A Guidance Algorithm for Unmanned Surface Vehicle Exhibiting Sternward Motion

Shu Du

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Daniel J. Stilwell, Chair
Craig A. Woolsey
Scott M. Bailey

September 27, 2013
Blacksburg, Virginia

Keywords: unmanned surface vehicle, guidance, sterward motion, backward path planning

A Guidance Algorithm for an Unmanned Surface Vehicle Exhibiting Sternward Motion

Shu Du

(ABSTRACT)

We propose a new dynamically feasible trajectory generation algorithm that incorporates sternward motion for unmanned surface vehicles. This work is motivated by riverine applications where the operating environment is large and poorly known. We extend a navigation approach for forward path planning into a more versatile framework that includes safe and dynamically feasible backward trajectories. We pose the backward trajectory generation problem as a finite-horizon optimal control problem and transform it into a nonlinear programming problem by utilizing the direct shooting method. The nonlinear programming problem is solved using the Hooke-Jeeves numerical algorithm. We provide successful simulation and field-trial results that demonstrate the performance of backward path planning algorithm.

Acknowledgments

I would like to send my greatest gratitude to my advisor Dr. Daniel Stilwell for his generous guidance and encouragement throughout my research. Additionally, I am grateful to my thesis committee members, Dr. Craig Woolsey and Dr. Scott Bailey for their invaluable advices and feedbacks.

My sincere gratitude also goes to the USV project leader, Dr. Aditya Gadre for his enormous help on this project. He has introduced me into a new world of robotics research. He is always more than willing to help whenever I have questions on path planning.

I also want to thank my parents, my grandparents, and my uncle for their constant support and encouragement. This work would not have been done without their love.

Thanks to other members in Autonomous Systems and Control Lab. It is a great pleasure knowing these nice people and working with them.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Contribution

It is a significant challenge to develop guidance algorithms for driving unmanned surface vehicles (USVs) to accomplish missions autonomously in dynamic and unstructured riverine environments. We assume that any *a-priori* environmental information is incomplete or inaccurate. Waterways may be narrow, non-uniform in depth, and even blocked. Occasionally this can lead to situations that USV may not be able to find a safe and dynamically feasible trajectory that consists of only forward motion. Thus the capability to plan safe backward motion is necessary for autonomous operations.

The main contribution of this work is the development of a fast and reliable backward planning algorithm that generates dynamically feasible trajectories in real time. The proposed algorithm extends the path planning approach that currently solves the forward planning problem [1]. We approach the backward planning problem as a finite horizon optimal control problem (OCP). We utilize direct shooting method to transform the OCP to a nonlinear programming problem which is then numerically solved by Hooke-Jeeves algorithm. Additionally, we introduce a straightforward switch scheme to switch the path planner between

forward and backward planning. The backward planning mode is activated whenever there does not exist a safe forward vehicle path. The path planner will continue generating backward paths until the USV reaches a vehicle state along the backward trajectory from which a dynamically feasible and safe forward path can be computed. The proposed backward planning algorithm has been successfully deployed on the VT USV.

1.2 Background

Over the past decade, USVs have drawn widespread attentions from academic labs, corporations and government. USVs have been increasingly deployed due to their unique characteristics. Currently, people can find USVs active in hydrological and environmental surveys, exploration in riverine and oceanic environments, and defense missions. An autonomous surface craft (ASC) was developed to collect bathymetry data in late 1990s at the MIT. It successfully accomplished a hydrographic survey in Boston Harbor after completing several field tests [2]. A USV named Owl MK II was developed based on a jet ski chassis equipped with a low-profile hull for stealth and payload capability. Its advanced version has been used for port security [3].

Riverine waterways can be inaccurate due to a variety of reasons, including shifting bathymetry, other vessels and floating debris, variable water levels and flow-rates. Such lack of precise knowledge of the operating environments poses demanding requirements for developing an autonomy framework combining sensing, guidance and control strategies. With respect to guidance algorithms specifically, it should consistently provide USVs with safe and dynamically feasible paths that lead to a desired location. Not only should vehicle paths satisfy dynamic constraints of the vehicle, they also need to be produced rapidly and within the real-time computational limits.

We choose to partition path planning algorithms into two classes, local planning and global planning. Local planning computes dynamically feasible paths within a very short planning

horizon in real-time [4][5][6], whereas global planning generates kinematic plans that can be exceptionally long. Concerning global planning, some approaches perform a sequential search of a graph whose edges consists of line segments, arcs, clothoids [7][8]. Recently [9] and [10] innovatively exploit the solution of level-set methods to compute an optimal global path which can be updated in the real-time manner. Combinations of these two categories takes advantage of the efficiently fast obstacle avoidance solution from the local planning, as well as optimal performance guarantees from the global planning. Zhang, et al. , in [11] present a path planning strategy that uses a combination of global planning and local planning. The global path planner provides a sequence of sub-goal-points used to guide the local planning. A similar approach is proposed in [12], where a robot navigates using precise local metric maps while a global plan is computed using a topological graph.

1.3 Related Work

The pioneering work to compute optimal paths including reverse motion was done by Reeds and Shepp [13]. They investigated the variant of Dubin's car with a minimum turning radius, which can move forward and backward in the absence of obstacles.

However, the literature for path planning involving backward motion, which is closely related to our application, is very limited. Most prior work is focused on unmanned ground vehicle platforms and addresses autonomous parking problems. In [14], a ground vehicle maneuvered using time-optimal trajectories that possessed both forward and backward segments. The algorithm uses different maneuvers to handle paths that may require both forward and reverse motions. The OTG (Optimal Trajectory Generation) solver is used to produce multi-maneuver trajectories by solving optimal control problems [14]. Similarly, the parking problem is considered as specialization of general optimal control problem in [15]. The initial computation of potential field allows collision avoidance, which introduces the possibility to compute minimum-time solutions for various complex parking configurations [15]. A parking

assistance system is developed by searching a database for the elementary movements that will make the complete parking maneuver in [16]. These approaches described in this section either suffer from relatively long computation time, or are only suitable for kinematic model which is not sufficient to model motions of an unmanned surface vehicle.

1.4 Organization

This thesis is organized as follows. We introduce a search algorithm and an optimization algorithm which we will apply to our proposed approach in Chapter 2. Chapter 3 presents the backward path planning for USV in detail after a brief introduction of currently used forward path planning method. We choose a cove on Claytor Lake in Virginia as the testbed, and we demonstrate that the results from both simulation and field trials validate the proposed backward path planning algorithm in Chapter 4. Finally, we present conclusions and potential research topics in Chapter 5.

Chapter 2

Preliminaries

In this chapter, a few algorithms and approaches involved in our proposed backward planning algorithm are presented briefly for the sake of clarification in the following chapters. We first introduce the search algorithm, hybrid-state A*, which is used to compute vehicle paths. Then a short survey on optimization algorithms is presented. Finally, the Hooke-Jeeves algorithm, a numerical solver to nonlinear programming problems, is discussed.

2.1 Search Algorithms

A* is a well-known search algorithm that finds a minimum cost path through a map discretized to uniform-sized cells. The optimality of A* in the sense that it expands the smallest number of nodes(cells) necessary to guarantee finding an minimum cost solution is shown in [8]. However, A* is only limited to piecewise-linear paths that are not executable by vehicles in practice. [17] proposes a variant of conventional A* algorithm that is named hybrid-state A*. It is essentially a heuristic search in continuous vehicle coordinates. Instead of only considering the centers of grid cells in traditional A*, hybrid-state A* associates with each grid cell a continuous vehicle state. During the expansion of a node, several control actions are

applied on the vehicle model to generate children states. The children states will be assigned to cells into which they will fall. The result of this expansion is that there exists an actual control action, hence a vehicle path, between each node on final path. Even though it is not assured to find the minimum cost solution using hybrid-state A*, it always finds a solution that lies in the neighborhood of the global minimum [18]. In our proposed backward path planning approach, we seek to minimize the length of forward paths, which are produced by using hybrid-state A*, starting anywhere along the backward path.

2.2 Trajectory Optimizations

Trajectory optimization problems can be partitioned into two classes, indirect and direct methods [19]. Indirect methods require an analytic expression for the optimal necessary conditions that involve the adjoint differential equations, the maximum principle, and associated boundary conditions. Indirect methods for optimizing a function with multiple variables often require an analytic expression for the gradient of the function that can be used to locate a set of variables using a root-finding algorithm. However, it is difficult to develop analytic expressions of the necessary conditions for complex nonlinear dynamics. The region of convergence for a root-finding algorithm may be small when dealing with adjoint variables that have no obvious physical interpretations. In contrast, direct methods directly adjust the state and control variables without requiring an analytic expression for the necessary conditions and initial guesses for the adjoint variables.

Two discretization schemes are often used in direct methods: the direct shooting method that only parameterizes the control functions, and the direct collocation method that parameterizes both control and state variables. The most common approach (direct shooting method) for converting an optimal control problem to a nonlinear programming problem is to choose an appropriate structure with finitely many parameters to approximate the control variables. This reduces the dimension of the control variables space, and the state variables

are obtained by solving the initial value problem.

We apply a direct shooting method in order to transform the optimal control problem, which formulates the backward path planning problem, into a nonlinear programming problem. Nonlinear programming is the name of an approach to solve an optimization problem defined by a collection of constraints and an objective function. Some of the constraints or the objective function are nonlinear [20]. We use piecewise constant functions rather than a piecewise Hermite approximation [21] to approximate the control variable. This reduces the computational load and improves performance of the proposed algorithm in real-time applications.

2.3 Numerical Solution to Trajectory Optimizations

Pattern search is a widely-used class of direct search methods for nonlinear optimization. The development of pattern search can be traced back to around 1960. Several features make it useful even today among a variety of numerical optimization algorithms. It has been proved in practice that pattern search provides faster solutions for some problems that classical methods can solve, and even some problems classical methods cannot solve. It does not require explicit computation or estimate of derivatives. So it is reasonably straightforward to implement and very useful for problems where derivatives or finite-difference derivatives are not available [22][23].

The Hooke-Jeeves algorithm is a pattern search methods that utilizes a series of exploratory moves accessing the behaviors of the objective function [24]. Suppose we seek argument $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ that minimizes a function $f(x) : \mathbb{R}^n \to \mathbb{R}$. The Hooke-Jeeves algorithm utilizes two types of adjustments on x. The first one is called the exploratory move for acquiring knowledge about the behavior of $f(x)$. Components of the variable are changed individually, i.e. $x_i \pm \Delta$, in every iteration. The second type is a pattern move in which the objective function is minimized by using the information (pattern) obtained in

exploratory moves. In each iteration, a pattern move follows a successful exploratory move. If an exploratory move fails to establish a pattern, the step size Δ is reduced, and then a new exploratory move will be tried in order to search new patterns. The algorithm will finally terminate when the step size is sufficiently small.

Due to the complicated objective function and nonlinear dynamics of the USV system, we choose the Hooke-Jeeves algorithm for computing a numerical solution of the nonlinear programming problem that was transformed from the original optimal control problem.

Chapter 3

Backward Path Planning for USV

In this chapter, we present our backward path planning algorithm in detail. This proposed guidance algorithm extends the forward path planning approach shown in [1] to address both forward and backward motion. A novel switch scheme is proposed to switch between forward and backward path planning. We first briefly introduce important details of the approach in [1] for the sake of clarity.

3.1 Forward Path Planning

The problem of planning forward vehicle trajectories over very large riverine areas that are potentially poorly known *a priori* was addressed using a receding horizon control strategy in [1]. We assume that the vehicle operates in an environment $\Omega \subset \mathbb{R}^2$, and the large size of Ω prohibits the use of occupancy grid maps for representing global information. In [25], a hybrid map structure for autonomous vehicle navigation is proposed. The map structure utilizes a topological map that is extracted from a binary map offline. The vehicle continuously maintains and updates a small local area around itself within the map which is represented by an occupancy grid map.

Presume that the vehicle dynamics are

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t_i) = x_0, \quad t \geqslant t_i \tag{3.1}$$

where $x(t) \in \mathbb{R}^n$ is the vehicle state, $u(t) \in \mathbb{R}^m$ is the control signal in the set of all feasible control actions $\mathbb{U}$. For the interval $t \in [t_i, t_i + T]$, we compute a sequence of control signals that minimizes the objective function

$$J_T(t_i, x_0) = \min_{u \in \mathbb{U}} \int_{t_i}^{t_i+T} q(\tau, x(\tau), u(\tau))d\tau + W(x(t_i + T)) \tag{3.2}$$

subject to the vehicle dynamics in (3.1). The integrand $q(t, x(t), u(t))$ is the integral cost calculated using traversal cost assigned to grid cells in local occupancy map. The terminal cost $W(x(t_i+T))$ is computed based on the global topological roadmap as in [25]. We denote $u_i^*(t)$ with the support $[t_i, t_i + T]$ to be a control signal that minimizes (3.2) and $x_i^*(t)$ to be the corresponding state trajectory. Using a standard receding horizon control strategy, $u_i^*(t)$ is implemented within $[t_i, t_i + \delta]$, where $\delta < T$. The optimal control problem (3.2) is solved again at $t_i + \delta$.

The convergence of the forward trajectory generation algorithm within the receding horizon control framework is guaranteed if a formal test, called the matching condition, is satisfied [1]. The matching condition

$$W(x(t_i + T)) - W(x(t_i + T - \delta)) \leqslant -\int_{t_i+T-\delta}^{t_i+T} q(\tau, x(\tau), u(\tau))d\tau \tag{3.3}$$

is used to indicate if there is a mismatch between the global roadmap and the computed local trajectory. The global roadmap is repaired when (3.3) fails.

The optimal control problem (3.2) is transformed into a nonlinear programming problem by applying a direct shooting method. We employ the Hooke-Jeeves algorithm to solve the nonlinear programming problem. It has been verified that significant performance gains can be achieved by seeding the Hooke-Jeeves algorithm with good initial guess. In [26], hybrid-state A* computes an initial guess of control signals whose corresponding state trajectory satisfies vehicle dynamics.

3.2 Backward Path Planning

In this section, we discuss the details of our proposed guidance algorithm for backward path planning. We consider backward path planning as a distinct problem, and solve it as a single optimal control problem. We presume that simply driving the USV straight-backward is too risky. Therefore it is essential to develop an intelligent way to guide USV backward as little as possible so that it can resume the mission with regular forward motions as quickly as possible.

The goal of developing backward path planning is to compute a dynamically feasible backward trajectory that will move the USV to a state from which a safe forward path can be generated. Thus, we formulate the problem as an optimal control problem

$$J_{T_b}(t_i, x(t_i)) = \min_{u \in \mathbb{U}} \left\{ \underset{t \in [t_i, t_i + T_b]}{\text{minimum}} \; l_{A^*} \right\} \tag{3.4}$$

considering vehicle dynamics for backward motion

$$\dot{x}(t) = f_b(x(t), u(t)) \tag{3.5}$$

where T_b is the backward path planning horizon and l_{A^*} is the length of the hybrid-state A* path computed at $x(t)$. We seek to minimize the length of the forward path computed using the hybrid-state A* algorithm, starting anywhere along the backward trajectory. In practice, the hybrid-state A* algorithm renders a safe and feasible forward path, and this path stays in a small neighborhood of the optimal path. Note that we utilize it as an initial guess for the forward trajectory optimization. The reason we choose to minimize the length of hybrid-state A* paths in backward path planning is that we intent to generate an optimal backward path by taking into account forward path planning information. Fig. 3.1 shows an illustration on computation of hybrid A* paths from different vehicle states along backward path.

We begin transforming the optimal control problem (3.4) into a nonlinear programming problem by applying the direct shooting method. This conversion is accomplished by param-

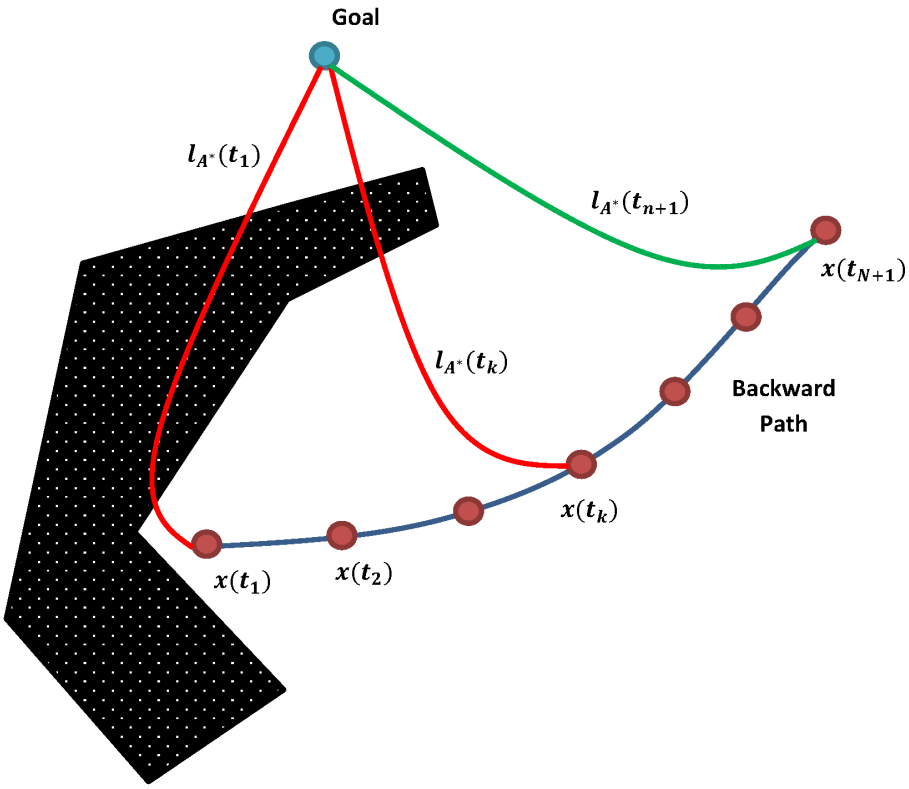

Figure 3.1: Computation of hybrid A* paths along backward path

eterization of the control signal $u(t)$. We discretize the backward planning horizon $[t_i, t_i + T_b]$ into $N-1$ subintervals,

$$t_i = t_1 < t_2 < \cdots < t_{N-1} < t_N = t_i + T_b \tag{3.6}$$

Instead of interpolating the control signal in the intervals by cubic splines, we approximate it to be piecewise constant in each subintervals in order to improve the computational performance. Thus the parameter vector Y of the nonlinear program consists of values of control variables at the discrete time instances t_k, $k = 1, \ldots, N$ in (3.6)

$$Y = [u(t_1),\, u(t_2),\, \ldots,\, u(t_N)] \tag{3.7}$$

for which the corresponding state trajectory is

$$X = [x(t_1),\, x(t_2),\, \ldots,\, x(t_{N+1})] \tag{3.8}$$

Given a parameter vector Y, we can compute the state trajectory vector X by solving the initial value problem within $[t_i, t_i + T_b]$, which enables us to re-formulate (3.4) and (3.5) in the form of nonlinear programming problem

$$g(Y) \tag{3.9}$$

subject to the constraints

$$c(Y) \tag{3.10}$$

and bounds

$$Y \in \mathbb{U} \tag{3.11}$$

Here we use $Y \in \mathbb{U}$ to indicate each $u(t_i) \in \mathbb{U}$. The nonlinear programming problem is to find the n-dimensional vector Y to minimize (3.9) subject to (3.10) and (3.11). Also note that the constraint in (3.5) is fully satisfied by solving the initial value problem, which ensures the feasibility of the control signal and the backward state trajectory.

Rather than using a nonlinear programming problem solver, we employ the Hooke-Jeeves algorithm to solve the nonlinear programming problem. Pattern search technique employed by the Hooke-Jeeves algorithm does not require the gradient of the objective function (3.9) and thus does not impose any smoothness conditions on it. It works by perturbing parameter vector Y and comparing the evaluations of the objective function for different values of Y to determine the pattern of reduction. With respect to calculating the objective function, it computes the lengths of the hybrid-state A* paths starting from $x(t_1)$, $x(t_2)$, $\ldots$, $x(t_{N+1})$ and picks the shortest one. In each iteration, the exploratory moves are implemented by sequential modifications on each coordinate of Y. It keeps the modifications which cause decrease in (3.9), denoted as $\Delta = [\Delta(t_1), \Delta(t_2), \ldots, \Delta(t_N)]$, and passes Δ to the pattern move where the algorithm makes real changes on Y, i.e. $Y = Y + 2\Delta$. The algorithm continues to modify parameter vector Y until it no longer can find a pattern that results in

a decrease of (3.9). The final parameter vector is the optimal solution given the best effort that Hooke-Jeeves algorithm makes,

$$Y^* = [u^*(t_1),\, u^*(t_2),\, \ldots,\, u^*(t_N)] \tag{3.12}$$

Because we consider and solve forward and backward path planning separately, we utilize a simple switching scheme to switch between them. In every iteration, the path planner first tries to compute a hybrid-state A* path initiating from the current vehicle state. If the hybrid-state A* algorithm fails to generate a path, which indicates the potential forward path passes over or too close to shoreline or obstacles, then the path planner switches to backward path planning mode. The USV moves astern along the generated backward path. The path planner switches back to forward path planning as soon as a hybrid-state A* path can be computed from the current vehicle state. Figure. 3.2 illustrates the integrated path planning framework.

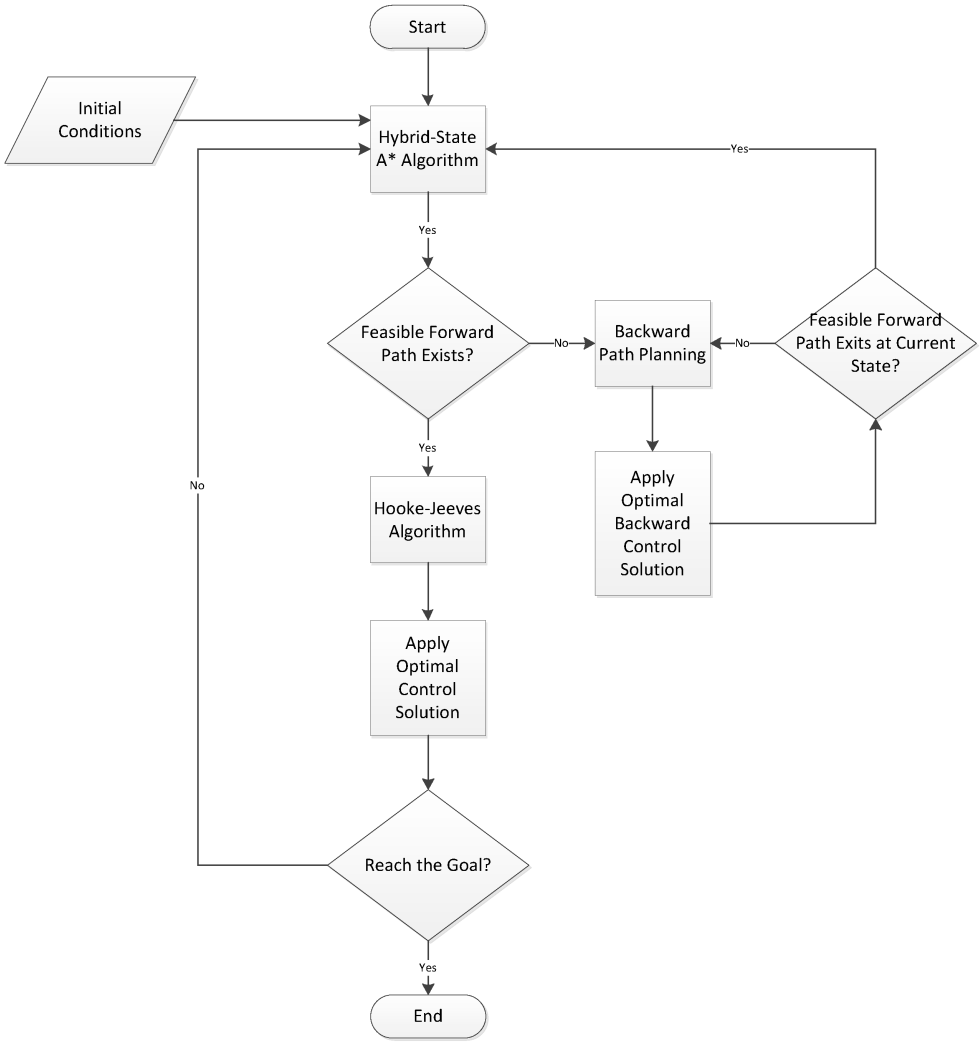

Figure 3.2: Integrated path planning framework

Chapter 4

Results

In this chapter, we present simulation results of the proposed backward path planning algorithm in a narrow cove. To demonstrate the capability of the backward path planning, the initial pose of the USV is such that the path planner cannot generate a safe and dynamically feasible forward path. We then present the result of a field trial performed in a cove in Claytor Lake in Dublin, VA. The result depicts that forward and backward path planning works seamlessly within the path planning framework.

4.1 Simulation Result

Fig. 4.1 shows an occupancy map of a cove where the USV operates autonomously. The grid cells are size of $1\,m \times 1\,m$ and we assume that the map is static and accurate. The white area represents traversable waterway. The vehicle dynamics for sternward motions used in (3.5) is

$$\begin{pmatrix} \dot{e} \\ \dot{n} \\ \dot{\psi} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} v\cos\psi \\ v\sin\psi \\ r \\ -\frac{1}{\tau_r}r + \frac{K_r}{\tau_r}\delta \end{pmatrix} \tag{4.1}$$

where e and n are the Cartesian coordinates of the vehicle position, representing easting and northing positions respectively, ψ is the heading, r is the turn rate, v is the vehicle speed, δ is the rudder angle as well as the control input. The time-invariant parameters K_r and τ_r take values from the results in system identification [27]. We limit backward path planning to low vehicle speed for safety, i.e. $v = 1\,m/s$. The backward path planning horizon is set to $T_b = 20s$.

Fig. 4.2 shows a close-up image of the backward path planning. Note that the triangles representing the USV are to scale. The USV starts very close to a shoreline and is required to reach the goal (red circle). Being too close to the shore, the path planner fails to find a feasible forward path (red triangle), and then computes a backward path (red line) by switching to backward path planning. A safe and dynamically feasible forward path (blue dot line) is generated at the end of backward trajectory (green triangle).

4.2 Field Trial Result

Field experiments involving environment mapping, trajectory planning, and control strategies calls for an autonomous vessel platform. The vehicle needs to be equipped with capabilities of sensing and navigation, as well as software-enabled steering and propulsion control. The Autonomous Systems and Controls Laboratory's USV was built based on a rigid hull inflatable Ribcraft 4.8. The vehicle has a 50 hp four-stroke Honda outboard engine which drives the vessel up to 22 knots.

Vehicle speed is controlled by throttle commands coming from an electronic throttle lever for manual control or an analog control interface that receives an analog voltage from 0 to 5V for computer control. A Glendinning Smart Actuator accepts throttle commands and responds by actuating the two metal cables connecting the gear and throttle levers of the outboard engine.

The steering system is achieved by installing a M81120 Raymarine hydraulic pump with a

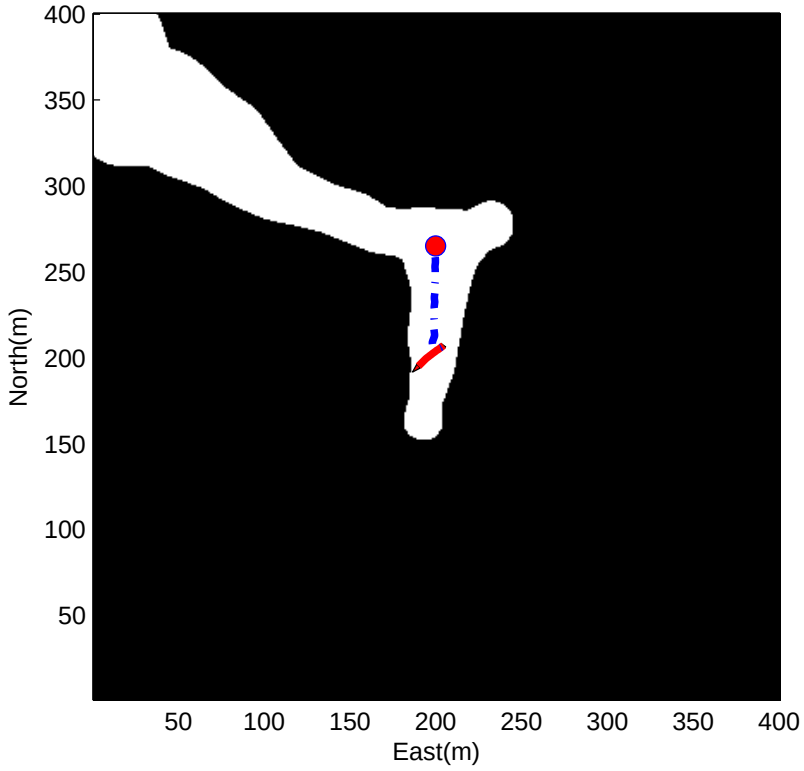

Figure 4.1: Occupancy map of a cove

RoboteQ AX500 motor controller that receives control signals from on-board computer. A Raymarine position sensor provides rudder position feedback allowing closed loop outboard orientation using the motor controller's built-in PID controller.

An Ibeo Alasca laser scanner is used to detect obstacles and the shorelines. It has a 150° angular field of view and uses four infrared lasers with 3.2° vertical separation. The laser scanner is pitched and rolled by a DirectedPerception PTU-D100 gimbal work with an attitude heading reference system (AHRS) unit. Closed loop control using orientation measurements from the AHRS can improve shoreline scans by compensating for vehicle motion in pitch and roll.

The localization of vehicle is estimated by a ComNav Vector G2 DGPS compass mounted

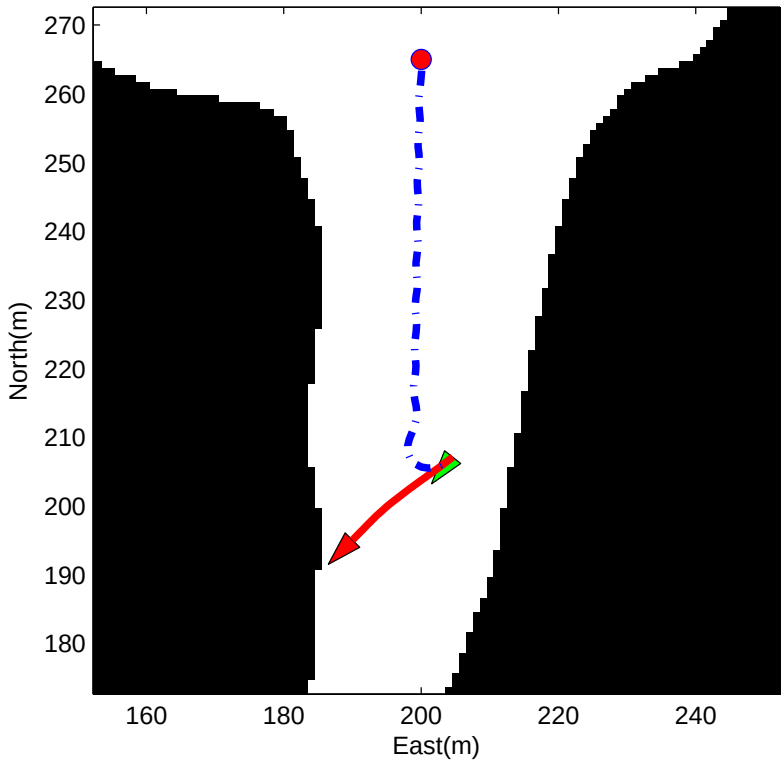

Figure 4.2: Close-up

on aluminum bars at the stern of the vehicle. It generates enhanced position information corrected by satellite. This device also outputs true heading and turn rate information by utilizing two internal GPS receivers.

Fig. 4.3 demonstrates the field results of backward path planning in a narrow cove in Claytor Lake in Dublin, VA. The position of the USV is shown by the red circles and the arrows indicate its heading. It clearly shows that the path planner continues to generate backward paths until it can compute a safe and feasible forward trajectory.

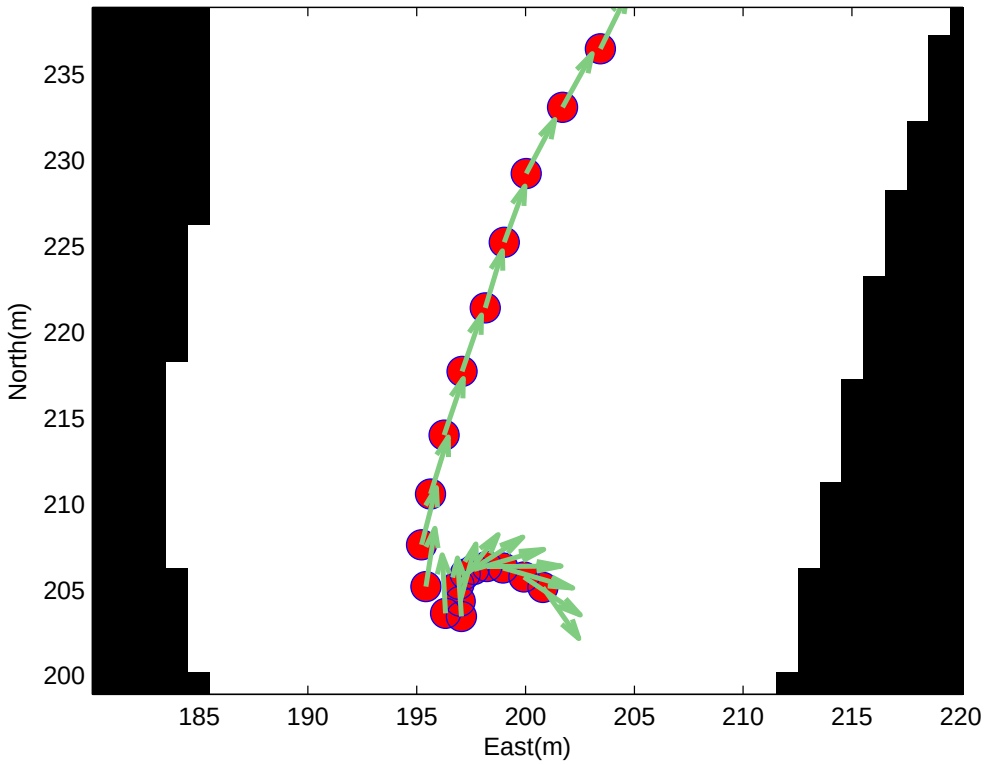

Figure 4.3: Field Trial Result

Chapter 5

Conclusions

In this work, we propose a backward path planning algorithm for unmanned surface vehicles. The principal contribution of this work is the real-time application of this algorithm and integration of forward and backward path planning. It operates in real-time and is integrated with a forward planning algorithm that was developed previously. Our new backward path planning algorithm solves the backward path planning problem as an optimal control problem, and we utilizes an adapted direct shooting method that makes real-time computation possible. The switching scheme links forward and backward path planning together to offer USV greater autonomous capabilities when it encounters more sophisticated riverine operating environments. The algorithm can also be deployed in other autonomous vehicle platforms since the system dynamics are implicitly incorporated in the generation of vehicle trajectories.

In future work, the path planner needs to generate both forward and backward paths that minimize potential probability of hitting obstacles. In another word, the USV should be able to find a path with lowest possibility of crashing into obstacles.

Bibliography

[1] D. J. Stilwell, A. S. Gadre, and A. J. Kurdila, "A receding horizon approach to generating dynamically feasible plans for vehicles that operate over large areas," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 1140–1145.

[2] J. E. Manley, "Unmanned surface vehicles, 15 years of development," in *OCEANS 2008*, Sept 2008, pp. 1–4.

[3] V. Bertram, "Unmanned surface vehicle - a survey," Skibsteknisk Selskab Copenhagen Denmark, Tech. Rep., 2008.

[4] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 583 – 601, July 2003.

[5] B. Nagy and A. Kelly, "Trajectory generation for car-like robots using cubic curvature polynomials," in *Field and Service Robots*, June 2001.

[6] T. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, February 2007.

[7] L. E. Dubins, "On curves of niminal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, pp. 497–516, 1957.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans on Systems Science and Cybernetics*, pp. 100–107, 1968.

[9] B. Xu, D. Stilwell, and A. Kurdila, "Efficient computation of level sets for path planning," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 4414–4419.

[10] B. Xu, A. Kurdila, and D. Stilwell, "A hybrid receding horizon control method for path planning in uncertain environments," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 4887–4892.

[11] Y. Zhuang, Y. Sun, and W. Wang, "Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating," *International Journal of Innovative Computing, Information and Control*, pp. 4095–4109, 2012.

[12] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *International Conference on Robotics and Automation*, May 2011, pp. 3041–3047.

[13] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathmatics*, pp. 367–393, 1990.

[14] J. D. Schwartz and M. Milam, "On-line path planning for an autonomous vehicle in an obstacle filled environment," in *IEEE Conference on Decision and Control*, Dec 2008, pp. 2806–2813.

[15] K. Kondak and G. Hommel, "Computation of time optimal movements for autonomous parkings of non-holonomic mobile platforms," in *IEEE International Conference on Robotics and Automation*, May 2001, pp. 2698–2703.

[16] C. Pradalier, S. Vaussier, and P. Corke, "Path planning for a parking assistance system: Implementation and experimentation."

[17] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics*, June 2008, pp. 32–37.

[18] ——, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, pp. 485–501, 2010.

[19] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, pp. 193–207, 1998.

[20] D. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[21] D. Kraft, "On converting optimal control problems into nonlinear programming problems," *Comput. Math. Programm*, pp. 261–280, 1985.

[22] R. M. Lewid, V. Torczon, and M. W. Trosset, "Why pattern search works," 1999.

[23] R. M. Lewis, V. Torczon, and M. W. Trosset, "Direct search methods: then and now," *Journal of Computational and Applied Methematics*, vol. 124, pp. 191–207, 2000.

[24] R. Hooke and T. A. Jeeves, "Direct search solution of numerical and statistical problems," *Journal of the ACM*, pp. 212–229, 1961.

[25] A. S. Gadre, S. Du, and D. J. Stilwell, "A topological map based approach to long range operation of an unmanned surface vehicle," in *American Control Conference*, June 2012, pp. 5401–5407.

[26] A. S. Gadre, C. Sonnenburg, S. Du, D. J. Stilwell, and C. Woolsey, "Guidance and control of an unmanned surface vehicle exhibiting sternward motion," in *OCEANS 2012*, Oct 2012, pp. 1–9.

[27] C. Sonnenburg, A. S. Gadre, D. Horner, S. Kragelund, D. J. Stilwell, and C. A. Woolsey, "Control-oriented planer motion modeling of unmanned surface vehicles," in *MTS/IEEE OCEANS*, 2010, pp. 1–10.