Bluefield State College
Vehicle Design Report
2004

Jarrod Snider • Amy Snider • Michael Hale • George Myers • Donald Walker • Jessica Brown

I, Dr. Robert Riggins, Professor of the Department of Electrical Engineering Technology Department at Bluefield State College do hereby certify that the engineering design of the new vehicle, Vasilius, has been significant and each team member has earned two semester hour credits for their work on this project.

Signed, Date

_____ _____

Bluefield State College
Phone: (304) 327-4134
E-mail: briggins@bluefieldstate.edu

1. Introduction

The Vasilius Ground Robotic Vehicle (GRV) team of Bluefield State College (BSC) is pleased to present a new and improved Vasilius robot to the 12th Annual Intelligent Ground Vehicle Competition (IGVC). Vasilius' basic platform is the same as the one that competed in the 11th Annual IGVC. For example Vasilius still uses seven independent types of sensors, differentially steered drive wheels, extra safety features, and algorithms based on the human decision making process. However, modifications on the new Vasilius are significant. The need for significant changes emerged for several reasons. First, Vasilius failed to move off the starting spot in the 11th Annual IGVC due to a hardware failure. Our re-design reduces possibilities of additional catastrophic failures. Second, in last year's competition we listed plans for incorporating Kalman Filtering into our algorithms. This we have done, and it has resulted in several journal articles and conference papers. Third, the judges last year suggested adding mapping and learning to our algorithms, which we have also done. Fourth, we decided to follow a design process that allowed more time on software development and testing. As a result we have a completely new and superior software package. Last but not least we simply wanted to improve our previous design and design process, and have the opportunity to show that Vasilius can perform all the tasks well in the 12th Annual IGVC.

2. The Original Platform

Before specifying our new design process, we will present Vasilius' mechanical and electrical systems as we had designed them for the 11th Annual IGVC. The original Vasilius mechanical and electrical designs are superb and therefore serve as a great "starting point" platform on which to plan the new Vasilius. We also include other design considerations in this section: safety, reliability, durability, and cost.

2.1 Mechanical System

The overall mechanical design of Vasilius focuses on simplicity, durability, compactness, maintainability and most importantly, safety. The vehicle is designed to operate and safely navigate in both indoor and outdoor environments. This small and versatile design provides the opportunity to test and develop the human-like system on a fully functional platform. The mechanical design can be divided into three separate categories: vehicle frame, drive system, and vehicle body.

2.1.1 Vehicle Frame

The vehicle frame is constructed of steel tubing. Steel tubing was chosen due to its light weight, durability, and ability to house wiring. The tubing acts as a conduit to conceal and organize connections as well as to shield vulnerable lines from RF noise. The rectangular design allows the frame to be strong while creating a protective carriage that houses the batteries, chargers, and other various components.

2.1.2 Drive System

Vasilius uses two 24-volt DC motors to power two drive wheels independently. The motors attach to the drive wheels at 90-degree angles and pivot vertically through brackets welded to the frame. The brackets prevent any horizontal movement reducing stress on the motors. The motors attach to the suspension system and travel with the wheels independently. The angles that the motors are mounted also vary as the vehicle travels across uneven ground. This ensures that a motor will not hit the ground

when its respective wheel enters a hole. The two rear wheels are free to rotate and change direction as the vehicle changes course. The rear wheels are mounted on a pivoting arm that allows the wheels to travel vertically, independent of the main drive wheels. The pivoting arm allows 30 degrees of rear wheel travel in both directions.

2.1.3 Vehicle Body

The vehicle's body framework is constructed of aluminum tubing. The exterior of the body consists of aluminum panels with lexan inserts around the entire surface. The panels are held in place with quarter-turn fasteners that can be removed by hand very quickly. Due to the number of panels and their positions, components can be added or removed easily. The body of Vasilius protects components from water and the internal heat that the vehicle generates. It is equipped with fans that cool and circulate the air inside the vehicle. Shelving inside the vehicle's body allows for component positioning and spacing, assisting in cooling the interior of the structure.

2.2 Electrical System

In the pursuit of a more human-like autonomous vehicle, a complex electrical system was designed to better imitate the human decision-making process. This imitation requires many sensors, multiple computers, a great deal of simultaneous processing, and various levels of redundancy. The system consists of four parts: the power system, sensors, computers, and vehicle control.

2.2.1 Power System

Two 12-volt deep cycle marine batteries connected in series provide the power to the controller, motors, vision computer and LMS. Two smaller 12-volt batteries power the sensors, emergency stop contactor, and a DC-DC converter. The on-board laptop is equipped with two batteries for its own power. The DC-DC converter provides +12V, -12V, and 5V for the various requirements of the electronics. After performing a power consumption analysis, the team balanced the power consumption across all batteries. In normal operation, the vehicle operates for six hours on a fully charged set of batteries.

Vasilius is equipped with its own onboard charging system. The charging system consists of one 24-volt charger and two 12-volt chargers. The on-board laptop also has its own charger. Once switched to a charging mode, all batteries and electronics are isolated.

2.2.2 Sensors

Vasilius is equipped with a multitude of sensing devices. This variety of sensors was chosen to provide various levels of data and redundancy similar to human senses. The following are the sensors onboard Vasilius, a brief summary and their respective data:

- **Stereoscopic Camera** – The stereo camera mimics human eyes. Like the human brain, Vasilius can take two slightly different images and create one image with depth information. Camera data contains the entire environment: lines, potholes, obstacles, etc.

- **LMS** – The LMS uses a laser to scan 180 degrees of the environment the vehicle is traveling towards. The LMS data contains the precise distance and angle of all obstructions in the plane of the laser.

- **DGPS** – The Differential Global Positioning System (DGPS) receiver uses global positioning satellites to obtain a position fix. It then uses a reference station and/or WAAS satellites to obtain corrections that improve accuracy. The DGPS data contains position (latitude, longitude), heading, and velocity.

- **Digital Compass** – The digital compass detects the earth's magnetic fields. The digital compass data contains very accurate heading when moving slowly or stationary.

- **Encoders** – The encoders detect movement of the motor shaft with great precision. The data from the encoders contain position, velocity, and azimuth.

- **Diffuse Sensors** – By emitting light that reflects from a surface back to the sensor, the frequency can be analyzed and compared to a programmed frequency. The sensors can be programmed to detect a particular frequency (color) on the ground.

- **Proximity Sensors** – By emitting light that reflects from a surface back to the sensor, an obstruction can be found.

2.2.3 Computers

Designing an electrical system modeled after human decision-making required a great deal of processing power. The human brain is divided into many sections that are responsible for different thinking processes. These processes occur simultaneously, yet separate from one another. However, the processes communicate and update each other constantly. The Vasilius team was interested in two of the processes; planning and reaction. The computing system was divided into two parallel systems to achieve this idea. A central computer is responsible for planning and vehicle control, while a second computer constantly checks for unforeseen situations and correct execution of the desired plan. Working together, the computers can provide a redundant, effective, and self monitoring means of navigation. A third, off-board, computer was also implemented to provide remote control, monitoring, and convenience. Figure 2.1 shows the two onboard computers as well as the off-board computer. The two onboard computers make up two parallel subsystems that create the entire electrical system.

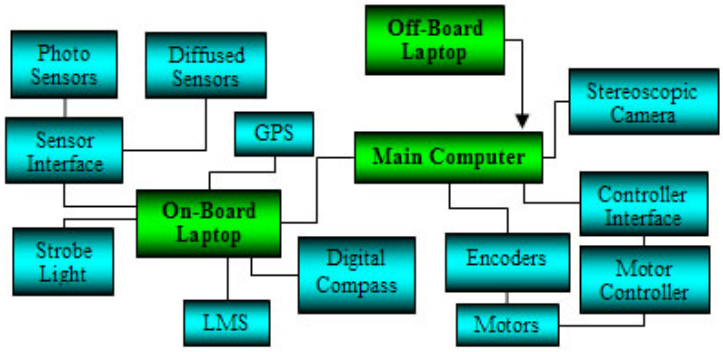

Figure 2.1 Computer Integration

2.2.4 Vehicle Control

Vasilius uses a closed-loop proportional control system. The team designed and built an interface that provides the controller with analog signals from the computer's digital signal. Encoders monitor the motors and provide feedback to the control algorithm. Figure 2.2 shows a block diagram of Vasilius' control system.

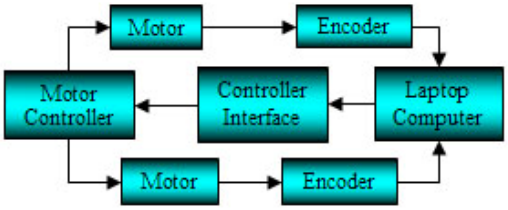

Figure 2.2 Control System

2.3 Other Design Considerations

2.3.1 Safety

Safety was an important concern in all aspects of the design, fabrication, and operation of Vasilius. This was accomplished through many different processes and was infiltrated throughout our design process. Especially important are the two manual pushbuttons located at the rear of the vehicle which, when pressed, disconnect power to the motors thereby effectively stopping the vehicle. In addition, we included a remote e-stop system consisting of a transmitter and

an onboard receiver. Fuses, circuit breakers, and disconnect switches protect all of Vasilius' components from overloads, noise spikes, and short circuits.

2.3.2 Reliability

We have stressed the reliability of Vasilius by using redundancy. Different sensor groups have redundant functions. Both the stereo camera and the diffused sensors detect the presence of lanes. The stereo camera algorithm also doubles with the LMS and proximity sensors in detecting objects.

2.3.3 Durability

Vasilius' solid mechanical design makes the vehicle very durable. Its framework houses and protects components. The exterior shell of the vehicle prevents water and debris from coming in contact with the electrical system. Components on the exterior of the vehicle are waterproofed and designed to withstand minimal damage. The vehicle can be operated under normal circumstances without fear of accidentally damaging vital components or affecting the vehicle's overall performance.

2.3.4 Cost

Item	Actual Worth	Our Cost
Main Computer/ Framegrabbers	$4,200.00	$3,410.00
LMS 220/interface	$8,852.46	$5,707.72
Two Laptops	$4,800.00	$4,232.00
DGPS	$3,000.00	$2,100.00
Six Diffuse Sensors	$517.50	$517.50
Two Encoders	$1,322.00	$1,322.00
Digital Compass	$806.00	$806.00

Item	Actual Worth	Our Cost
Two board-level cameras Two 6mm lenses	$960.00	$960.00
Six Proximity Sensors	$600.00	$90.60
Steel for Frame	$200.00	$0.00
Aluminum/Lexan	$500	$0.00
Miscellaneous	$600.00	$600.00
Total	$25,357.96	$19,745.82

Table 2.1 Estimated Cost by Component

3. Design Process

Since Vasilius was originally designed, fabricated, and tested in preparation for the 11th Annual IGVC, a customized design process that could optimize an existing vehicle had to be developed. Our goal was to come up with a process that would encompass the improvement of an already excellent platform and allow considerable time for software development. We developed the process shown in Figure 3.1 immediately following last year's competition.

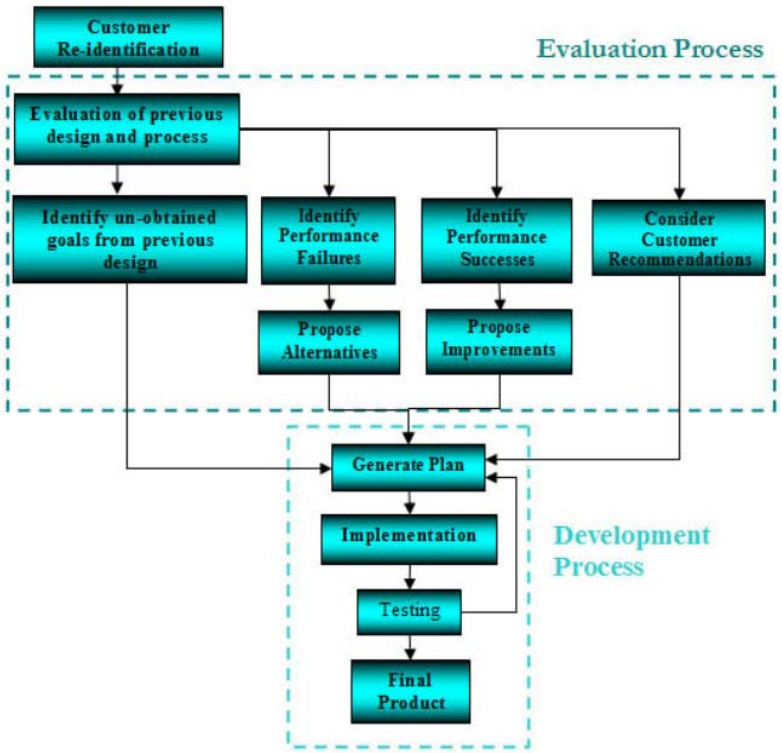

Figure 3.1 Design Process

3.1 Customer Re-identification

The first step in our design process was to re-identify our customer. In the 11th Annual

competition we identified our customer as actually being a group of customers. Again we found that we could not pinpoint a single customer. This project is significant to many, such as the IGVC judges and our team. We strive to do our best for our sponsors and for our community. Again we consider all of these people a part of the customer. In addition, this year we really strived to be innovative and apply some modern theoretical approaches to autonomous navigation. We focused on intelligence and conducted research in many areas of robotics. Therefore, we also consider the robotics research community as part of the customer and we hope to provide significant contributions to the field of robotics through our research.

3.2 Evaluation Process

After identifying a customer, we decided to hold a series of meetings to evaluate the original vehicle and the original design process. This evaluation was necessary to formulate any plans to improve the vehicle. We chose to conduct this evaluation in a systematic way such that both problems and solutions could be exploited. First we identified all of the goals set in the original design process that were not obtained due to time constraints. We wanted to continue the original process as well as improve the vehicle with a new one. The implementation of an Extended Kalman Filter (EKF) and learning algorithms were two goals not met the previous year. Next we identified all the performance failures as well as all of the performance successes. The team members individually proposed alternative methods to overcome failures. They also proposed improved methods that could make Vasilius' successful qualities even better. The performance failures consisted mostly of hardware vulnerabilities and processing speed problems. Some performance successes were the maneuverability, compactness, sensor selection, dual processing capabilities, and algorithms designed to be more human-like in decision-making. Finally we considered customer recommendations. For example, the IGVC judges were disappointed in the lack of environment mapping and learning capabilities of the vehicles. This preliminary evaluation allowed us to understand the problems encountered in previous years and to have a better understanding of what would be necessary to improve Vasilius.

3.3 Development Process

The next portion of the design process is the development process. Considering we already had a suitable platform, the development section was included in the design process to really focus on the intelligence of the vehicle. Therefore, this section consumed most of the time allowed for the project.

Once a complete evaluation and analysis of the original vehicle was conducted and proposed improvements presented, a group discussion was held yielding a plan for the re-design of Vasilius. This plan would act as a "road map" of the work to be completed and fully tested in time for the 12th annual competition. It included mostly software changes. The original software was to be completely rewritten. We chose to develop new software that had a better means of system and sensor integration. We added an EKF algorithm to integrate all of the localization and orientation sensors, a correlation algorithm to integrate LMS and vision data, a dynamic memory mapping algorithm to integrate all of the data to produce a map, and a new path-planning algorithm to determine long range and short range paths through the mapped environment as well as to determine the appropriate vehicle dynamics. Some additional platform changes were also made to support the new software and improve sensing capabilities. The platform changes and software design are discussed in detail in later sections of this paper.

After formulating a plan, we began the implementation phase. This is the process of completing the necessary platform changes and writing the above-mentioned software. Next the new changes were tested and a new plan developed

to continuously improve the capabilities of the vehicle. This loop was designed to continue until the day of the competition. We have been in the development process loop since mid September and are very excited with our progress. We have a great deal of confidence in our vehicle due to this carefully thought out design process.

4. Platform Changes

Based on our evaluation process, the Vasilius team proposed a number of changes to the original platform, especially in software. Our new software design focuses on two primary functions: sensor information integration and path finding. Section 5 discusses the software re-design in detail. In order to implement our software objectives, we also proposed a number of hardware changes.

The first hardware change involved Vasilius' two on-board computers. The old platform used a decentralized control scheme and both computers shared control decisions and sensor interfaces. The new platform has the same two computers, but one is dedicated solely to vision while the other controls everything else. The main computer views the vision computer and camera operating together as an ensemble vision sensor. Our analysis shows that dedicating one computer to vision distributes computer resources more efficiently. The Ethernet interface between the two computers gives adequate speed for transferring processed vision information between computers.

Another hardware change is in the vision system itself. The original platform used stereoscopic vision. These cameras were gray-scaled, and had no automatic lighting or focusing, making it difficult to operate in different environments. We replaced these two cameras with a single camcorder. Now the vision system has automatic lighting and focusing as well as color information. We can also add a second camcorder for stereoscopic vision, but that is unnecessary for this competition.

One of the alternatives we proposed as a result of the evaluation process was to re-design the connection between the main computer and all the non-vision sensors. Our original platform used a serial port distribution center that failed during the 11th Annual IGVC, causing Vasilius to fail in the autonomous and GPS competitions. Our re-design uses two pc-card adaptors with eight serial ports. Now reliability is higher and programming is easier.

Finally, we also decided from the evaluation process to increase the input/output capabilities of Vasilius. The original platform had a small number of binary inputs and outputs, limiting the addition of on/off reaction sensors and lights. Vasilius now has a new 96 input/output port that accommodates our sensor needs and allows for considerable upgrades.

5. Software Design

The software design of Vasilius was chosen to optimize the human-like senses, redundancy, and parallelism while maintaining reasonable processing time and providing the control system with inputs at least once per 100 milliseconds (ms). There are four main components of Vasilius' software design. The first software component is the main navigation algorithm. This algorithm is responsible for integrating all sensor data, building a map, and path-planning. The main navigation algorithm requires inputs from the other three components. These three components are the application algorithms which are responsible for providing the main navigation algorithm with details about what is to be detected, if a target is to be found, *etc*. The application algorithms used for the competition are: lane following, obstacle avoidance, and waypoint navigation.

5.1 Main Navigation Algorithm

5.1.1 Sensor Integration

5.1.1.1 Integration of Localization and Orientation Sensors

For Vasilius to take advantage of its wide array of sensors, the software must integrate all sensor information in some optimal or near-optimal scheme. We have designed an algorithm to do just that. The algorithm stores a dynamic 3-D map in a data file. It is "dynamic" because this data file is accessible in real-time. An EKF provides estimates of robot trajectory and obstacle positions based on difference equations representing the dynamics of Vasilius. The algorithm then places these estimates on the map. In the remaining portion of Section 5.1.1.1 we describe the design of this algorithm.

5.1.1.1.1 Deterministic Dynamic Model

To provide estimates of the robot trajectory we must first develop an appropriate deterministic (*i.e.*, "ideal," with no uncertainties) model describing the dynamics of Vasilius. We must also develop the ideal equations relating the sensor measurements to the trajectory. Using both the dynamic model and the set of measurement equations will enable the algorithm to determine the best estimate of robot trajectory in the presence of uncertainties. The parameters of the robot trajectory, or states, are as follows:

$H \equiv Current\ heading\ of\ robot$
$x \equiv Current\ x\ position\ or\ longitude\ of\ robot$
$y \equiv Current\ y\ position\ or\ latitude\ of\ robot$

The position (x,y) is the same as (longitude, latitude) of the robot, and so we will refer to the coordinate frame as the "earth frame." In the earth frame, the y-axis points to the north and the x-axis points to the east. Vasilius uses differentially controlled steering with velocity commands to each motor:

$U_L \equiv Velocity\ command\ to\ left\ motor$
$U_R \equiv Velocity\ command\ to\ right\ motor$

The differential equations that govern the robot motion are

$$\begin{aligned} \frac{dH}{dt} &= \frac{U_L(t) - U_R(t)}{b} \approx \frac{H_{K+1} - H_K}{\Delta t} \\ \frac{dx}{dt} &= \left(\frac{U_L(t) + U_R(t)}{2}\right)\sin(H(t)) \approx \frac{x_{K+1} - x_K}{\Delta t} \\ \frac{dy}{dt} &= \left(\frac{U_L(t) + U_R(t)}{2}\right)\cos(H(t)) \approx \frac{y_{K+1} - y_K}{\Delta t} \end{aligned} \quad (1)$$

where b is the width of the robot, t is the time variable, $\Delta t = t_{K+1} - t_K$, and t_K represents an instant in time parameterized by the index K. Rearranging yields the following difference equations:

$$\begin{aligned} H_{K+1} &= H_K + \frac{(U_{LK} - U_{RK})\Delta t}{b} \\ x_{K+1} &= x_K + \Delta t\left(\frac{U_{LK} + U_{RK}}{2}\right)\sin\left[H_K + \frac{(U_{LK} - U_{RK})\Delta t}{b}\right] \\ y_{K+1} &= y_K + \Delta t\left(\frac{U_{LK} + U_{RK}}{2}\right)\cos\left[H_K + \frac{(U_{LK} - U_{RK})\Delta t}{b}\right] \end{aligned} \quad (2)$$

The parameters U_{LK} and U_{RK} represent the velocity inputs to the left and right motors respectively at time K. For Vasilius we set the time interval Δt to 100ms and apply new velocity inputs at the beginning of each time interval.

During each interval we assume constant velocity inputs. However, the other two remain approximate since heading may change throughout the interval. Note, when $U_{LK} = U_{RK} = 0$, Equation Set 2 becomes linear: $V_{K+1} = AV_K$ where

$$V \equiv \begin{bmatrix} H & x & y \end{bmatrix}^T \qquad 3$$

and A is the 3 by 3 identity matrix. Therefore, in the absence of inputs, Vasilius maintains position and heading as described by the linear matrix difference Equation $V_{K+1} = AV_K$. In the presence of inputs, Vasilius changes position and heading according to the nonlinear difference relations of Equation Set 2.

The DGPS receiver gives $(H_{K+1}, x_{K+1}, y_{K+1})$ directly. However, the encoders give the displacements (n_{K+1}, m_{K+1}), which indirectly yields $(H_{K+1}, x_{K+1}, y_{K+1})$ with the use of the following relations:

$$\begin{aligned} H_{K+1} &= H_K + \frac{(n_{K+1} - n_K) - (m_{K+1} - m_K)}{b} \\ x_{K+1} &= x_K - \frac{b}{2}\left[\frac{(n_{K+1} - n_K) + (m_{K+1} - m_K)}{(n_{K+1} - n_K) - (m_{K+1} - m_K)}\right]\left[\cos(H_{K+1}) - \cos(H_K)\right] \\ y_{K+1} &= y_K + \frac{b}{2}\left[\frac{(n_{K+1} - n_K) + (m_{K+1} - m_K)}{(n_{K+1} - n_K) - (m_{K+1} - m_K)}\right]\left[\sin(H_{K+1}) - \sin(H_K)\right] \end{aligned} \qquad 4$$

where n_K and m_K are the left and right encoder wheel position measurements at time t_K. Therefore, we have six measurements of the state components: three $(H_{K+1}, x_{K+1}, y_{K+1})$ from DGPS and three $(H_{K+1}, x_{K+1}, y_{K+1})$ from the encoders. Setting this up in matrix form with Z_{K+1} as a 6 by 1 measurement vector, we have the following linear matrix equation:

$$Z_{K+1} = CV_{K+1} \qquad \text{where} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T \qquad 5$$

The first three components of Z_{K+1} come from DGPS and the remaining three components come from the encoders using Equation Set 5.

5.1.1.1.2 Stochastic Model

Equation Set 2 and Equation 5 represent the dynamics of Vasilius and the measurement relationships without considering uncertainties (wheel slippage, modeling approximations, measurement biases and higher order errors.) In order to properly integrate the measurement data and the model dynamic equations, we will have to determine the relative "goodness" of the data and model by characterizing these uncertainties. For our model uncertainty we assume additive white Gaussian noise:

$$\begin{aligned} H_{K+1} &= f_1(H_K, U_{LK}, U_{RK}) + w_{1K} \\ x_{K+1} &= f_2(H_K, U_{LK}, U_{RK}) + w_{2K} \\ y_{K+1} &= f_3(H_K, U_{LK}, U_{RK}) + w_{3K} \end{aligned} \qquad 6$$

where f_1 represents the first (linear) equation of Equation Set 2, and f_2 and f_3 represent the nonlinear equations of Equation Set 2. Each difference equation contains a unique additive noise. We assume a diagonal and constant noise covariance matrix *Q*:

$$Q = E\left[ww^T\right] = diagonal\left[E\left(w_1^2\right) \quad E\left(w_2^2\right) \quad E\left(w_3^2\right)\right] \qquad \text{where} \qquad w \equiv \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}^T \qquad 7$$

and E is the expected value. Our assumption of a diagonal covariance matrix means we assume no cross-covariance between noise components.

For our sensor data (or measurement data) uncertainty, we also assume additive white Gaussian noise:

$$Z_{K+1} = CV_{K+1} + v_{K+1} \tag{8}$$

where $v_{K+1} \equiv \begin{bmatrix} v_{1(K+1)} & v_{2(K+1)} & v_{3(K+1)} & v_{4(K+1)} & v_{5(K+1)} & v_{6(K+1)} \end{bmatrix}^T$

and $Z_{K+1} \equiv \begin{bmatrix} Z_{1(K+1)} & Z_{2(K+1)} & Z_{3(K+1)} & Z_{4(K+1)} & Z_{5(K+1)} & Z_{6(K+1)} \end{bmatrix}^T$

Matrix Equation 8 represents six unique measurement equations: three measurement equations from DGPS and three measurement equations from the encoders. Each measurement equation contains a unique additive noise term. Again, we assume a diagonal covariance matrix R:

$$R = E\left[vv^T\right] = diagonal\left[E\left(v_1^2\right) \quad E\left(v_2^2\right) \quad E\left(v_3^2\right) \quad E\left(v_4^2\right) \quad E\left(v_5^2\right) \quad E\left(v_6^2\right)\right] \tag{9}$$

5.1.1.1.3 Extended Kalman Filter Algorithm

In the previous two sections we presented sets of relations describing the dynamics of Vasilius. In these relations, the future state V_{K+1} of Vasilius depends on the past state V_K and input U_K. We also developed equations relating measurement data at t_{K+1} to V_{K+1}. Now we will present an Extended Kalman Filter algorithm that estimates the state by combining two sources of information: the model and the measurements. We present an algorithm that combines all available information to produce our best possible estimate of robot trajectory at each time.

Step 1. Set up initial conditions.

- Set $K = 0$ (Vasilius has not started moving)
- Average DGPS data to obtain initial position, $\hat{x}_0$ and $\hat{y}_0$.
- Average compass data to obtain initial heading, $\hat{H}_0$.
- If necessary, type changes to the default values of the nine tuning parameters in matrices Q and R on the screen.
- Set the initial error covariance matrix to all zeros:

$$P_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

By setting this matrix to zero, we assume perfect initial state values obtained from averaging.

Step 2. Propagate the state ahead for 100 milliseconds.

- Give velocity commands to each wheel, U_{LK} and U_{RK}, either to direct the robot toward a target or to avoid an obstacle. Now Vasilius is moving. Start the 100ms timer.
- Propagate the state forward in time for 100ms using Equation Set 1:

$$\hat{H}_{K+1}^- = \hat{H}_K + \frac{(U_{LK} - U_{RK})(0.1)}{b}$$

$$\hat{x}_{K+1}^- = \hat{x}_K + (0.1)\left(\frac{U_{LK} + U_{RK}}{2}\right)\sin\left[\hat{H}_K + \frac{(U_{LK} - U_{RK})(0.1)}{b}\right]$$

$$\hat{y}_{K+1}^- = \hat{y}_K + (0.1)\left(\frac{U_{LK} + U_{RK}}{2}\right)\cos\left[\hat{H}_K + \frac{(U_{LK} - U_{RK})(0.1)}{b}\right]$$

The relations above give *a priori* estimates before the next measurement at time K+1 comes in.

Step 3. Propagate the error covariance ahead for 100 milliseconds.

- The covariance propagation technique in an EKF provides a method to propagate the error covariance forward in time for 100ms:

$$P_{K+1}^- = AP_KA^T + Q = P_K + Q$$

This relation gives the *a priori* state error covariance before the next measurement comes in.

Step 4. Compute the Kalman gain.

- Determine which measurements will come in at t_{K+1}. There are four possible sets of measurements:

  Case 1…....DGPS and encoder measurements
  Case 2……Encoder measurements
  Case 3……DGPS measurements
  Case 4……No measurements available

  Encoder measurements come in every 100ms while DGPS measurements come in every one-second. If the algorithm detects bad or missing measurements for both encoders and GPS, the algorithm chooses *Case 4*.
- Determine the proper *C* matrix. The *C* matrix of Equation 5 changes depending on what measurements are available
- Compute the Kalman gain for time *K*+1:

$$KG_{K+1} = P_{K+1}^{-} C^T \left(C P_{K+1}^{-} C^T + R \right)^{-1}$$

where *KG* is the Kalman gain. The size of the matrix inside the inverse is 6 by 6. Normally, taking the inverse of such a large matrix is computationally burdensome. However, for this application, the matrix inside the inverse takes on a special form making computations simple.

Step 5. Input new measurements at time K+1.

- Input DGPS Measurements. Three measurements may come from the DGPS receiver if available:

$$\begin{aligned} Z_{1(K+1)} &= GPS\_H_{K+1} \quad \textit{Heading from the DGPS receiver} \\ Z_{2(K+1)} &= GPS\_x_{K+1} \quad \textit{Longitude from the DGPS receiver} \\ Z_{3(K+1)} &= GPS\_y_{K+1} \quad \textit{Latitude from the DGPS receiver} \end{aligned} \tag{10}$$

- Input encoder measurements. Three other measurements come from the encoders if available. To get heading and position from the encoders, we must first use Equation Set 4 with the actual displacement measurements n_K and m_K:

$$\begin{aligned} ENC\_H_{K+1} &= \hat{H}_K + \frac{(n_{K+1} - n_K) - (m_{K+1} - m_K)}{b} \\ ENC\_x_{K+1} &= \hat{x}_K - \frac{b}{2}\left[\frac{(n_{K+1} - n_K) + (m_{K+1} - m_K)}{(n_{K+1} - n_K) - (m_{K+1} - m_K)}\right]\left[\cos(ENC\_H_{K+1}) - \cos(\hat{H}_K)\right] \\ ENC\_y_{K+1} &= \hat{y}_K + \frac{b}{2}\left[\frac{(n_{K+1} - n_K) + (m_{K+1} - m_K)}{(n_{K+1} - n_K) - (m_{K+1} - m_K)}\right]\left[\sin(ENC\_H_{K+1}) - \sin(\hat{H}_K)\right] \end{aligned} \tag{11}$$

- Combine all measurements into one vector. Combining Equation Sets 10 and 11, we can write the entire measurement vector:

$$Z_{K+1} = \begin{bmatrix} Z_{1(K+1)} = GPS\_H_{K+1} \\ Z_{2(K+1)} = GPS\_x_{K+1} \\ Z_{3(K+1)} = GPS\_y_{K+1} \\ Z_{4(K+1)} = ENC\_H_{K+1} \\ Z_{5(K+1)} = ENC\_x_{K+1} \\ Z_{6(K+1)} = ENC\_y_{K+1} \end{bmatrix} \tag{12}$$

Step 6. Update the state estimate with the new measurements.

- Update the state estimate. Now we can combine the *a priori* state estimate of Step 2 and actual measurement of Equation 12 by using the Kalman gain of Step 4:

$$\begin{bmatrix} \hat{H}_{K+1} \\ \hat{x}_{K+1} \\ \hat{y}_{K+1} \end{bmatrix} = \begin{bmatrix} \hat{H}_{K+1}^{-} \\ \hat{x}_{K+1}^{-} \\ \hat{y}_{K+1}^{-} \end{bmatrix} + KG_{K+1}\left(Z_{K+1} - C\begin{bmatrix} \hat{H}_{K+1}^{-} \\ \hat{x}_{K+1}^{-} \\ \hat{y}_{K+1}^{-} \end{bmatrix}\right) \tag{13}$$

- Place the *a posteriori* state estimate on the map. Matrix Equation 13 gives the *a posteriori* state estimate at time *K*+1. To compute this estimate, the filter uses all available information at time *K*+1. Therefore, under the assumptions we have made throughout, Equation 16 gives us the best possible estimate of Vasilius' heading and position at t_{K+1}. This data can now be placed on the map.

Step 7. Update the state error covariance.

- Finally, we can use the Kalman gain of Step 4 to compute the *a posteriori* state error covariance from the *a priori* state error covariance of Step 3:

$$P_{K+1} = (I - KG_{K+1} C) P_{K+1}^{-} \tag{14}$$

where I is a 3 by 3 identity matrix. Again, under our assumptions, P_{K+1} represents our confidence in Vasilius' state estimate at time K+1.

Step 8. Prepare for the next 100- millisecond interval.

- Increment K for the next 100ms interval and loop back to Step 2.

5.1.1.2 Integration of Obstacle and Lane Marker Sensors

5.1.1.2.1 Image Processing Using Regional Color Vector Hypotheses

Each pixel in each captured image contains information in the form of three 8-bit binary numbers for red, green, and blue (R_p, G_p, B_p). One way to characterize each pixel graphically is to create a color vector in 3-dimensional space with pure red, pure green, and pure blue as orthogonal axes. Figure 5.1 illustrates such a color. The magnitude of this color vector represents the overall brightness of the pixel, and the direction of the vector represents the relative color in the pixel. Written in vector form with c as the pixel column and r as the pixel row, the pixel color vector is:

$$\vec{P}_{RGB_{c,r}} = R_{P_{c,r}}\vec{i} + G_{P_{c,r}}\vec{j} + B_{P_{c,r}}\vec{k}$$

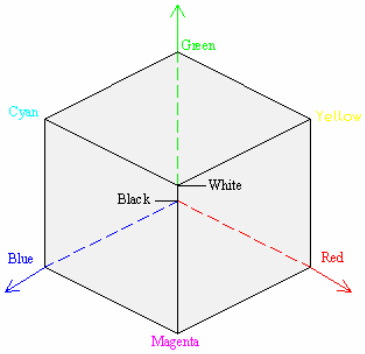

Figure 5.1 Color Cube

where $(\vec{i}, \vec{j}, \vec{k})$ are orthogonal unit vectors.

The algorithm filters spurious noise by using m by m regions of pixels. Averaging the pixel color vectors in each region creates a regional color vector. This process can be shown mathematically by

$$\vec{R}_{RGB_{j,k}} = \frac{1}{m^2}\sum_{j=0}^{j=\frac{w}{m}-1}\sum_{k=0}^{k=\frac{h}{m}-1}\sum_{c=mj}^{c=m(j+1)-1}\sum_{r=mk}^{r=m(k+1)-1}\vec{P}_{RGB_{c,r}}$$

where j is the column number of the region and k is the row number of the region. The regional color vector contains averaged color distributions for the m^2 pixels in the region specified by j and k.

In general the GRV will only have to recognize and navigate through and around certain known objects and surfaces such as sand, bridges, grass, tarps, construction barrels, and path markings. Suppose there are n such surfaces and objects. Each of these surfaces and objects has its own color vector as long as there is uniform distribution of color throughout a region of the surface or object image. The algorithm assumes each region of each image must fall on or close to one of these n hypothesized surfaces or objects. Each has a possibility of being the correct one, so the hypothesized regional color vectors are given by:

$$\vec{H}_{RGB_n} = R_n\vec{i} + G_n\vec{j} + B_n\vec{k}$$

Any number of hypotheses are possible as long as no two vectors are collinear. The amount of separation of the hypothesized color vectors relates directly to the accuracy possible from this algorithm. The algorithm exploits the property that the same pixel in shade or bright sun would have the same general direction in color space even if the magnitudes were very different. Therefore, the important quantities of interest are the angles between the hypothesized vectors and the actual regional color vector. The dot product between these vectors gives this angle:

$$\theta_n = \cos^{-1}\left[\frac{\vec{R}_{RGB_{j,k}} \bullet \vec{H}_{RGB_n}}{\left|\vec{R}_{RGB_{j,k}}\right|\left|\vec{H}_{RGB_n}\right|}\right]$$

The algorithm chooses the hypothesis with the smallest angle θ_n for each region. Since the actual angle is not important, the decision can also be made using only the cosine of the angle:

$$r_n = \frac{\vec{R}_{RGB_{j,k}} \bullet \vec{H}_{RGB_n}}{\left|\vec{R}_{RGB_{j,k}}\right|\left|\vec{H}_{RGB_n}\right|}$$

For each region, the algorithm chooses the hypothesis with r_n closest to one. Our testing shows that this image processing works well as a stand-alone algorithm, but for better reliability we combine this with a traditional sobel edge detection.

5.1.1.2.2 Integrating and Filtering of LMS and Vision Data

Due to the particular hardware design of Vasilius, laser and vision data formats are very different. The LMS mounts in front eight inches above the ground and sweeps horizontally 180 degrees from left to right. The video camera mounts on a post five feet high pointing to the front and pointing down at an angle of 45 degrees. As discussed before, this arrangement of camera and laser system is not so peculiar to just this robot since many autonomous robots use similar sensor placements. This section describes the process of integrating these two sets of data and describes how to place the combined data on a map. The map is a horizontal semicircle five meters in radius and centered at the front of the robot.

Data from the LMS is a 1 by 181 vector of distances where each element represents a distance to an obstacle at an angle equal to the index. At a top speed of 5 mph the robot needs only to look ahead a maximum of five meters (>2.5 seconds travel time). Therefore, LMS data is directly suited for the map described above.

Vision, on the other hand, is not quite so easy. The rectangular picture box on the computer screen translates to a footprint in the shape of an "Aladdin's Lamp" on the ground due to radial distortion and due to the height and angle of the camera. The following method of integration converts the obstacle pixel values of each vision frame to a 1 by 181 vector of distances with each element representing the distance to an obstacle at an angle equal to the index (0 to 180 degrees) of the vector. This format is exactly the same format as the LMS. Converting vision format to LMS format allows the algorithm to place both the LMS and vision data directly onto the semicircle map.

Curve-fitting analyses show that vision pixels translate to the map on parabolic curves. Each pixel translates to the map on a pair of parabolas: a vertical parabola and a horizontal parabola. The horizontal parabolas are nearly straight lines, so with little loss of accuracy the algorithm assumes straight lines for the horizontal parabolas. The following equations relate the pixels to the map:

$$x_{map} = \left(x_{pixel} - \frac{image\ \ width\ \ in\ \ pixels}{2}\right)\left(\frac{line\ \ width\ \ in\ \ meters}{image\ \ width\ \ in\ \ pixels}\right)$$

$$y_{map} = \left(\frac{b-a}{c^2}\right)x_{map}^2 + a$$

$$r_{map} = \sqrt{x_{map}^2 + y_{map}^2}$$

$$\vartheta_{map} = \tan^{-1}\left(\frac{y_{map}}{x_{map}}\right)$$

where (x_{pixel}, y_{pixel}) is the pixel location of the obstacle or marking, and (x_{map}, y_{map}) represents the location of the obstacle on the map. The distances and angles, $(r_{map}, \vartheta_{map})$, form a vector similar to the LMS vector. Curve-fitting analyses give the following equations for the parameters *a*, *b*, and *c* :

$$a = -.3911 + 4.453\, e^{-.008\ y_{pixel}}$$

$$b = -.5385 + 4.9745\, e^{-.00819\ y_{pixel}}$$

$$c = .9494 + 2.9771\, e^{-.01037\ y_{pixel}}$$

Although the numbers are specific to the camera, the lens, and the position of the camera, the general form of these equations work for many different cameras, lenses, and locations.

Test results show that this integration method works very well. The tests include two different types of cameras and two types of lenses. Objects sensed by both camera and LMS map to the same location on the map.

5.1.1.3 Generation of Map Using All Sensor Data

5.1.1.3.1 Conversion to the Earth-Frame

Next we must transform the obstacle data consisting of angle and distance relative to the robot to obstacle data consisting of latitude and longitude relative to the earth frame. We do this transformation in two steps: rotation and translation. We define a robot coordinate frame with the origin at the front center of the robot, with the y-axis pointing toward the front of the robot, and with the x-axis pointing to the right side of the robot. The obstacle coordinates with respect to this coordinate frame are (x_L, y_L). We next rotate the robot frame of reference through the angle $\hat{H}_K$, the estimated heading of Vasilius. Estimated heading comes from the EKF. We use a two-dimensional direction cosine matrix to do the rotation:

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \begin{bmatrix} \cos \hat{H}_K & \sin \hat{H}_K \\ -\sin \hat{H}_K & \cos \hat{H}_K \end{bmatrix} \begin{bmatrix} x_L \\ y_L \end{bmatrix}$$

where (x_R, y_R) are the coordinates of the obstacle in the new rotated frame. This rotated frame has its origin at the front center of the robot, its y-axis pointing to the north, and its x-axis pointing to the east.

The coordinates *x* and *y* are not yet the same as longitude and latitude of the obstacle because the origin of this rotated frame is at the robot. Now we must translate the rotated frame from the robot to the earth-frame origin (we assume a flat earth frame since Vasilius operates in a relatively small region of the earth's surface.) To translate, we add the estimated position of the robot with respect to the earth-frame to the coordinates of the obstacle with respect to the rotated frame:

$$\begin{bmatrix} x_T \\ y_T \end{bmatrix} = \begin{bmatrix} Longitude\ \ of\ \ Obstacle \\ Latitude\ \ of\ \ Obstacle \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} + \begin{bmatrix} \hat{x}_K \\ \hat{y}_K \end{bmatrix}$$

where (x_T, y_T) are the coordinates of the obstacle in the earth frame, and $(\hat{x}_K, \hat{y}_K)$ are the estimated coordinates of the robot computed by the EKF.

5.1.1.3.2 The Dynamic Memory Map

At this point we have both robot position/heading and obstacle positions in the same coordinate frame. We continually store all this information in a dynamic memory map such as the one depicted in Figure 5.2. The input to the path determination algorithm uses the map as its dynamic information source.

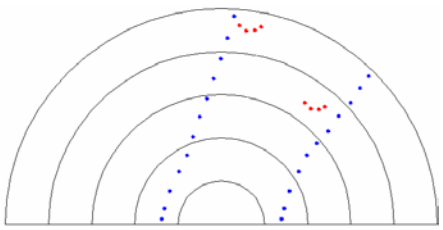

Figure 5.2 Dynamic Memory Map

5.1.2 Path Determination

The map provides a "birds-eye" view of all obstacle locations and marking locations the robot "sees". The next task is to navigate intelligently around obstacles and markings using the map. The idea is to mimic the human decision-

making process when navigating through a field of obstacles and markings. A human driver will aim the car in the general direction toward the middle of the road at some distance in front of the car, with the distance depending on speed; however, if an obstacle or marking is encountered in-between, the driver will slow down and navigate the obstacle or marking. In the same way, the robot will attempt to find an open corridor to a point five meters down the path. If a path wide enough for the robot is not found, then the robot will lower its "aim" distance and again attempt to find a path. If the robot cannot find a path even at close distances, then, just like the human, it will back away and try again somewhere else.

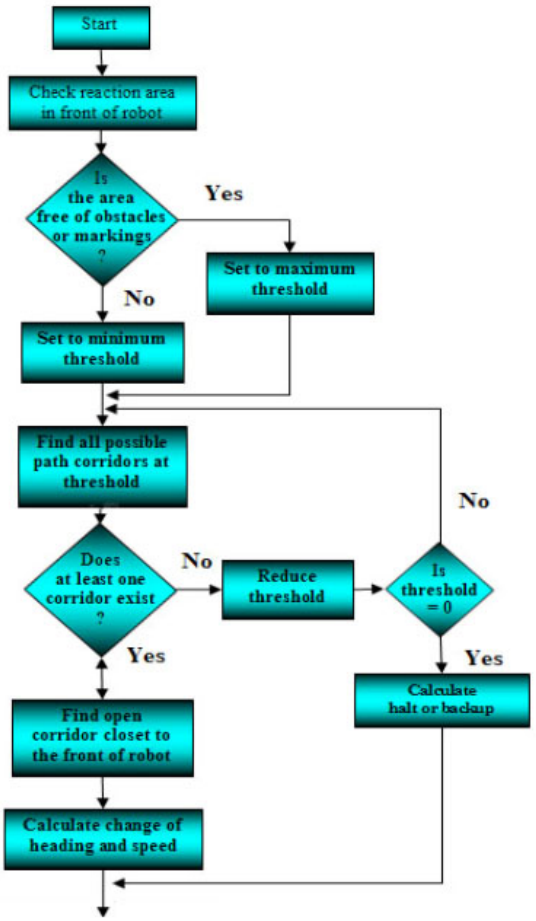

Figure 5.3 Path Determination Flow Chart

Extensive testing of Vasilius proves that this method works well with traps because the robot initiates path planning at 5 meters. The algorithm also handles dashed-lines correctly. If the algorithm detects "lines" over a threshold in length and the lines end abruptly, then the algorithm will extend those "lines" a certain length in either direction.

The flowchart in Figure 5.3 depicts the decision process the robot uses in navigating. The path-planning algorithm finds all possible path corridors that are wide enough for the robot up to the threshold. The algorithm marks each candidate path corridor with a left angle and a right angle. For example, Figure 5.4 shows one candidate path corridor using a threshold of five meters. In this example the corridor is wide enough for the robot all the way to the five-meter threshold and is marked by a left angle and a right angle (a_2 and a_1). If no candidate corridors exist at that threshold, the threshold is dropped by one meter and the search repeated. When the threshold drops to zero then no paths are possible so the robot backs out of the situation and tries the algorithm again.

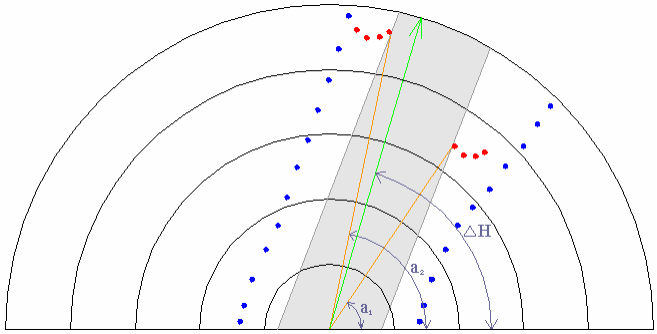

Figure 5.4 Candidate Path Corridor

As the robot approaches the opening it will adjust the heading depending on distance to the opening and angle of the opening with respect to heading. The algorithm uses the following function to calculate change of heading:

$$\Delta H = \left(\frac{a_2 - a_1}{2} \right) * 2^{-(R-1)} + a_1$$

where a_1 is the left angle , a_2 is the right angle, and

$$R = \frac{Dist.\ at\ a_1}{Dist.\ at\ a_2}.$$

Note when the opening is at a sharp angle relative to heading, then ΔH is closer to the most distance point of the opening since R <<1 or R>>1. If the opening is perpendicular to the heading, then $R = 1$, and ΔH will be halfway between the angles a_1 and a_2 (the robot would head toward the center of the gap.) Speed of the robot is greatest at long range but

decreases as the robot approaches the gap. This simple but effective approach gives the robot "good dynamics" in maneuvers and allows smooth transitions during changes in heading.

5.2 Application Algorithm

The following algorithms were designed to allow a user to customize Vasilius navigation for specific applications. They can be used together, independently, or in any combination.

5.2.1 Lane Following

The lane following algorithm is designed for an application that requires marked lanes to be detected and a path that follows the lane to be determined. It supplies inputs to the main navigation algorithm that triggers the proper image processing to find lane markers and sets the path-planning algorithm to follow lanes without getting turned around.

5.2.2 Obstacle Avoidance

The obstacle avoidance algorithm supplies inputs to the navigation algorithm that triggers the proper image processing to find obstacles and correlate them with LMS measurements. This allows the path-planning algorithm to "know" what to avoid when generating a path corridor.

5.2.3 Waypoint Navigation

The waypoint navigation algorithm supplies inputs to the path-planning algorithm that cause the search for a clear path to start in the direction of a target.

6. Analysis of Predicted Performance and Results

Design and testing indicates that Vasilius should perform as indicated in Table 6.1. The table also indicates our results. Each prediction listed in the table comes from analyses of components as well as overall performance**.**

Performance Measure	Performance Prediction	Performance Results
Speed	5 mph	4.9 mph
Ramp Climbing	20-degree incline	20-degree incline
Turn Reaction Time	360 degrees/ second	315 degrees/second
Battery Life	8 hours	6 hours
Stop Reaction Time	Almost Immediate	Almost Immediate
Object Detection	0 to 8 meters	0 to 8 meters
Waypoint Accuracy	2 feet one sigma	2 feet one sigma
Dead-End Detection	0 to 5 meters	0 to 5 meters
Pothole/Lane Detection	0 to 5 meters	0 to 5 meters

Table 6.1 Performance Results

The tool chosen to test and improve the software design was environment simulation software that is specifically designed for Vasilius. The simulation software contains a model of Vasilius' dynamics and sensor characteristics. A randomly generated environment provides a course to test the algorithm. The simulation provides simulated measurements from all sensors as the robot moves through the virtual environment. The simulation exists on a separate computer that requires inputs from the control algorithm and provides fictitious feedback data to Vasilius' algorithm. Random noise that is characteristic of the sensor measurements and control system is injected into the algorithm to make the simulation more realistic. The simulation software also contains a truth model in which a comparison of the actual environment and Vasilius' map can be made. This will allow us to test the algorithm numerous times in a realistic way very quickly and without using the actual vehicle. It will also enable us to cause various problems within the system such as a sensor failure or corrupt measurement. The simulation software is still in development, therefore all test data has not yet been obtained. Our goal is to use the simulation as a tool to expose any problems that exist in the navigation algorithm and to tune the filtering software yielding estimates that are closer to optimal.