Trajectory Generation and Control Methodology for an Autonomous Ground Vehicle

John Connors and Gabriel Hugh Elkaim[*†]

The Overbot, originally designed to compete in the DARPA Grand Challenge, has been retasked as a testbed for autonomous vehicle research. A complete platform, the vehicle allows for development and validation of new control algorithms, actuators and sensors by allowing integration into an existing architecture. An overview of the vehicle is provided, describing the interaction of hardware and software. A new method is developed for manipulating spline-based paths in obstacle rich environments. The generated trajectories are forwarded to a new controller developed for the vehicle to follow the given paths. Various control techniques are tested on the vehicle and both the path planning and control are validated on a $600m$ offroad course.

Nomenclature

c	Curvature
$dist(\cdot,\cdot)$	Euclidean distance
f	Trajectory curve
i	Integral
k	Gain
Pos	Position
W^{GPS}	Waypoint in GPS frame
WP	Initial waypoint

[*]This work was supported by the National Science Foundation under NSF grant 0521675

[†]Autonomous Systems Laboratory, Computer Engineering Department, University of California, Santa Cruz

Y	Crosstrack
θ	Heading
ξ	Position
Subscript	
$carrot$	Projected along heading
$Error$	Error
$Goal$	Goal or Target
$Proj$	Projected along path
V	Current

I. THE OVERBOT

The Overbot, shown in Figure 1 was designed and built by a team of Silicon Valley engineers to compete in the 2004 and 2005 DARPA Grand Challenge events. Though selected for both competitions, the Overbot was unable to complete the qualifying events for either race. In 2005, the vehicle was donated to the University of California, Santa Cruz, for use as a research tool in the Autonomous Systems Laboratory.

The Overbot is a fully autonomous vehicle, designed to follow a series of GPS waypoints. The goal of the vehicle is to traverse each waypoint, in order, while staying within a given radius of the nominal straight line path and avoiding all obstacles. To accomplish this task, the vehicle is equipped with various sensors and actuators, as well as custom designed software to navigate the vehicle. During autonomous operation, the vehicle is completely self contained and receives no external human input.

I.A. Hardware Architecture

The Polaris Ranger is a commercially available utility vehicle designed for off-road environments. The Ranger is aptly suited for traversing mountainous and rocky terrain with a top speed of around 40 mph and the ability to climb over large obstacles.

The Sick Lidar unit is a highly accurate laser rangefinder. Using a rotating mirror, the Lidar emits an infrared beam across a 180° viewing angle. As the beam reflects back to a sensor, the range can be calculated for each point in the beam's rotation [10]. On the Overbot, the Lidar unit has been mounted above the vehicle on a custom gimbaled mount. This setup allows the Lidar to project downward onto the ground, and thus provide a contour of the road ahead. The mount itself is driven by a servo motor and can be pitched relative to

Figure 1. The Overbot autonomous vehicle.

the vehicle itself. Additionally, the Overbot incorporates a Unibrain Fire-I 400 color camera, a CCD based camera capable of 659(H) x 494(V) resolution. The camera is mounted atop the vehicle near the Sick Lidar unit for a similar vantage point of the terrain.

The Crossbow AHRS 400CB-200 is an attitude and heading reference system incorporating a 3-axis accelerometer, rate gyros, and a three axis magnetometer. By fusing these measurements the unit is able to estimate roll, pitch and yaw as well as their relative rate of change. Together, these measurements determine the orientation and motion of the vehicle dynamics. The AHRS 400CB-700 is accurate to $\pm 2.5^\circ$ for pitch and roll, and $\pm 4^\circ$ for yaw [8], though in practice, it has been found to be highly susceptible to noise.

For the vehicle to autonomously maneuver from location to location, it is essential to maintain an accurate measurement of the vehicle's position. The Overbot relies on a commercial Global Positioning System (GPS) device from Novatel. The Novatel ProPak-LB, when coupled with the Omnistar L-Band service, use differential corrections to reduce typical position errors to less than 0.10 meters [9].

On the Overbot chassis, Galil motion controllers are used to control the steering, throttle, brakes and shifting. The steering shaft is rotated by an industrial servo motor and the original steering rack is used to control the vehicle. A modified cruise control unit actuates the throttle to regulate fuel to the Ranger's engine. A screw drive and cable actuates the vehicle's stock brake pedal to regulate brake pressure in the master cylinders. This setup always manual application of the brake system if necessary. An additional motor is used to actuate the transmission linkage to select various forward and reverse gears.

Each subsystem on the vehicle is connected together through an Ethernet network and a series of routers. Communication is handled by transmitting UDP packets from device to device. Some devices, like the Galil controllers, are Ethernet enabled, while others, such as the LiDAR and GPS unit, have USB, Firewire or serial ports, and are adapted to communicate on the network.

I.B. Software Hierarchy

The Overbot uses a single Pentium class processor for all onboard computations. The machine runs the QNX real time operating system to run the custom software that controls the vehicle. The software is divided into several process threads, each serving a different task. Together, the threads define a hierarchy with the sensor information input at the top, and actuator commands output at the bottom (Figure 2). Each thread, referred to as a server, generally only communicates with the threads above and below it, building a level of abstraction that assumes the input data is both recent and accurate.

The highest level of the software hierarchy is the GPSINS Server, which communicates

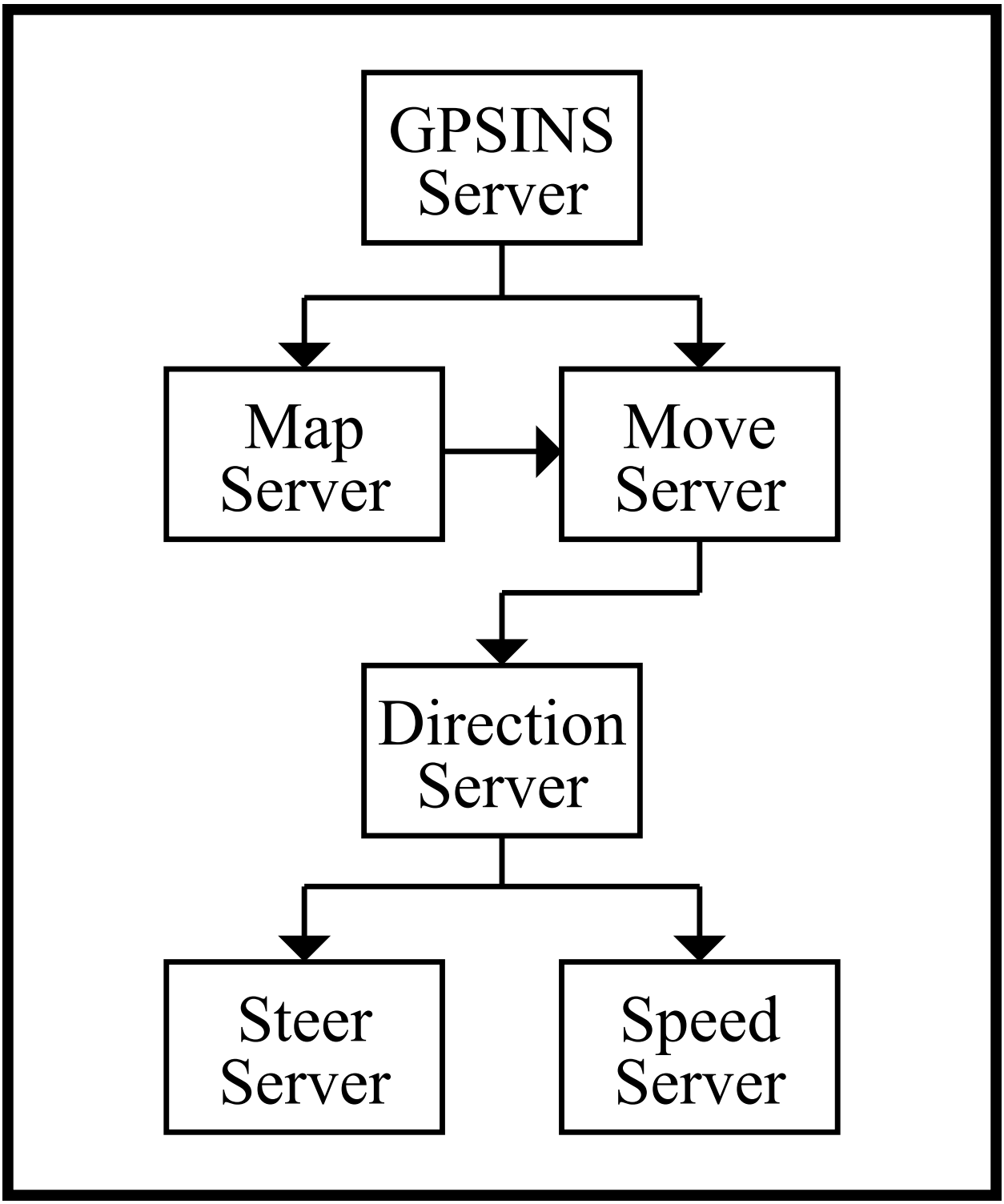

Figure 2. The software threads on the Overbot form a hierarchy to command movements based on sensor inputs.

with the Novatel GPS receiver to determine the vehicle's position. With the addition of the Omnistar High Performance subscription, the position measurements are accurate to decimeter levels. The GPSINS Server also interfaces to the Crossbow AHRS 400CB-200 roll, pitch, yaw and angular rate estimator. The Crossbow unit internally calculates Kalman filtered angles at a rate of 60hz. Together, the GPS receiver and AHRS provide the position and orientation of the vehicle.

The computer vision techniques employed by the Overbot are implemented in the Map Server. The laser range finder is used as the primary tool to monitor the environment and detect obstacles. Distance information is measured for discrete angular increments within the vision plane of the range finder. When compared to the vehicle's position and orientation, and the tilt angle of the range finder, these measurements represent points in the physical environment. Over time, as the range finder scans over the ground, these points are aggregated to represent surfaces and ultimately define obstacles. All such obstacles are projected onto the surface of the Earth to create a two dimensional, discretized map.

The Steer Server acts as the path planning control for the vehicle. Using the information provided by the GPSINS Server, the Steer Server plots a path to the desired destination. Obstacle information is included and trajectories are planned to avoid these obstacles as well as the imposed corridor constraints. The resulting path is evaluated and fed forward to the Move Server.

The majority of the vehicle dynamics and error checking occur inside the Move Server. Desired *distance*, *max speed* and *curvature* are commanded from the Steer Server. In general, the Move Server attempts to keep the vehicle moving as fast as possible while obeying the *max speed* limitation. Simple vehicle dynamics are computed with the orientation information calculated above. These dynamics limit the vehicle's speed for the given *curvature* and terrain. Final speed, gearing and curvature decisions are made and send to the Speed and Direction Servers.

The main purpose of the Speed Server is to interface with the hardware motion controllers. The Speed Server has control over the throttle, brakes and gear selection. Commands are accepted from the Move Server and converted to actuator position to change the speed, acceleration and gearing.

The steering counterpart to the Speed Server is the Direction Server. The software acts as a translation between the the Move Server commands and the instructions sent to the hardware controller. The Ranger's steering box is actuated by a servo motor and the resulting motion is captured through the use of an optical encoder.

II. PATH PLANNING

As described previously, the Overbot is designed to follow GPS waypoints. These points define a nominal straight line path for the vehicle to follow. Each point must be traversed in order, while maintaining a maximum radius from the nominal path. This distance defines a corridor about the path, constraining the space the vehicle is allowed to occupy. Vision sensors on the vehicle continually scan and map obstacles into a bitmap stored on the vehicle. At discrete time steps, a trajectory is generated through the defined corridor and obstacle map using the vehicle's current position and orientation. Two path planning techniques implemented on this vehicle are briefly described below. More detail about these algorithms is available in previous work [7] [5].

II.A. Arc Based Paths

The Overbot's original path planning technique follows a goal based approach [7]. Paths are generated relative to a GPS reference frame, with East and North axes. A goal point, W_{Goal}^{GPS} is computed, and the resulting path is defined as the circular arc through W_{Goal}^{GPS} and vehicle's current position, W_{V}^{GPS}, and constrained to be tangent to the vehicle's current heading, θ_V. For any given planning step, the vehicle's position and orientation are fixed. Therefore the path is uniquely defined by the goal point. Relocating W_{Goal}^{GPS} alters the path length, as well as the curvature.

For each arc generated, a wedge is defined about the arc with a width equal to that of the vehicle. For the path to be valid, the entire wedge must be clear of obstacles. If a suitable wedge is not found, the goal point is moved until a clear path is found. Moving W_{Goal}^{GPS} closer to W_{V}^{GPS} results in a smaller wedge and becomes more likely to be clear of obstacles. The wedge is evaluated by pixelizing arcs within the wedge using Bresenham's algorithm [2]. Each point is compared to the obstacle map to determine wether an obstacle collision will occur.

The simplicity of this algorithm limits the running time required to compute a trajectory. Running time is important in a real time system to ensure that events happen when necessary. However, the restriction of using a singular circular arc limits the ability to maneuver in complex environments. To guarantee obstacle free paths, the path length will be reduced and may not plan a complete route through the obstacle field. As a result, this approach often fails to traverse environments with numerous obstacles.

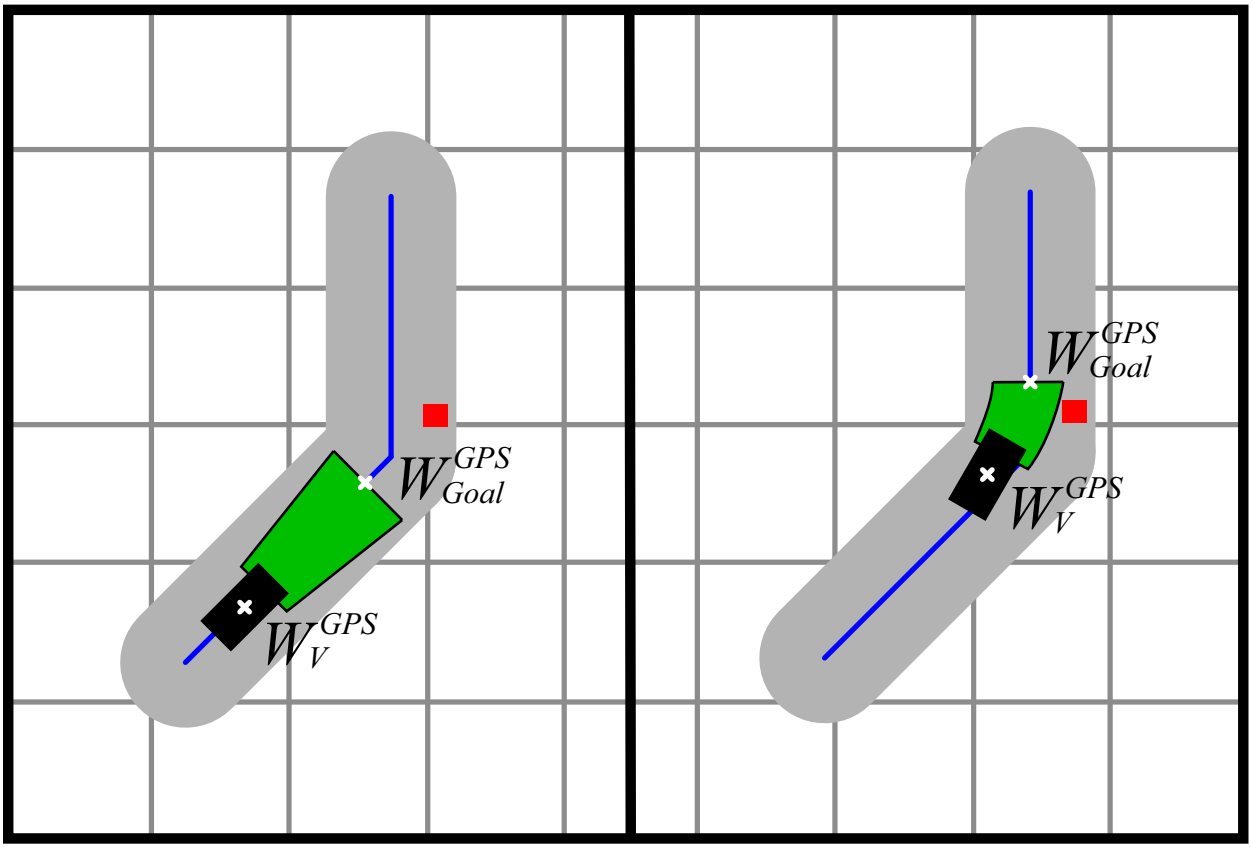

Figure 3. Curved wedge indicates a possible path.

II.B. Spline Based Paths

A second approach is implemented to overcome the shortcomings of the arc based solution [5] [1]. For this algorithm, the resulting trajectory is represented by cubic splines. Splines are piecewise polynomial functions. For use as trajectories, the spline is constrained to be continuous through its second derivative to guarantee the continuity of the path, heading and steering angle.

To overcome the short sightedness of the arc based method, paths are planned through three consecutive waypoints, ie two path segments. Paths further than three consecutive points are generally beyond the range of the sensor inputs and are not considered. By using three points, a rotation is guaranteed that will result in the needed monotonicity to calculate a spline path [5]. The rotation defines a new reference frame and allows a cubic spline to be interpolated through the given waypoints.

To validate the path, the spline is mapped back into the GPS reference frame. The path is discretized to the resolution of the obstacle map and each pixel evaluated. For this implementation, each obstacle detected by the vision sensors is inflated when stored in the map. Enlarging the obstacles compensates for the width of the vehicle, and only the path of the center of the vehicle must be evaluated. Obstacle collisions and corridor violations are recorded and isolated into individual violations.

The curve is manipulated to bend around a given collision. A new point is introduced into the cubic interpolation in the vicinity of the violation [5]. A new spline is generated through the new control point. By introducing additional points, the curve can be manipulating to avoid the obstacle or stay within the defined corridor. The process is repeated, with the new curve being evaluated within the obstacle map. The curve is iteratively refined until a collision free trajectory is found (Figure 4).

This method uses more complex curves than the arc based approach, but still guarantees properties required to be traversable by a ground vehicle. Additionally, this approach is able to consider longer paths and consider multiple obstacles. As a consequence, the validation of longer paths requires more running time than the previously described algorithm [3]. The spline based method requires more time, but generates more usable paths in typical applications.

III. CONTROL

Control techniques must be implemented to track the paths generated by the path planning algorithms described above. The path acts as a feed forward term and the vehicle attempts to match its steering angle to the curvature of the path. This open loop method

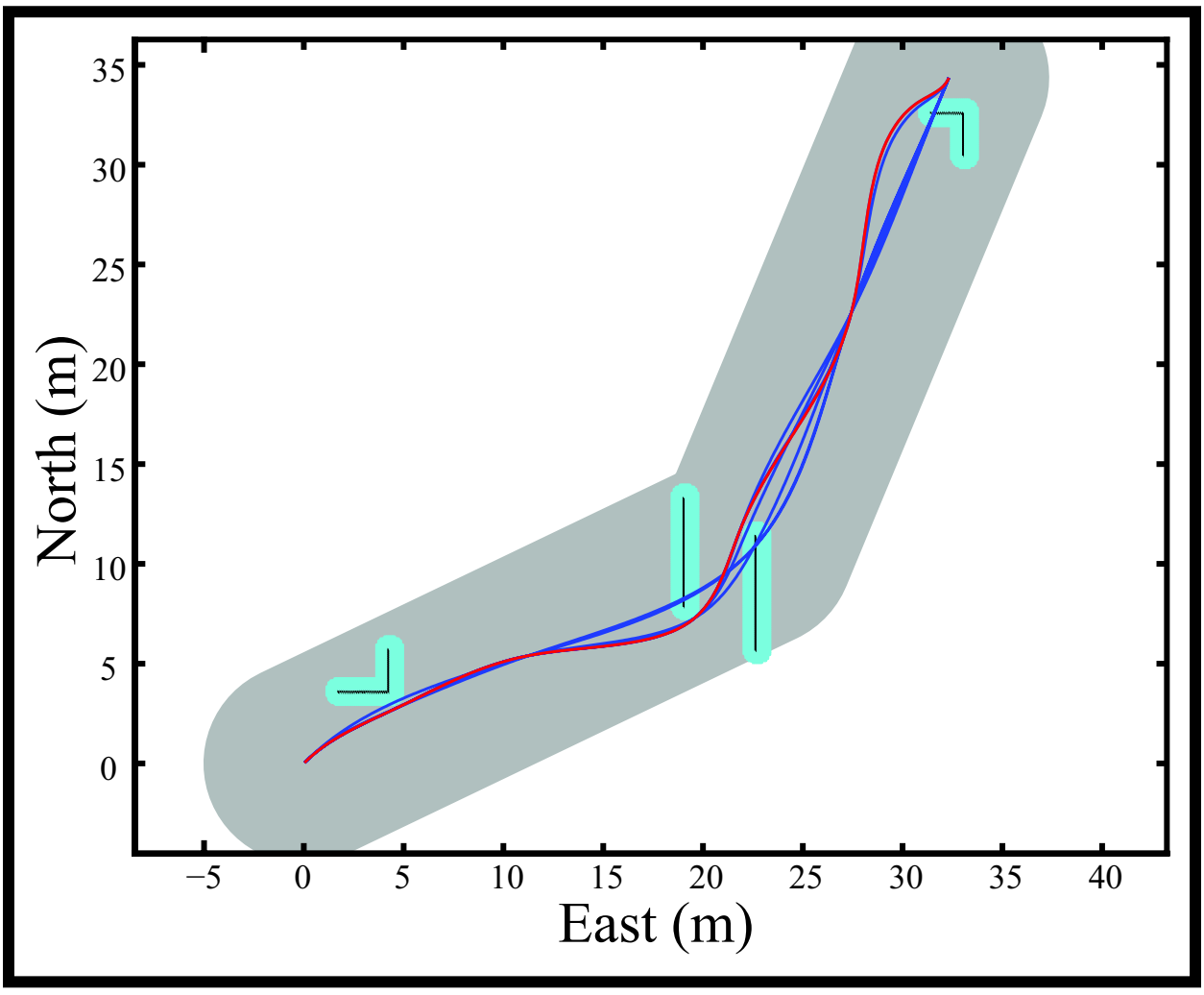

Figure 4. Smooth curves (blue) are iteratively generated, bending around obstacles until a collision free path (red) is found.

requires commands to be followed exactly as it has no method of compensating for sensor or actuator error. More advanced methods are typically used to track the vehicle's movement to the desired behavior and adjust for any errors.

The actual position of the vehicle, ξ_V, is projected in front of the vehicle to a point ξ_{Carrot}. This point introduces lead into the system to compensate for lag inherent in the actuators. ξ_{Carrot} is projected onto the curve, f, at ξ_{Proj} where ξ_{Carrot} lies on a line normal to f at ξ_{Proj}. The projected point, ξ_{Proj}, exists for any continuous function, but may not be unique. However, the maximum distance between any two projected points is $2 * dist(\xi_{Carrot}, \xi_{Proj})$. For well behaved systems, $dist(\xi_{Carrot}, \xi_{Proj})$ is minimal and the possible solutions are roughly equivalent. For this controller, ξ_{Proj} is calculated as the point of f closest to ξ_{Carrot} (Figure 5).

The curvature c_{ff} of f is evaluated at ξ_{Proj} and issued as a steering command for strict feed forward control. In addition, feedback can be used to better track the curve by comparing the relative values of ξ_V and ξ_{Proj}. The cross track error, Y_{Error}, is defined as the distance between ξ_{Carrot} and ξ_{Proj} and the heading error, θ_{Error} as the difference between θ_V and the heading of f at ξ_{Proj} (Figure 5). i_{Error} represents the integral of the cross track error over distance. Feedback gains can be calculated such that the commanded curvature is

$$c_{Command} = c_{ff} + k_y * Y_{Error} + k_\theta * \theta_{Error} + k_i * i_{Error}$$

The feedback control was developed and tested in two separate environments. The controller was first adapted to the vehicle and tested on a straight line course of approximately $80m$. These tests attempted to remove the errors inherent in the open loop implementation of the arc based path planning algorithm. When the spline based approach was implemented, the controller was retested and adjusted for complex curves.

III.A. Straight Line Control

A series of tests were conducted on an $80m$, straight path over level, unpaved ground without obstacles. The vehicle used the arc based path planning described above to generate paths twice per second in a receding horizon fashion. Errors calculations were made every 100ms and used to adjust the commanded steering curvature. Open loop, PD and PID controllers were tested, each completint the course 10 times. Performance was evaluated by measuring the crosstrack error of the vehicle relative to the straight line path defined for the vehicle. The initial and final segments of each test were truncated to include only the middle $50m$ section of the test. The removed sections are not indicative of actual performance as they include the effects of the conversion time of the internal GPS algorithms and transient effects

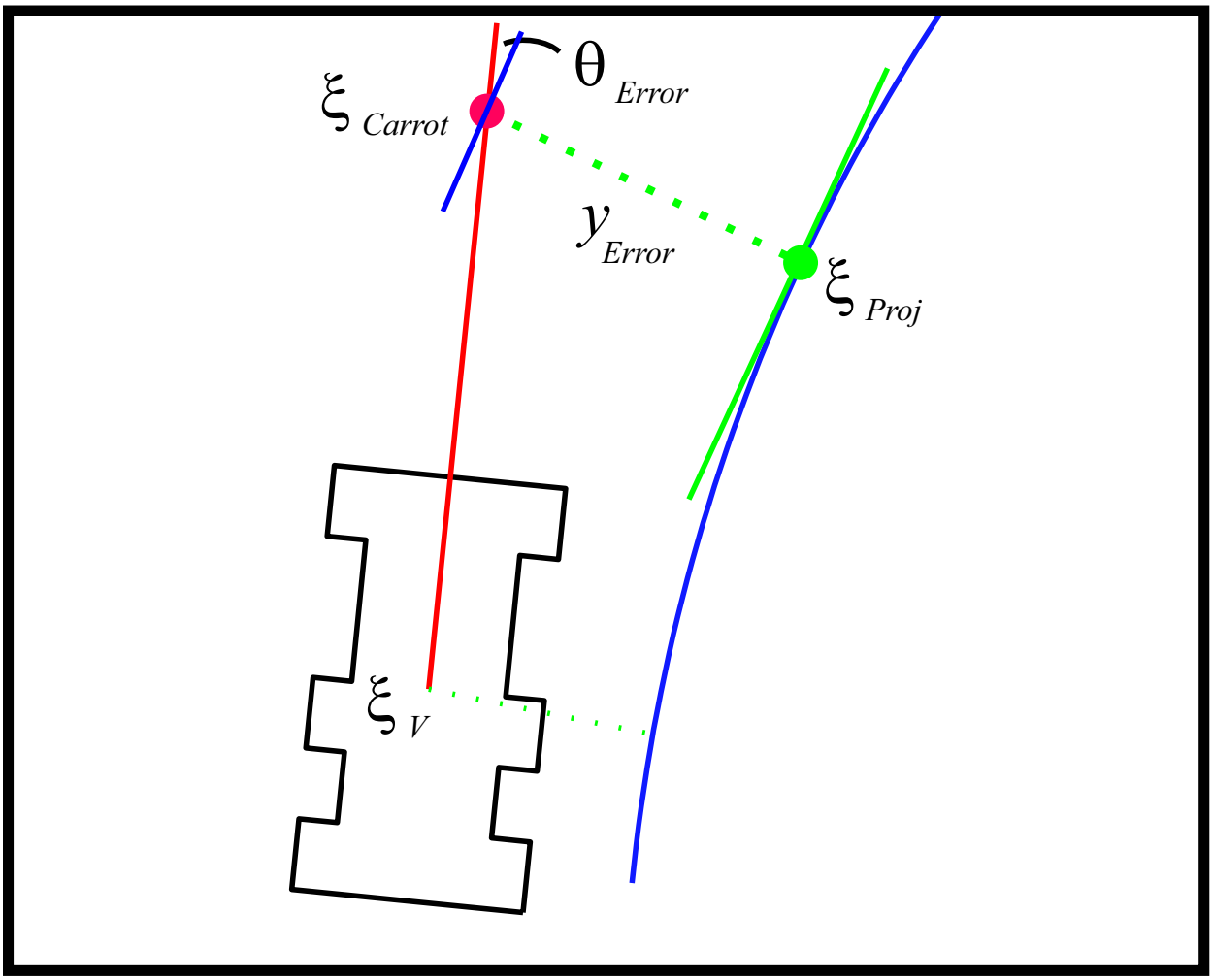

Figure 5. The actual vehicle position is projected onto the curve to calculate the curvature, cross track error and heading error.

associated with the various controllers.

The cross track error was calculated as the length of the line normal to the path and passing through the vehicle position, which is unique for a straight line path.

$$error_t = sin(\Psi_{Path} - \Psi_{V(t)}) * dist(WP, Pos_t),$$

where Ψ_{Path} is the heading along the path, $\Psi_{V(t)}$ is the angle between the initial point of the path and the position of the vehicle at time t, WP is the initial point of the path, and Pos_t is the position of the vehicle at time t. These results are analyzed below and presented in Table 1.

Controller	Cross track Error(cm)	Standard Deviation(cm)
Open Loop	114.5	25.4
PD Control	62.3	7.3
PID Control	14.8	10.3

Table 1. The use of feedback control on the Overbot decreases the vehicle's cross track error.

III.A.1. Open Loop Control Results

The open loop control commands steering angles based on the curvature of the computed trajectory. This approach is subject to actuator errors and biases. As no feedback terms were calculated for this test, new paths were generated every 100ms, as opposed to every 500ms, as used for the other tests. By default, the arc based path planner starts the goal point on the straight line path. As a result, as the crosstrack error grows, the curvature of paths returning to the straight line path increases, eventually canceling the effects of steering biases. This effect results in the vehicle tracking parallel, but offset, to the straight line path.

The effects of these errors can be seen in test results, presented in Figure 6. A misalignment in steering causes the vehicle to be offset from its nominal path in one direction. Once significantly off course, the adjustment of the path planning results in a fairly parallel path, as can be seen in the graph. The results indicate an average cross track error of 114.5cm and a standard deviation of 25.4cm.

III.A.2. Proportional and Derivative Control Results

With the open loop results as a baseline, a second controller was tested that included feedback of the error terms defined above. The cross track and heading errors were included proportionally into the steering control along with the original feed forward term to create

define the control law

$$c_{Command} = c_{ff} + k_y * Y_{Error} + k_\theta * \theta_{Error}$$

The results presented in Figure 6 demonstrate this controller's ability to correct for some of the errors introduced by the vehicle dynamics. The control techniques actively track the nominal path and demonstrates a smaller average cross track error of 62.4 cm. The error that still exists as a constant offset in the cross track is due to a steering bias in the vehicle. This problem can be further diminished by the introduction of integral control. In addition to reducing the offset, the use of the PD controller also reduced the standard deviation to 7.4 cm, approximately two and half times better than the purely open loop control.

III.A.3. Proportional, Integral and Derivative Control

To eliminate the remaining crosstrack error, an additional error term was included in the feedback calculations. An integral error was computed as the area between the actual path and the ideal curve. This term was included in the control in the same way as the proportional and derivative terms to create a new control law

$$c_{Command} = c_{ff} + k_y * Y_{Error} + k_\theta * \theta_{Error} + k_i * i_{Error}$$

The use of an integral gain tends to reduce overall error at the expense of increasing standard deviation. While the results show an average crosstrack error of 14.8 cm and standard deviation of 10.3 cm and correlate well with these expectations, the cross track error was not eliminated. In analyzing the data, other errors were found to be reducing performance in addition to the steering bias. Inaccuracies in the heading sensors, when included in the derivative gain, k_θ, and when magnified by the projection, ξ_{Proj}, resulted in similar bias issues.

III.B. Control for Complex Curves

A new test route was laid out to evaluate the newly implemented spline based path planning algorithm. The course followed a 600m rutted dirt road with a right turn approximately halfway through. The course spanned a hillside and various sections of the route were inclined, both up and down, as well as pitched side to side. The final 200m of this course represents a continual incline, climbing approximately 30m.

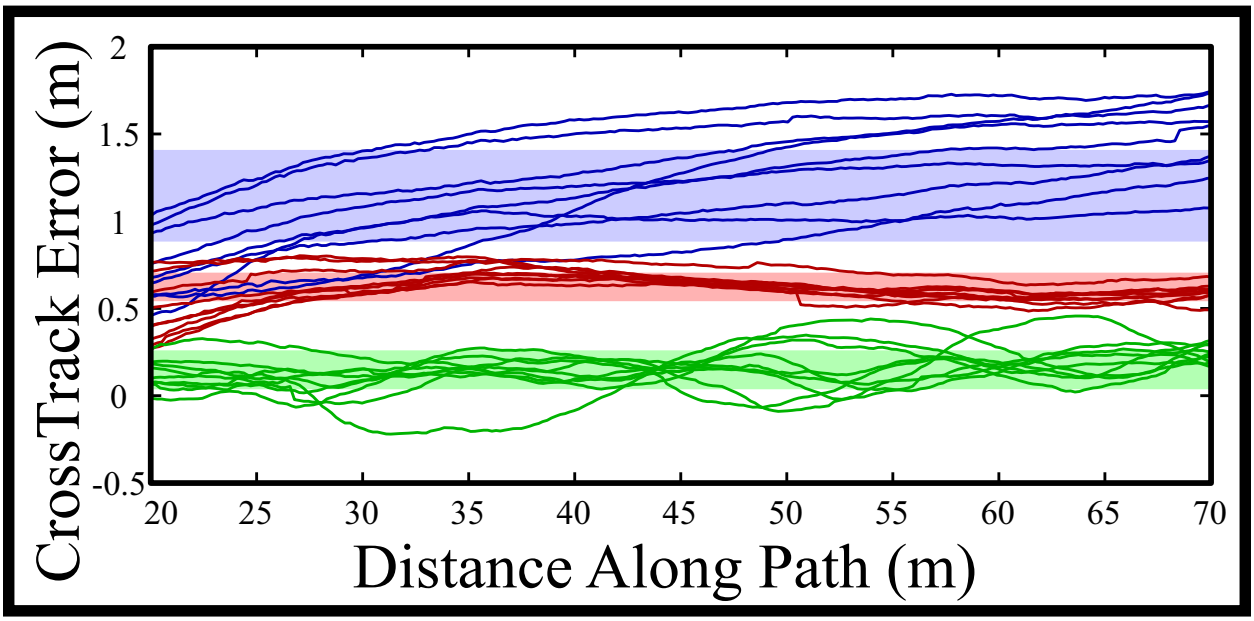

Figure 6. Cross track errors were recorded for each of the controllers tested. The error for each run is plotted over a band denoting the average ± standard deviation for each controller, blue showing open loop, red showing PD control and green showing PID control.

III.B.1. Controller Variations

The controllers used for the straight line tests were found to be unstable on the complex curves required to navigate this course. A variety of controllers were developed using different error terms, gains and projection lengths. A PD controller was selected for simplicity and to avoid windup time required to make an integral term useful.

For each controller tested, the use of the projected point, ξ_{Proj}, resulted in oscillatory behavior. Because of noise in heading measurements, the accuracy of ξ_{Proj} is unreliable. Errors in the heading are magnified by the projection and corrupt the crosstrack error, used for proportional feedback. As a result, the projection was not used and all errors were calculated based on the vehicle's measured position, ξ_V.

As a further refinement, the replanning step was reduced from 500ms to 1s. Error calculations and feedback correction were still computed every 100ms. Due to that fact that paths are continually replanned from the vehicle's present position, resulting in no crosstrack or heading error, increasing the time between path generation made it easier to see the effects of the feedback control. A final set of control gains were selected empirically for the best performance. This controller was found to be suitable for both path planning techniques described in this work.

III.B.2. Successful Paths

Initial tests were conducted on the previously mentioned course without the presence of obstacles. The feedback controller was refined to more accurately track the curves involved in this course as described above. Both the original arc-based approach, and the new spline method presented here were tested. Each were able to consistently complete the defined course. This test demonstrates the functionality of the algorithm within the context of the vehicle and confirms the satisfiability of the corridor constraint.

The completion of the multiple waypoint course also indicates the successful implementation of the receding horizon control. Log files from the vehicle confirm the replanning of paths as the vehicle progressed through the course. In general, the new paths were very similar to previous routes but were able to compensate for errors in vehicle control or GPS drift. In the absence of obstacles, both algorithms performed nearly identical to one another. It should be noted that the modified controller gains greatly increased the performance of the arc-based approach. This method is now empirically more stable and more consistent than in prior experiments.

A virtual environment was constructed with 10 randomly placed obstacles used as a simulated environment for the Overbot. Again, both methods were run through the entire course. The spline-based algorithm was able to successfully find clear paths around the new obstacles and traverse the course. The test data, shown in Figure 7, show paths adapting to obstacles and maneuvering around them.

An additional test was run on an environment containing 20 obstacles. Due to various issues, discussed in other work [4], the test was only run up to the main turn, approximately 300m. The results continue to verify the success of the algorithm on the vehicle. For this course, the obstacles were denser than previously tested. The vehicle maneuvered around these obstacles, even when avoiding one collision would normally have led the vehicle toward another. This test demonstrates the algorithm's ability to wind paths between obstacles. Again the vehicle traverses the course while satisfying the corridor constraint and the avoiding obstacles, demonstrating the success of the path planning and the control.

IV. CONCLUSION

The Overbot, once a DARPA Grand Challenge entry, is now a capable test platform for autonomous vehicle research. The vehicle is actuated by a series of motors and controllers. Vision and positioning devices are outfitted on the vehicle for navigation and sensing. A pentium class PC runs custom software to monitor the environment, plan trajectories and actuate the vehicle.

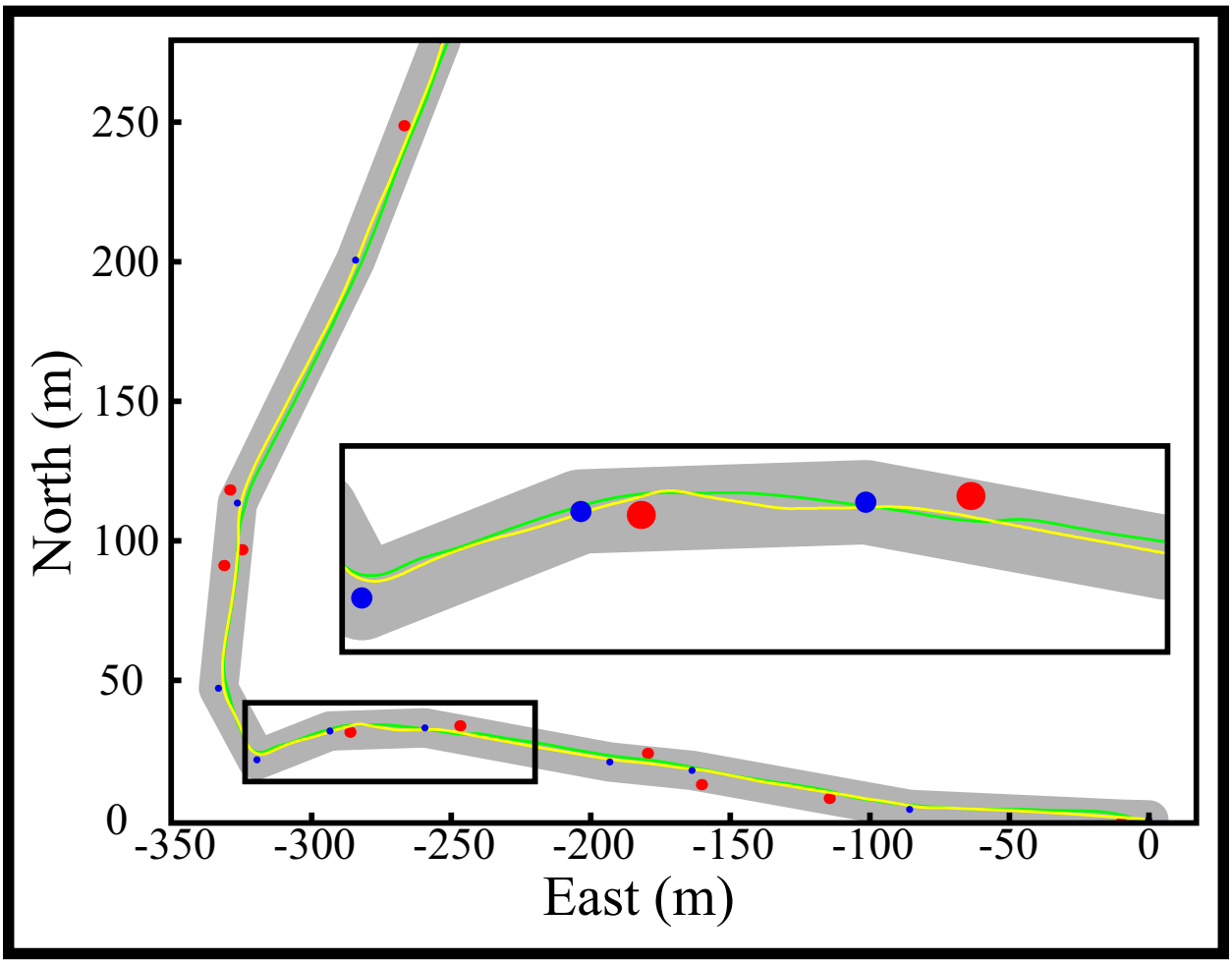

Figure 7. The vehicle was also run through an environments with 10 randomly placed obstacles (red). The paths followed by the arc (green) and spline (yellow) methods are shown.

Two different path planning techniques are discussed, one based on singular circular arcs, the other on cubic splines. Both methods are implemented on the Overbot and validated on an offroad course. Feedback control is developed for the vehicle and tested on various environments. The vehicle control is improved dramatically and able to follow complex curves.

V. FUTURE WORK

Further work will improve the spline based algorithm and eliminate issues resulting from extended running times. The algorithm could also be expanded to include time information be used for dynamic obstacles. For control purposes, bounds will be imposed on the curvature of the resulting paths [4].

Additionally, improvements to the vehicle as a testbed platform will aid in future development of control algorithms, sensor suites and actuators. The Overbot will be adapted to be more modular and more easily reconfigurable. The flexibility of the ethernet communication allows for easy addition of new hardware, but the software structure of the vehicle is heavily coupled. The architecture will be redesigned to allow easier substitution of new software, drivers and controllers.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of the Autonomous Systems Laboratory and the National Science Foundation.

References

[1]Tomas Berglund, Hkan Jonsson, and Inge Sderkvist. An obstacle-avoiding minimum variation b-spline problem. *Proceedings of the 2003 International Conference on Geometric Modeling and Graphics*, 2003.

[2]Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25, 1965.

[3]John Connors and Gabriel Hugh Elkaim. Analysis of a spline based, obstacle avoiding path planning algorithm. *Proceedings of the Vehicular Technology Conference*, 2007.

[4]John Connors and Gabriel Hugh Elkaim. Experimental results for spline based obstacle avoidance of an off-road ground vehicle. *Proceedings of the ION Global Navigation Satellite Systems Conference*, 2007.

[5]John Connors and Gabriel Hugh Elkaim. Manipulaiting b-spline based paths for obstacle avoidance in autonomous ground vehicles. *Proceedings of the ION National Technical Meeting*, 2007.

[6]Carl de Boor. *A Practical Guide to Splines*. Springer, revised edition, 1978.

[7]Gabriel Elkaim, John Connors, and John Nagle. The overbot: An off-road autonomous ground vehicle

testbed. *Proceedings of the ION Global Navigation Satellite Systems Conference*, 2006.

[8]Crossbow Technology Incorporated. Ahrs400 series users manual. http://www.xbow.com, 2005.

[9]NovAtel Incorporated. Novatel propak-lb plus user manual. http://www.novatel.com, 2005.

[10]SICK Incorporated. Laser measurement systems technical description. http://www.mysick.com, 2003.