Low-Cost Autonomous Vehicles Using Just GPS

Michael E. Holden
San Francisco State University

Introduction

The Navigator is an autonomous ground vehicle. Using a commercial handheld GPS as its only sensor, it is able to follow a preprogrammed path and speed with remarkable accuracy. Assembled from a model car chassis, inexpensive 8-bit microcontroller and GPS, it is about as simple as autonomous ground vehicles can be. This simplicity makes it easy for students unfamiliar with autonomous vehicles to understand the system, keeps the system cost low, and allows for a truly miniature guidance and navigation electronics package. The simplicity also provides engineering challenges, such as navigating using only the infrequent sensor data from the GPS, and working within the limits of GPS accuracy and discretization.

Originally designed for a middle-school after-school program for disadvantaged students[1] (as a consulting project for the MLB Company[2]), the Navigator has been used for undergraduate and graduate education as well. It has an LCD display and 4 button interface that make its operation easy even for the non-engineer, while engineering students can use the interface to adjust feedback gains without reprogramming the controller. For graduate students starting autonomous vehicle research projects, the Navigator is a good starting point from which to build more complex vehicles.

The Navigator is primarily an educational tool—although it is a capable platform, its lack of external sensors makes it useful only in areas free of obstacles. The electronics are extendable to other platforms, such as boats (which are less likely to be obstructed) and robotic bases with external sensors for collision avoidance. There are many remote sensing applications that can make use of an inexpensive autonomous ground or water vehicle, and the author is pursuing collaborations that will use the Navigator as a research tool instead of merely as a research subject.

This paper will describe the Navigator, its control algorithms and electronic assembly, as well as the educational uses it has been a part of to date. Data from using the Navigator electronics as a boat controller will also be shown. Figure 1 shows an example of an operational Navigator.

Vehicle Description

The Navigator's purpose is to drive itself along a pre-programmed path at a given speed. The path is made up of straight-line legs between waypoints, and the car follows the path at a speed specified for each leg. Figure 2 shows a sample path. The waypoints define the path, and the speed associated with each waypoint defines the speed the Navigator will hold while heading for that waypoint.

Figure 1: The Navigator
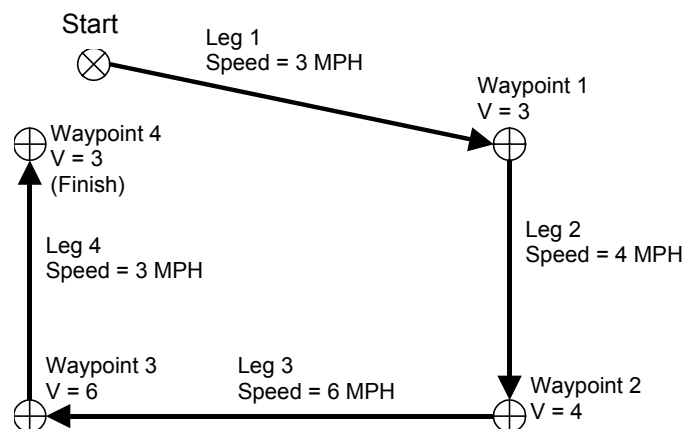
Figure 2: Sample Autonomous Path

The goal of the project was to create an autonomous car using a commercially available GPS receiver as its sole sensor. The single sensor arrangement has two main advantages: low cost by eliminating unneeded sensors, and ability for miniaturization by eliminating additional sensors and signal conditioning circuitry from the controller board. Miniaturization is an important goal for groups like the armed forces, whose desire for a backpackable unmanned vehicles lead to the DARPA micro-air-vehicle program, which created aircraft no larger than 6 inches in any dimension and required extensive miniaturization work[3].

Thinking along the small/cheap lines also led to the decision to eliminate telemetry equipment from the system, as transceivers with range that matches the vehicle's tend to be quite expensive (on the order of the cost of the rest of the vehicle)[4]. Also, smaller vehicles do not require lower power telemetry equipment, because power consumption for a given telemetry technology does not scale with the size of the vehicle, unlike actuators and other systems. The fixed telemetry

system size poses an impediment to miniaturization. Without telemetry equipment, the vehicle must have its route pre-programmed, and there is no way to modify or interrupt its mission once in motion except by physically pushing a button on the vehicle, but this was an acceptable trade in this simple vehicle project.

Figure 3: Autonomous Boat Underway

The single-sensor constraint leads to challenges in the design, both because of the accuracy of the sensor used and the lack of other information to augment the GPS. The handheld GPS used for the project, a Garmin Etrex Legend, is not a perfect sensor—it lacks both accuracy and bandwidth as an autonomous vehicle sensor. The accuracy of the GPS, according to its display, is usually in the 15-30 foot range (depending on satellite coverage). The update rate is approximately 0.5 Hz, allowing a vehicle traveling 5 MPH to cover almost 15 feet between position measurements. The GPS rounds its position data to approximately the nearest 7 feet, giving a large discretization error. In order to navigate to the precision of a neighborhood street, the GPS information must be augmented with estimation techniques, as described below.

A model car is the easiest platform to start with, because it can be acquired easily and operated nearly anywhere. The Navigator autopilot is not limited to model cars, however, and has been successfully installed and used in a model boat (Figure 3). Actually, any stable platform may be used, including snow mobiles, hovercraft, or even aircraft. The platform must either be open-loop stable or with very slowly diverging unstable modes[5] in order to be successfully controlled with the low-bandwidth sensing used in the system. Data showing the boat's performance will be shown along with that from the car, to indicate the multi-disciplinary nature of the Navigator controller.

Algorithm description

The autopilot algorithm can be broken into three major subsystems: the state estimation routines, the navigation routines, and the control loops.

State Estimation

The GPS sensor gives the vehicle's position (latitude/longitude), speed (MPH) and compass heading (degrees true) approximately every 2 seconds. This data is combined with a simple vehicle model to provide continuous position and heading estimation between GPS fixes.

The GPS speed is used directly for speed control without estimation—this approach worked better than speed estimates that incorporated the vehicle's throttle, since most speed changes are caused by external phenomena such as hills.

Figure 4 shows a block diagram of the position estimator. It first combines the speed and the heading estimate (described below) to form Easterly and Northerly velocity components. The velocity is integrated, using the GPS position as an initial condition, to give the current position estimate. Whenever a new GPS position and speed is available, the position estimate is updated to use this information and the integration begins again from the new GPS position.
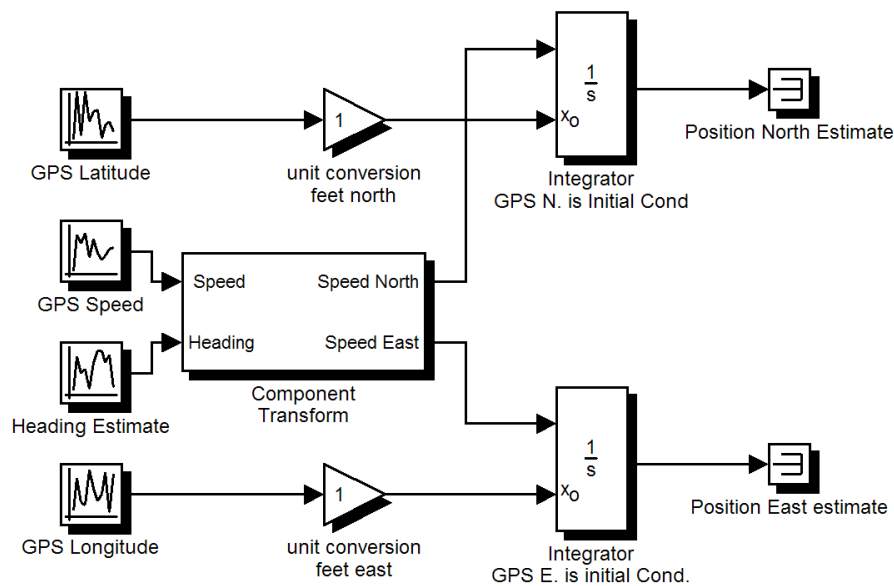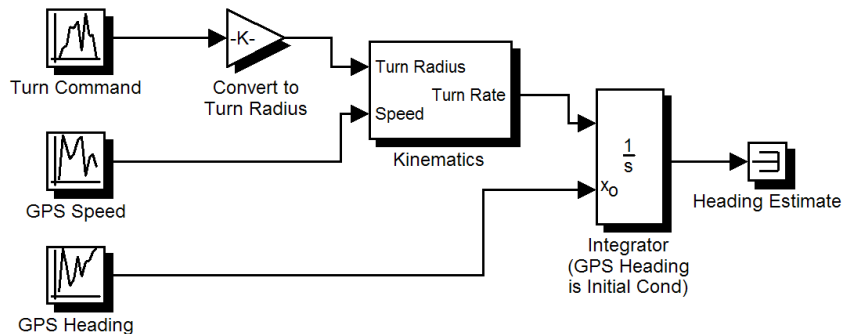

Figure 4: Position Estimator

Figure 5 shows the heading estimator block diagram. This estimator is critical to the operation of the vehicle, because when using GPS heading alone only the very lowest feedback gains could be used for a stable controller. Like the position estimator, the heading estimator uses GPS data as it is available and integrates to create an estimate of heading until the next GPS update. The heading estimator integrates a turn rate estimate, with the GPS heading as the initial condition.

The turn rate estimate is created from the control loop's turn command (which can be converted into a turn radius using the vehicle geometry), and the vehicle speed.

In both the position and heading estimation algorithms, no blending of the old estimates with the new GPS data is done (as a Kalman filter[6] would do), for three reasons: as a teaching tool for undergraduates this simple estimator is easy to understand without prior filtering experience; also it is important to keep the mathematics simple enough to program on a single-chip microcontroller and retain a reasonable control-cycle time; and some filtering techniques such as complementary filters[7] add lag to the GPS data which was found to hurt performance more than the smoothing helped.

Figure 5: Heading Estimator

Navigation

The navigation algorithm is the outer control loop. It uses the position estimate and the waypoints that define the route to determine which leg of the route the vehicle is on, and to calculate a heading command. The Navigator moves to the next waypoint in the route once it is within 10 feet of the current waypoint; when it reaches the last waypoint the vehicle stops. The navigation algorithm keeps the vehicle on the line between waypoints, rather than heading directly at the waypoint from the vehicle's current position. Figure 6 illustrates the navigation parameters and geometry, and Figure 7 shows the navigation control law. The navigation loop calculates the cross-track error, which is the vehicle's perpendicular distance from the course line. The heading command differs from the course leg's heading by an amount proportional to the cross-track error, up to a maximum of 90 degrees. For example, when the cross-track error is zero, the heading command is the course-leg heading, and as the cross-track error increases, the heading command changes to point more and more towards the course line, until it is perpendicular to the course line for very large cross-track error.
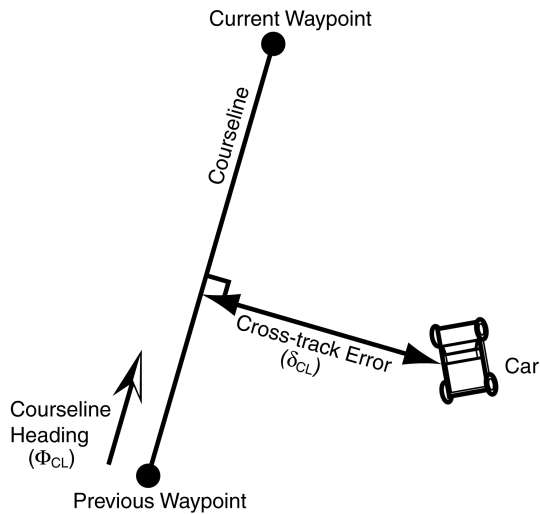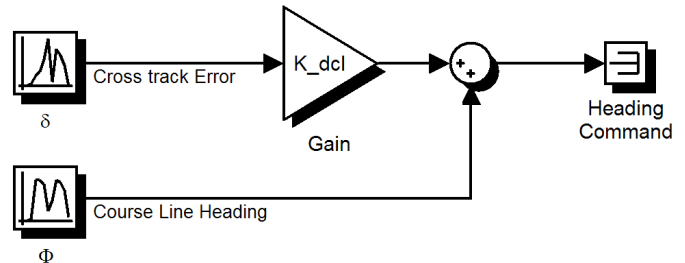
Figure 6: Navigation Parameters
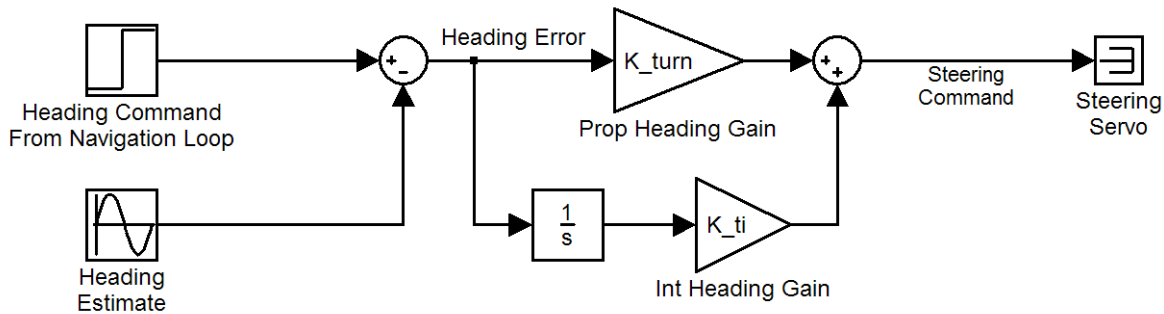

Figure 7: Navigation Control

Figure 8: Heading Control Block Diagram

Control Laws

The heading control loop takes the heading command from the navigation loop and the estimated compass heading from the state estimator, and forms a steering command based on simple proportional-integral feedback[8]. Figure 8 shows a block diagram of the heading control.

The speed control loop is used to maintain the programmed speed on each leg. It is important to regulate speed carefully, since the state estimator uses the GPS speed both in the position and heading updates, and if the car's speed is very different from the GPS readings the navigational errors will increase. Figure 9 shows the block diagram for the speed control proportional-integral feedback loop. A feed-forward gain[9] is used to set the throttle to an approximate equilibrium position based on the commanded speed.

The feedback gains were selected initially using a simple Matlab simulation of the vehicle dynamics, including appropriate sensor discretization and lag. These gains were modified slightly during testing in order to achieve the best performance of the vehicle.
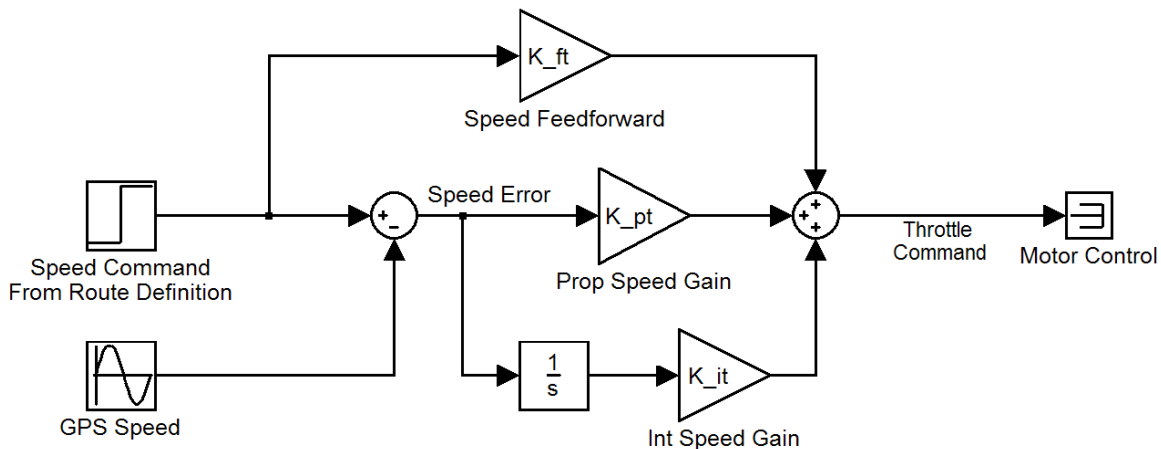
Figure 9: Speed Control Block Diagram

Electronics and Software Description

The Navigator controller was implemented on an Atmel 8-bit microcontroller, the Mega 32L[10]. This microcontroller was chosen because it is a single-chip computer that includes enough speed and memory for the math required, a serial port for communication with the GPS, as well as non-volatile data storage to store the programmed path and feedback gains between runs. The control algorithm was programmed in C using the Codevision compiler. In spite of being a puny 8-bit controller, the majority of the control law uses floating point calculations, which makes the programmer's job much more tractable than purely integer math.

As mentioned previously, a Garmin Etrex Legend was the sensor used. Although many other handheld GPS units output their data at 1 Hz, which is faster than the Etrex, the compass heading algorithm used internally in the Etrex seems particularly well-suited to the speeds of the autonomous vehicles tested. In testing with a few other GPS units that were available, lower feedback gains were required and larger errors encountered than with the Etrex.

Much of the program space available on the controller chip is used to run a user-interface for entering the route for the vehicle, calibrating the steering and throttle, and changing the feedback gains. A 4-button interface and a 2 line LCD text display are used to gather and display data. The ability to change feedback gains without reprogramming the chip is a key to the Navigator's use as an educational tool, since engineering students may test various gain combinations very quickly.

The user interface allows three methods of setting up a route or path for the vehicle. The simplest method is to carry the car around the desired route in the "Mark Route" mode. Pressing the "OK" button at each waypoint after selecting the speed for the leg with the +/- buttons defines the route using the current GPS position. Alternately, a route may be set up on a handheld GPS using the GPS's waypoints, and downloaded into the Navigator using the NMEA protocol. Any commercial software for creating routes on a handheld GPS may be used to create routes for the Navigator, since once they are downloaded to the GPS they may be downloaded to

the controller. The final method of setting up a route is to manually edit it using the buttons, which is useful for minor changes but too tedious to be recommended for creating a new path.

Figure 10: User Interface

The total electronics parts cost, without the GPS, is approximately $50. The Garmin Etrex GPS family varies in price from around $100-$300 depending on features such as mapping, with the Etrex Legend used for the prototypes priced at $200. The car used for the prototypes was a Duratrax Evader truck which costs around $120. This puts the total parts cost of the autonomous car at approximately $370, which most people would consider to be inexpensive for an autonomous vehicle.   Comparison with existing educational-market and research-level autonomous vehicle components such as the basic stamp, lynxmotion, and Innovation first show that the components used here provide a good balance of cost and capability for an off-road capable GPS-guided vehicle. Table 1 compares potential guidance computers to that used in the system, while Table 2 compares some common mobile platforms, and Table 3 compares some complete systems (although there are not many similar products available for comparison).

Part	Cost	Advantages/Disadvantages
Navigator Computer	$50	Comparison Baseline—custom electronics parts cost
Basic Stamp 2[11]	$49	Easy programming, no display or buttons, slow computations
Micropilot MP 1100[12]	$2000	Very sophisticated and capable, many sensors including GPS, no display or buttons
Innovation First Robot Controller[13]	$565	No display

Table 1: Comparison of Guidance Computers

| Part	Cost	Advantages/Disadvantages
Duratrax Evader[14]	$119.99	Comparison Baseline—prebuilt with speed control 2WD Off-road model truck
Lynxmotion[15] Carpet Rover Basic kit	$94.95	Not off-road capable
Lynxmotion[16] 4WD2 Basic Kit	$211.80	4 Wheel drive base, no speed controls
Max robot base '99[17]	$275	Can carry big load

Table 2:   Comparison of Mechanical Bases

| Part	Cost	Advantages/Disadvantages
Navigator	$370	Comparison Baseline—total cost
Parallax Boe-bot[18]	$249	Not off-road capable, no GPS, no display
Talrik Jr[19]	$189	Not off-road capable, no GPS, no display

Table 3: Comparison of Complete Vehicles

Performance

Figure 11 and Figure 13 show the performance of the Navigator when installed in a car and a boat, respectively. The data was gathered by the GPS unit's track-logging capability, so it is not the estimated position used by the control law. The jaggedness of the figures is due to the GPS discretization—this is the accuracy of the sensor (position is discretized into approximately 7 ft increments). The car was commanded to drive 3 MPH over the route shown, the boat is slow and always runs at full throttle (about 2.5 MPH). The final waypoint of the boat's path was placed far up on the shore of the lake where the boat was operated, so that the boat drove itself up on shore for easy retrieval, which is why the path never reached the final waypoint. Figure 12 shows the cross-track error for the car's path in Figure 11—note the transient error after turning around a waypoint onto a new leg of the course. The jaggedness of this plot is due to the discretization of the GPS position, but it is clear that the error is smaller than the GPS can accurately measure.

Several generalizations and observations of the performance of the car can be made. With the accuracy of the GPS, including its update rate, it is just barely possible to run the car in a suburban street without hitting the curb. A GPS route created down the middle of the street one day may be down the far edge of the street the next day due to different satellite configurations and the baseline GPS error. In a parking lot (or on a lake with the boat), the vehicles navigate precisely enough for a variety of interesting paths and missions.
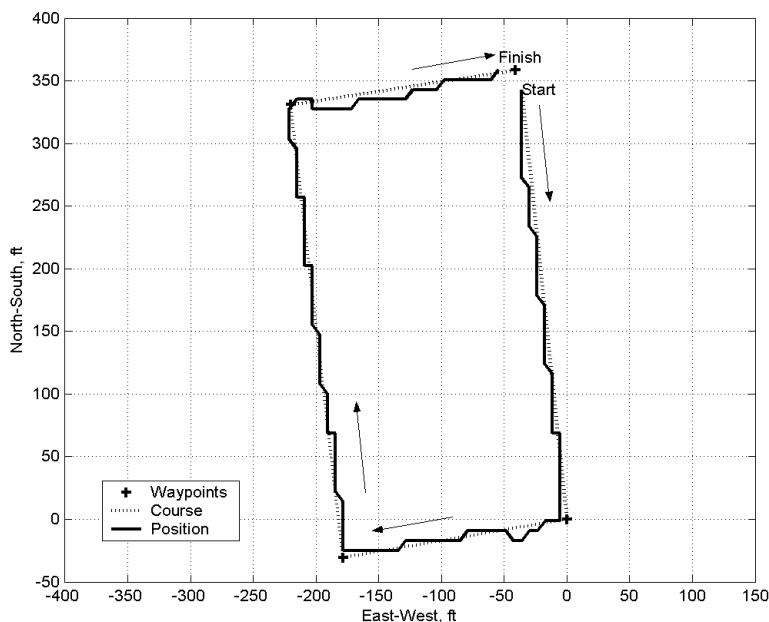
Figure 11: Autonomous Car Path
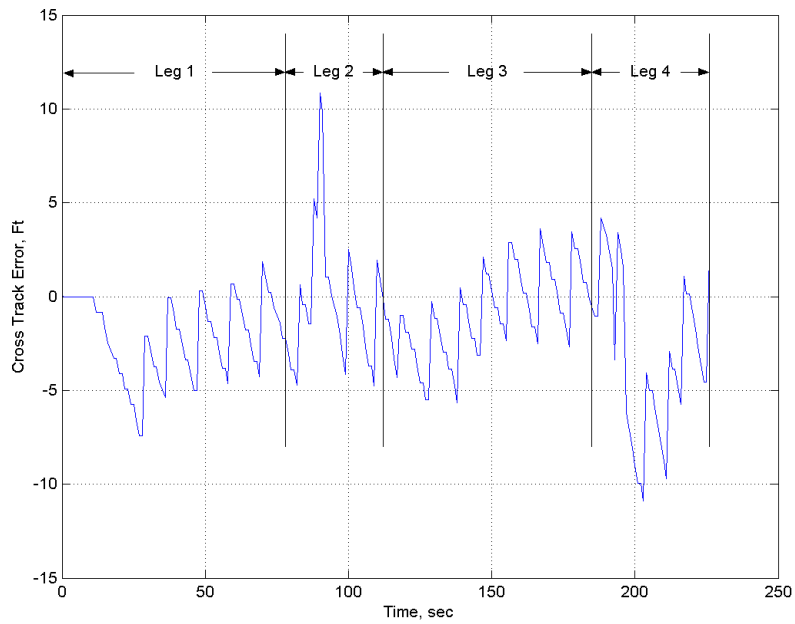
Figure 12: Car Cross Track Error
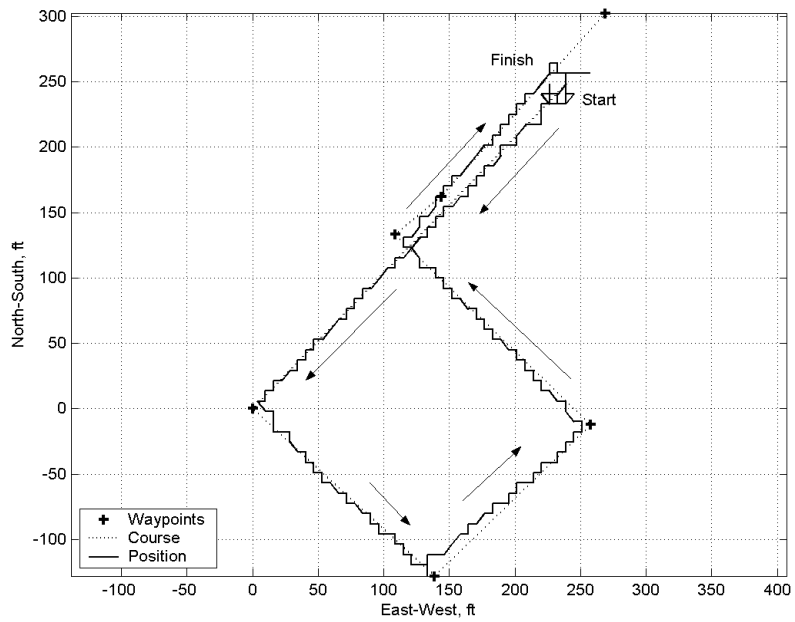
Figure 13: Autonomous Boat Path

Educational Uses

The original Navigator system was designed for a middle-school after school program[1]. The program allows students to build their own radio-controlled car to race, with various lessons embedded into the construction process. The Navigator car was used to extend the technology from radio-controlled cars to robotic cars, in order to motivate the students to learn more about engineering and technology.

Undergraduate engineering students find the Navigator system interesting, and it makes a good example for undergraduate classes. Undergraduates new to control systems can understand the simple proportional-integral feedback loops, and by setting individual gains to zero they can operate the car with either purely proportional or purely integral control. The estimator used is also easy to understand without much background beyond a dynamics class, as it can be derived from the principles of kinematics, and can provide a good example of the importance of filters and estimators.

The Navigator system is used in San Francisco State's undergraduate mechatronics class, as an example of navigation and feedback control. With a single sensor, a single chip controller, and two actuators, it is a good introduction to autonomous systems. The system and its control loops can be described in a few lectures, and the concepts are general enough that students can directly apply them to their own projects. The control software is also a good example of simple real-time control, that can be reused in other mechatronics projects.

For graduate students starting out in the field of autonomous systems, the Navigator is an excellent base upon which to build more complex systems. There is room to make the Navigator a more accurate navigation platform by adding additional sensors, such as speed sensors and turn rate or heading sensors, as well as obstacle avoidance sensors such as vision or ultrasonics. The sensors would need to be integrated into the state estimation, providing an opportunity to use advanced techniques such as Kalman filtering. A graduate student at SFSU is currently porting the Navigator control code to a prefabricated single-board computer, a good exercise to learn both the control code and the new controller. The student hopes to incorporate some added sensors to create a more capable platform. Results from the graduate student work will be published as it is completed.

Conclusion

As a single-sensor autonomous vehicle, the Navigator is remarkably successful. It can navigate to within the accuracy of the GPS unit that it is guided by, and may become a useful remote sensing tool. It is also a novel teaching tool that captures the interest of many engineering students. Careful combinations of classical feedback control and estimation techniques are all that are required for smooth operation, and are readily implemented in inexpensive controllers.

Future work is centered around finding practical work for the autonomous vehicles to do, as remote sensing platforms or transportation and delivery vehicles. A grant was recently received to extend the Navigator electronics to an aircraft, which will complete the Land-Sea-Air inexpensive autonomous vehicle trilogy.

Author Biography

Michael E. Holden is an Assistant Professor of Mechanical Engineering at San Francisco State University, where he has been teaching since August of 2003. Prior to this, he was Vice President of the MLB Company, and involved in many unmanned air vehicle programs. He can be reached at meholden@sfsu.edu, and would be happy to talk to people interested in using the Navigator system.

References

[1] Student Racing Partners: www.studentracing.com

[2] The MLB Company: www.spyplanes.com

[3] S. Morris and M. Holden, "Design of Micro Air Vehicles and Flight Test Validation", Conference on Fixed, Flapping and Rotary Wing Vehicles at Very Low Reynolds Numbers, University of Notre Dame, June 2000

[4] See for example the excellent hardware at www.microhardcorp.com

[5] K. Ogata, Modern Control Engineering, 4th edition, Prentice Hall, 2002

[6] M. Grewal and A. Andrews, Kalman Filtering, 2nd edition, Wiley, 2001

[7] K. Murphy and S. Legowik, "GPS Aided Retrotraverse For Unmanned Ground Vehicles", Proceedings of the SPIE 10th Annual AeroSense Symposium, Conference 2738, Navigation and Control Technologies for Unmanned Systems, Orlando, FL, April 1996

[8] G. Franklin, J. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, 3rd edition, Addison – Wesley 1994

[9] K. Ogata, Modern Control Engineering, 4th edition, Prentice Hall, 2002

[10] www.atmel.com

[11] www.parallaxinc.com

[12] www.micropilot.com

[13] www.innovationfirst.com

[14] www.towerhobbies.com

[15] www.lynxmotion.com

[16] www.lynxmotion.com

[17] www.robotstore.com

[18] www.parallaxinc.com

[19] www.firstrobotics.com