Motion Planning for Nonlinear Underactuated Vehicles using H^∞ Techniques

Gregory J. Toussaint Tamer Başar Francesco Bullo

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, IL 61801

Abstract

This paper presents a new solution to the motion planning problem for nonlinear underactuated vehicles. The approach solves the problem by generating a polynomial curve connecting the desired initial and final positions for the vehicle and then using the curve to estimate the vehicle's complete configuration along the trajectory. The algorithm introduces an iterative H^∞-filter to improve upon the initial estimate for the trajectory. The solution to the basic motion planning problem can be coupled with randomized path planning algorithms to solve the obstacle avoidance and multiple vehicle versions of the problem. Numerical simulations illustrate the algorithm's performance on an underactuated planar vehicle.

1 Introduction

Automated motion planning is a key element in developing vehicles that can operate independently in an unexplored environment. When the vehicles are both nonlinear and underactuated, the motion planning problem becomes very challenging because we must account for the dynamics of the vehicle, as well as the physical constraints in the environment. We formulate such a few motion planning problems and present novel solutions based on certain classes of polynomial curves and on an iterative H^∞-filter.

A good starting point to understand the motion planning problem (considered in the setting of systems with no dynamics) and some of the common solution techniques is a recent survey paper and the textbook by Latombe [1, 2]. Another general reference is a collection edited by Li and Canny [3] which focuses on nonholonomic systems, and it is therefore more closely associated with motion planning for underactuated vehicles. From the literature (see also [4] for a complete review), we learn that randomized techniques are the best algorithms for fast planning problems with obstacles and multiple vehicles.

In this paper, we present a technique that can quickly generate paths for nonlinear underactuated vehicles, that can handle obstacles in the environment, and that can accommodate planning for multiple vehicles. We develop an original solution to the basic motion planning problem which uses a deterministic approach to handle the nonlinear underactuated dynamics of the vehicle, but cannot directly accommodate the other aspects of the problem. We extend the solution to the basic motion planning problem by coupling it with an existing randomized planning techniques that allow us to consider obstacles in the environment and plan trajectories for multiple vehicles. From the various randomized techniques available, we select rapidly-exploring random trees (RRTs), as the algorithm [5, 6] and its variations [7, 8] have shown to be fast and widely applicable.

The paper is organized as follows. In Section 2, we formally state the problem. Sections 3 and 4 describe the solution to the basic motion planning problem and its extensions. We present simulation results in Section 5.

2 Problem Formulation

In the basic motion planning problem, we assume we are given the equations of motion for the vehicle we want to control. We are also given the initial and final configurations for the vehicle as well as a time interval for the motion. Denote the initial and final configurations as $\mathbf{q}_0$ and $\mathbf{q}_f$, respectively, and the time interval as $[t_0,\ t_f]$. Our goal is to find the control inputs to move the vehicle from $\mathbf{q}_0$ to $\mathbf{q}_f$ during the time interval $[t_0,\ t_f]$. We can then apply these control inputs to the equations of motion for the vehicle to generate a feasible trajectory. A feasible trajectory is one that the underactuated vehicle could follow in the absence of any disturbances and the vehicle's initial configuration aligned with the trajectory.

For the basic motion planning problem, there are no obstacles in the environment and no other requirements for the vehicle, except that its motion must satisfy the dynamic equations. We introduce obstacles and additional vehicles in Section 4 after solving the basic problem. Even though the vehicle is underactuated, there are still an infinite number of control laws that move

the vehicle from one configuration to the other in the given time. As a final aspect of the problem formulation, we present the vehicle model used for this research effort. The model was used earlier by Pettersen and Nijmeijer [**?**] and we have made minor changes to simplify the notation. The relevant equations of motion are

$$\begin{aligned}
\dot{u} &= m_u\, v\, r - d_u\, u + u_1 && (1)\\
\dot{v} &= m_v\, u\, r - d_v\, v && (2)\\
\dot{r} &= m_r\, u\, v - d_r\, r + u_2 && (3)\\
\dot{x} &= u\, \cos(\psi) - v\, \sin(\psi) && (4)\\
\dot{y} &= u\, \sin(\psi) + v\, \cos(\psi) && (5)\\
\dot{\psi} &= r && (6)
\end{aligned}$$

where u, v, and r are the body-frame velocities in surge, sway, and yaw respectively, x and y are the inertial positions, and ψ is the inertial rotation angle. The coefficients m_i and d_i represent combined mass terms, including added mass, and damping coefficients. The two control inputs are u_1 and u_2.

3 Basic Motion Planning

The basic motion planning algorithm consists of four main steps to generate a feasible trajectory that moves the vehicle from $\mathbf{q}_0$ to $\mathbf{q}_f$. The first step generates a polynomial curve in the x-y plane that connects the initial and final positions and has the proper orientation at each endpoint. The second step uses this curve to estimate the complete trajectory and a corresponding set of control inputs. We call these results the *candidate* trajectory and the *candidate* inputs, respectively. The third step uses the candidate inputs to generate a feasible trajectory for the vehicle. If this feasible trajectory approaches the desired final configuration, then the fourth step is not required. If the feasible trajectory is not satisfactory, the fourth step implements an H^∞-filter which uses the underactuated dynamics of the vehicle to update the estimate for the trajectory and the control inputs.

Step 1: Polynomial position curve generation The first step in the motion planning process uses one of two techniques to generate a polynomial position curve connecting the initial and final positions for the vehicle with the correct orientation at each endpoint. The first technique uses cubic splines to generate the curve and is sufficient for most problems. The second technique uses Pythagorean hodograph curves to connect the points and is slightly more complicated than the cubic spline approach. The additional complexity associated with the second approach may be justified if we want to quickly calculate the curve length or the curvature along the curve [9].

Both curve generation techniques require knowledge of the initial and final configurations for the vehicle, which are given as $\mathbf{q}_0$ and $\mathbf{q}_f$, respectively. We compute the initial and final velocities in the inertial frame, $\dot{x}(t)$ and $\dot{y}(t)$, by using $\mathbf{q}_0$ and $\mathbf{q}_f$ to evaluate (4) and (5) at t_0 and t_f, respectively. When combined with the orientation information, the initial and final velocities determine the direction of motion for the vehicle at the endpoints. We use this information along with the x-y coordinates of the endpoints to find an appropriate polynomial position curve. Once we have the polynomial expressions for $x(t)$ and $y(t)$, we can use them to estimate the other states of the system.

Step 2: Candidate trajectory estimation The second step in the algorithm uses the polynomial position curve to estimate the complete trajectory between the endpoints and to estimate a pair of control inputs corresponding to the trajectory. To develop accurate estimates, we use the equations of motion for the system and make some reasonable assumptions about the motion of the vehicle.

For the first step of the trajectory estimation, we use the polynomials for $x(t)$ and $y(t)$ as the estimates for the position of the vehicle, so that $\hat{x}(t) = x(t)$ and $\hat{y}(t) = y(t)$, where the 'hat' notation denotes an estimated value. These position estimates may not represent a feasible motion for the underactuated vehicle, but they do represent the type of curve we would like the vehicle to follow to satisfy the boundary conditions. We treat the difference between the x-y curve and a similar feasible path as a disturbance for the motion planning algorithm to attenuate. The iterative H^∞-filter attenuates this disturbance as it improves the estimate for the trajectory.

To find estimates for the other states, we perform a local analysis of the vehicle's motion. The instantaneous motion of the vehicle is in the direction given by the vector combination of $\dot{x}(t)$ and $\dot{y}(t)$ and is tangent to the x-y curve. If the vehicle was moving with a steady motion along a straight line, the orientation would align with the direction of motion. Since the vehicle is underactuated, it cannot maintain an orientation that is tangent to the x-y curve if the motion contains any turn. To develop an estimate for ψ, we assume the vehicle is turning in a circle at a constant velocity and account for the offset angle in the turn caused by the underactuated dynamics. The following result is based on one presented by Godhavn [10] for a similar type of underactuated vehicle. See [4] for a detailed derivation. The resulting orientation estimate is

$$\hat{\psi}(t) = \bar{\psi}(t) - \arctan\left[\frac{m_v \bar{r}(t)}{d_v}\right] \qquad (7)$$

where

$$\bar{\psi}(t) = \arctan\left[\dot{x}(t), \dot{y}(t)\right], \qquad \bar{r}(t) = \frac{d}{dt}\bar{\psi}(t).$$

The function $\arctan(a, b)$ with two arguments gives the argument of the complex number $a + ib$, so that the value of $\bar{\psi}(t)$ can range from $-\pi$ to π. We can use the

estimate for $\hat{\psi}(t)$ to generate the estimate for $r(t)$ as follows:

$$\hat{r}(t) = \frac{d}{dt}\hat{\psi}(t). \tag{8}$$

We now have expressions for $\hat{x}$, $\hat{y}$, $\hat{\psi}$, and $\hat{r}$. To estimate the remaining two states, we can use (4) and (5) to arrive at the following expressions for $\hat{u}(t)$ and $\hat{v}(t)$:

$$\hat{u}(t) = \cos\left[\hat{\psi}(t)\right]\frac{d}{dt}\hat{x}(t) + \sin\left[\hat{\psi}(t)\right]\frac{d}{dt}\hat{y}(t) \tag{9}$$

$$\hat{v}(t) = -\sin\left[\hat{\psi}(t)\right]\frac{d}{dt}\hat{x}(t) + \cos\left[\hat{\psi}(t)\right]\frac{d}{dt}\hat{y}(t). \tag{10}$$

Now that we have estimates for the complete trajectory $\hat{\mathbf{q}}$, we can estimate the two control inputs by using equations (1) and (3) and the derivatives $\frac{d}{dt}\hat{u}(t)$ and $\frac{d}{dt}\hat{r}(t)$:

$$\hat{u}_1(t) = \frac{d}{dt}\hat{u}(t) - m_u\hat{v}(t)\hat{r}(t) + d_u\hat{u}(t) \tag{11}$$

$$\hat{u}_2(t) = \frac{d}{dt}\hat{r}(t) - m_r\hat{u}(t)\hat{v}(t) + d_r\hat{r}(t). \tag{12}$$

The state estimation procedure fully exploits the information in the equations of motion to provide an initial trajectory estimate from a polynomial x-y position curve. We denote this candidate trajectory by $\hat{\mathbf{q}}(t)$. We also have made an initial estimate for the two control inputs, $\hat{u}_1(t)$ and $\hat{u}_2(t)$, which we denote as $\hat{\mathbf{u}}(t)$. Finding these estimates marks the end of the second step in the motion planning algorithm.

Step 3: Generating the feasible trajectory The next step in the motion planning process is to use the inputs $\hat{\mathbf{u}}$ to numerically integrate the equations of motion given by (1) through (6) starting at $\mathbf{q}_0$. This numerical integration generates a feasible trajectory, $\mathbf{q}(t)$, for the underactuated vehicle. If the feasible trajectory approaches the desired final configuration, then we have found a solution to the basic motion planning problem. In general, the feasible trajectory will not follow closely the estimated trajectory $\hat{\mathbf{q}}$. Large differences between the two trajectories might arise because the position curve is significantly different from a straight line or a circular arc. In addition, there are differences because we are using the initial condition $\mathbf{q}_0$ to generate the feasible trajectory and it may not match the estimated initial condition $\hat{\mathbf{q}}(t_0)$ obtained from the previous step.

One way to reduce the differences between the feasible trajectory and the estimated trajectory is to use the estimated initial configuration $\hat{\mathbf{q}}(t_0)$ to initiate the numerical integration. This adjustment introduces an error between the initial configurations for the vehicle and the feasible trajectory. If the initial configuration error is small enough, then the tracking controller is able to compensate and we have a suitable feasible trajectory. If adjusting the initial condition introduces a significant configuration error, then a tracking controller may not be able to recover from the initial error to follow the feasible trajectory. We remark that tracking controllers for general nonlinear control system do in general have a limited stability region.

Step 4: Iterative H^∞-filtering We propose an H^∞-filter for the setting in which the feasible trajectory generated by the first three steps does not closely approach the desired final configuration. The objective of the H^∞-filter is to iteratively improve the estimate for the control inputs and the resulting feasible trajectory. In summary, the filtering approach relies on the fact that we can decompose the system equations into two subsystems that are each affine if the state of the other subsystem is known. The interlaced subsystems allow us to compute improved estimates for the state of the system that account for the nonlinear underactuated dynamics of the vehicle. We then use the new state estimates to update the estimates for the control inputs that generate a feasible trajectory.

We derive the iterative H^∞-filter starting with the equations of motion for the vehicle. We have polynomials representing the x-y positions we want the vehicle to track, but since it is an underactuated vehicle, it may not be able to do so exactly, because the motion may be infeasible. Therefore, we treat the x-y curve as an imperfect measurement of the position of the vehicle. We now use this measurement of the vehicle's position and the equations of motion to estimate the remaining four states of the system that are consistent with the measurements. We let $\mathbf{y}$ represent the measurement vector as follows:

$$\mathbf{y} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} w_x \\ w_y \end{bmatrix} = C\mathbf{q} + E\mathbf{w}_{xy} \tag{13}$$

where w_x and w_y represent disturbance terms that capture the error in the measurements. To develop the H^∞-filter, we first decompose the nonlinear system into two subsystems with the property that each one is affine if the state of the other subsystem is known. We let $\mathbf{q} = [\mathbf{q}_1^{\mathrm{T}}, \mathbf{q}_2^{\mathrm{T}}]^{\mathrm{T}}$, where

$$\mathbf{q}_1 = \begin{bmatrix} u & v & x & y \end{bmatrix}^{\mathrm{T}}, \qquad \mathbf{q}_2 = \begin{bmatrix} r & \psi \end{bmatrix}^{\mathrm{T}}.$$

The two subsystems can be written compactly as

$$\dot{\mathbf{q}}_1 = A_{11}(\mathbf{q}_2)\mathbf{q}_1 + \alpha_1(\mathbf{q}_2) + B_1\mathbf{u}_1 \tag{14}$$

$$\dot{\mathbf{q}}_2 = A_{22}(\mathbf{q}_1)\mathbf{q}_2 + \alpha_2(\mathbf{q}_1) + B_2\mathbf{u}_2. \tag{15}$$

In (14) and (15), the $\alpha_i(\mathbf{q}_j)$ variables represent the affine terms. We would like to identify a first-order approximation for each affine term, so we calculate the linear portion of α_1 and α_2 as follows:

$$A_{12} = \frac{\partial\alpha_1(\mathbf{q}_2)}{\partial\mathbf{q}_2}, \qquad A_{21} = \frac{\partial\alpha_2(\mathbf{q}_1)}{\partial\mathbf{q}_1}.$$

We can now rewrite equations (14) and (15) as

$$\dot{\mathbf{q}}_1 = A_{11}(\mathbf{q}_2)\mathbf{q}_1 + A_{12}\mathbf{q}_2 + [\alpha_1(\mathbf{q}_2) - A_{12}\mathbf{q}_2] + B_1\mathbf{u}_1$$
$$\dot{\mathbf{q}}_2 = A_{22}(\mathbf{q}_1)\mathbf{q}_2 + A_{21}\mathbf{q}_1 + [\alpha_2(\mathbf{q}_1) - A_{21}\mathbf{q}_1] + B_2\mathbf{u}_2.$$

and combine them into a single set of equations as

$$\dot{\mathbf{q}} = \begin{bmatrix} A_{11}(\mathbf{q}_2) & A_{12} \\ A_{21} & A_{22}(\mathbf{q}_1) \end{bmatrix} \mathbf{q} + \begin{bmatrix} \alpha_1(\mathbf{q}_2) - A_{12}\mathbf{q}_2 \\ \alpha_2(\mathbf{q}_1) - A_{21}\mathbf{q}_1 \end{bmatrix} + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \mathbf{u}$$

where $\mathbf{u} = [u_1, u_2]^{\mathrm{T}}$. We can then write the combined subsystem equations compactly as

$$\dot{\mathbf{q}} = A(\mathbf{q})\mathbf{q} + \alpha(\mathbf{q}) + B\mathbf{u}. \tag{16}$$

We do not know the value of $\mathbf{q}$ in advance, so we use the estimate $\tilde{\mathbf{q}}$, to be defined shortly, in the terms $A(\mathbf{q})$ and $\alpha(\mathbf{q})$ to get

$$\dot{\mathbf{q}} = A(\tilde{\mathbf{q}})\mathbf{q} + \alpha(\tilde{\mathbf{q}}) + B\mathbf{u} + [A(\mathbf{q})\mathbf{q} - A(\tilde{\mathbf{q}})\mathbf{q} + \alpha(\mathbf{q}) - \alpha(\tilde{\mathbf{q}})].$$

We can characterize the nonlinear term in square brackets as a disturbance term so that we can write

$$\dot{\mathbf{q}} = A(\tilde{\mathbf{q}})\mathbf{q} + \alpha(\tilde{\mathbf{q}}) + B\mathbf{u} + D\mathbf{w} \tag{17}$$

where $\mathbf{w}$ is the disturbance vector. We take (17) to be the affine differential equation that describes the motion of the vehicle. We can measure the states x and y and want to use those measurements to estimate the remaining states. We use the following H^∞-filter differential equations to estimate the states and compute the corresponding covariance matrix [11]:

$$\dot{\tilde{\mathbf{q}}} = A(\tilde{\mathbf{q}})\tilde{\mathbf{q}} + \alpha(\tilde{\mathbf{q}}) + B\mathbf{u} + \Sigma C^{\mathrm{T}} N^{-1}(\mathbf{y} - C\tilde{\mathbf{q}}) \qquad \tilde{\mathbf{q}}(t_0) = \mathbf{q}_d(t_0) \tag{18}$$

$$\dot{\Sigma} = A(\tilde{\mathbf{q}})\Sigma + \Sigma A^{\mathrm{T}}(\tilde{\mathbf{q}}) - \Sigma(C^{\mathrm{T}} N^{-1} C - \gamma^{-2} Q)\Sigma + DD^{\mathrm{T}} \qquad \Sigma(t_0) = Q_0^{-1}. \tag{19}$$

In (18), the $A(\tilde{\mathbf{q}})$ matrix and the affine term $\alpha(\tilde{\mathbf{q}})$ depend on the state estimate, which is available information and allows us to capture the nonlinear dynamics in an affine differential equation.

We need to know $\mathbf{u}$ to compute the state estimates in equation (18), but $\mathbf{u}$ represents the control inputs we are trying to determine to generate a feasible trajectory. To resolve this circular problem, we use the estimate $\hat{\mathbf{u}}$ in place of $\mathbf{u}$ in (18). We then implement the H^∞-filter to develop a new estimate for the configuration vector for the vehicle. With the new configuration estimates, we use (11) and (12) to update the estimates for the control inputs. Finally, we repeat the H^∞-filtering to improve the results. It is important to remark that for all iterations the output signal $\mathbf{y}$ is the polynomial curve generated during **Step 1**. We run the iterative H^∞-filter until the final configuration error no longer improves, but we limit the number of iterations to guarantee that the iteration terminates. We use the final results of the H^∞-filter to generate a feasible trajectory for the vehicle.

Assuming the system is linear, and the polynomial curve for the center of mass is a feasible output, then there exists a input/state trajectory which is a fixed point for the iteration, and the fixed point is attractive. Since the system is nonlinear, we rely on a linearization along the estimated trajectory as a way to still use linear filtering. The H^∞-filter provides some disturbance attenuation as a means to deal with the neglected nonlinearities. Finally, we use a particular way of linearizing the equations, based on the interlaced structure. In our numerical experimentation, exploiting this structure is helpful. Future research will focus on analytically characterizing the properties of this iterative filtering technique.

4 Extensions to Basic Motion Planning

We consider two extensions to the basic motion planning problem. The objective is to solve planning problems when obstacles are present and when other vehicles are in the environment. Before we extend the planning techniques to handle these problems, we briefly describe a useful randomized planning algorithm.

4.1 Rapidly-Exploring Random Trees

From the many techniques available to address the obstacle avoidance and multiple vehicle planning problems, we selected the rapidly-exploring random trees (RRTs) approach developed by LaValle and Kuffner [5, 6] as the one to use for our application. This approach offers the advantages of easily handling the underactuated nature of our problem, automatically accounting for obstacles, and, with minor modifications, handling the multiple vehicle case.

There are many variations of the RRT algorithm, so we briefly describe the one that we implemented for our problem. The algorithm builds two search trees, with one starting at the initial configuration and the other starting from the goal configuration. The trees consist of nodes that represent configurations the vehicle could achieve after applying an input for an incremental time step. The objective is to build the two trees towards each other so that they can eventually be linked to form a complete trajectory between the initial and goal states. At each time step, a random configuration in the space of all possible configurations is selected. The algorithm then searches one of the trees to find the closest node to the new configuration. The distance between two configurations is determined by a metric, which is a key element in the design process. Once the algorithm identifies the closest node in the tree, it samples a list of possible inputs to find values that, in one time step, move the configuration as close as possible to the new node without colliding with an obstacle. The resulting configuration is then added as a new node to the tree. The process is then repeated with the roles of the trees swapped and the whole algorithm is iterated until the number of nodes is exhausted or the trees are connected.

4.2 Obstacle avoidance and multiple vehicles motion planning

We extend the basic motion planning problem to the setting of environments with obstacles. The objective is to find a feasible trajectory for the underactuated vehicle that avoids collisions with the (fixed) obstacles. A similar problem is that of planning motions for multiple vehicles: the goal is to identify feasible trajectories for each vehicle that satisfy the motion requirements and do not cause collisions. If few vehicles are present, the multiple vehicles problem can be cast in a centralized planning setting as a larger dimensional planning problem with obstacles.

We solved the problem of planning motions around obstacles by using the RRT algorithm and carefully designing one of its key elements: the metric needed to evaluate the separation between configurations. The standard metrics in [5, 6] are based on a Euclidean-type distance between the configurations. As suggested in [6], these metrics are not appropriate for an underactuated vehicle, because it is possible that two configurations separated by a small Euclidean distance whereas the underactuated vehicle would have to travel a relatively long trajectory to move between them. The new metric tries to compensate for this deficiency by including a measure of the path length for the trajectory the vehicle would actually follow. Given two configurations, the new metric computes the length of a Pythagorean hodograph curve connecting the two positions, with the curve having the proper tangent at each endpoint. The tangents are determined by the vehicle's velocity and orientation at the endpoints. The new metric sums the square of the path length with the squares of the errors in the other four state variables and then takes the square root of the result. This metric attempts to provide a more reasonable estimate of the separation between two configurations than the simple Euclidean metrics and our simulations indicate it can improve the results of the RRT algorithm.

We solved an obstacle avoidance problem for the underactuated vehicle using the modified RRT software with both the standard metric and the new Pythagorean hodograph curve metric. Our simulations indicate there are three advantages to using the Pythagorean hodograph curve metric compared to the standard metric. First, the algorithm could find solutions with fewer nodes in the trees, which translates into fewer exploration steps. Second, the resulting trajectories are usually shorter than with the standard metric, so the motion is more efficient. Third, the maximum jump between any two nodes is usually smaller with the new metric, so the paths are smoother and it is easier for the tracking controller to follow the trajectories.

Finally, we smooth the trajectories generated by the RRT algorithm to improve the motion of the vehicle. The trajectories generated by the RRT algorithm satisfy the motion planning requirements, but the resulting motion may be very rough because the inputs are randomly selected at each time step. There are two simple methods for smoothing that are based on the basic motion planning algorithm. The first smoothing method applies the iterative H^∞-filter to the feasible trajectory to refine the results from the RRT approach. The second smoothing method samples the original RRT-generated trajectory and uses the basic planning algorithm to design motions between the sampled configurations. With either technique, the resulting trajectory is smoother than the original and retains the overall shape, so it likely avoids the obstacles and other vehicles, as originally planned.

5 Simulation Results

We now present the simulations that implement the motion planning algorithms described in the previous sections. The simulations used the following values for the coefficients in the underactuated vehicle model given in Section 1: $m_u = 0.5$, $m_v = -2.0$, $m_r = 0.5$, $d_u = 1.0$, $d_v = 2.0$, and $d_r = 1.0$. We applied the motion planning algorithms to two example problems to demonstrate its features and additional examples can be found in [4].

The first example is shown in Figure 1 and illustrates how the iterative H^∞-filter improves the feasible trajectory to make the vehicle's final configuration approach the goal configuration. In this example, we left the initial configuration of the vehicle as its *actual* configuration for the motion planning algorithm. This design choice appears in Figure 1 as a difference between the candidate position curve (dotted line) and the feasible planned position curves along the initial segment of the path. This difference does not diminish as the iterative H^∞-filter operates on the system. Using the estimated initial configuration to generate the feasible trajectory would significantly reduce this initial error.

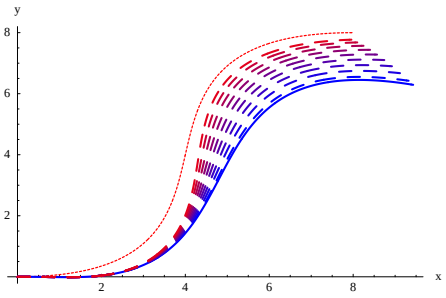

Figure 1: Basic motion planning results. The dotted line shows the candidate position curve, the solid line shows the first feasible trajectory and the dashed lines show the feasible trajectories as the H^∞-filter iterates on the results.

Figure 2 provides another example of a feasible motion developed using the basic motion planning algorithm. The results Figure 2 show how the vehicle would make

a lateral displacement from the origin. In this case, we used the estimated initial condition to develop the feasible trajectory and we did not need the iterative H^∞-filter to achieve reasonable performance.

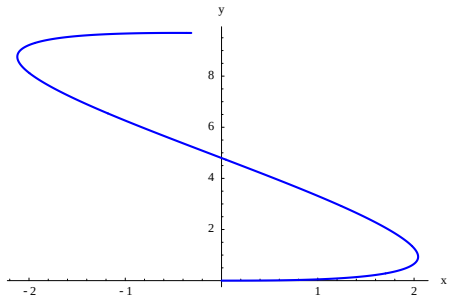

Figure 2: Basic motion planning results using only the first three steps. The required motion is sideways translation.

For the third example, Figure 3 illustrates the results of planning motions for two vehicles around obstacles and using sampling to smooth the trajectories. In this simulation, we required a minimum amount of sampling to capture the important aspects of the vehicles' motions to avoid collisions. The two vehicles achieve the motion planning objectives with natural maneuvers while avoiding each other and the obstacles.

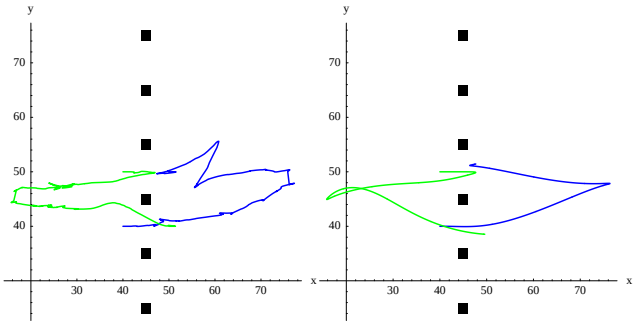

Figure 3: Motion planning for multiple vehicles with obstacles. The left figure shows the results of the RRT algorithm and the right figure shows how smoothing the trajectories with sampling improves the results. Vehicle 1 moves from $(40, 40)$ to $(50, 50)$ and vehicle 2 moves from $(40, 50)$ to $(50, 40)$.

6 Summary

We have presented motion planning techniques to generate feasible trajectories for an underactuated vehicle. The solutions address the basic motion planning problem and the situations where obstacles or other vehicles are present in the environment. The solution to the basic problem uses a polynomial position curve to estimate the trajectory the vehicle follows and incorporates an iterative H^∞-filter to improve the trajectory estimate. The obstacle avoidance and multiple vehicle problems rely on rapidly-exploring random trees to generate an initial solution, which can later be smoothed to find a more realistic trajectory.

References

[1] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[2] J.-C. Latombe, *Robot Motion Planning*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1991.

[3] Z. Li and J. F. Canny, eds., *Nonholonomic Motion Planning*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1993.

[4] G. J. Toussaint, *Robust Control and Motion Planning for Nonlinear Underactuated Systems using* H$^\infty$ *Techniques*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, June 2000. Available electronically at `http://black.csl.uiuc.edu`.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *IEEE Int. Conf. on Robotics and Automation*, (Detroit, MI), pp. 473–479, May 1999.

[6] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Workshop on Algorithmic Foundations of Robotics*, (Dartmouth, NH), pp. 293–308, Mar. 2000.

[7] E. Frazzoli, M. A. Dahleh, and E. Feron, "A hybrid control architecture for aggressive maneuvering of autonomous helicopters," in *IEEE Conf. on Decision and Control*, (Phoenix, AZ), pp. 2471–6, Dec. 1999.

[8] T. Karatas and F. Bullo, "Randomized searches and nonlinear programming in trajectory planning," in *IEEE Conf. on Decision and Control*, (Orlando, FL), pp. 5032–5037, Dec. 2001.

[9] H. Bruyninckx and D. Reynaerts, "Path planning for mobile and hyper-redundant robots using pythagorean hodograph curves," *Proceedings of the 1997 8th International Conference on Advanced Robotics*, pp. 595–600, July 1997.

[10] J.-M. Godhavn, "Nonlinear tracking of underactuated surface vessels," in *IEEE Conf. on Decision and Control*, (Kobe, Japan), pp. 975–80, Dec. 1996.

[11] T. Başar and P. Bernhard, *H^∞-Optimal Control and Related Minimax Design Problems*. Boston, MA: Birkhäuser, 2 ed., 1995.