-10: Rover position estimates for two-rover octagon test.

(7.33, 3.49). The settling times of all the turns were approximately 4 seconds. Some were slightly more due to the turn being slightly sharper than 45 degrees causing the heading controller to saturate. However, the rovers still produced good results.

Chapter 4

Conclusion

4.1 Indoor GPS Sensor

This thesis demonstrates the use of a new indoor positioning sensor for estimation of a vehicle's state. By using Kalman filtering techniques, velocity and position estimates are made and used for closed loop control of multiple rovers. The sensor proved to provide precise enough data to reliably navigate a vehicle to a desired point in the testbed. Since the indoor sensor system puts the computational burden of position calculation at each sensor, multiple vehicles can use the positioning system without affecting one another. Thus, an iGPS testbed can be scaled effectively to accommodate several vehicles.

4.2 Future Work

4.2.1 System Model

Some work needs to be done to improve the way the system is being modeled. The single-axle simplification assumed in this thesis works well, but is somewhat crude. If higher precision control is to be achieved, a double-axle model must be derived. This model must take into account the way that the two axles interact with each other when a turns of different radii are being made. Since this analysis can get quite

complex it was not explored in this thesis. However, it is necessary for high precision control work in the future.

4.2.2 Estimation Synchronization

The synchronization issue that causes the spikes in the velocity estimate also needs to be addressed. As discussed, the sample rate of the iGPS can simply be made significantly higher relative to the estimation loop rate. However, optimally, both iGPS and estimator should be running at the same rate so skipped samples never occur. In future versions of this testbed, an alternative real-time operating system, such as BeOS, should be considered. This would eliminate the velocity spikes while minimizing the noise associated with a high iGPS sample rate (as compared to the estimator loop rate).

4.2.3 Vehicle Expansion

Although this thesis only demonstrates the simultaneous use of 2 trucks, the decentralized nature of the indoor sensor allows the testbed to be easily expanded to many trucks. All computation is isolated on the computer on each rover, making it elegantly scalable. Since the low-level controls of each vehicle are abstracted away from the user, the testbed can be expanded to other types of vehicles, such as blimps or helicopters, without having to change the underlying testbed architecture. A control system and model for a new vehicle type can replace that used for the P3-AT rover.

Bibliography

[1] T. Schouwenaars, B. Mettler, E. Feron and J. P. How, "Robust Motion Planning Using a Maneuver Automaton with Built-in Uncertainties," Proceeding of the IEEE American Control Conference, June 2003.

[2] J. S. Bellingham, M. Tillerson, M. Alighanbari and J. P. How, "Cooperative Path Planning for Multiple UAVs in Dynamic and Uncertain Environments," Proceeding of the IEEE Conference on Decision and Control, Dec. 2002.

[3] T. Corazzini, A. Robertson, J.C. Adams, A. Hassibi, and J.P. How, "GPS Sensing for Spacecraft Formation Flying," Fall issue of *The Journal of The Institute Of Navigation*, Vol. 45, No. 3, 1998, pp. 195–208.

[4] Pohlman, Nick. *Estimation and Control of a Multi-Vehicle Testbed Using GPS Doppler Sensing,* S.M. thesis in the Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Sept. 2002.

[5] White paper 063102, "Constellation 3Di Error Budget and Specifications." Copyright 2002, Arc Second, Inc. Dulles, VA.

[6] J.P. How. Course notes from AA271a, "Dynamics and Control of Aircraft and Spacecraft." Spring 1999.

[7] Prof. J. How and Prof. J. Deyst. Course notes from 16.982, "Advanced Estimation for GPS and Inertial Systems." Fall 2002.

[8] Grewal, Mohinder S., and Andrews, Angus P. *Kalman Filtering Theory and Practice.* New Jersey: Prentice Hall, 1993.

[9] L. A. Gould, W. R. Markey, J. K. Roberge, and D. L. Trumper. *Control Systems Theory*. Unpublished, 1997.