Manipulating B-Spline Based Paths for Obstacle Avoidance in Autonomous Ground Vehicles

John Connors and Gabriel Elkaim
Jack Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, California, 95064

BIOGRAPHY

John Connors completed B.A. degrees in Computer Science and Mathematics at the University of California at Berkeley, Berkeley CA, in 2004. He worked as a PCB engineer and consultant designing audio/video switching devices before starting as a graduate student in 2005 in the Jack Baskin School of Engineering, at the University of California, Santa Cruz, Santa Cruz CA, working towards a M.S. in Computer Engineering. His research interests include electric vehicles, robotics and autonomous control. His current project centers around outfitting an autonomous offroad ground vehicle as a reconfigurable autonomous testbed for comparative analysis of various control techniques and sensor suites.

Gabriel Elkaim received his B.S. degree in Mechanical/Aerospace Engineering from Princeton University, Princeton NJ, in 1990, and both M.S. and Ph.D. degrees from Stanford University, Stanford CA, in Aeronautics and Astronautics, in 1995 and 2002 respectively. In 2003, he joined the faculty of the Computer Engineering department, in the Jack Baskin School of Engineering, at the University of California, Santa Cruz, Santa Cruz CA, as an Assistant Professor. His research interests include control systems, sensor fusion, GPS, system identification, and autonomous vehicle systems. His research focuses on intelligent autonomous vehicles, with an emphasis on robust guidance, navigation, and control strategies. Specifically, he has founded the Autonomous Systems Lab at UC Santa Cruz, and is currently developing an autonomous wing-sailed marine surface vehicle and off-road autonomous ground vehicles.

ABSTRACT

The Overbot is one of the original DARPA Grand Challenge vehicles now being used as a platform for autonomous vehicle research. The vehicle, equipped with a complete actuator and sensor suite, provides for an extremely capable robotic platform with computing infrastructure and software framework already in place to create a reconfigurable testbed.

For point to point navigation, calculating suitable paths is computationally difficult. Maneuvering the vehicle safely around obstacles is essential, and the ability to generate safe paths in a real time environment is crucial for vehicle viability. We present a method for developing feasible paths through complicated environments using a baseline smooth path based on cubic splines. A method for adjusting and bending the spline to avoid obstacles is developed. This method is able to iteratively refine the path to more directly compute a feasible path and thus find an efficient, collision free path in real time through an unstructured environment. This method, when implemented in a receding horizon fashion, becomes the basis for high level control.

INTRODUCTION

The Overbot, originally designed to run the DARPA Grand Challenge, has been retasked as an off-road autonomous vehicle testbed. The base chassis is a Polaris Ranger 6x6 gasoline powered off-road vehicle. It has a single cylinder 30 HP engine capable of propelling the vehicle to over 40 mph. In addition to the chassis, the Overbot is equipped with a Sick Lidar on a custom gimbaled mount, a color firewire camera in a waterproof housing, a Crossbow AHRS, a Novatel differential GPS system and several modified 802.11.b wireless hubs. In terms of actuation, there are five Galil Motion Controllers which drive servomotors that handle shifting (H-L-N-R), braking, throttle, steering and sensor angle. Software which performs obstacle detection, path planning and low-level control is implemented in a Real Time Operating System (RTOS) on a standard Pentium class PC. The vehicle is presently being used to further research in autonomous vehicles at the University of California, Santa Cruz. Various elements are being improved and upgraded, including substantial efforts to improve path planning algorithms.

Research projects in autonomous vehicles have benefited from a recent surge of interest, due in part to the DARPA sponsored events. In addition to obvious military applications, autonomous vehicles have generated significant interest for commercial applications. The removal of human interaction from everyday driving could lead to safer roads, fewer accidents and increased traffic bandwidth. Even basic vehicle operation requires significant path planning to provide an accurate, feasible and smooth path from point to point through a potentially complex environment. The introduction of obstacles, such as buildings, pedestrians and other vehicles increase the complexity and limit traversable paths.

This paper details the implementation of basis splines (b-splines) as a tool for path planning. Splines are piecewise polynomial functions defined through a finite set of control points. Given n cartesian control points, (x_i, y_i), we create a spline, or series of polynomial functions, $(P_1, \ldots, P_{n-1})$ to traverse the given points. The use of splines allows us to constrain the polynomials such that the first derivative of $(P_1, \ldots, P_{n-1})$ is continuous across each boundary. Hence, $D^1 P_i(x_{i+1}, y_{i+1}) = D^1 P_{i+1}(x_{i+1}, y_{i+1})$. Furthermore, restrictions can be enforced on the starting and ending points to enforce a particular derivative value, $D^1 P_1(x_1, y_1) = C_1, D^1 P_{n-1}(x_n, y_n) = C_n$.

These characteristics of basis splines offer a number of convenient properties when discussing vehicle path planning. By enforcing $D^1 P_1(x_1, y_1) = C_1$, we can generate a path starting from the current vehicle position, (x_1, y_1), and having a given heading described by C_1. Therefore new paths can be generated and initialized from the current vehicle position and heading. As the first derivative of the path is proportional to the vehicle heading, a non-continuous derivative would result in an infeasible path for this type of vehicle, but the second derivative is proportional to the vehicle steering angle and any discontinuities would force the vehicle to stop at each control point to adjust its steering. By creating a path with continuous derivatives, we can guarantee smooth vehicle control and remain in motion throughout the vehicle path.

The mathematics involved in creating splines allow for easy construction of smooth paths through a given set of control points. We examine the initial construction of a path, the investigation of obstacles and methods for dealing with such obstacles. We introduce additional control points in the neighborhood of each obstacle and develop methods to shift these points away from obstacles and into clear areas. By moving control points, we can bend and shape our curve to safely traverse the environment.

Early methods of path planning involved straight lines or circular arcs[5][7] while later methods progressed to 3rd and 4th order polynomials[15]. Other methods take advantage of splines to produce smooth curves and can be designed to avoid obstacles but often require a great deal of computational power as they evaluate the entire path space[1][13]. In a real time environment it is beneficial to directly compute feasible paths continuously to allow for variations in the environment, control error and unmodelled sensor error.

We demonstrate successful path generation through a variety of environments while avoiding obstacles. We demonstrate abilities to compute paths in real time which makes this method practical for autonomous vehicle control.

THE OVERBOT

An overview of the vehicle's hardware and software is included for completeness though the method presented in this paper is not vehicle dependant.

Fig. 1. Polaris Ranger 6x6

HARDWARE

The Overbot is built from the Polaris Ranger, shown in Figure 1. The Ranger is a commercially available utility vehicle with a 3/4 ton towing capacity, the ability to drive through water up to 27 in. deep and over 6 inches of suspension travel front and rear. The 30 HP engine allows a top speed of around 40 mph and offers enough torque to overcome large obstacles[12].

On the Overbot, a Sick Lidar unit has been mounted above the vehicle on a custom gimbaled mount. The Lidar unit is a highly accurate laser rangefinder. The Lidar is an active device meaning it does not require any external illumination. Using a rotating mirror, the Lidar emits an infrared beam in 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[14]. This setup allows the Lidar to project downward at and angle to the ground, and thus provide a contour of the road ahead.

To aid in detecting obstacles and road boundaries, the Overbot incorporates a Unibrain Fire-I 400 color camera. The Fire-I is a CCD based camera capable of 659(H) x 494(V) pixels. The camera is mounted atop the vehicle near the Sick Lidar unit for a similar vantage point down onto the road[16].

The Crossbow AHRS 400CB-200 is an attitude and heading reference system, estimating roll, pitch and yaw using a 3-axis accelerometer, rate gyros, and a three axis magnetometer. Together, these measurements can be used to determine the orientation and rotation rates of the vehicle in motion[3].

The Overbot also incorporates a commercial Global Positioning System (GPS) device from Novatel. The Novatel ProPak-LB features a 24 channel "all in view" GPS receiver. When coupled with the OmniSTAR L-Band service, differential corrections can applied to reduce typical errors to less than 0.10 meters[10].

The DMC-1400 series Motion Controllers from Galil Motion Control inc. are designed to offer convenient control of brushed, brushless and servo motors. With digital and analog inputs, and digital outputs, the Galil devices are some of the most versatile controllers available[6]. On the Overbot chassis, the Galil controllers are used to control the steering, throttle, brakes and shifting.

The Polaris Ranger has hydraulic disc brakes on four of its six wheels. They are actuated by a screw drive device driven by a brush-type servomotor with two limit switches on the actuator, and a pressure switch and pressure sensor on the front hydraulic system. The actual throttle actuator is a modified cruise control unit. The cruise control motor has been replaced by a small servomotor. The stock Ranger has manual steering through a steering box on the front axle with an upward-pointing lower steering shaft mounted to a universal joint at each end. The lower steering shaft connects to a large Faulhaber servomotor through a shock coupling. The actuator for the steering system is a planetary gear head driven by a brush-type servomotor with encoder feedback[11].

SOFTWARE

The highest level of software is the GPSINS Server which determines the vehicles coordinates. With the addition of the Omnistar High Performance subscription, the position readings are accurate to decimeter levels.

The Steer Server acts as the path planning control for the vehicle. Using the current location of the vehicle 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. Using this path, a turn radius and speed are calculated and sent to the Move Server as commands. We will discuss the details of the path planning in later sections.

The majority of the vehicle dynamics and error checking occur inside the Move Server. Limits such as pitch and acceleration rates are enforced through simple vehicle dynamics. These measurements come from the accelerometer included in the Crossbow AHRS unit[9]. The Move Server decides on ideal speed, gearing and curvature before issuing commands 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. The Direction Server controls the lower level hardware controllers attached to the Ranger's steering rack. The vehicle's steering box is actuated by a servo motor and the resulting motion is captured through the use of an optical encoder.

INTRODUCTION TO SPLINES

We present the basics of splines using the notation given by de Boor[4]. We start with a non-decreasing sequence of real numbers $\xi \triangleq (\xi_1 \ldots \xi_{l+1})$, that we will refer to as control points, a positive integer k and a sequence of l polynomials

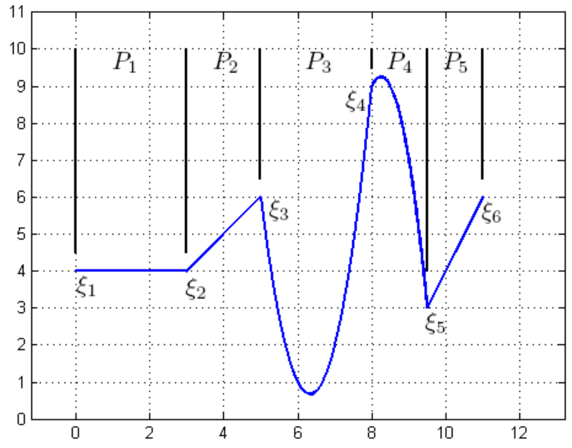

Fig. 2. A continuous, piecewise polynomial function.

$P_1, \ldots, P_l$, each of degree $< k$. We now define f (Figure 2) to be a piecewise polynomial(pp) function of order k

$$f(x) \triangleq P_i(x) \quad \text{for } x \in (\xi_i, \xi_{i+1}) \text{ for } i = 1, \ldots, l. \tag{1}$$

This equation defines the most general form of a spline and is often extended to include the entire real line, in which case the first and last polynomial is extended such that

$$f(x) \triangleq \begin{cases} P_1(x) & \text{if } x \le \xi_1, \\ P_l(x) & \text{if } \xi_{l+1} \le x. \end{cases} \tag{2}$$

Note that for f to be continuous, it must hold that $P_{i-1}(\xi_i) = P_i(\xi_i)$ for $i \in (1, \ldots, l)$.

We now define $\Pi_{k,\xi}$ to be the linear space of pp functions defined by the sequence ξ of polynomials of degree $< k$. A given polynomial P_i can be defined by its coefficients $(c_1, \ldots, c_k)$ or likewise by its right derivatives $D^{j-1}P_i(\xi_i)$ for $j = 1, \ldots, k$. The set $\Pi_{k,\xi}$ is therefore uniquely defined by

$$\Pi_{k,\xi} \triangleq \begin{cases} \text{integers } l \text{ and } k, \\ \text{control points } \xi, \\ \text{and derivatives } C_{ji} = D^{j-1}P_i(\xi_i) \\ \quad j = 1, \ldots, k \quad i = 1, \ldots, l. \end{cases} \tag{3}$$

In different applications it may be easier to work in an environment where derivatives are more easily computed. In other situations, one can continue to work in the coefficient environment which makes evaluation of f more efficient.

As a linear space, $\Pi_{k,\xi}$ can be defined by the basis elements $B_{i,k,\xi}$. Therefore we can define the Basis-form of f as

$$f \triangleq \begin{cases} \text{integers } k \text{ and } n = kl, \\ \text{control points } \xi, \\ \text{and coefficients } \alpha_i \text{ of } f \text{ for basis} \\ \quad B_{i,k,\xi} \text{ for } i = 1, \ldots, n. \end{cases} \tag{4}$$

In this way we can define piecewise polynomial functions through any set of cartesian points. In practice however, we usually wish to include additional constraints on the function.

HOMOGENEOUS CONDITIONS

In many applications splines are desired to be continuous and often have numerous continuous derivatives. We therefore define a subspace of $\Pi_{k,\xi}$

$$\Pi_{k,\xi,v} \triangleq \begin{cases} \Pi_{k,\xi} \\ \text{and constraints} \\ \quad D^{j-1}P_{i-1}(\xi_i) = D^{j-1}P_i(\xi_i) \\ \quad \text{for } j = 1,\ldots,v_i \text{ and } i = 2,\ldots,l \end{cases} \tag{5}$$

where v_i defines the number of continuous conditions at ξ_i. These conditions are enforced through the use of duplicate control points, or rather the absence of them. We construct a new sequence $(t_1,\ldots,t_n)$ where $n = kl - \Sigma_i v_i$ and each ξ_i appears in the sequence $k - v_i$ times. The Basis-form to include these constraints is as follows

$$f \triangleq \begin{cases} \text{integers } k \text{ and } n = kl - \Sigma_i v_i, \\ \text{sequence } (t_1,\ldots,t_n), \\ \text{and coefficients } \alpha_i \text{ of } f \text{ for basis} \\ \quad B_{i,k,\xi} \text{ for } i = 1,\ldots,n. \end{cases} \tag{6}$$

The coefficients α_i are calculated based on the desired output of f at the control points. Hence, given a sequence of cartesian (x_i, y_i), such that x is non-decreasing, we can construct a piecewise polynomial with continuous derivatives that passes through each pair. By letting $\xi = x$, and y describe the behavior of f, we can calculate α, t and subsequently f, which becomes our path.

In the context of path planning it is obvious that f needs to be continuous, or v must be at least $(1,\ldots,1)$. Furthermore, when considering vehicles such as the Overbot, which are unable to change heading, $D^1 f$, at a fixed position we must enforce that v is at least $(2,\ldots,2)$ in order to guarantee a feasible path. The additional restriction that v is at least $(3,\ldots,3)$ allows the vehicle to remain moving throughout its path. A discontinuity of $D^2 f$ at ξ_i would require a vehicle like the Overbot to stop and reposition its front wheels in order to alter its steering angle, $D^2 f$.

UNIQUENESS

We consider a simple case of constructing a spline through $l+1$ control points. Solving for the polynomials is equivalent to solving for the $k \cdot l$ coefficients for the l polynomials. Letting each interior control point be continuous through $D^{k-2}P_i$, yields $(k-2)(l-1)$ constraints. The end points of each polynomial introduce an additional $2l$ constraints. A further $k-2$ constraints on the derivatives at ξ_1 and ξ_{l+1} result in a total of $k \cdot l$ equations to solve for $k \cdot l$ variables yielding a unique solution. Therefore, by choosing cubic polynomials and constraining the spline to begin and end with a desired first derivative, we can guarantee a unique solution to our problem.

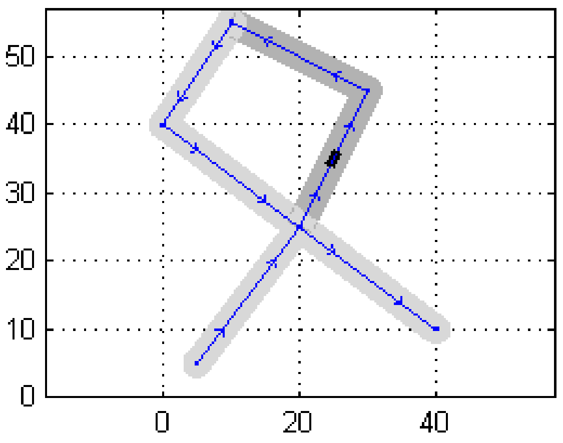

Fig. 3. GPS waypoints, corridor (light grey) and current segment (dark grey).

Furthermore, given a spline f_1, constructed with the above constraints through $(\xi_1,\ldots,\xi_{l+1})$, we also construct a spline f_2 with the same constraints, but passing through $(\xi_1,\ldots,\xi_i,\xi_{New},\xi_{i+1},\ldots,\xi_{l+1})$, where ξ_{New} lies on f_1 between ξ_i and ξ_{i+1}. Since f_1 satisfies the constraints for f_2, and because f_2 must be unique by construction, we can conclude that $f_2 \equiv f_1$ and the introduction of additional control points along the spline do not alter the curve.

PATH PLANNING

PATH OBJECTIVE AND COORDINATE SPACES

The Overbot was constructed to function according to the guidelines of the DARPA sponsored Grand Challenge. The vehicle, therefore, accepts a sequence of GPS waypoints used to define the high level mission. The vehicle's task is to traverse through each waypoint, in order, while staying within a given radius of the nominal straight line path (Figure 3). This radius defines a corridor bounding the vehicle's possible positions. Given a vehicle able to rotate in place, and an obstacle-free environment, the trivial path planning solution would be a straight line path. However, as this is rarely the case, we discuss path planning for situations when these conditions do not hold.

For a ground vehicle, such as the Overbot, all GPS waypoints are considered to lay on the surface of the Earth. For localized path planning, we assume a flat Earth model and therefore work in a cartesian plane with North and East axes. We refer to this plane as the GPS Frame. Due to the nature and defined sequence of the waypoints, there are no guarantees about the sequence of either North or East points, nor the uniqueness of pairs (Figure 3). We will impose the one constraint that no three consecutive waypoints can define an angle of 180° (*i.e.*: a path can not double back upon itself).

For the detection of obstacles, the Overbot includes various imaging devices such as a color camera and laser rangefinder. As all sensors are fixed on the vehicle, obstacles are mapped relative to the vehicle position and heading. Because the vehicle is constantly in motion, the obstacles are rotated and shifted into a constant reference frame which we refer to as the Map Frame.

INITIAL PATH SEGMENTS

The Overbot is a dynamic system with motion, control and sensor error as well as limited sensor range. It is naive to pre-plan the entire vehicle path from the outset. Thus, we replan the path from the the current position and heading, and using the most current obstacle map and do so in a receding horizon fashion. We also limit our path to only look ahead past the next waypoint. In this way, we help ensure the next turn will be feasible and once executed, a new path will be generated to maneuver the vehicle into a suitable position for the following turn. For this reason, we only work with three consecutive waypoints at a time, the most recently passed, $W_1^{GPS} = (x_1^{GPS}, y_1^{GPS})$ and the following two points, W_2^{GPS} and W_3^{GPS} (Figure 4).

In order to construct a spline through these points, we must guarantee that $(x_1^{GPS}, x_2^{GPS}, x_3^{GPS})$ is strictly increasing. We let W_M^{GPS} be the midpoint of the line segment between W_1^{GPS} and W_3^{GPS}. We now define the Path Frame to be the GPS Frame rotated and shifted such that W_i^{GPS} maps to W_i^{Path}, W_1^{Path} is at the origin, x_3^{Path} is positive, and the line segment between W_M^{Path} and W_2^{Path} is vertical (Figure 4). We constrain the waypoints to not double back, therefore $(x_1^{Path}, x_2^{Path}, x_3^{Path})$ is strictly increasing. Though we can now construct a spline using these points, we have found it useful to include additional points along the straight line path. The quantity and number of these points depend greatly on the relationship between the individual path lengths and the corridor width. By including additional points, the initial path is kept closer to the nominal straight line path and is therefore less likely to violate the corridor constraint(Figure 5). These points are only used for the initial path estimate and will not be rigid constraints in the final path. For demonstration, we have included points W_a^{Path} and W_b^{Path}, midpoints of the first and second path segments respectively.

At this point we can apply our spline function in order to generate an initial path. For the construction of $f(x)$, $l = 2$ is fixed while the degree k is free, though cubic functions have been found to produce smooth curves. We consider the vehicle's current position, W_V^{Path} and then the sequence $(t_1, \ldots, t_n)$ can be construct using $\xi = (x_V^{Path}, x_a^{Path}, x_2^{Path}, x_b^{Path}, x_3^{Path})$ and a suitable vector v as described above. The coefficients α are chosen based on the desired behavior given by y_i^{Path} and the initial heading C_V^{Path}, equal to the heading of the vehicle at W_V^{Path} and final heading C_F^{Path}, tangent to the second path segment. Figure 5 shows two splines generated for the current path segment, one using

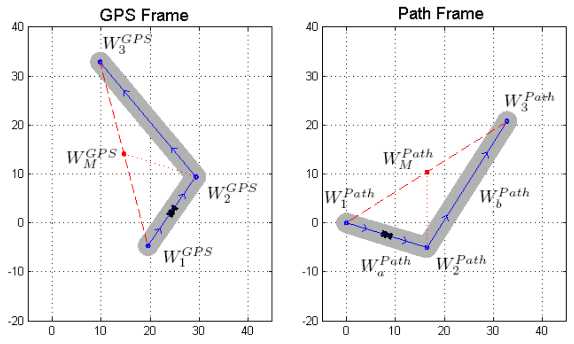

Fig. 4. Translation of path segment.

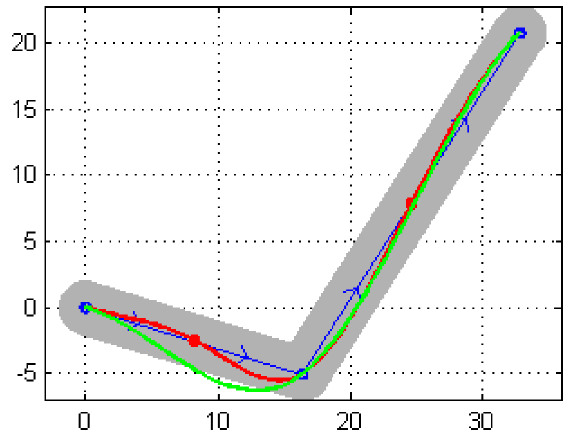

Fig. 5. The introduction of additional control points can keep the initial path from violating the corridor constraints. The green path uses only the GPS waypoints whereas the red path also includes the midpoints.

only the initial waypoints, and the other adding midpoints to each segment.

OBSTACLE CONSTRAINTS

DETECTING COLLISIONS

As described above, obstacles are mapped from various sensors into the Map Frame, which maintains a constant rotation and shift relative to the GPS Frame. The map is stored as a pixelated map, each pixel denoting the presence or absence of an obstacle (binary), or the severity of the terrain as required by the mission. To check our path against the map, we define a transformation $z^{Map} = H(z^{Path})$ to be the needed rotation, shift and scaling required to translate from the Path Frame to the Map Frame. Therefore our path in the Map Frame become $H \cdot f$ evaluated over $[z_V^{Path}, z_3^{Path}]$.

To evaluate our curve against the map, we implement a modified variant of Bresenham's line drawing algorithm. Originally developed to control plotters, Bresenham's algorithm reproduces lines and curves in a pixelized environment and is a popular tool in computer graphics[2]. The converted spline can be evaluated at each pixel and compared to the map to

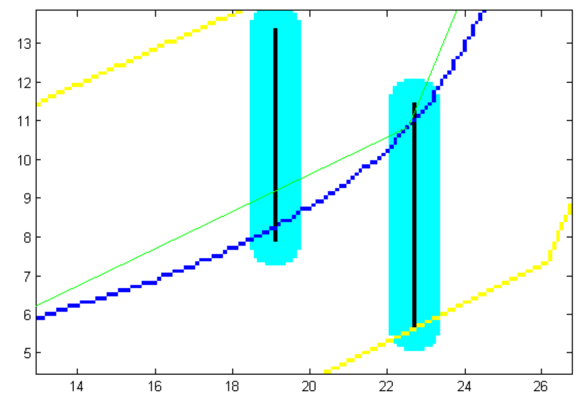

Fig. 6. The co-existance of a path and obstacle pixel results in the detection of a collision

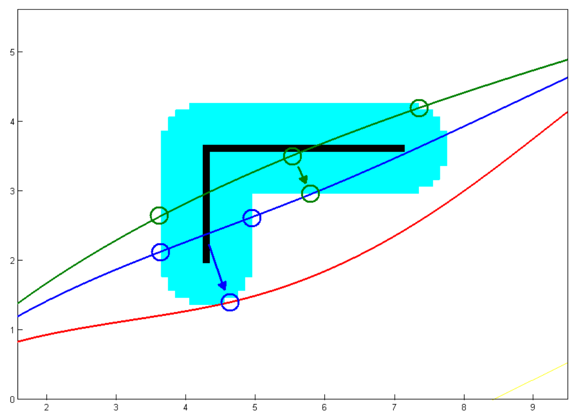

Fig. 7. The midpoint of the collision is moved out until a clear path is found.

detect a collision. It should be noted that each obstacle must be inflated by at least half the width of the vehicle to guarantee that a collision does not occur. One could also define a corridor having the curve of the spline but the width of the vehicle. Using this method, one would have to then evaluate every pixel within the corridor instead of the single pixel path. The faster solution would rely on the relationship between path length and the size and density of the obstacles. In this work, we inflate our obstacles within the map. The co-location of a path and obstacle pixel indicates a collision (Figure 6), which is stored for later use. The path is also compared to the corridor constraint and any violations are treated as an obstacle collision.

PATH MANIPULATION

Once it has been determined that the path collides with an obstacle, we manipulate the spline to avoid that obstacle. As multiple collision may have occurred, we first edit the list of collision pixels to include only the first collision. The first and last pixels of this collision define the entry and exit points of the collision, C_{Entry}^{Map} and C_{Exit}^{Map}. We add an additional control point to our spline to guide the path around the obstacle. We use the midpoint, $C_{Adj}^{Map} = (C_{Entry}^{Map} + C_{Exit}^{Map})/2$ as our point of manipulation. We move C_{Adj}^{Map} perpendicular to the line $[C_{Entry}^{Map} C_{Exit}^{Map}]$ by a distance of approximately one pixel per step. The point is continually updated and checked against the map until C_{Adj}^{Map} is free of collision. This means that a path through C_{Adj}^{Map} will no longer collide with our obstacle at that point. C_{Adj}^{Map} is mapped into C_{Adj}^{Path} and added to the list of control points. A new spline is computed through these points still having our desired characteristics but also passing through the new point (Figure 7).

Each new spline is mapped back into the Map Frame and the process is repeated. If the new spline still intersects the obstacle, the collision will be shorter and closer to the edge of the obstacle. Thus the algorithm will continue to displace the spline around the remaining portion of the obstacle. Because the points are fit with cubic functions, along with the constraint that x_i^{Path} be strictly increasing, a large number of control points within close proximity to each other can cause large deviations in the path. To avoid this, we include a simple adjustment to the control points; when adding a new point, C_{Adj}^{Path}, we remove any other control points within a given radius of C_{Adj}^{Path}. In essence, when an adjustment fails to avoid the obstacle, the new point replaces the old one. All points, except the initial position, are removable. Deletion of the endpoint is acceptable as the path will be replanned after the next way point, and the second segment of the three waypoint set will never be fully executed.

There are, however, instances when both adjacent control points may be necessary. It is easy in these cases to detect the cyclic insertion and deletion of points and the radius can be adjusted to allow for both points. Another method, which we use, to determine when to keep the adjacent control points is as follows. Though C_{Adj}^{Map} is moved in increments of approximately one pixel, we also enforce a minimum radius for the first adjustment. This addresses situation when the spline is moved away from one section of obstacle and onto another. If the adjustment for the second segment is to move the curve back, a cycle will develop (see Figure 8). By detecting these cycles, and increasing the minimum adjustment radius, we take larger and larger steps away from the obstacle, eventually breaking the cycle. In this way we can overcome the irregularities inherent in certain obstacles.

Once the final obstacle-free spline is found, it can be mapped back into the GPS frame to be used by the vehicle. The path, as defined by the spline, ensures safe passage of the vehicle through its environment and contains all necessary information. The steering angle can be derived from the second derivative of the path. Feedback is applied to keep the vehicle on the spline path.

DETECTING INFEASIBLE PATHS

It is important to detect situations in which this algorithm will not converge onto a feasible path. The cyclic behav-

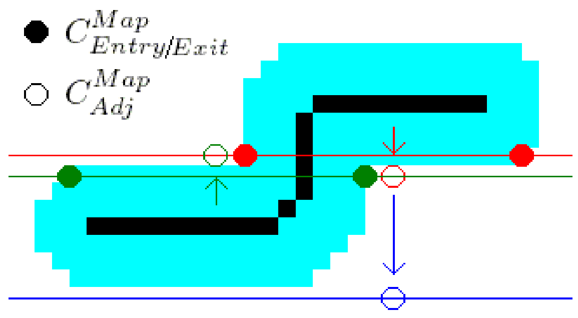

Fig. 8. Some obstacles can cause adjustments to oscillate, red to green, so an large minimum adjustment may be necessary to find a feasible path, blue.

ior described above may be an indication of an inability to converge. One method of improving this behavior is by increasing the distance of the first adjustment, as opposed to starting closer to the obstacle. Should this approach also fail to generate a feasible path, further improvement may be gained by decreasing the radius applied to the removal of old points, thus increasing the number and density of new control points on the spline. This has the effect of allowing a much tighter radius of curvature for the path, which may be required to navigate around the obstacles. Additional points will more tightly control the shape of the curve but the additional constraints will force the path into sharper curves. As such, we enforce a lower bound on the proximity radius to avoid undesired and unnavigable path characteristics (such as a curvature too large for the vehicle steering). Increases in initial adjustment followed by decreases in proximity without convergence suggests an inability of the algorithm to generate a feasible path.

Simpler cases of infeasibility can also be detected. Moving C^{Map}_{Adj} across the entire width of the corridor without finding a clear area indicates the presence of path blocking obstacle. Similarly, if the adjustment process moves the spline along a path that is sliding along an angled obstacle, continued collision of neighboring pixels may indicate an impassable obstacle. In order to deal with these situations, limits are enforced on the initial adjustment radius, the proximity radius and the overall number of iterations that the spline algorithm is allowed. When no feasible path is found, other methods can be used to find localized solutions in a difficult area, assuming such a feasible solution is possible.

CONCLUSIONS

We have presented a method for solving path planning problems by bending smooth cubic splines to avoid obstacles collisions. We have developed a simple method for generating maneuverable paths through potentially complex environments (Figure 9). In the context of real time systems, this approach is designed to run efficiently by converging toward feasible solutions instead of investigating the entire path space. The use of cubic splines guarantees continuous vehicle heading and turning angle which is ideal for many ground based vehicles.

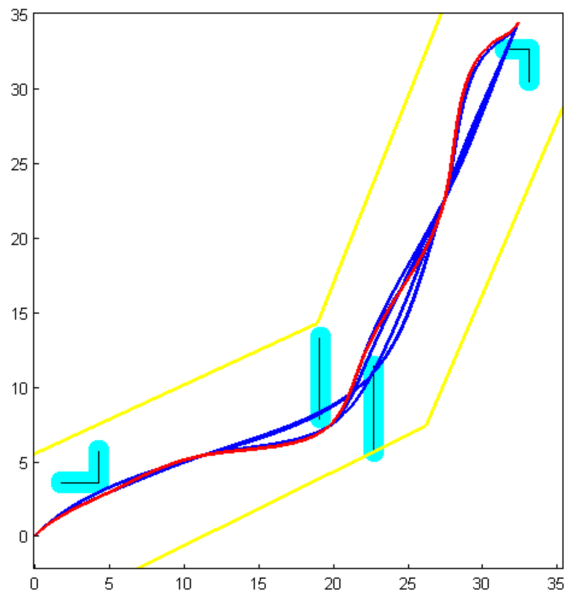

Fig. 9. We iteratively generate smooth curves(blue) bending them around obstacles until a collision free path(red) is found.

FUTURE WORK

It has not been proven that the algorithm can guarantee a feasible path if one exists. We plan further work to investigate this topic and make the method more robust in difficult environments. It can be noted that a heading can also be constrained at any point on the curve by simply breaking the curve into two sections, thereby allowing the new endpoint to have first derivative constraints. This feature makes it possible then to not only force the vehicle through a narrow opening, but control its direction through that opening. Doing so, however, loses the continuity of the second derivative at that points, and thus requires a vehicle stop to readjust its steering angle. Note that in practice, the control system feedback would simply treat the discontinuity as a plant disturbance, and regulate to the path as required. We also plan to explore the effects of moving obstacles using this method; there are interesting tradeoffs that develop due to the relationships of vehicle and the dynamic obstacle speed (note that refresh rates of both the obstacle map and the path itself become relevant with these tradeoffs).

Finally, experimental validation of this algorithm will be implemented on the real time OS of the Overbot to perform its path planning. As stated previously, planning past the next two waypoints is wasteful, unless the waypoints are very closely spaced. As the vehicle passes the first of these two waypoints, a new path is generated from its current position and proceeding two more waypoints ahead. The portion between the first and second waypoints may or may not remain the same, but

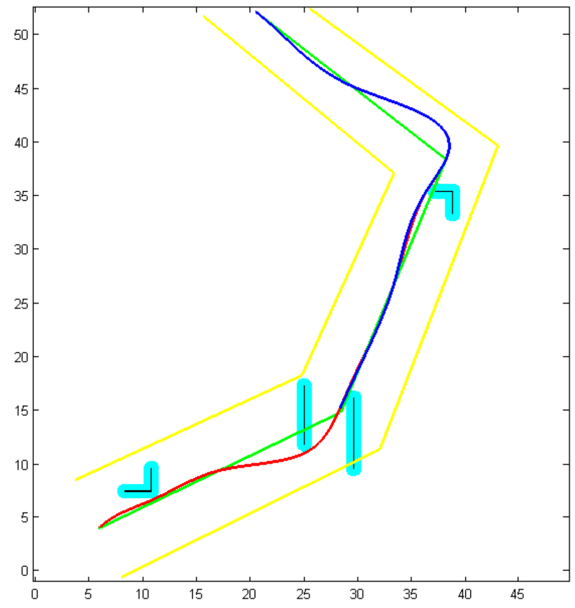

Fig. 10. The vehicle follows the first path(red) until the first turn, then plans a new path(blue).

will now compensate for any changes that need to be made to navigate around the second waypoint as opposed to arriving there (Figure 10). Furthermore, continually replanning along the segments can help compensate for positioning errors and allows for the discovery of new obstacle information.

ACKNOWLEDGEMENTS

We would like to thank Team Overbot for all the time and energy they put into the construction of the Overbot, and John Nagle for his generosity and continuing support of the vehicle and project. We would also like to thank everyone in the Autonomous Systems Lab for their time and support.

REFERENCES

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

[2] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, p. 25, 1965.

[3] C. T. Incorporated, "Ahrs400 series users manual," 2005, www.xbow.com/Support/Support_pdf_files/AHRS400_Series_Users_Manual.pdf.

[4] C. de Boor, *A Practical Guide to Splines*, revised ed. Springer, 1978.

[5] G. Elkaim, J. Connors, and J. Nagle, "The overbot: An off-road autonomous ground vehicle testbed," *Proceedings of the ION Global Navigation Satellite Systems Conference*, 2006.

[6] G. M. Control, "Dmc-14x5/6 user manual," 2005, www.galilmc.com/support/manuals/man14x5_16.pdf.

[7] Y. Kanayama and B. I. Hartman, "Smooth local path planning for autonomous vehicles," *International Journal of robotics Research*, vol. 16, no. 3, p. 263, 1997.

[8] P. M. Kinney, M. Dooner, J. Nagel, and P. G. Trepagnier, "Kat-5: Systems based on a successful paradigm for the development of autonomous ground vehicles," *Position, Location, and Navigation Symposium*, p. 378, 2006.

[9] K. H. Lim, "Position estimation for team overbot," *Stanford University*, 2003.

[10] N. Incorporated, "Novatel propak-lb plus user manual," 2005, www.novatel.com/Documents/Manuals/om-20000046.pdf,.

[11] J. Nagle, A. Reddy, and K. H. Lim, "Team overbot," www.overot.com.

[12] P. Industries, www.polarisindustries.com.

[13] Z. shiller and Y.-R. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, p. 241, 1991.

[14] S. Incorporated, "Laser measurement systems technical description," 2003, www.mysick.com/saqqara/im0012759.pdf.

[15] S. Thompson and S. Kagami, "Continous curvature trajectory generation with obstacle avoidance for car-like robots," *Proceedings of the 2005 International Conference on Computaional Intelligence for Modelling, Control and Automation and International Intelligent Agents, Web Technologies and Internet Commerce*, 2005.

[16] U. Incorporated, "Fire-i 400 industrial camera technical specifications," www.unibrain.com/Products/VisionImg/tSpec_Fire_i_Industrial.htm.

[17] C.-H. Wang and J.-G. Horng, "Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic b-spline functions," *IEEE Transactions on Automatic Control*, vol. 35, no. 5, p. 573, 1990.

[18] A. R. Willms and S. X. Yang, "An efficient dynamic system for real-time robot-path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 36, no. 4, p. 755, 2006.