9

SciAutonics-Auburn Engineering's Low Cost High Speed ATV for the 2005 DARPA Grand Challenge

Robert Daily[1], William Travis[1], David M. Bevly[1], Kevin Knoedler[2], Reinhold Behringer[2], Hannes Hemetsberger[3], Jürgen Kogler[3], Wilfried Kubinger[3], and Bram Alefs[4]

[1] Auburn University, Auburn, Alabama 36849
[2] SciAutonics, Thousand Oaks, California 91360
[3] ARC Seibersdorf Research GmbH, A-1220 Vienna, Austria
[4] Advanced Computer Vision, A-1220 Vienna, Austria

Summary. This paper presents a summary of SciAutonics-Auburn Engineering's efforts in the 2005 DARPA Grand Challenge. The areas discussed in detail include the team makeup and strategy, vehicle choice, software architecture, vehicle control, navigation, path planning, and obstacle detection. In particular, the advantages and complications involved in fielding a low budget all-terrain vehicle are presented. Emphasis is placed on detailing the methods used for high-speed control, customized navigation, and a novel stereo vision system. The platform chosen required a highly accurate model and a well-tuned navigation system in order to meet the demands of the Grand Challenge. Overall, the vehicle completed three out of four runs at the National Qualification Event and traveled 16 miles in the Grand Challenge before a hardware failure disabled operation. The performance in the events is described, along with a success and failure analysis.

9.1 Introduction

The 2005 DARPA Grand Challenge was a competition to spur the development of autonomous ground vehicle capabilities. It consisted of a 132 mile course that had to be completed in less than 10 h by vehicles with no human intervention. The course was mostly desert terrain including dry lake beds, rough roads, long tunnels and underpasses, and numerous obstacles. Initially, 195 teams entered the Challenge; 43 were invited to the National Qualification Event (NQE). SciAutonics-Auburn Engineering was one of 23 teams chosen from these semifinalists to compete in the final 132 mile course.

SciAutonics formed to compete in the initial DARPA Grand Challenge in 2004. The core technical team was initially comprised mainly of engineers at Rockwell Scientific Corporation (RSC) and received a large portion of its funding from RSC. ATV Corporation donated the vehicle platform and a second test vehicle, and provided continual technical support throughout the project. Auburn University joined the team to develop the vehicle control and navigation

M. Buehler, K. Iagnemma, and S. Singh (Eds.): DARPA'05, STAR 36, pp. 281–309, 2007.
springerlink.com

aspects of the system (Behringer, Gregory et al., 2004). In 2005, the team name changed to SciAutonics- Auburn Engineering, and more collaborators joined to complement the existing expertise. Seibersdorf Research provided a stereo vision system for object detection and road segmentation. The City of Thousand Oaks was also a partner, providing an area of land for performing vehicle tests and the required DARPA site visit.

This paper discusses the SciAutonics-Auburn Engineering effort in the DARPA Grand Challenge 2005. In particular, it covers both the components that comprised the entry vehicle and the strategies that allowed the team to compete successfully in the Challenge. Particular emphasis is placed on the localization, obstacle detection, and vehicle control algorithms.

9.1.1 System Development Strategy

A system that exhibits autonomous driving capability is, by its very nature, quite complex and consists of subsystems with a high degree of interdependence. Since the team was mostly a volunteer effort comprised of people working on this project in their spare time, it was a challenge to map the system structure onto the various team members in a way that was efficient and could allow the team members to work on subtasks independently.

The vehicle team initially set up the hardware: Vehicle components and the lowest level actuation controllers. This could be done relatively independently as other subteams addressed different technical challenges. As the hardware implementation progressed, the work on the control system became more relevant. Two students from Auburn University worked remotely on identifying the vehicle model for throttle and steering control, as well as the navigation algorithm. The sensor team addressed the feasibility of sensor systems and performed independent sensor tests and characterization in desert conditions. ATV Corporation gave the team access to a second vehicle, which had the same driving characteristics but was not equipped with any means for automatic driving. This vehicle served as a platform for mounting and testing sensors while not removing the capability to manually drive. The software team built the framework for communication, sensor data acquisition, and processing. The intelligent behavior subteam addressed the issues related to the design of the autonomous concept, such as path planning and obstacle avoidance.

The team met regularly on evenings and weekends to test, implement, and integrate the different modules in the complete system context. Trials were performed at the Lang Ranch area, where the City of Thousand Oaks had given permission to conduct these tests.

9.1.2 Entry Vehicle

A small All Terrain Vehicle (ATV) platform (Figure 9.1) was chosen for the entry vehicle due to its size, agility, and ruggedness (Behringer, Sundareswaran et al., 2004). The Prowler by ATV Corp. is an ATV modified for military use

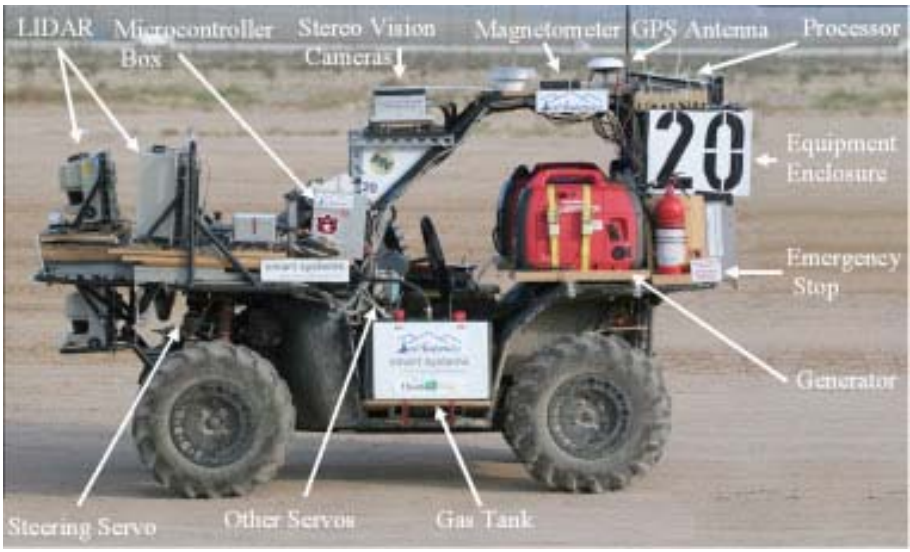

Fig. 9.1. RASCAL

that is equipped with a 660 cc Yamaha engine, enhanced suspension, full roll cage, run-flat tires, and cargo rack. This combination of power, ruggedness, and useable space proved to be an excellent foundation for an off-road autonomous vehicle. The 1,000 lb payload capacity and heavy duty suspension handled the multitude of motors, sensors, and computers that were mounted on the cargo rack and in the roll cage. The independent suspension with 8-9 in. of travel, and high 12.5 in. ground clearance, allowed the vehicle to traverse difficult terrain with relative ease.

Modifications were made to the ATV, dubbed RASCAL (Robust Autonomous Sensor Controlled All terrain Land vehicle), for automation. A servomotor was installed in the engine bay and attached to the steering system to actuate the front wheels. The motor output, 6.5 ft lb of torque, was fed into a 14:1 gearbox, placing a total of 90 ft lb of torque on the steering rack. The throttle, brake, and gear were controlled with smaller servos, which output 27 ft lb. All of the servos were directed by microcontrollers, which communicated with a computer via serial ports. An emergency stop mechanism, with ultimate authority over the microcontrollers, was wired in series to the vehicle's power and the brake and throttle servos to eliminate the possibility of losing control of the vehicle in the event of a software or hardware failure. Two 15 gallon gas tanks were added to the side of the ATV and gave RASCAL more than enough fuel to finish the course. Two 2,000 W generators provided additional power needed to operate the on-board electronics. An enclosure mounted in the rear contained the delicate hardware.

9.2 Software Architecture

9.2.1 Autonomous System Concept

One of the main ideas in the autonomous system was the modularity of the architecture (Behringer et al., 2005). The vehicle control and GPS/INS navigation were the core modules providing the basic autonomous functions. The other modules were "optional add-ons" (Figure 9.2). They provided information about the environment, as well as objects to be avoided. If these modules were disconnected or failed, the core vehicle control still continued to operate using solely GPS and inertial input for computing the control output. Of course, in this mode, the vehicle operated blindly; therefore, the maximum speed was reduced to 2 m/s, a compromise between avoiding heavy damage in a collision and being able to continue driving to fulfill the given mission.

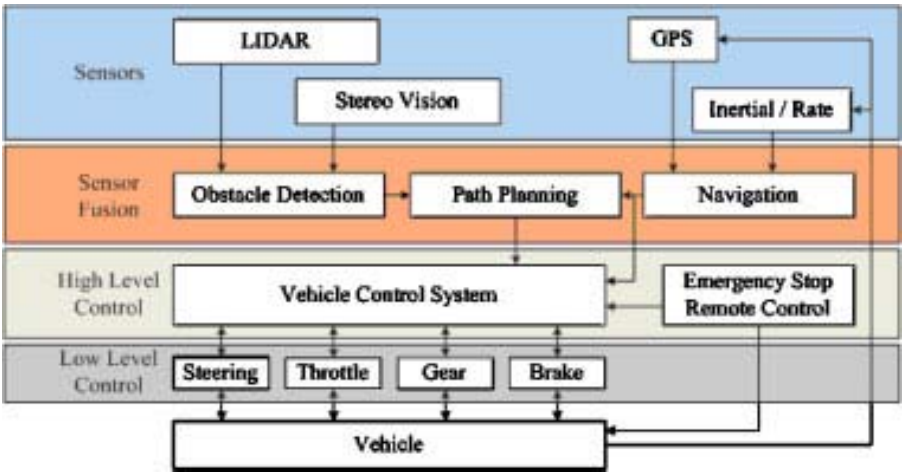

Fig. 9.2. System architecture for autonomous driving on RASCAL

9.2.2 Software Structure

The software was structured to allow easy partitioning among physical central processing units (CPUs) and offline debugging. The overall structure of the software was a collection of modules running as independent processes in the Linux operating system. The modular design allowed fault isolation and parallel development of the modules. A network was set up to allow communication between the modules, via a user datagram protocol (UDP) (Postel, 1980) with timeouts for detecting fault conditions, but no acknowledgements. Running each module as a process provided CPU and memory usage isolation, as well as easy

partitioning among the physical processing units. This was important as the CPU utilization of each module was not known in advance. The easy partitioning also allowed all modules to be run on a single system for debugging, or any one module to be run on a debug system on or off the vehicle, as long as it was on the vehicle network. For the events, two laptops were used to run the vehicle.

Linux was chosen as the operating system for a number of reasons. It provided acceptable real-time behavior; while the default 2.6 kernel is certainly not a hard real-time kernel, it provided consistent cycle times for control loops and good isolation of the processes from one another. Linux could also be run on developer desktops, as well as the actual vehicle, which improved development efficiency. With the use of the Gazebo simulator (Koenig & Howard, 2004) and tools for playing back recorded vehicle data, much of the debugging and development could be carried out on individual laptops; so development work could continue when the vehicle was not available. During vehicle test sessions, results from one run could be analyzed and changes could be made, while a different set of tests were running on the vehicle.

The real-time nature of the vehicle did result in some challenges, and highlighted some of the limitations of Linux for this application. In some cases, code modules would go into tight loops due to bugs. This would cause other modules running on the same CPU to get insufficient processor time. In other cases, too many modules or overly complex algorithms were run, again using too many CPU cycles. The control loops, starved of cycles, would either timeout or in borderline cases not behave as desired. A more interesting case was when one of the CPUs would reduce its frequency based on temperature limits. The behavior would be correct in most cases; but for this application, the system would not have enough CPU cycles to complete all tasks. A hard real-time operating system would limit the impact of these issues to only one process. However, if that process was a key process (as it often was) the vehicle would still be disabled. Debugging these issues required looking at the timestamps of messages passed between the modules.

9.3 Localization

9.3.1 Hardware

RASCAL contained a variety of sensors to determine states critical to the vehicle controller, such as position, heading, and speed. A strategy of redundancy was employed to provide measurements from some sensors when others were not available or in the event of the failure of a particular sensor.

The cornerstone of vehicle localization was a single antenna Navcom differential global positioning system (DGPS) receiver with Starfire corrections broadcast by Navcom. It generated unbiased measurements of position (north and east), velocity (north, east, and up), and course at 5 Hz. It is important to note that

vehicle course is the angle from north created by the vehicle's velocity vector, not the vehicle's pointing vector. With the corrections broadcast by Navcom, this GPS receiver is capable of producing position measurements accurate to less than 10 cm. However, the output rate of the receiver was too low to adequately control the vehicle.

An inertial measurement unit (IMU) was used to obtain high update rate measurements. A Rockwell Collins GIC-100 tactical-grade six degrees of freedom IMU measured translational accelerations and angular rates at 50 Hz. These measurements, however, were inherently corrupted with biases and noise. Dead reckoning with the unit provided acceptable results for a short period of time if the initial biases were eliminated. However, the biases did not remain constant and therefore had to be continually updated and removed from the measurement to provide a reliable navigation solution.

The ATV's onboard speedometer was used as an additional speed sensor. The output rate of this sensor was dependent upon vehicle speed, so compensation was needed to provide a more consistent measurement to the controller. In addition, the measurement was corrupted by wheel slip, which appeared as a sudden large change in the bias. The sensor also contained a calibration error that would corrupt the speed estimate during a GPS outage.

Magnetometers are often used in aerial applications to provide orientation information. They sense the earth's magnetic field, thus all measurements are referenced from magnetic north and not true north. This difference can easily be calibrated and remains fairly constant if the sensor remains in a region near its calibration point. RASCAL utilized two magnetometers to measure the vehicle's heading, roll, and pitch angles. A TCM2 magnetometer provided 16 Hz measurements that contained high noise, but a slow bias drift rate. A Microstrain 3DM-GX1 IMU and magnetometer provided 50 Hz measurements that had very little noise, but the bias drifted quickly (the IMU was not used). It was discovered that the magnetometers could help initialize the navigation algorithm, but once the vehicle started moving they were of little use because the magnitude and drift rate of their biases were greater than that of the other sensors.

9.3.2 Algorithms

Kalman filtering is a proven method for blending measurements to eliminate various sensor deficiencies while utilizing the strengths of each sensor by statistically weighting each measurement. The localization algorithm used was an extended Kalman- Bucy filter (EKF), outlined in detail by Stengal (1994). This algorithm handled the system nonlinearities by continuously propagating the system model to calculate the time update, and discretely propagating the measurement update. The EKF combined the bias-free low update GPS measurements with the other measurements to produce a bias-free high update (50 Hz) navigation solution. An EKF is as accurate and less computationally intensive

Table 9.1. Variable definitions

V – Velocity	$b_{[r,\dot{\phi},\dot{\theta}]}$ – Rate gyro bias (yaw, roll, pitch)
ψ – Heading	$b_{M[\psi,\phi,\theta][1,2]}$ – Magnetometer bias (heading, roll, pitch) (TCM2,Microstrain)
ϕ – Roll	g – Gravity
θ – Pitch and longitudinal accelerometer bias	a_x – Measured longitudinal acceleration
N – North	r – Measured yaw rate
E – East	$\dot{\phi}_{meas}$ – Measured roll rate
θ_g – Road Grade	$\dot{\theta}_{meas}$ – Measured pitch rate

as some higher-order filters if the sample rate is high enough (St. Pierre & Gingras, 2004). This efficiency was an advantage with the algorithm because the 16 element state vector already imposed a moderate computational burden. GPS and inertial measurements were loosely coupled, meaning the inertial errors were corrected with a computed GPS solution. A tightly coupled system, where the GPS pseudo-range measurements amend the IMU errors (Farrell & Barth, 1999), was considered, but due to time constraints and the overall satisfaction with the loosely coupled system, it was not constructed.

The estimated states were chosen to provide the vehicle controller with the necessary position, velocity, and heading data; to fully orient the vehicle; and to calculate sensor biases. The nonlinear differential equations of the states are listed below [Eq. (9.1)], and the variable definitions are described in Table 9.1:

$$\begin{aligned}
&\dot{V} = a_x - g\theta - g\theta_g; && \dot{\phi} = \dot{\phi}_{meas} - b_{\phi}; && \dot{b}_{M\psi 1} = 0;\\
&\dot{\psi} = r - b_r; && \dot{b}_{\dot{\phi}} = 0; && \dot{b}_{M\phi 1} = 0;\\
&\dot{b}_r = 0; && \dot{\theta} = \dot{\theta}_{meas} - b_{\theta}; && \dot{b}_{M\theta 1} = 0;\\
&\dot{N} = V\cos(\psi); && \dot{b}_{\dot{\theta}} = 0; && \dot{b}_{M\psi 2} = 0;\\
&\dot{E} = V\sin(\psi); && \dot{\theta}_g = 0; && \dot{b}_{M\phi 2} = 0;\\
& && && \dot{b}_{M\theta 2} = 0;
\end{aligned} \tag{9.1}$$

These equations were continuously integrated in the time update of the EKF. Noise terms do not appear in Eq. (9.1) because they are included in the time update of the EKF by utilizing the process noise covariance matrix. Methods defined by Bevly (2004) were used to derive these equations of motion. The coordinate frame used is depicted in Figure 9.3.

The bias states have no dynamic response according to the equations of motion, but in actuality, biases randomly drift over time. The process noise covariance

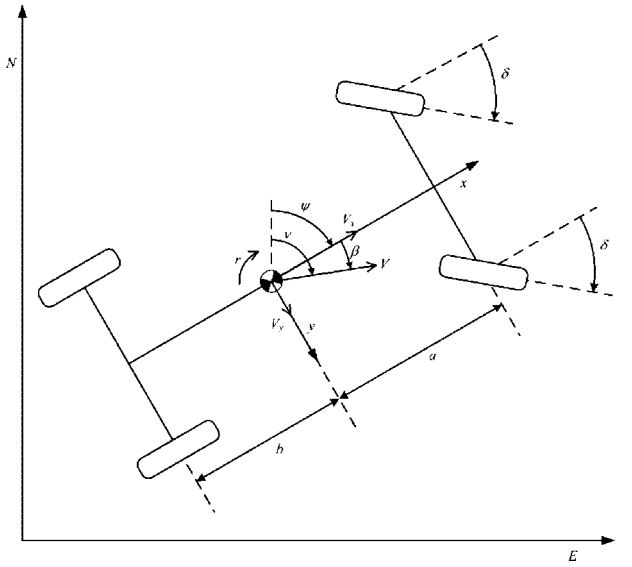

Fig. 9.3. Vehicle body coordinates and reference frame

Table 9.2. Process noise statistics

$\sigma_{az}^2 = 0.65^2 dt$	$\sigma_r^2 = 0.038129^2 dt$	$\sigma_{br}^2 = 10^{-10}$	$\sigma_{\dot{\phi}}^2 = 0.17851^2 dt$
$\sigma_{b\phi}^2 = 10^{-10}$	$\sigma_{\dot{\theta}}^2 = 0.088147^2 dt$	$\sigma_{b\theta}^2 = 10^{-8}$	$\sigma_N^2 = 0.000001^2 dt$
$\sigma_E^2 = 0.000001^2 dt$	$\sigma_{b\theta_g}^2 = 10^{-7}$	$\sigma_{bM\psi 1}^2 = 10^{-4}$	$\sigma_{bM\phi 1}^2 = 10^{-6}$
$\sigma_{bM\theta 1}^2 = 10^{-8}$	$\sigma_{bM\psi 2}^2 = 0.01$	$\sigma_{bM\phi 2}^2 = 0.1$	$\sigma_{bM\theta 2}^2 = 0.1$

matrix included values for the bias states to account for this random bias walk. These were used as tuning parameters for the EKF because they directly influenced the amount of filtering on the estimated states. The other entries in the matrix captured the system noise, which was determined during static tests. The process noise covariance matrix is a diagonal matrix with the covariances in Table 9.2. The time update in the EKF requires this matrix to be continuous; therefore, the measured discretized values are multiplied by the sample rate (dt) as presented by Stengal (1994). This discrete to continuous conversion is only valid for very small sample rates.

The discrete measurement update in the EKF utilized statistically weighted measurements, from the sensors listed in Section 9.3.1, to overcome integration errors. The measurement matrix was adjusted accordingly depending upon the availability of the different measurements. Two calculations were included in addition to the raw sensor measurements. GPS forward and vertical velocities were used to solve for road grade [Eq. (9.2)], and linear equations of vehicle roll

Table 9.3. Measurement noise statistics

$\sigma_V^2 = 0.05^2$	$\sigma_{VWS}^2 = 0.3^2$	$\sigma_\psi^2 = \left(\sigma_V / V\right)^2$	$\sigma_N^2 = 0.1^2$	$\sigma_E^2 = 0.1^2$
$\sigma_{\dot{\theta}_g}^2 = \left(\sigma_V / V\right)^2$	$\sigma_{\theta_{calc}}^2 = 0.08135^2$	$\sigma_{M\psi 1}^2 = 0.19995^2$	$\sigma_{M\phi 1}^2 = 0.024445^2$	$\sigma_{M\theta 1}^2 = 0.036297^2$
	$\sigma_{M\psi 2}^2 = 0.003^2$	$\sigma_{M\phi 2}^2 = 0.003^2$	$\sigma_{M\theta 2}^2 = 0.003^2$	

as a function of lateral acceleration were derived using knowledge of the vehicle's dynamics [Eq. (9.3)]:

$$\theta_g = \tan^{-1}\left(\frac{V_{GPSup}}{V_{GPSx}}\right) \approx \frac{V_{GPSup}}{V_{GPSx}} \tag{9.2}$$

$$\phi = \frac{1}{g}\left(a_y - V\left(r - b_r\right)\right) \tag{9.3}$$

Using these two measurements, the magnetometer roll and pitch biases were observable. It should be noted that the roll estimate contained the lateral accelerometer bias because of the method defined in Eq. (9.3). With the existing sensor suite, there was no measurement available to observe and remove the lateral accelerometer bias.

Noise statistics were found by recording long periods of static data. The covariance values shown in Table 9.3 were loaded into the diagonal measurement noise covariance matrix for use in the measurement update.

9.3.3 Experimental Validation

The algorithm's performance was evaluated based on the amount of error during a simulated GPS outage. Two reasons for this evaluation method are as follows: First, when enough satellites are in view and the receiver is outputting valid messages, the EKF successfully tracks the GPS measurements; and second, a real GPS outage would eliminate the truth measurement, degrading the accuracy of the error analysis. Figure 9.4 is a plot of GPS and estimated position during a test run. The circles signify the beginning and end of the outage, starting before the first turn and concluding at the end of the straight section.

Clearly, error growth occured at the onset of the outage. Figure 9.5 is a plot displaying the magnitude of the error. Over the 25 s outage, the vehicle was traveling 3.2 m/s. The maximum error for this period of time at this speed was slightly over 1.5 m. Since the vehicle was required to remain in a 10 m corridor, this level of error was acceptable.

The IMU and speedometer biases had the most negative impact on the navigation solution. As stated earlier, the magnetometers were not of much use during a run, and were statistically weighted out of the EKF after initialization.

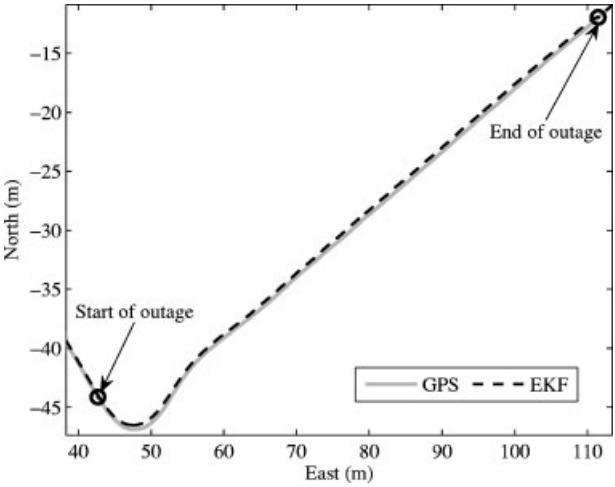

Fig. 9.4. Position estimation during an artificial GPS outage

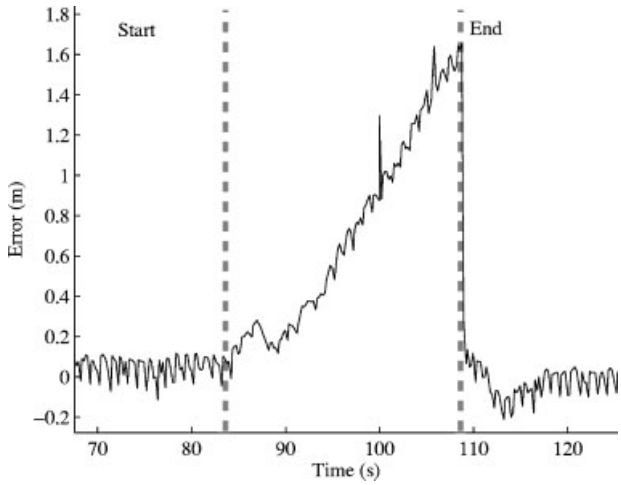

Fig. 9.5. Position error growth during a GPS outage

The tactical-grade IMU contained mechanical rate gyros with a bias drift rate of less than 1° per square-root-h, so the bias error over this period of time is 1 min. The speedometer contained a nonlinear scale factor that could be estimated as a bias. However, when GPS was lost, bias states were held constant because they were nonobservable. If the vehicle slightly changed speeds during an outage, this bias estimate would have been incorrect. This specific algorithm without the

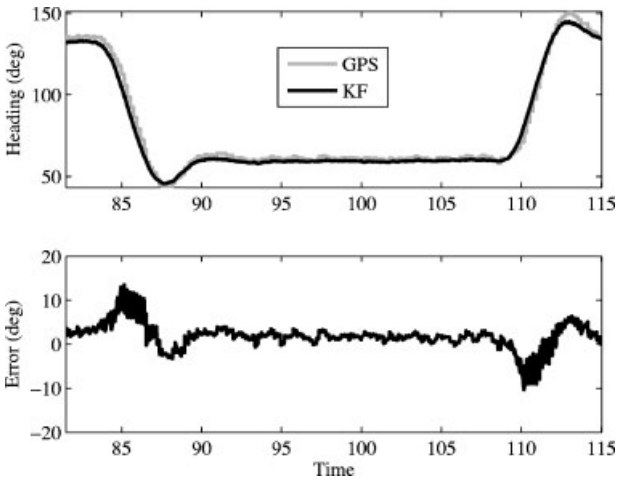

Fig. 9.6. Vehicle heading and discrepancies due to sideslip

estimated bias state was compared with another that included this bias state. It was determined there was no benefit to including the bias state, because the errors in both algorithms grew similarly. The leading error source during this run was due to the incorrect scale factor on the speedometer.

Another source of navigation error is vehicle slip. Vehicle slip can disrupt a navigation algorithm even in the presence of GPS. Longitudinal and lateral slip occur on moving ground vehicle's. Longitudinal slip is generated by a difference in the vehicle's velocity and the wheel's velocity, and lateral slip is created when the vehicle translates laterally. Wheel slip can corrupt the speedometer measurement by causing a sudden jump in the estimated bias (if the modeled bias dynamics have a high enough bandwidth). In addition, it reports a false speed value to the Kalman filter, which can directly inject error into the speed and position estimates. Sideslip also leads to a less accurate estimate, because GPS and integrated IMU measurements differ. The GPS course measurement and an integrated yaw rate gyroscope are typically used to estimate the vehicle's heading. In reality, the sensors are providing two different measurements; the measurements just happen to be similar for the majority of the time on ground vehicles. GPS course (ν) is the direction of the vehicle's velocity vector, while an integrated yaw gyro is the direction of the vehicle's pointing vector (ψ) when roll and pitch angles are absent. Evidence of sideslip-induced error can be seen in Figure 9.5 when the vehicle enters a turn. Figure 9.6 is a plot of the estimated heading when GPS is available. A discrepancy due to sideslip can be seen between estimated heading and GPS course. The error caused by vehicle slip is seen on multiple estimated states and is influenced by the initial tuning of the EKF. This phenomenon is discussed more in depth by Travis (2006).

9.4 Obstacle Detection

One of the key tasks for RASCALwas to remain in the predefined corridor while choosing the fastest and/or easiest path through the corridor. Five sensors were used to search for obstacles within the corridor: The stereo vision system (SVS) from Seibersdorf Research and four SICK LMS-221 light detection and ranging device (LIDAR) units as shown in Figure 9.1.

9.4.1 Stereo Vision

9.4.1.1 Hardware

A high-performance embedded platform was chosen as the processing unit for the vision system. A sealed camera box, hardware platform, and pair of cameras with a 300 mm fixed baseline were the three main components comprising the embedded vision system. Figure 9.7 shows the system mounted on RASCAL during test runs in the desert. Two communication types were necessary for operation. One was the communication with RASCAL, especially with the path planner software module, and the other was communication with the cameras. The communication of the stereo sensor system with RASCAL was carried out using 100 mbits Ethernet; the messages were sent via UDP packets. The cameras were controlled with the DCAM standard 1.31 using the IEEE1394a FireWire bus. The images acquired from the cameras were also transferred using the FireWire bus.

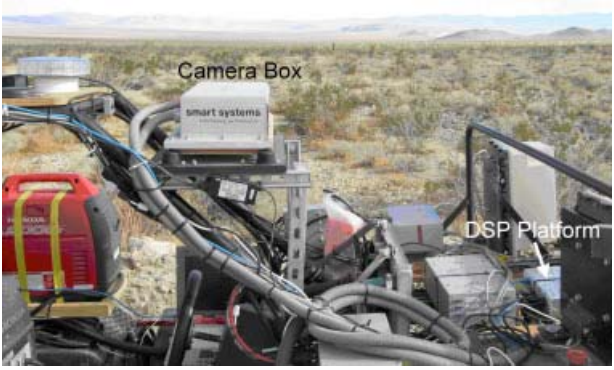

Fig. 9.7. The stereo vision system mounted on RASCAL

9.4.1.2 Algorithms

The obstacle detection algorithm detected relevant objects within the field of view. It used a predictive variant of the V-disparity algorithm (Labayrade, Aubert & Tarel, 2002), which provided a coarse grid for searching the disparity space efficiently based on prior knowledge about vehicle state and road geometry. This met real-time constraints with a processing time of about 20 ms for

two images of 640 X480 pixels. The algorithm consisted of two modules: Pitch determination and obstacle detection. First, the pitch was determined from the vertical movements of far objects. Second, the ground plane was determined using this pitch information. Third, obstacles were detected that were above the ground plane and met constraints of width and height. Hence, this method combined dynamic pitch determination based on image features and the detection of obstacles near the ground plane.

9.4.1.2.1 Pitch Determination

Vehicle pitch was defined with respect to world coordinates. For obstacle detection, the relative pitch with respect to the road surface was relevant, not the absolute pitch. Since the relative pitch depends on the scene in front of the vehicle, it cannot be determined from vehicle dynamics or GPS data only. An algorithm was developed to determine relative pitch from the image sensor by tracking vertical changes of image features for a specified region of interest (ROI).

The relative pitch was defined by the vertical angular difference between the optical axis, i.e., the axes perpendicular to the image plane and the track to be followed on the terrain in front of the vehicle. As the relative pitch was defined with respect to the elevation of the track to be followed, it could be shown that the long-term average relative pitch for reaching a point is zero. In fact, for RASCAL, changes in pitch (absolute or relative) were dominated by angular vibrations of the vehicle itself. These changes included the camera system with respect to the wheels due to a vibration isolation system. Such system vibrations were typically periodic and corresponded primarily to changes in velocity. Figure 9.8 shows an example of the change of the estimated pitch, which was dominated by the vehicle dynamics. The solid line indicates the change

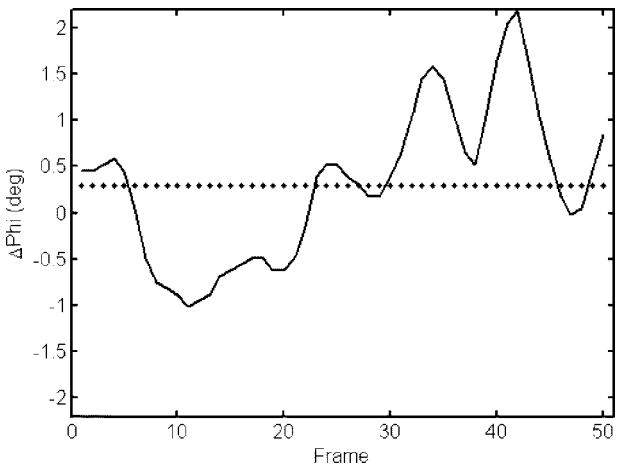

Fig. 9.8. Pitch determination with stereo vision cameras

of pitch during a typical test that included velocity variations; the dashed line indicates the average. As seen in Figure 9.8, pitch varied around a constant equilibrium position, depending only on the camera mount.

Assuming planar terrain geometry and change in pitch due to vehicle vibrations, the relative pitch could be determined by summing changes in the relative pitch and correcting it with its long-term average. The pitch determination module was implemented by choosing a ROI in the center of the field of view, which predominantly consisted of objects near the horizon or the horizon itself. In the case of high obstacles and terrain elevations at about camera height, the change in vertical position due to the vehicle movement in the forward direction was given by the following:

$$\Delta z_{vehicle} = f \frac{\Delta z}{x} \tag{9.4}$$

where $\Delta z_{vehicle}$ was the vertical change of objects in image coordinates and f was the focal length in pixels. z and x were the vehicle coordinates as shown in Figure 9.3. $\Delta z_{vehicle}$ could be calculated as follows:

$$\begin{aligned} \Delta z_{vehicle} &= \left(z_2 - z_1\right) = f \cdot H \left(\frac{1}{x - \Delta x} - \frac{1}{x} \right) \\ &= f \cdot H \frac{\Delta x}{x(x - \Delta x)} \end{aligned} \tag{9.5}$$

where H was the camera height above the ground plane and Δx was the forward change in position of the vehicle; z_1 and z_2 were positions of an obstacle at t=1 and t=2, respectively. For this region, the horizontal edges were determined using a gradient filter, and for three horizontally divided subregions, a vector was determined with the edge density as a function of the vertical image position. For each region, these vectors were matched between two subsequent frames and the total vertical shift was given by

$$\begin{aligned} \Delta z &= \Delta z_{vehicle} + \Delta z_{pitch} \\ \Delta z &\approx \Delta z_{pitch} \end{aligned} \tag{9.6}$$

where Δz_{pitch} was the vertical shift of image features due to the pitch change between two subsequent frames. The relative pitch was determined by integrating Δz over time and each 10 frames correcting with the average value of the previous 30 frames.

9.4.1.2.2 Obstacle Recognition

It was assumed the path trajectory basically followed a drivable track or road. Obstacles that would naturally block the trajectory, such as ravines, rock faces, or landslides, were not to be expected. Expected obstacles included objects, natural or not, which were put there by man. Such obstacles, consisting of boxes,

vehicle tires, traffic cones, trunks, etc., were of limited size and shape, i.e., artificial to their surroundings.

Compactness, together with positioning, described the size and position of the evaluation window. Typical object sizes were 75×75 cm, positioned on the ground plane. Using the epipolar geometry and assuming a minimal detection range of 10 m and a minimal window size of 10×10 pixels in the image, relevant disparities ranged from a 26 down to 4 pixel shift between the left and right images.

The pronouncement of objects was given by the strengths of its edges. For the matching of images with a horizontal baseline, only vertical edges were relevant; the horizontal edge strength was omitted. Edges were determined using a gradient filter, as with the pitch determination. For each time instance, t, and evaluation window, the correlation between the left and right edge image was determined, providing a matrix $C_t(y, d)$ consisting of 114×23 elements, for each horizontal image position (y) and disparity value (d).

The object stability was determined by tracking different modes in $C_{t-2}(y, d)$, $C_{t-1}(y, d)$ and $C_t(y, d)$, considering a constant velocity, and the reciprocal relation between the distance and disparity value. Objects, whose summed correlation

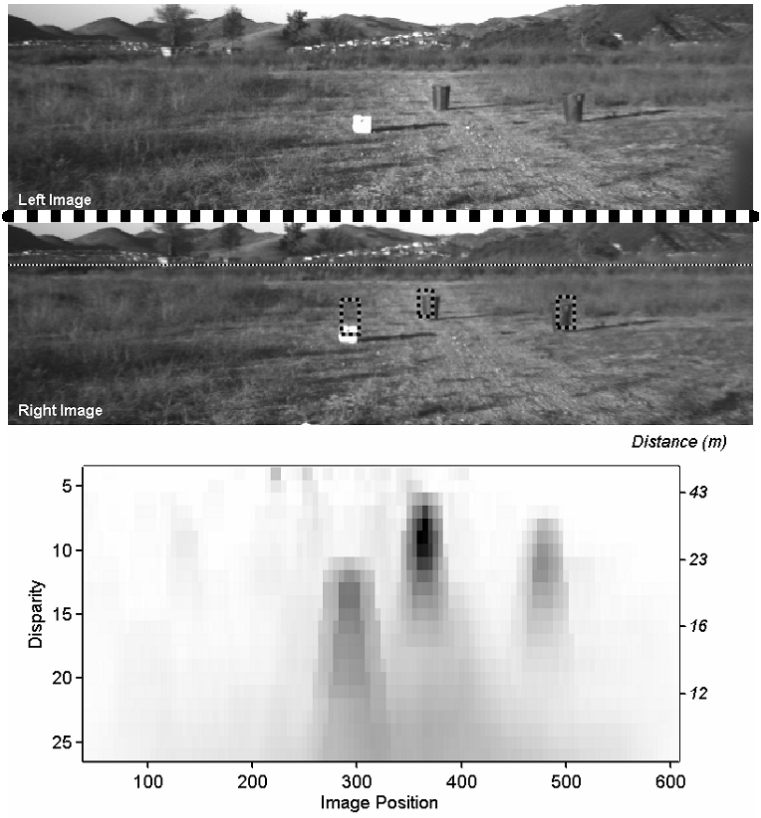

Fig. 9.9. Left and right camera images (top) with correlation magnitude (bottom)

exceeded an adaptive threshold, were included as obstacles. Figure 9.9 shows the left and right images with three obstacles, and the projected horizon from pitch determination (horizontal line). Each of the three obstacles within range was detected, as indicated with dashed rectangles in the right image of Figure 9.9. The lower plot is an example of the correlation magnitude for 2,622 image regions at different disparity values; between 4 (far end) and 26 (near end). A dark color indicates a high correlation result for the specific window.

9.4.2 LIDAR

9.4.2.1 Hardware

A LIDAR sends pulses of infrared light and records the reflection time and intensity to measure the distance and angle to an object. The scans range from 0° to 180° at 75 Hz to produce a two-dimensional image of the environment in polar coordinates. The range of the scan is adjustable in software, and can reach 80 m if desired.

Two of the LIDARs were mounted above the front tires to scan vertically, one was mounted on RASCAL's "hood" so that the scanned line was parallel to the ground (horizontal LIDAR), and one was mounted below the hood so that the scanned line intersected the ground at approximately 5 m in front of the vehicle. Each sensor data set was processed by an algorithm specialized to that sensor/orientation, and the processed data were sent to the path planner.

9.4.2.2 Algorithms

The horizontal LIDAR was processed to filter out grass and weeds while still detecting actual obstacles. This was done based on three principles. First, weeds did not present a consistent surface as an obstacle or berm. Within a LIDAR scan line, the points could be checked if the distances for adjacent points were within a threshold of one another. Second, the LIDAR image of the weeds varied as the vehicle moved. A solid object provided a consistent LIDAR return from scan to scan while weeds varied dramatically as different stalks were hit based on the angle. By comparing multiple consecutive scans, weeds could be differentiated from real obstacles based on the correlation between scans. Third, weeds (at least small ones) tended to be narrow while obstacles (fence posts, trees, telephone polls, cars, berms, etc.) were wide. LIDAR scans that showed very narrow obstacles (< 5cm) were considered weeds. While the techniques were quite effective, it was important to be aware of the limitations. A thin steel pole (0.7 cm in diameter), as used for an electric fence, would be considered a weed. In this application, driving over a steel pole would not harm RASCAL; but in other applications, knocking down fences that may contain livestock would be discouraged.

Figure 9.10 shows an obstacle map on a section of a test track with and without weed filtering. The only real obstacles on the course are the cylinders seen in the photo.The arrow on the graphs indicates the direction of the vehicle; the circles represent the cylinders.

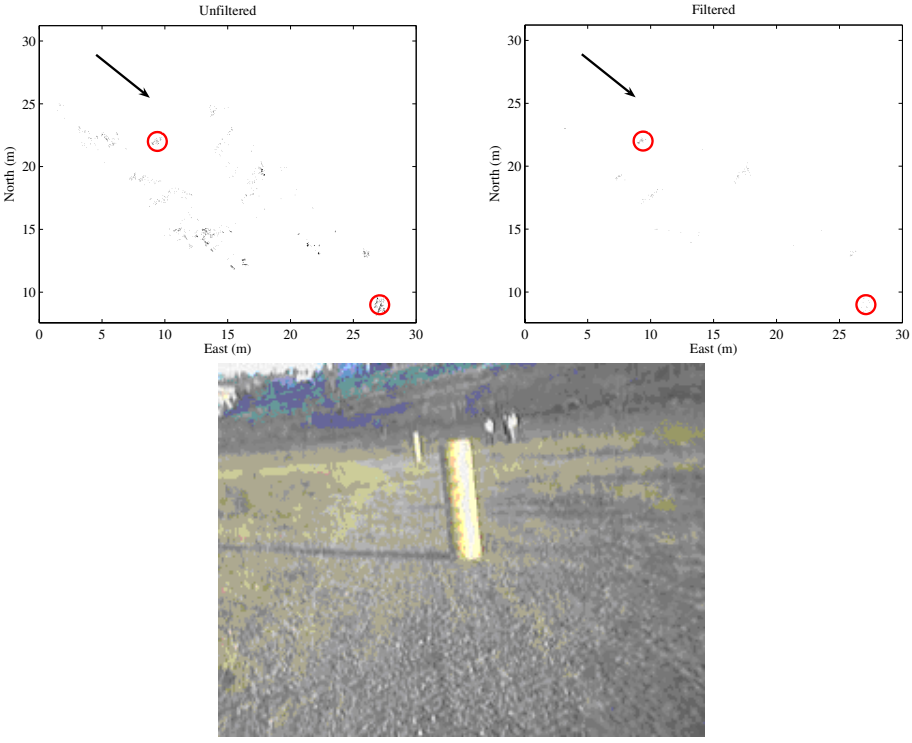

Fig. 9.10. Filtered and unfiltered LIDAR data (top) of obstacles (bottom)

When RASCAL pitched up, the downwardfacing LIDAR was processed as if it were a horizontal LIDAR. Normally, the downward-facing LIDAR saw the ground approximately 5 m in front of RASCAL, and was processed to reject debris thrown up by the front wheels and to detect berms or dropoffs on the edges of the road. The vertical LIDARs were similarly processed to detect obstacles and dropoffs in front of the vehicle.

9.4.3 Experimental Validation

The obstacle detection systems were evaluated in many stand-alone tests; however, the real testing and experimental verification of robust and correctly working algorithms could only be carried out during realistic trials. The NQE (described below) offered the opportunity to verify the obstacle detection in real-world scenarios. Under these conditions, there were several differences between the detection capabilities of the LIDAR and stereo vision systems.

Figure 9.11 shows the tunnel section and the detected obstacles of the NQE. Obstacles that were detected by the stereo vision sensor are represented by light dashes, and the obstacles detected by the LIDAR are shown with dark lines.

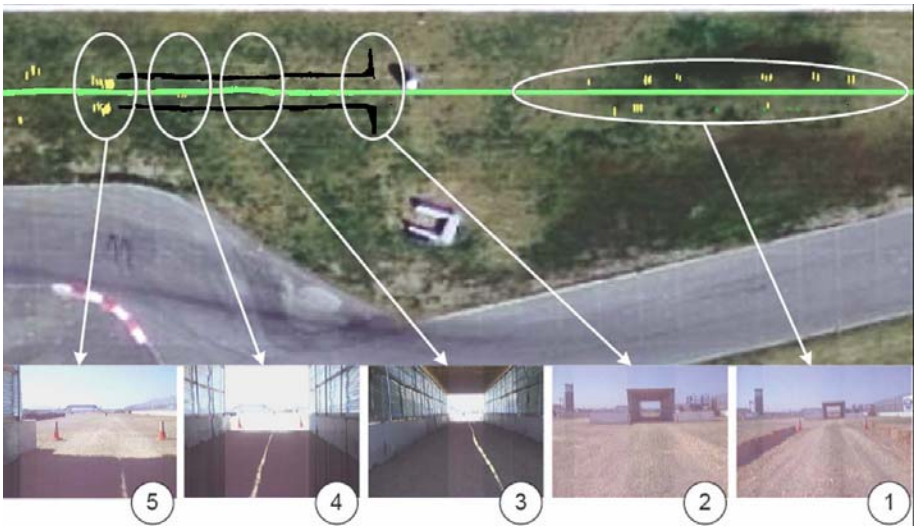

Fig. 9.11. Section with the tunnel from NQE Run 3

Pictures captured by a webcam mounted in front of RASCAL, are also depicted. Picture No. 1 shows the hay bales; these obstacles were only seen by the SVS because they were too low for the LIDAR to perceive. The SVS was able to detect the hay bales if they had any vertically edges in their textures. The left (lower) hay bale border had a long consistent shadow; therefore, detection was not consistent. However, on the right side, more visible transitions between the hay bales existed, and thus more obstacles were detected. Picture No. 2 shows the tunnel entry, which was only seen by the LIDARs. The stereo vision sensor was not able to detect this as an obstacle, because the algorithm was configured to detect obstacles with a width less than 1 m. Picture No. 3 shows the inside of the tunnel, and picture No. 4 displays the exit point of the tunnel. In the tunnel, the stereo vision sensor detected a "ghost obstacle." This happened because of the sunbeams that shone through a crack in the tunnel. In picture No. 5, the two detected cones that were placed on the exit of the tunnel can be seen. The repeated detection of cones (more than one line per cone) was due to the vibrating sensor head. The vibrating differences between the left and the right cameras created differences in distance calculations.

9.5 Path Planning

9.5.1 Initial Path

The initial route definition data file (RDDF) given to the team by DARPA, had points spaced variably from a few meters out to hundreds of meters. The control algorithms needed more consistent information than this offered. Two

approaches were used to generate the nominal path that the vehicle would follow through DARPA's corridor: Speed optimization and map tuning. The speed optimization worked on the principle of maximizing the attainable speed, while staying within the corridors defined by DARPA. To this end, a path made up of a series of lines and arcs was generated. This path took into account the turning capability, acceleration, deceleration, and maximum speed of RASCAL to minimize the time through the course. Each line/arc segment included information about allowable speed, as well as position. The mapping team then adjusted the path based on map data, such as location of roads, cliffs, etc. The map tuning also included modifications of allowable speed to account for such factors as passing through and exiting tunnels, etc. As the vehicle traversed the course, this initial path was modified to take into account previously unknown obstacles.

9.5.2 Path Regeneration

The processed output of the five obstacle detection sensors was used to modify the initial path as RASCAL drove. The processed output of each sensor was a set of obstacles and a weight associated with each obstacle. That weight was an indication of how likely it was that RASCAL would need to avoid the obstacle. For example, in the LIDAR weed filtering, items that passed the filter were given a high weight, while items categorized as weeds were given a very low weight. The intent was to always avoid real obstacles, but not to drive through weeds if a perfectly clear path was also available.

The currently planned path was then compared against the list of obstacles. This list contained a global map of the obstacles currently in sight, as well as past obstacles that had moved out of view. If RASCAL was on a path that would intersect an obstacle, a new path was produced. A number of alternative paths were generated using several different methods. One method generated the sharpest possible turns based on the current velocity and deceleration capability of the vehicle. This allowed the vehicle to avoid near obstacles in the shortest distance possible. Other alternative paths were generated that gradually shifted the current path to the left and right, again taking into account the vehicle velocity.

The alternative paths and the original path were then scored. Paths that would certainly take RASCAL out of bounds were eliminated. Since the position is not known precisely, paths that might have taken RASCAL out of bounds, or nearly out of bounds, were penalized based on an estimated position error, i.e., paths that came closer to the corridor boundary were penalized. Paths that would have taken longer to traverse the same distance were penalized. Paths that went through low probability obstacles were penalized. Paths that intersected high probability obstacles were rejected. All of the paths were then ranked, and the best was chosen. If no path was found that was above a threshold, the current path was held and velocity was reduced to 1 m/s. RASCAL would then continue to look for alternative paths, while traveling at a safe speed. This fall-back state was incorporated to handle false obstacles, so that RASCAL could continue even

with uncertain sensor data. Further work was planned to integrate ultrasonic and contact sensors to discriminate between false and real obstacles, but was not implemented.

9.6 Vehicle Control

The vehicle control module consisted of three parts: Path interface, speed control, and heading control. The path interface took the path segment from the path planner module, and determined where the vehicle should drive based on the vehicle's position, orientation, and speed. The speed and heading control then took this information and determined what throttle and steering inputs to apply to the vehicle.

9.6.1 Path Interface

The path planner module sent a series of waypoints to the vehicle controller. Each waypoint contained a position and desired speed. The vehicle's current position was known from the navigation solution. Using the current position and path segment, a waypoint to drive toward was chosen. The waypoint chosen had to meet two conditions, illustrated in Figure 9.12. First, it had to be in front of the vehicle, and second it had to be at least a given distance from the vehicle. If no waypoints satisfied these two criteria, RASCAL slowly circled to find a valid point. This distance increased with speed and made the steering smoother by effectively adding damping to the heading controller. Humans have a similar response: At low speeds, focus is near the front of the vehicle; as the speed increases, however, it becomes necessary to look further ahead.

Once the point was chosen, a heading error was computed as the angle between the vehicle heading and the line from the vehicle to the chosen waypoint. Speed error was also computed as the difference in the desired and actual speed. These errors were fed to the speed and heading controllers.

This method of path following (waypoint tracking) does have limitations. In particular, because of the necessity to look ahead to upcoming points, it can turn too tightly on corners, particularly tight ones, causing oscillations as the error is reduced. The fact that the look ahead distance is reduced with speed mitigates this problem to a large degree; for tight corners, the car is already moving slowly, thus not looking ahead much, and not cutting the corner. Another problem may arise if the vehicle is significantly off the path. In this situation, the controller tends to drive straight toward the nearest point on the path, instead of smoothly merging onto the path, possibly leading to oscillation about the desired path. However, this method of designating the path allows the steering controller to be much more robust to uncertainties or changing parameters in the vehicle model.

9.6.2 Speed Controller

The speed controller set the throttle position to drive the speed error to zero. Braking was also used to slow the vehicle; however, because the brakes were

either on or off (not progressive), they were only used in more extreme situations. The dynamics, from throttle position to vehicle speed, are generally modeled as a first-order lag described by a DC gain and a time constant. The time constant was identified by performing step inputs into the system; from a stop, the throttle was set to a known position, and the time to reach a percentage of the steady-state speed was recorded. In general, this method should also identify the DC gain of the system, but the drive train of RASCAL contained a continuously variable transmission (CVT) making the DC gain variable. The CVT, along with other engine/vehicle dynamics, make the time constant also somewhat dependent on speed.

Once a rough time constant and DC gain were identified, the controller was designed to obtain a smooth yet responsive reaction to disturbances or changes in desired speed. The controller was a PI type. The integrator was needed to counteract the uncertainty in the DC gain. Special care was taken to ensure the integrator did not wind up and harm the engine or drive train. In particular, a limit was placed on the integrated error, and the error did not accumulate unless the actual vehicle speed was within a bound of the desired speed. This bound was tuned based on the uncertainty in the DC gain.

9.6.3 Heading Controller

The heading controller was more complicated than the speed controller. The complication was due to the dynamics between steer angle (δ) and yaw rate (r); heading (ψ) simply added an integrator to the system (Figure 9.3). The

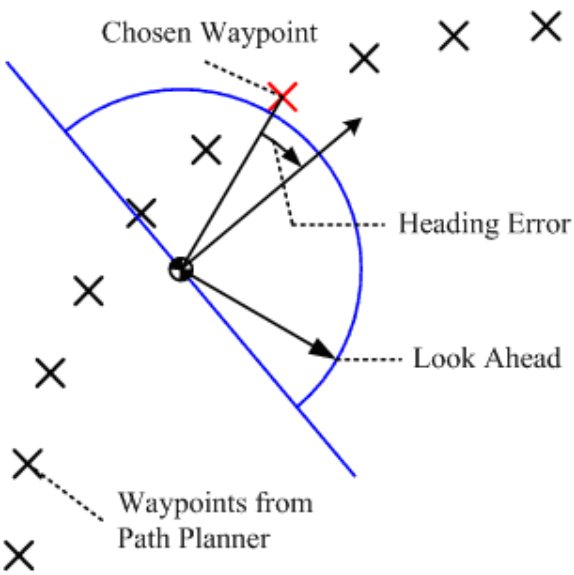

Fig. 9.12. Vehicle control waypoint selection geometry

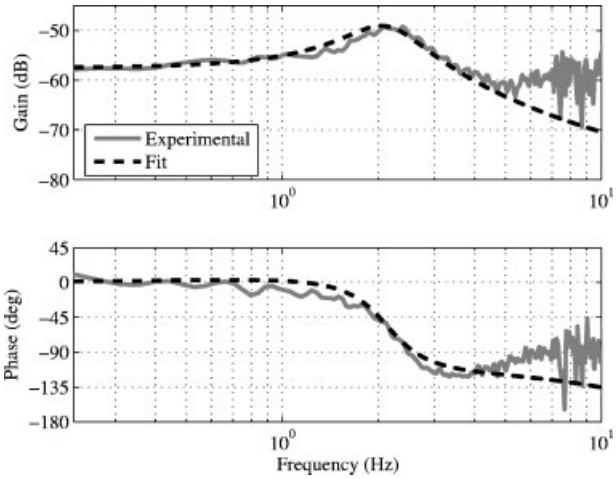

Fig. 9.13. RASCAL open loop steering response

simplest model that captures all of the important lateral vehicle dynamics is a second-order model with one zero as shown below:

$$\frac{r}{\delta} = \frac{k(s+n)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{9.7}$$

The coefficients in the transfer function needed to be identified through dynamic testing. To capture the full range of the vehicle's response, a wide range of frequencies needed to be applied as inputs; a chirp signal was used to meet this requirement. To fit the data, an ARMAX model (Ljung, 1999) produced the least residual errors. One frequency response, along with its fit, is shown in Figure 9.13. The small gain is due to the steer angle input being in counts applied to the servo; 32,000 counts were approximately 30° at the tire. The chirp signal stopped at 7 Hz, meaning the experimental data above this frequency are not valid. A higher-order model could fit the data better; particularly, the higher frequencies. However, in order to keep the controller simple, the fit was left as second order.

Terrain and other varying parameters could also create inaccuracies in the identified model. In general, the parameters that capture vehicle models, tire cornering stiffness, in particular, are assumed to be terrain independent. The tire saturation force is variable, but this does not become a factor, except in extreme cases (Gillespie, 1992).

One characteristic of vehicles is that the lateral dynamic characteristics change with speed, so the chirp input and fit were performed over a range of constant speeds. The identified parameters as functions of speed are shown in Figure 9.14. A curve fit was applied to each of the identified parameters. These curve fits were then used to design the steering controller. It is interesting to note that the trend of these parameters with velocity did not follow that of typical models for ground

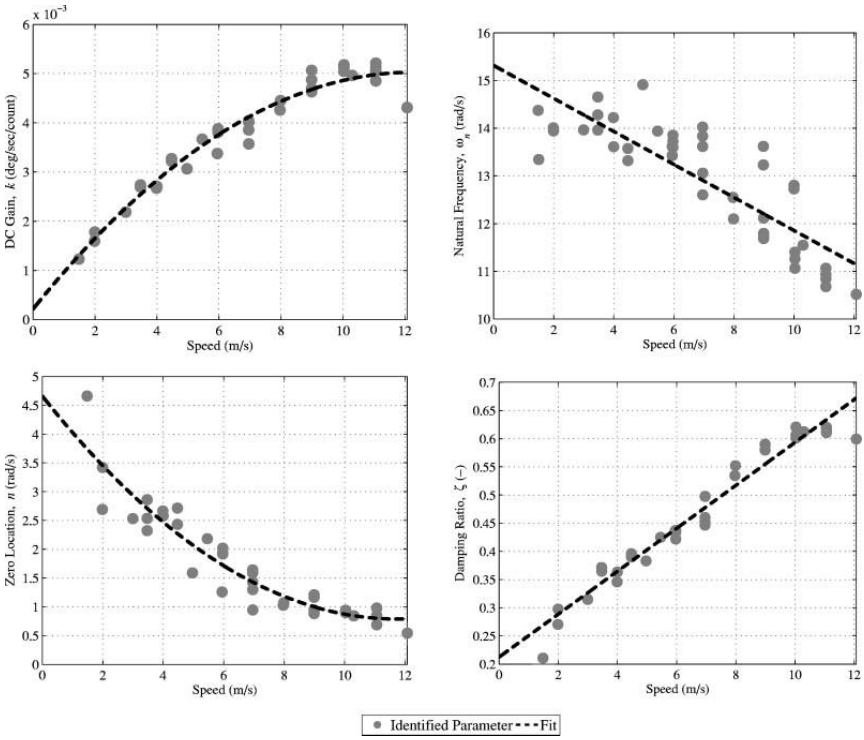

Fig. 9.14. Steering parameters behavior with speed

vehicles, in particular the bicycle model (Gillespie, 1992). The most likely cause was the size of the vehicle. The natural frequency was higher than that of most ground vehicles (2 Hz as opposed to 1 Hz); this meant that other typically ignored factors, such as tire dynamics, were affecting the system response.

A PD controller was used to drive the heading error to zero. As was previously mentioned, the transfer function from steer angle to heading is given by Eq. (9.7) with an additional integrator, making it third order. Two states were fed back; heading error and yaw rate error; therefore, two closed-loop poles could be chosen for the system. Over the identified range, the PD gains were scheduled with velocity, based on the parameter curve fits, to keep these closed-loop poles constant. The third closedloop pole was floating; it was checked to guarantee stability for the entire range of speeds. To account for any model inaccuracies, the controller gains were conservatively designed. This also necessitated the simpler path structure discussed earlier.

9.6.4 Experimental Validation

Figure 9.15 shows the response of the vehicle and throttle controller to steps in the reference speed. In general, the controller performed well; the error is

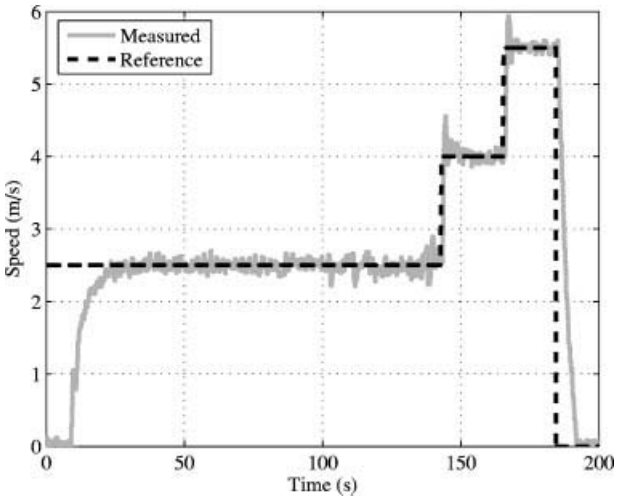

Fig. 9.15. Controlled speed response

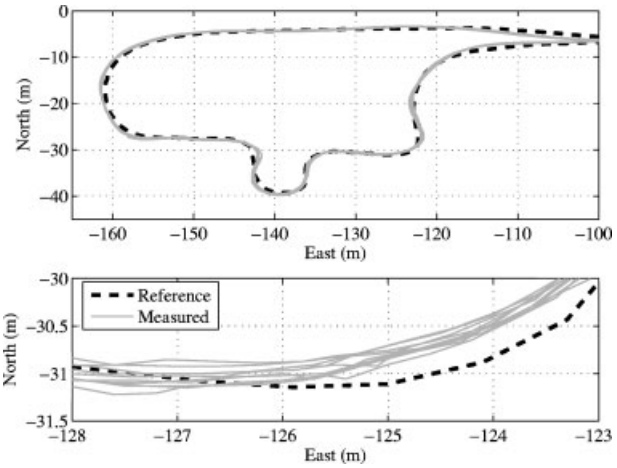

Fig. 9.16. Waypoint tracking response

typically around 0.1 m/s. It is interesting to note that the initial step (to 2.5 m/s) is overdamped, while the other two (to 4 and 5.5 m/s) are underdamped. The difference is due to the unmodeled dynamics moving the effective closed-loop pole and the fact that the integrator is switching on and off based on the magnitude of the speed error. The controller was tested at much higher speeds (up to 18 m/s) and still performed with no noticeable degradation.

Figure 9.16 shows a portion of the path created for the application video required by DARPA, along with laps run by RASCAL around this path. There are some areas where RASCAL does not exactly follow the desired path; after one particularly tight high-speed corner, the error was 1 m. This error was mostly due to look ahead in the steering controller as discussed above. It could have been reduced with a higher fidelity controller/vehicle model at the expense of robustness; however, with the parameters defined by DARPA (i.e., typical corridor width), the error was deemed acceptable. The pass-to-pass repeatability was quite accurate; typically around 20 cm. RASCAL's response to a given situation was very predictable.

While the vehicle was at Auburn University, the controllers were also stress tested for over 100 miles at high speeds; between 25 and 40 mph. For a vehicle the size of an ATV, these speeds were significant, as they corresponded to a sports utility vehicle traveling at highway speeds. These tests were conducted without the obstacle detection algorithms in order to concentrate on the behavior of the navigation and control modules alone. The error grew as the speed increased, reaching 2 m at 40 mph; some of this was due to an incorrect calibration in the steering servo. However, at these speeds, RASCAL would have been on a wide road during the Grand Challenge, and that amount of error was acceptable.

9.7 Results

9.7.1 The National Qualification Event

The NQE was the final measure for entry in the Grand Challenge. Of the 195 initial entrants, 43 teams successfully completed a preselection qualification and

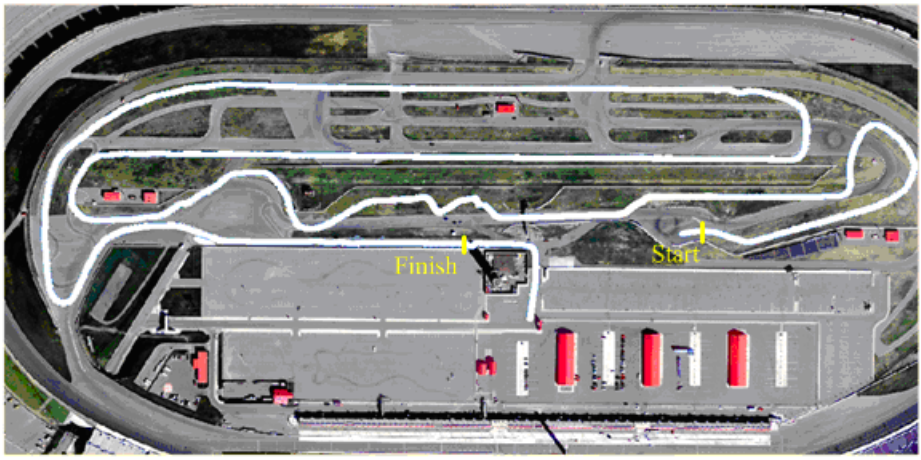

Fig. 9.17. Aerial picture of an NQE obstacle course

Table 9.4. RASCAL NQE results

	NQE1	NQE2	NQE3	NQE4
Time	Did not finish	16min	14min	14min
Passed Gates	22/50	46/50	48/50	48/50
Passed Obstacles	1/4	4/4	4/5	5/5

participated at the NQE. After the NQE, the best 23 teams got the opportunity to start the Grand Challenge. The main goal of the NQE was to complete four different tracks, each approximately 2.5 miles long. Figure 9.17 shows an aerial picture of an NQE track. In these tracks, DARPA officials composed courses with different sets of obstacles that could be found in the desert. Each track consisted of a small hill, a high-speed section, a tunnel where the GPS signal was blocked, gates, and several manmade obstacles. The four tracks differed in types and location of obstacles. A run consisted of 50 gates, a tunnel section, and four or five obstacles. As presented in Table 9.4, RASCAL completed three of the four runs in the NQE. This success rate allowed RASCAL to be one of ten teams selected for early qualification.

The initial run ended soon after the tunnel; Figure 9.18 shows the cause of the failure. The light thin line indicates the estimated position from the localization algorithm, and the light dots represent the recorded GPS position. As expected, RASCAL lost GPS immediately upon entering the tunnel and began to dead reckon to estimate position. The dead reckoning worked exceptionally well for 2 min, during which time RASCAL avoided several obstacles. GPS measurements were reacquired as RASCAL was initiating its obstacle avoidance procedure to avoid a parked car. The GPS receiver incorrectly reported its measurements valid to within 10 cm. Instead, the position measurement was off by 10 m to the north and east, and the velocities were reported as pure zeros causing the localization algorithm to crash. The addition of a few simple lines of code ignored these false messages, at the expense of having to dead reckon for approximately 4 min after a GPS outage. Once this fix was applied, the remaining NQE runs were successfully completed.

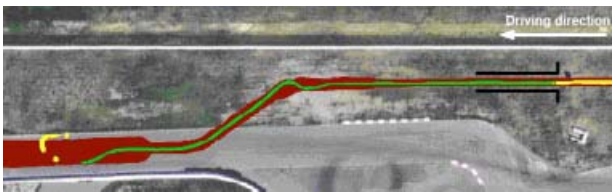

Fig. 9.18. Dead reckoning through NQE Run 1

9.7.2 DARPA Grand Challenge

RASCAL was the tenth vehicle to start the 2005 Grand Challenge. It efficiently negotiated the first segments of the course, passing two competitor vehicles within the first few miles. The initial portion of the course was smoothly traversed with no difficulty. However, problems developed and officials had to stop RASCAL because of severe performance degradation 16 miles into the course. For the duration of the run, the vehicle traveled an average of 4.6 m/s, and reached maximum speeds of 11.5 m/s. Although the vehicle was capable of operating at higher speeds, the obstacle detection system could not process the data reliably beyond 11 m/s. Figure 9.19 displays the whole course, along with a closeup of the portion RASCAL completed.

After an analysis of the recorded data, the cause of failure is fairly certain. Shortly before RASCAL went off road, it lost all LIDAR data. Soon thereafter, it lost all vehicle state data. The LIDAR and the internal sensors were connected via USB hubs to the processing computers. Speculation is that one of the following faults occurred and ended RASCAL's day: USB hubs lost power terminating the connection between the computer and sensors, or the USB hubs overheated and ceased to function.

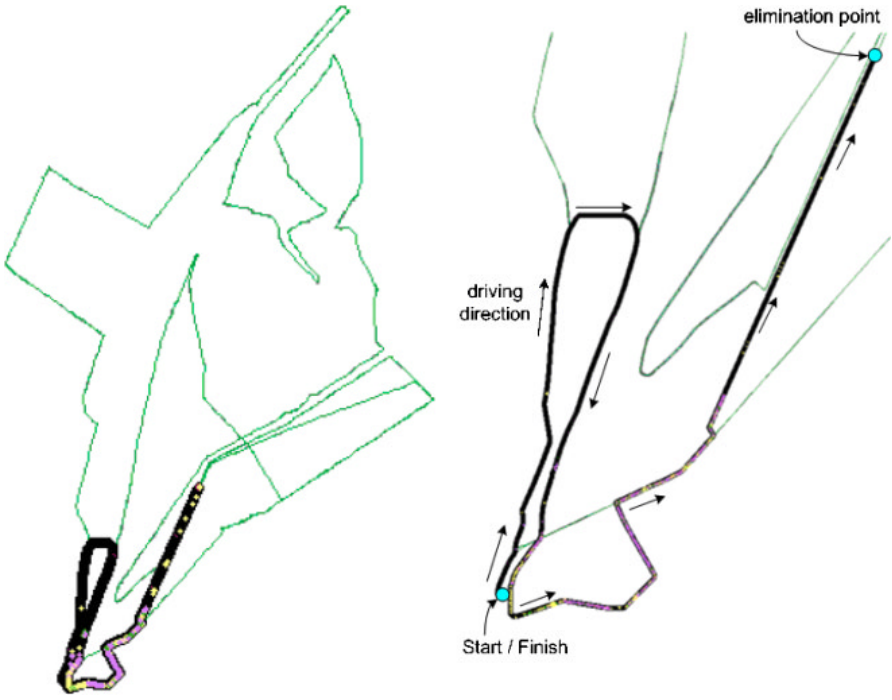

Fig. 9.19. Grand Challenge race course

9.8 Conclusion

SciAutonics-Auburn Engineering, along with Seibersdorf Research, managed to build a strong contender for the DARPA Grand Challenge 2005. A team of volunteer engineers with limited resources managed to stay competitive among teams with more time, money, and resources. Out of 195 initial entrants, 23 teams started at the DARPA Grand Challenge, and five teams finished the race. The 16 mile effort by RASCAL was the 16th longest run among the remaining entrants. The vehicle successfully demonstrated the cohesive hardware and software integration at the NQE, but minor events harmed the vehicle's endurance and ended its run in the 2005 Grand Challenge.

The key to the Grand Challenge was not necessarily the incredibly accurate sensing technology or immense amounts of computing power. RASCAL was able to detect and avoid the same obstacles as teams with twice the number of sensors and considerably more than twice the computing power. With intelligent yet efficient algorithms and a few key sensors, these hurdles could be overcome. The failure that finally disabled RASCAL was a simple hardware connection malfunction. More time needed to be spent by the team to harden the vehicle. RASCAL's concept was validated by its showing in the Grand Challenge; with more time, the realization of its potential would also have been reached.

Acknowledgments

The authors acknowledge the financial contributions to the SciAutonics, LLC team, from Rockwell Scientific (RSC), issued by CEO Derek Cheung. They also acknowledge the provision of SICK LIDAR sensors by Rockwell Collins, thanks to the support by Patrick Hwang and his group. They further acknowledge the strong support of their team by ATV Corporation through Amos Deakon. Thanks are extended to the stockholders for the monetary support and enthusiasm they provided. The Auburn University College of Engineering and Mechanical Engineering Department must also be thanked for their support of this project. In addition, without the help of all the volunteers, this project and the achievements would not have been possible. Special thanks go to the team members who could not contribute to this specific paper, but without whom RASCAL would not have driven, particularly: Team leader— John Porter, Wayne Guthmiller, Dick Elsley, Bob Addison, Gary Bostrup, Monte Khoshnevisan, Don Harris, Ron Dilley, James Whiteley, and Alon Yaari.

References

Behringer, R., Gregory, B., Sundareswaran, V., Addison, B., Elsley, R., Guthmiller, W., deMarchi, J., Daily, R., Bevly, D., Reinhart C. (2004, July). Development of an Autonomous Road Vehicle for the DARPA Grand Challenge. IFAC Symposium on Intelligent Autonomous Vehicles 2004. Lisbon, Portugal.

Behringer, R., Sundareswaran, V., Gregory, B., Elsley, R., Addison, B., Guthmiller, W., Daily, R., Bevly D. (2004, June). The DARPA Grand Challenge – Development of an Autonomous Vehicle. IEEE Symposium on Intelligent Vehicles 2004. Parma, Italy.

Behringer R., Travis, W., Daily, R., Bevly, D., Kubinger, W., Herzner, W., Fehlberg, V. (2005, September). RASCAL – An Autonomous Ground Vehicle for Desert Driving in the DARPA Grand Challenge 2005. IEEE ITSC 2005. Vienna, Austria.

Bevly, D. (2004). Global Positioning System (GPS): A Low-Cost Velocity Sensor for Correcting Inertial Sensor Errors on Ground Vehicles. Journal of System Dynamics, Measurement, and Control, Vol. 126. pp 255–264.

Farrell, J., Barth, M. (1999). The Global Positioning System and Inertial Navigation. New York, NY: McGraw-Hill Companies, Inc.

Gillespie, T. (1992). Fundamentals of Vehicle Dynamics. Warrendale, PA: Society of Automotive Engineers.

Koenig, N., Howard, A. (2004, September). Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. IEEE International Conference on Intelligent Robots and Systems. Sendai, Japan.

Labayrade, R., Aubert, D., Tarel, J. (2002, June). Real Time Obstacle Detection in Stereo Vision on Non-Flat Road Geometry Through "V-Disparity" Representation. IEEE International Vehicle Symposium. Versailles, France.

Ljung, L. (1999). System Identification: Theory for the User, 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR.

Postel, J. (1980). RFC 768 – User Datagram Protocol. St. Pierre, M., Gingras, D. (2004, June). Comparison Between the Unscented Kalman Filter and the Extended Kalman Filter for the Position Estimation Module of an Integrated Navigation Information System. IEEE Symposium on Intelligent Vehicles 2004. Parma, Italy.

Stengal, R. (1994). Optimal Control and Estimation. Mineola, NY: Dover Publications.

Travis, W. (2006). Methods for Minimizing Navigation Errors Induced by Ground Vehicle Dynamics. M.S. thesis, Auburn University, Auburn, Alabama.