A dynamic programming approach for the control of autonomous vehicles on planar motion

Jorge Estrela da Silva
Electrical Engineering Department
Institute of Engineering of Porto
Porto, Portugal
jes@isep.ipp.pt

João Borges de Sousa
Electrical and Computer Engineering Department
Faculty of Engineering of Porto University
Porto, Portugal
jtasso@fe.up.pt

***Abstract*—The problem of path following for autonomous vehicles under adversarial behavior is considered. The objective is to keep the cross-track error to the reference path inside a given tolerance interval. The adversarial behavior models system uncertainty and unknown or poorly estimated bounded disturbances. The first step to that objective is the computation of an invariant set, namely the maximal set of states that the vehicle may enter while ensuring that the cross-track error will never exceed the tolerance interval. This is done through dynamic programming. Two modes of operation are then considered: when the vehicle is inside the invariant set, the objective is to stay inside it while minimizing a combination of the actuation effort and cross-track error; otherwise, the objective becomes to reach the invariant set in minimum time. Each mode corresponds to a different optimal control problem which is dealt independently; thus, each mode has a corresponding control law. We discuss efficient ways of computing and implementing those control laws on currently available computational systems. For the purpose of the dynamic programming approach, the autonomous vehicles are modeled as an unicycle. Simulations with a six degree of freedom nonlinear model of an autonomous submarine are performed in order to illustrate the robustness of the control strategy.**

***Index Terms*—dynamic programming, path following, autonomous vehicles**

I. Introduction

The path following problem has been studied for wheeled mobile robots (see [1] and [2] for early approaches), for under-actuated marine vehicles (see [3] for instance) and aerial vehicles (see [4] for instance). The path following formulation, as considered here, takes into account the vehicle's current state in order to compute the reference posture. The first aspect to be dealt in a path-following algorithm is how to choose the reference point at each time instant. The simplest approach to this reference generation is to choose the posture (position and heading) associated to the closest point in the path. This intuitive approach has been studied before (see [5] and [4] for instance). It must be remarked that the reference generator may be a dynamic system itself; in this case, the reference configuration acts as a virtual moving target that takes in account the vehicle's past trajectory and dynamic constraints (see [6], for instance). That approach is explored in order to cope with more general paths, for which the closest point approach may lead to singularities (e.g., when the vehicle is at the center of an arc). The approaches may also differ on whether the path curvature must be known or not. In [6], the authors develop a controller that may follow arbitrary paths without knowing its curvature profile.

The existing approaches to path following vary on the degree of complexity of the considered system model. In general, the vehicle longitudinal speed (surge) is considered constant. In [4], the authors assume an essentially kinematic model, with no sideslip and with first order linear dynamics for yaw rate. For vehicles with non-null lateral speed (sway) some kind of dynamic model must be considered (see [5], [3] and [7] for examples on marine vehicles). In general, determination of an accurate dynamic model is a difficult task. In [6], the authors disregard the dynamics of lateral speed and model it as a bounded disturbance. In [7], the authors describe an approach to add some robustness against model uncertainty.

The approach described in this work is independent of the system model. However, for simplicity, we will consider a model with 3 state variables, where the lateral speed is modeled as a bounded disturbance.

Most papers on the subject seek the perfect following of the reference path. In some cases, that objective may be impractical or simply unnecessary. For instance, autonomous underwater vehicles have strong limitations in what concerns the estimation of their exact position. Given that uncertainty, it does not make sense to be very stringent in what concerns the following of a path that the vehicle can not determine very accurately. Therefore, the designer of the control system may seek a more relaxed objective: to keep the vehicle inside a given tube around the reference path. This approach is followed in [6]. We follow similar guidelines in the current work. The main difference between these two works resides in the controller design. In [6], the authors derive a control law with guaranteed bounded error and convergence. However, the performance of the proposed control law is not discussed. In the current work, we present a methodology to design *robust optimal* path following controllers with guaranteed bounded error for paths with bounded curvature. Since the methodology is based on dynamic programming [8], [9], the derived controllers are globally optimal up to the accuracy of the employed numerical scheme.

The paper is organized as follows. Section II describes the system model. In section III, we propose a general approach for controlling systems ensuring bounded error against bounded

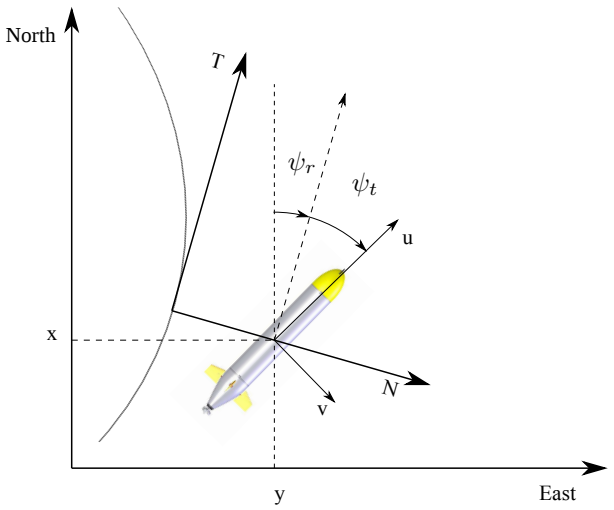

Fig. 1. Coordinate system (North-East-Down convention).

disturbances; the problems associated to that approach are formulated. In section IV, these problems are solved in the framework of dynamic programming. In section V, we describe how to use a publicly available numerical solver to derive the control laws; the process is then applied to the problem of path following. In section VI, we discuss some practical aspects of implementation. Finally, in section VII, the conclusions are presented.

II. System Model

Consider the following vehicle model

$$\dot{x} = u\cos(\psi) - v\sin(\psi) + c_x \tag{1}$$

$$\dot{y} = u\sin(\psi) + v\cos(\psi) + c_y \tag{2}$$

$$\dot{\psi} = r \tag{3}$$

where (x, y) describes the vehicle's position on a earth fixed frame, ψ is the vehicle's heading direction, u and v are the longitudinal and lateral speeds, $r \in [-r_{\max}, r_{\max}]$ is the vehicle's angular speed and (c_x, c_y) models the effect of environmental disturbances (such as constant winds and currents). For practical operation, $u > ||(c_x, c_y)||_\infty$ is assumed. In general, we have $|u| \gg |v|$.

Consider a frame with its origin at the nearest point of the path. We assume there is some scheme to disambiguate the choice in case of multiple candidates. This frame (the Frenet frame) has a T axis tangent to the path and a N axis normal to the path. The orientation of the T axis with respect to the earth fixed frame is ψ_t and we define the vehicle's angle relative to the path as $\psi_r = \psi - \psi_t$ (see Fig. 1). The cross-track error d is the shortest distance between the vehicle and the path. Thus, the cross-track error dynamics may be described by the following model:

$$\dot{d} = u\sin(\psi_r) + v\cos(\psi_r) + c_n \tag{4}$$

$$\dot{\psi}_r = r - (u\cos(\psi_r) - v\sin(\psi_r) + c_t)\kappa \tag{5}$$

where $c_n = -c_x\sin(\psi_t) + c_y\cos(\psi_t)$, $c_t = c_x\cos(\psi_t) + c_y\sin(\psi_t)$ and κ is the path's curvature (for a straight path $\kappa = 0$) at the closest point (origin of the Frenet frame).

In the general case, it is reasonable to assume that u is constant and that it may act as an input to the system. However, the same can not be assumed for v for vehicles with no lateral actuation and no sideslip constraint (i.e., that may skid). In these cases, the dynamics of v have to be considered. However, an accurate identification of the system's dynamic model may be an expensive task. In what concerns the environmental disturbances, this task is even more difficult, since the disturbances may vary both temporally and spatially. In order to cope with this model uncertainty and keep the system model simple, we consider a bounded virtual input c_v that may take values in $[-c_{v,\max}, c_{v,\max}]$ such that

$$c_{v,\max} \geq \|v(.)\cos(\psi_r(.)) + c_n(.)\|_\infty \tag{6}$$

Additionally, we consider the following input

$$r_v \in [-r_{v,\max}, r_{v,\max}] \supset [-U_{t,\max}/R_{\min}, U_{t,\max}/R_{\min}] \tag{7}$$

where $U_{t,\max}$ is the maximum expected tangential velocity (i.e., the projection of the vehicle's velocity on the T axis). This input models possible variations in the path's curvature, i.e., it models the assumption that the minimum radius of curvature is $R_{\min}$; it may also encompass the effect of modeling errors for the r dynamics. At each instant of time, c_v and r_v may take the worst-case values for the current control objective. In the framework of differential games [10], these are adversarial inputs. Under the stated assumptions, the model becomes as follows:

$$\dot{x} = f(x, a, b) = \begin{cases} u\sin(\psi_r) + c_v \\ r - r_v \end{cases} \tag{8}$$

where $x = \begin{bmatrix} d & \psi_r \end{bmatrix}'$, $a = \begin{bmatrix} r \end{bmatrix}$ and $b = \begin{bmatrix} c_v & r_v \end{bmatrix}'$.

III. Problem Formulation

The considered systems have smooth dynamics, described by $\dot{x} = f(x, a, b)$, where $x \in \mathbb{R}^n$ is the state vector, a is a bounded controlled input and b is a bounded adversarial input that always acts against the defined control objectives. The main objective is to keep the system inside a neighborhood $\mathcal{S}$ of a reference point x_{ref}. The set $\mathcal{S}$ models the maximum tolerance for system operation. When outside $\mathcal{S}$, the state of the system should be driven to $\mathcal{S}$ in minimal time. When inside $\mathcal{S}$, the system may use any control strategy as long as it stays inside $\mathcal{S}$. One such strategy could be to make the system's state converge to x_{ref} while avoiding abrupt changes in the system's trajectory; this may be important, for instance, for data sampling applications or to maximize comfort in transportation systems. The main objective is accomplished by the solution of the three problems formulated below. We recall the definition of weakly invariant set:

Definition 1: A set S is weakly invariant for the dynamic system $\dot{x} = f(x, a, b)$ (as defined above) if for every $x_0 \in S$ there is at least one trajectory such that $x(0) = x_0$ and $x(.)$

never leaves S, i.e., for each input $b(.)$ there is at least one input $a(.)$ such that $x(.)$ never leaves S.

It is clear that $\mathcal{S}$ must meet the requirements of a weakly invariant set. However, in many scenarios, the tolerance for system operation may be defined in a rather loose sense; it may just reflect, for instance, some constraint in the value of some system output. Therefore, we assume that the input for our control design is the set $\mathcal{R}$ which defines the acceptable region of operation. Since $\mathcal{R}$ is not necessarily invariant, we define the first problem.

Problem 1: Find the maximal invariant set $\mathcal{S} \subset \mathcal{R}$.

Let us resort to the path following problem in order to illustrate this aspect. In the present approach to the path-following problem, one simply states the requirement of keeping the cross-track error below a given threshold. However, in order to accomplish that, there are certain states that must be avoided. For instance, consider that the vehicle is over the desired path (i.e., accomplishing null cross-track error) but still perpendicular to it; then, due to its limited radius of curvature, the vehicle might not be able to respect the desired maximum cross-track error when following the remaining path. Thus, although a set $\mathcal{R} = [-d_{\max}, d_{\max}] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ may be initially considered as an acceptable region of operation, the system will not be able to remain inside that set for the entire duration of the operation. Therefore, it is necessary to find a subset of $\mathcal{R}$ composed only of those states the system may enter while ensuring that it is still able to remain inside $\mathcal{R}$ afterwards. That subset of $\mathcal{R}$ is the invariant set $\mathcal{S}$. Given $\mathcal{S}$, two additional problems are considered:

Problem 2: Derive a control law such that if the system is outside $\mathcal{S}$ then it should drive the system into $\mathcal{S}$ as soon as possible.

Problem 3: Derive a control law such that if the system is inside $\mathcal{S}$ then it should make the system stay inside $\mathcal{S}$ while minimizing the time integral of a given function $l(x, a) \geq 0$ (the running cost) for the whole operation time T. The running cost is assumed to be bounded for $x \in \mathcal{R}$.

More formally, Problem 2 consists of finding a control law $f_{\text{mt}}(x)$ such that, for any given initial condition $x(0) = x_0$, $a(.) = f_{\text{mt}}(x(.))$ is the solution of the following time-optimal differential game [10]:

$$\min_{a(.)} \max_{b(.)} t_f \tag{9}$$

subject to:

$$\dot{x}(t) = f(x(t), a(t), b(t)) \tag{10}$$

$$x(t_f) \in \mathcal{S} \tag{11}$$

$$\forall t \in [0, t_f] : a(t) \in U_a, b(t) \in U_b \tag{12}$$

where $a(.)$ belongs to the set of measurable nonanticipating strategies.

Similarly, Problem 3 consists of finding a control law $f_S(x)$ such that, for any given initial condition $x(0) = x_0$, $a(.) = f_S(x(.))$ is the solution of the following finite horizon optimal control problem with state constraints:

$$\min_{a(.)} \max_{b(.)} \int_0^T l(x, a) \tag{13}$$

subject to:

$$\dot{x}(t) = f(x(t), a(t), b(t)) \tag{14}$$

$$x(t) \in \mathcal{S} \tag{15}$$

$$\forall t \in [0, T] : a(t) \in U_a, b(t) \in U_b \tag{16}$$

where $a(.)$ belongs to the set of measurable nonanticipating strategies. The choice of T is discussed in the next section. Note that our system, described by (8), meets the usual conditions for the existence of solution and, moreover, it meets the Isaacs condition (see [10, Ch. 1]):

$$\begin{aligned} &\min_{a \in U_a} \max_{b \in U_b} [p \cdot f(x, a, b) + l(x, a)] \\ =&\max_{b \in U_b} \min_{a \in U_a} [p \cdot f(x, a, b) + l(x, a)] \end{aligned} \tag{17}$$

for any $p \in \mathbb{R}^n$ and $(t, x) \in [0, T] \times \mathbb{R}^n$.

IV. Solution Approach

Our approach to problem 1 is inspired by [11]. We define the value function $V_1(x, t)$ as follows: $V_1(x, -\tau) = \tau$ if there is a trajectory leaving x at $t = 0$ and reaching $\mathcal{R}$ at $t = \tau$ such that $\forall t \in [0, t] : x(t) \in \mathcal{R}$; $V_1(x, t) = \infty$ otherwise. Therefore, $\mathcal{S} = \{x \in \mathbb{R}^n : V(x, -T_1) \neq \infty\}$, where T_1 is chosen such that the following condition is met:

$$\forall x \in \{x : V_1(x, -T_1) \neq \infty\} : \frac{\partial}{\partial t} V_1(x, -T_1) = -1 \tag{18}$$

This means that the set $\{x \in \mathbb{R}^n : V(x, t) \neq \infty\}$ is not changing anymore for $t < -T_1$.

The problem now consists of computing $V_1(x, t)$. This can be done by computing the solution of the following Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE).

$$\frac{\partial}{\partial t} V(x, t) + H(x, \nabla V(x, t)) = 0 \tag{19}$$

$$V(x, 0) = 0, x \in \mathcal{R} \tag{20}$$

$$V(x, 0) = \infty, x \notin \mathcal{R} \tag{21}$$

$$V(x, t) = \infty, x \notin \mathcal{R} \tag{22}$$

with

$$H(x, p) = \min_a \max_b p \cdot f(x, a, b) + 1. \tag{23}$$

Problem 2 is also solved using dynamic programming techniques. Consider the following static HJI PDE:

$$\max_b \min_a \nabla V_{\text{mt}}(x) \cdot f(x, a, b) + 1 = 0, x \notin \mathcal{S} \tag{24}$$

$$V_{\text{mt}}(x) = 0, x \in \mathcal{S} \tag{25}$$

Then, $V_{\text{mt}}(x)$ is the minimum time to reach the target. The desired optimal control law is

$$f_{\text{mt}}(x) = \text{argmin}_a [\nabla V_{\text{mt}}(x) \cdot f(x, a, b)] \tag{26}$$

Finally, in order to solve Problem 3, the following HJI PDE is considered:

$$\frac{\partial}{\partial t}V_S(x,t) + H_S(x, \nabla V_S(x,t)) = 0, x \in \mathcal{S} \quad (27)$$

$$V_S(x,0) = 0, x \in \mathcal{S} \quad (28)$$

$$V_S(x,t) = \infty, x \notin \mathcal{S} \quad (29)$$

where

$$H_S(x,p) = \max_b \min_a p \cdot f(x,a,b) + l(x,a). \quad (30)$$

The solution of this PDE is $V_S(x,t)$, the minimum cost to keep the vehicle inside $\mathcal{S}$ during $-t$ units of time. We are interested in finding a time-invariant control law. That is because we do not want to rely on a estimate of the system operation time. We assume that, for a sufficiently large T, $V_S(x,-T)$ gives a reliable measure of the cost associated to steady state operation. Thus, if $V_S(x_1,-T) > V_S(x_2,-T)$ then we assume that, for the general case, its preferable to drive the system from x_1 to x_2 than staying in x_1. Therefore, the following control law is defined

$$f_S(x) = \text{argmin}_a[\nabla V_S(x,-T) \cdot f(x,a,b) + l(x,a)] \quad (31)$$

Therefore, the global optimal control strategy is

$$a(t) = \begin{cases} f_{\text{mt}}(x(t)), x(t) \notin \mathcal{S} \\ f_S(x(t)), x(t) \in \mathcal{S} \end{cases} \quad (32)$$

V. Numerical Results

A. The Toolbox of Level Set Methods

The major challenge of dynamic programming is the computation of the value function, i.e., finding the solution of the HJI PDE's. In this work, the computations were performed using Ian Mitchell's Toolbox of Level Set Methods (ToolboxLS) [12], [13]. The ToolboxLS provides a set of solvers for the numerical solution of the Hamilton-Jacobi equation in its several forms. It is implemented as a set of Matlab™m-files.

The ToolboxLS solves the following HJI PDE:

$$\frac{\partial}{\partial t}\Phi(x,t) + \min[0, H(x, \nabla\Phi(x,t))] = 0 \quad (33)$$

$$\Phi(x,0) = g(x) \quad (34)$$

where

$$H(x,p) = \max_a \min_b p \cdot f(x,a,b). \quad (35)$$

and $g(x)$ is the signed distance to a given target set. Notice that this is not the standard HJI PDE usually considered in dynamic programming. It is through the level sets of $\Phi(x,t)$ that the ToolboxLS computes the value function associated to static HJI PDE's of the form

$$\max_b \min_a \nabla V(x) \cdot f(x,a,b) + l(x,a), x \in \mathbb{R}^n \setminus \mathcal{S} \quad (36)$$

$$V(x) = 0, x \in \partial\mathcal{S} \quad (37)$$

Furthermore, in order to accomplish that, one has to feed the toolbox with the following Hamiltonian:

$$H(x,p) = \max_b \min_a \frac{\nabla V(x) \cdot f(x,a,b)}{l(x,a)} \quad (38)$$

Then, when using the toolbox, the value function associated to the original static HJI PDE is given by

$$V(x) = \min\{t : \Phi(x,t) \leq 0\}. \quad (39)$$

See [12] for additional technical details. Problems 1 and 3 presents us with a time-dependent HJI PDE. Our approach is to cast them as static HJI PDE's. This is done by augmenting the state of the original problem with a new state variable s with dynamics $\dot{s} = 1$ and initial condition $s(0) = 0$, and making the change of variable $t = s$ in the HJI PDE.

In what follows, all computations were performed using the fifth order upwind weighted essentially non-oscillatory scheme for the spatial derivatives and the third order strong stability preserving Runge-Kutta scheme for explicit time integration. The Hamiltonian is approximated using a Lax-Friedrichs scheme.

B. The path-following problem

A vehicle with constant surge $u = 1$ m/s and maximum angular velocity $r_{\max} = 0.26$ rad/s is assumed in the numerical examples. We assume $|c_v| < 0.25$ m/s and $r_v = 0$ rad/s (straight line following). These parameters were chosen in order to mimic the dynamics of the Autonomous Underwater Vehicle (AUV) described in [14] in a typical operational scenario. The expression for the running cost is

$$l(x,a) = K_d d^2 + r^2 \quad (40)$$

A maximum cross-track error of 2 meters is imposed; therefore, an obvious initial choice for $\mathcal{R}$ is $[-2,2] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$. It must be remarked that the method is completely general and these parameters can be easily modified.

The ToolboxLS requires the user to provide the analytical expression of the Hamiltonian. For problem 1, the Hamiltonian takes the following expression:

$$\begin{aligned} H(x,p_1) =& p_d u \sin(\psi_r) + |p_d| c_{v,\max} \\ &+ |p_{\psi_r}|(r_{v,\max} - r_{\max}) + p_s \end{aligned}$$

where $p_1 = \begin{bmatrix} p_d & p_{\psi_r} p_s \end{bmatrix}'$ and u is the constant surge velocity. The computational domain was defined to be an hypercube that encloses $\mathcal{R} \times [-T_1, 0]$ and its interface with the forbidden set. This domain was discretized by a grid of $47 \times 97 \times 21$ points (the forbidden set is discretized by 3 grid points in each dimension). The toolbox iterates until the solution converges to a fixed point. Some trial and error might be necessary in order to choose a sufficiently high value for T_1 such that (18) is verified and $\mathcal{S}$ can be accurately retrieved. Notice that the computation must be repeated for each new value of T_1. The boundary of $\mathcal{S}$ can be easily identified on Fig. 3 and Fig. 5.

For problem 2, a larger computational domain was considered. Ideally, all the state space would be considered. Of course this is not possible. Therefore, we compute the solution for a partition of the state space, trying to gain some insight about the global solution. We used the entire range for the ψ_r state variable and the interval $[-20, 20]$ for the d state variable. In order to avoid errors due to the finite resolution of the grid, the

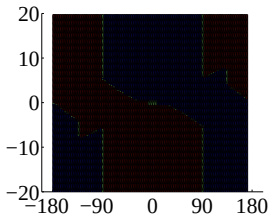

Fig. 2. Control law to reach $\mathcal{S}$ in minimum time, for the considered scenario. Vertical axis is for state x, in meters, and horizontal axis is for state ψ_r, in degrees. For instance, if $d > 8$ and $|\psi_r| < 90$ then $f_{\text{mt}}(x) = r_{\max}$.

target set is a subset of $\mathcal{S}$. The Hamiltonian for this problem is

$$H(x, p_2) = p_d u \sin(\psi_r) + |p_d| c_{v,\max} - |p_{\psi_r}|(r_{\max} - r_{v,\max})$$

where $p_2 = \begin{bmatrix} p_d & p_{\psi_r} \end{bmatrix}'$. In this case, the algorithm iterates (i.e., proceeds with the time integration) until the minimum time is computed for each point of the domain. The computation of an horizon of n seconds for a grid of 80×51 nodes took roughly $n/3$ seconds in terms of real time, on a Intel Core 2 Duo T7250 based system. Increasing the number of nodes by a factor of 16 increased the computation time by a factor of approximately 32.

The optimal control law (26) is derived through the numerical derivative of the resulting value function such that

$$f_{\text{mt}}(x) = -\text{sign}\left(\frac{\partial}{\partial \psi_r} V_{\text{mt}}(x)\right) r_{\max} \tag{41}$$

The image of the control law for the considered example is represented on Fig. 2. Analysis of (41) shows that the optimal control must be either $-r_{\max}$ or $r_{\max}$ except on those cases where V_{mt} reaches a minimum with respect to ψ_r. The inspection of Fig. 2 shows that, for $|d| > 8$, it is trivial to identify and define an explicit expression for the switching surface between those modes of operation. Therefore, in the implementation of the controller, we only use the numerical data of the value function for $|d| \leq 8$. Actually, it would also be possible to compute a polynomial approximation of the switching surface for $|d| \leq 8$, but we found that implementing the optimal control law as a lookup table was an acceptable solution for the final implementation.

The Lax-Friedrichs scheme requires the user to provide an estimate of the maximum $|\frac{\partial}{\partial p_i} H(x, p)|$ for each dimension i in order to control the accuracy of the numerical scheme. For problem 3, the considered Hamiltonian is affected by the term $l(x, a)^{-1}$, which may become very large (or even infinity) for small values of d or r (see (38)). These singularities would impact the computation time and the accuracy of the results. Therefore, we add a constant term ϵ to the running cost $l(x, a)$. In order to retrieve the value function associated to the original running cost, it is necessary to subtract ϵt from the obtained value function. Therefore, for problem 3, the following local minimization must be performed for each grid point in each iteration:

$$H(x, p_1) = \min_r (K_d d^2 + r^2 + \epsilon)^{-1}(p_s + p_d u \sin(\psi_r) + |p_d| c_{v,\max} + |p_{\psi_r}| r_{v,\max} + p_{\psi_r} r)$$

The assumed time horizon (the third dimension) was 44 seconds. The computation runs until all points of the invariant set are reached at time $t = -T$. The computation time is directly related to the maximum cost at $t = -T$. The computation took 2 hours and 45 minutes for a grid of $43 \times 93 \times 45$ points.

The expression for the optimal control law defined in (31) is

$$f_S(x) = -\frac{1}{2}\frac{\partial}{\partial \psi_r} V_S(x), \left|\frac{\partial}{\partial \psi_r} V_S(x)\right| \leq 2 r_{\max} \tag{42}$$

$$f_S(x) = -\text{sign}\left(\frac{\partial}{\partial \psi_r} V_S(x)\right) r_{\max}, \text{otherwise} \tag{43}$$

Fig. 3 illustrates a sequence of level sets of $f_S(x)$ for $K_d = 0.01$. The thick line represents the optimal trajectory for initial $d = 1.5$ meters and $\psi_r = 10$ degrees, assuming no disturbances; in this case, the trajectory converges to the origin. Fig. 4 shows the corresponding $d(.)$ and $r(.)$ signals. Assuming the worst case disturbance (i.e., $c_v(.) = \text{argmax}_b \nabla V_S \cdot f(x(.), f_S(x(.)), b)$), it can be observed that the trajectories no longer converge to the origin. The equilibrium point corresponds to the vehicle getting an orientation such that the projection of its longitudinal velocity on the N axis cancels the disturbance. Fig. 5 shows these trajectories, for $K_d = 1$.

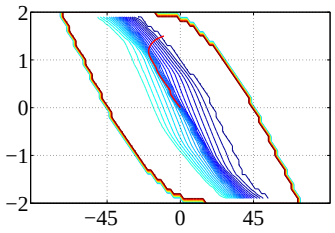

Fig. 3. Level sets of $f_S(x)$ for the considered scenario. Vertical axis is for state x, in meters, and horizontal axis is for state ψ_r, in degrees. The thick line represents the system trajectory for initial $d = 1.5$ meters and $\psi_r = 10$ degrees, with no disturbances. The outer contour delimits $\mathcal{S}$.

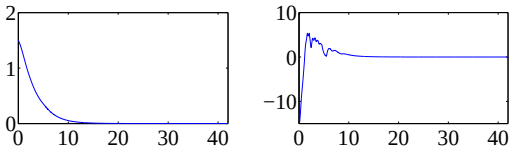

Fig. 4. Evolution of $d(t)$ (meters) and $r(t)$ (degrees/second) for the trajectory represented on Fig. 3.

VI. Controller implementation

The controllers were also implemented in C++ and tested, for the control of an AUV operating at constant depth, in the simulation environment described in [14]. This simulation environment allows the user to test the control system using a virtual vehicle whose motion is described by a 6 degree

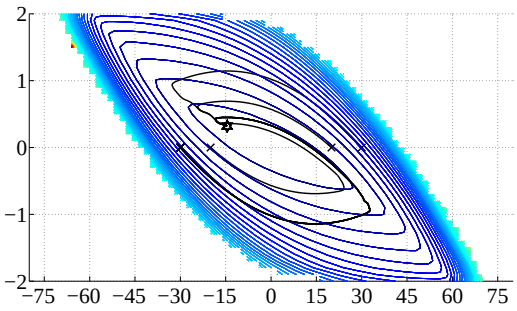

Fig. 5. System trajectories (thick lines) starting from different initial states, with the vehicle subject to the worst-case disturbance signal $c_v(.)$. The level sets of $V_S(x, -T)$ (thin lines) are shown in the background. Vertical axis is for state x, in meters, and horizontal axis is for state ψ_r, in degrees.

of freedom nonlinear model. In general, the entire simulation may even run on the final computational system. The control rate was set to 40Hz, the same used in the real time control system of the mentioned above AUV. It was observed that the controllers were able to drive the vehicle into the invariant set and to keep the vehicle inside it, even when constant currents were considered. We remark that the computation times mentioned in the previous section are related only to the computation of the value functions V_S and V_{mt}. The actual controller consists only of the implementation of (32) which is much less computationally demanding. The implementation uses three 2-dimensional arrays. The elements of the first array are nonzero for those points (d, ψ_r) inside $\mathcal{S}$ and zero otherwise. The elements of the second array are the optimal controls to reach the safe set in minimum time, i.e., the discrete version of (26). Finally, the third array contains the optimal controls to keep the vehicle inside the safe set, i.e., it represents the discrete version of (31). At each control cycle, the software checks whether the vehicle's state is inside $\mathcal{S}$ or not and then uses the corresponding optimal control law. Due to the finite resolution of the discrete value functions, the actual control is interpolated from a stencil composed of the closest four grid nodes. With carefully chosen discretizations of the value functions, these controllers are simple enough to be implemented on most types of micro-controllers, being the amount of non-volatile memory the main concern. If a storage device is used, the arrays may be stored as files. The files can be loaded entirely to main memory or read on demand at each control cycle. This makes it possible to use large arrays (meaning higher accuracy) on machines with small amounts of main memory. In those cases, the files are stored on a solid-state drive (SSD) since the seek times of this type of devices are small enough to meet the real-time requirements of most applications.

VII. Conclusion

The described approach provides an efficient solution for optimal path-following. The major computational burden resides in the offline computation of the optimal controls. The memory requirements for the real time application are easily met by most current computational platforms. For scenarios where the vehicle departs far from the desired path, the control strategy can be seen as an integrated approach for optimal path generation and following. Most work in the literature tackles this two problems independently. However, even in cases for which there is an analytical expression for the optimal path, it might be simpler to implement a lookup table based algorithm than some set of cumbersome expressions, such as those in [15].

It can be argued that this approach is too conservative since in practice the disturbances are not actively fighting the normal system operation. However, we remark that the method can take in account all the information provided by the state estimation. Therefore, the adversarial input is bounded by the degree of uncertainty of the system's state: the more the confidence in the state estimate, the lesser the effect of the adversarial input.

References

[1] C. Samson, "Path following and time-varying feedback stabilization of a wheeled mobile robot," in *Int. Conf. ICARCV'92*, Singapore, 1992, pp. RO–13.1.

[2] O. Sordalen and C. Canudas de Wit, "Exponential control law for a mobile robot: extension to path following," May 1992, pp. 2158 –2163 vol.3.

[3] P. Encarnação, A. Pascoal, and M. Arcak, "Path following for marine vehicles in the presence of unknown currents," in *Proceedings of SYROCO'2000 - 6th International IFAC Symposium on Robot Control*, vol. II, Vienna, Austria, 2000, pp. 469–474.

[4] D. Nelson, D. Barber, T. McLain, and R. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519 –529, June 2007.

[5] G. Indiveri, M. Aicardi, and G. Casalino, "Nonlinear time-invariant feedback control of an underactuated marine vehicle along a straight course," in *Proceedings of the 5th IFAC Conference on Manoeuvring and Control of Marine Crafts, MCMC 2000*, Aalborg, Denmark, August 2000, pp. 221–226.

[6] M. Aicardi, G. Casalino, G. Indiveri, A. Aguiar, P. Encarnação, and A. Pascoal, "A planar path following controller for underactuated marine vehicles," in *Proc. 9th Mediterranean Conference on Control and Automation*, Dubrovnik, Croatia, 2001.

[7] L. Lapierre and B. Jouvencel, "Robust nonlinear path-following control of an auv," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 2, pp. 89 –102, April 2008.

[8] R. Bellman, *Dynamic programming*. Princeton University Press, 1957.

[9] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhauser, 1997.

[10] M. Bardi, T. E. S. Raghavan, and T. Parthasarathy, Eds., *Stochastic and differential games: theory and numerical methods*, ser. Annals of the International Society of Dynamic Games. Basel, Switzerland: Birkhäuser Verlag, 1999, vol. 4.

[11] A. B. Kurzhanskii and P. Varaiya, "Dynamic optimization for reachability problems," *Journal of Optimization Theory & Applications*, vol. 108, no. 2, pp. 227–51, 2001.

[12] I. M. Mitchell, "A toolbox of level set methods," UBC Department of Computer Science, Tech. Rep. TR-2007-11, June 2007.

[13] ——, "The flexible, extensible and efficient toolbox of level set methods," *Journal of Scientific Computing*, vol. 35, no. 2-3, pp. 300–329, June 2008.

[14] J. E. da Silva, B. Terra, R. Martins, and J. B. de Sousa, "Modeling and simulation of the lauv autonomous underwater vehicle," in *13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics*, Szczecin, Poland, August 2007.

[15] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.