Control and Intelligent Systems, Vol. 39, No. 4, 2011

A DYNAMIC PROGRAMMING BASED PATH-FOLLOWING CONTROLLER FOR AUTONOMOUS VEHICLES

Jorge Estrela da Silva* and João Borges de Sousa**

Abstract

The problem of path following for autonomous vehicles under adversarial behaviour is considered. The objective is to keep the cross-track error to the reference path inside a given tolerance interval. The adversarial behaviour models system uncertainty and unknown or poorly estimated bounded disturbances to ensure that the concept of weakly invariant set is used, *i.e.*, the set of states that the vehicle may enter while ensuring that the cross-track error will never exceed the tolerance interval. 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, there is one different control law for each mode. The control laws are synthesized using a dynamic programming approach. Simulation results with a full nonlinear dynamical model illustrate the performance and robustness of the control strategy.

Key Words

Dynamic programming, path following, autonomous vehicles, robust control

1. Introduction

Operations with autonomous vehicles often entail the following of prescribed paths such as roads, aerial corridors or simple concatenations of way-points. The main characteristic of path following control relies on its independence with respect to time, as opposed to trajectory tracking problems, where the desired position is a function of the time variable. This time independence provides additional freedom which can be explored to provide a control strategy with increased robustness.

The path following problem has been studied for wheeled mobile robots (see [1] and [2] for early approaches), under-actuated marine vehicles (see [3] for instance) and aerial vehicles (see [4] for instance). 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. Approaches accounting lateral speed can also be found (see [3], [5] and [6] for examples on marine vehicles). However, determination of an accurate dynamic model for the lateral dynamics may be a difficult task. In [6], the authors describe an approach to add some robustness with respect to model uncertainty for underwater vehicles. In [7], the authors disregard the dynamics of lateral speed and model it as a bounded disturbance. The approaches may also differ on whether the path curvature must be known or not. In [7], the authors develop a controller that may follow arbitrary paths (with bounded curvature) without *a priori* knowledge of the curvature profile.

Most path following approaches seek perfect following of the reference path. However, 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 following a path that the vehicle cannot determine very accurately. Moreover, the disturbances may also be able to prevent stabilization at the origin, as will be verified here. Finally, if the control system is implemented in a computer-based platform, limitations due to the finite sampling rate and discretization of the actuator commands make stabilization to a single point (in the Lyapunov sense, *e.g.*, [8]) an impossible task. 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. In [7], the authors derive a control law with guaranteed bounded error and convergence. However, the performance of the proposed control law is not discussed.

Several other control strategies can be found in the literature, including for the integrated path following and obstacle avoidance problem (see also [9]–[11] and references therein). However, the authors are not aware of any

* Institute of Engineering of Porto, Porto, Portugal, e-mail: jes@isep.ipp.pt
** Faculty of Engineering of Porto University, Porto, Portugal; e-mail: jtasso@fe.up.pt
Recommended by Prof. Fakhri Karray
(DOI: 10.2316/Journal.201.2011.4.201-2318)

work describing a practical procedure for the implementation of optimal closed loop path-following controllers with provable bounded error.

This paper presents a methodology for the automated design of robust and practically optimal path following controllers with guaranteed bounded error for paths with bounded curvature. The methodology is based on the numerical solution of the dynamic programming equations [12], [13]. It is well known that the dynamic programming principle provides a sufficient condition for global optimality. When numerical methods are used, the derived controllers are optimal up to the accuracy allowed by the discrete nature of the computation (this is designated here as *practical optimality*). In what concerns robustness, bounded environmental disturbances and model uncertainty are accounted for in the design stage as adversarial inputs with worst case behaviour (min-max approach). For that purpose, the problem is formulated in the framework of deterministic differential games [14], [15]. This approach leads to more conservative results than stochastic approaches (*e.g.*, the LQG problem).

The approach described in this work is independent of the system model. However, to reduce the computational burden, a simple kinematic model is considered. This is a common simplified model for underactuated vehicles. The impact of neglecting the dynamics for the angular and lateral velocities is investigated with numerical simulations. Moreover, the results described here can be seen as a first benchmark for developments with more accurate models.

The paper is organized as follows. Section 2 presents some background on the subject of dynamic programming. Section 3 describes the system model. Section 4 formulates the path following problem as an optimal control problem. Section 5 describes the design of the optimal state feedback controller in the framework of dynamic programming. Section 6, presents simulation results with a numerical example. Finally, in section 7, some conclusions are drawn.

2. Background

Apart from some changes in notations, the material presented in this section can be found in [13] and [16]. Consider the following cost functional:

$$J(t_0, x_0, a, b) = \Psi(x(t_f)) + \int_{t_0}^{t_f} L(x(\tau), a(\tau), b) d\tau \quad (1)$$

subject to:

$$\dot{x}(t) = f(x(t), a(t), b(t)) \quad (2)$$

$$x(t_0) = x_0 \quad (3)$$

$$x(t) \in \mathbb{X} \quad (4)$$

where $x_0 \in \mathbb{R}^n$ is a given initial state; $L(x, a, b)$ and $\Psi(x)$ are the running and terminal costs, respectively; (2) describes the system dynamics, *i.e.*, the system flow at state $x(t)$ when subject to inputs $a(t)$ and $b(t)$; a is drawn from $\mathcal{U}_a$, the space of measurable input sequences such that $a(t) \in U_a$; likewise, b, the adversarial input, is drawn from $\mathcal{U}_b$, the space of measurable input sequences such that $b(t) \in U_b$; $\mathbb{X}$ defines the state constraints.

The optimal control problem (OCP) is defined as follows:

$$\sup_{\beta \in \Delta_b} \inf_{a \in \mathcal{U}_a} J(t_0, x_0, a, \beta[a]) \quad (5)$$

where $\Delta_b : \mathcal{U}_a \rightarrow \mathcal{U}_b$ is the set of nonanticipating strategies for the adversarial input b. Basically, this corresponds to choosing, among all possible actions, the action for which the maximum cost that can be imposed by the adversarial is minimal (min–max). On the other hand, it is assumed that the adversarial knows this strategy and acts accordingly. For the general case, this is the only way to ensure the minimization of the "losses" imposed by the adversarial input. Note that the adversarial input will be used to model the effect of disturbances and model uncertainty.

Consider also the following variation of the OCP presented above where:

$$t_f = \min_t x(t) \in \mathcal{T} \quad (6)$$

and $\mathcal{T}$ is a given target set. This version of the OCP has a static solution, in the sense that it does not depends on t_0.

The fundamental object of the dynamic programming approach is the value function. By the same reasons discussed for (5), the value function corresponding to the *upper value* of the differential game is considered:

$$V(t_0, x_0) = \sup_{\beta \in \Delta_b} \inf_{a \in \mathcal{U}_a} J(t_0, x_0, a, \beta[a]) \quad (7)$$

The value function can also be defined in a constructive form. This is done by the dynamic programming principle (DPP). The DPP for deterministic discrete-time systems is expressed as follows:

$$V(t, x) = \inf_{a \in U_a} \sup_{b \in U_b} \left\{ \int_0^{\Delta} L(y_\Delta(x, \tau, a, b), a, b) d\tau + V(t + \Delta, y_\Delta(x, \Delta, a, b)) \right\}, t < t_f \quad (8)$$

with terminal condition $V(t_f, x) = \Psi(x)$; $y_\Delta(x, t, a, b)$ is the state of the system at time t, starting from $x(0) = x$, when subject to a constant inputs a and b during a period Δ. For the static case, the DPP is simplified to

$$V(x) = \inf_{a \in U_a} \sup_{b \in U_b} \left\{ \int_0^{\Delta} L(y_\Delta(x, \tau, a, b), a, b) d\tau + V(y_\Delta(x, \Delta, a, b)) \right\}, x \notin \mathcal{T} \quad (9)$$

with boundary condition $V(x) = \Psi(x), x \in \mathcal{T}$.

Given a time-independent value function $V(x)$, the corresponding optimal control can be determined in state feedback form in the following way:

$$f(x) \in \arg\min_{a\in U_a}\max_{b\in U_b}\left\{\int_0^{\Delta} L(y_\Delta(x,\tau,a,b),a,b)d\tau + V(y_\Delta(x,\Delta,a,b))\right\} \quad (10)$$

For a fully discrete implementation (*i.e.*, discrete time and discrete space), in general, $y_\Delta(x,\Delta,a,b)$ will not coincide with any grid node. Therefore, $V(y_\Delta(x,\Delta,a,b))$ is computed by interpolation of the value function at the neighbouring nodes of $y_\Delta(x(t),\Delta,a,b)$.

For analysis purposes, note that the limit of (10) as Δ goes to 0 is

$$f(x) \in \arg\min_{a\in U_a}\max_{b\in U_b}[\nabla V(x)\cdot f(x,a,b) + L(x,a,b)] \quad (11)$$

If $V(x)$ is non-differentiable, $\nabla V(x)$ must be interpreted as some form of generalized gradient [17].

3. System Model

Consider the following planar vehicle model:

$$\dot{x} = u\cos(\psi) - v\sin(\psi) + c_x \quad (12)$$

$$\dot{y} = u\sin(\psi) + v\cos(\psi) + c_y \quad (13)$$

$$\dot{\psi} = r \quad (14)$$

where (x,y) describes the vehicle's position with respect to an 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 velocity 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, $|u| \gg |v|$.

Consider a frame with its origin at the nearest point of the path with respect to the vehicle (*i.e.*, with respect to the origin of the vehicle's body fixed frame). Assume there is some disambiguation scheme 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 the vehicle's angle relative to the path is defined 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:

$$\dot{d} = u\sin(\psi_r) + v\cos(\psi_r) + c_n \quad (15)$$

$$\dot{\psi_r} = r - (u\cos(\psi_r) - v\sin(\psi_r) + c_t)\kappa \quad (16)$$

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 a system parameter.

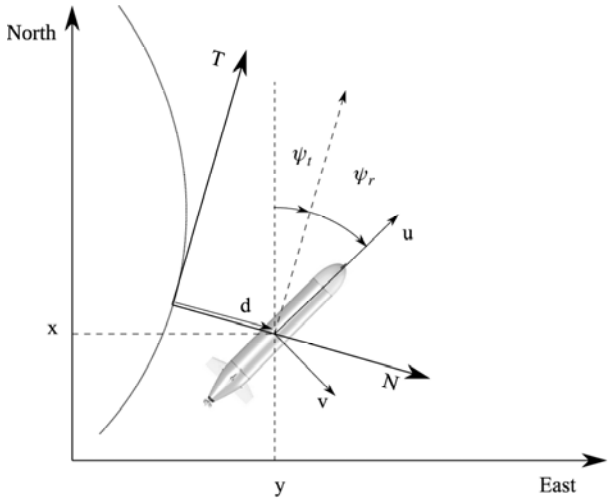

Figure 1. Coordinate system (North–East–Down convention).

On the other hand, for vehicles with no lateral actuation and no sideslip constraint (*i.e.*, that may skid), the same cannot be assumed for v. In these cases, the dynamics of v may have a considerable impact in the system performance. 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. To cope with this model uncertainty and keep the system model simple, a bounded virtual input c_v is considered; this input may take values in $[-c_{v,\max}, c_{v,\max}]$ such that

$$c_{v,\max} \geq \|v(\cdot)\cos(\psi_r(\cdot)) + c_n(\cdot)\|_\infty \quad (17)$$

Additionally, the following input is considered:

$$r_v \in [-r_{v,\max}, r_{v,\max}] \in [-U_{t,\max}/R_{\min}, U_{t,\max}/R_{\min}] \quad (18)$$

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 modelling 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 [14], [15], these can be modelled as adversarial inputs. Under the stated assumptions, the model becomes as follows:

$$\dot{x} = f(x,a,b) = \begin{cases} u\sin(\psi_r) + c_v \\ r - r_v \end{cases} \quad (19)$$

where $x = [d \;\; \psi_r]'$, $a = [r]$ and $b = [c_v \;\; r_v]'$.

4. Problem Formulation

Consider the path-following model (19). The control design for this system is driven by a set of state constraints and by the running cost for regular operation. Regular operation means that the vehicle is respecting the state constraints. The state constraints are the following:

- $|d(t)| \leq d_{\max}$, where $d_{\max}$ is the maximum acceptable cross-track error.
- $|\psi_r(t)| \leq \pi/2$; this constraint enforces forward motion along the path.

or, in a more compact form,

$$x(t) \in \mathcal{R} \equiv [-d_{\max}, d_{\max}] \times [-\pi/2, \pi/2] \tag{20}$$

Whenever possible, the vehicle should follow the path as closely as possible. On the other hand, for some inspection applications (*e.g.*, when using a camera), it is important to minimize the occurrence of fast changes in the vehicle's heading (*i.e.*, high values of r). These requirements lead to the following expression for the running cost:

$$L_{IH}(x, a, b) = d^2 + \psi_r^2 + K_r r^2 \tag{21}$$

In practice, one cannot assume that the system will always meet the state constraints defined in (20): unforeseen disturbances may drive the system outside $\mathcal{R}$; the system may start outside $\mathcal{R}$; finally, even if the system starts inside $\mathcal{R}$, there may be no trajectory inside $\mathcal{R}$ for all the desired time horizon. The last point is central to this approach. To meet the constraint (20) at all times, the system state must be brought to the interior of the weakly invariant set $\mathcal{S}$. The weakly invariant set is defined in the sense of, *e.g.*, [18]: the set of states from which the system is still able to respect the state constraints afterwards.

Whenever the system state is outside $\mathcal{S}$, the controller must drive the system to the interior of $\mathcal{S}$ in minimal time. Thus, the following state feedback control strategy is defined:

$$a(x) = \begin{cases} f_{\text{mt}}(x), x \notin \mathcal{S} \\ f_S(x), x \in \mathcal{S} \end{cases} \tag{22}$$

The strategy comprises two modes of operation, one corresponding to "travelling into $\mathcal{S}$ (in minimal time)" and the other to "staying inside $\mathcal{S}$". The control system selects the appropriate feedback control law ($f_{\text{mt}}(x)$ and $f_S(x)$) for each mode of operation.

5. Control Synthesis

5.1 A Specification of the Control Laws

To derive $f_S(x)$ and $f_{\text{mt}}(x)$, two value functions are defined:

1. $V_S(x)$, associated to the infinite horizon OCP defined by (5) with $t_0 = -\infty$, $t_f = 0$, $L(x, a, b)$ given by (21), system dynamics given by (19) and state constraints given by (20);
2. $V_{\text{mt}}(x)$, associated to a minimal time to reach (MTTR) OCP defined by (5) with $t_0 = 0$, $t_f = \min\{t : x(t) \in \mathcal{T}\}$, $L(x, a, b) = 1$, system dynamics given by (19) and target set $\mathcal{T} \in \mathcal{S}$.

The control laws $f_S(x)$ and $f_{\text{mt}}(x)$ are obtained by applying the dynamic programming equation (10) to $V_S(x)$ and $V_{\text{mt}}(x)$, respectively.

Note that the general solution of the constrained infinite horizon OCP leads to a time-dependent value function $V_{IH}(t, x)$. However, it will be shown that the desired static value function, $V_S(x)$, can be obtained from $V_{IH}(t, x)$. Analysis of the optimally controlled system shows that the trajectories always reach an equilibrium. This equilibrium can be either of two types: stable, of the form

$$\lim_{t \to \infty} (d(t), \psi_r(t)) = \pm(d_e(K_r), -\arcsin(c_{v,\max}/v)) \tag{23}$$

where $d_e(K_r)$ is some positive constant and the sign of the expression depends on the initial conditions or a limit cycle.

The type of equilibrium is strongly related to the running cost. For lower values of K_r, the worst case disturbance action will be to make the vehicle stay far from the path; to compensate the disturbance, the vehicle must point in the opposite direction, as given by (23). Note that the vehicle cannot risk steering too much towards the path because, that way, a change in the disturbance could lead to an even bigger cross-track error (due to the need of making a bigger turn in the opposite direction). For higher values of K_r, the solution is a limit cycle. From the point of view of the adversarial input, it will be profitable to induce the limit cycle if the cost due to r compensates the loss incurred by the passages through the origin; in that case, the adversarial input will change consistently its value and the controller will be forced to steer the vehicle constantly to avoid leaving $\mathcal{S}$. Finally, when no disturbance is present, the controller stabilizes the system to the origin.

Taking that into account, the value function associated to the constrained infinite horizon OCP will be of the form $V_{IH}(t, x) = V_S(x) + ct$, where c is associated to the average cost at the equilibrium. Therefore, $V_S(x)$ can be computed as for the static case by considering the running cost $L(x, a, b) = L_{IH}(x, a, b) - c$.

Given the exact $V_S(x)$, $\mathcal{S}$ could be determined by simple inspection, *i.e.*, $\mathcal{S} = \{x : V_S(x) \neq \infty\}$. However, due to inevitable inaccuracies in the the numerical solution, that is not possible. Instead, a discrete sub-approximation of $\mathcal{S}$ is considered. This sub-approximation is computed by simulation of the optimal trajectories associated to $V_S(x)$. The optimal trajectory emanating from each grid node in $\mathcal{R}$ is evaluated until it either reaches a neighbouring region of the equilibrium or leaves $\mathcal{R}$; if a trajectory leaves $\mathcal{R}$, the corresponding initial grid node is marked as not belonging to $\mathcal{S}$.

The minimum cross-track error imposed by the disturbances can be computed by repeating the computation of $\mathcal{S}$ with decreasing values for the constraint on d. For $r_v = 0$, this minimum can be easily derived by analysis of the system trajectories: assume that the vehicle is at an equilibrium, *i.e.*, $\psi_r = -\arcsin(c_{v,\max}/u)$; if the disturbance inverts its sign, then the cross-track

error changes until a new equilibrium is reached (this takes $t = 2\arcsin(c_{v,\max}/u)/r_{\max}$); integration of the system trajectory and centring around the origin results in $d = c_{v,\max}\arcsin(c_{v,\max}/u)/r_{\max}$. At the first sight, this minimum could seem a good candidate for the maximum allowed cross-track error. However, sensor noise could easily make the system leave the corresponding invariant set thus leading to the frequent execution of the more abrupt and undesirable MTTR control law.

5.2 Numerical Scheme

The numerical scheme for the computation of value functions is based on the iterative scheme from [19]. The value function is computed at the nodes of a regular grid that samples the desired region of the state space. On each iteration, the algorithm computes (9) for each grid node. In general, $y_\Delta(x, \Delta, a, b)$ will not coincide with any grid node. Therefore, $V(y_\Delta(x, \Delta, a, b))$ is computed by interpolation of the values at the neighbouring nodes. The current implementation uses bilinear interpolation.

In what concerns the solution of the ordinary differential equations (ODE) for the computation of $y_\Delta(x(t), \Delta, a, b)$, all computations were performed with a fixed step fourth order Runge–Kutta scheme. The algorithm also allows the choice of any of the variable step ODE solvers from the GNU Scientific Library (GSL) [20]. These can provide more accurate results, specially if large time steps are considered, at the expense of higher computation times. For systems of low dimension, as it is the case, it is possible to compute these trajectories just once and to store $y_\Delta(x(t), \Delta, a, b), \forall(a, b)$ in memory, thus avoiding repeating the computation in every iteration. In fact, $y_\Delta(x(t), \Delta, a, b)$ is stored in terms of barycentric coordinates with respect to the nodes used for the bilinear interpolation. For a grid with N nodes, a system with M possible input values and n state variables, the required memory for this cache is the size of the floating point data type (*e.g.*, 8 bytes) multiplied by $NM2^n$.

The algorithm was implemented as a multi-thread application to take advantage of the now common multiprocessor systems with shared memory (*e.g.*, the typical "dual core" personal computers fall on this classification). This is done by computing the value function for a different subset of the grid on each thread. Except for eventual bottlenecks on the memory bus, this algorithm scales linearly with the number of processors.

Our implementation keeps in memory the data from the current and previous iterations, which effectively duplicates the otherwise required memory to store the grid. The objective of this is twofold:

1. To allow comparison of the results between iterations. Define $\tilde{V}_k$ as the approximation of the value function at iteration k. Then, it is possible to establish different stopping condition based on $\tilde{V}_k - \tilde{V}_{k-1}$.
2. To eliminate potential data access conflicts between the application threads. Note that each thread may have to read data outside its own partition, namely near the boundaries. To avoid conflicts, at each iteration k, $\tilde{V}_k$ is computed based only on the results from the previous iteration ($\tilde{V}_{k-1}$).

For the MTTR problem, the nodes corresponding to the target $\mathcal{T}$ are kept with zero value at all iterations (they are not evaluated). For the computation of $V_S(x)$, no target is defined; the state constraints are enforced by setting $\tilde{V}_0 = 0, \forall x \in \mathcal{R}$ and $\tilde{V}_0 = K_\infty, \forall x \notin \mathcal{R}$, where K_∞ is a large constant such that $V(x) < K_\infty, \forall x \in \mathcal{S}$. Nodes with the value K_∞ are not evaluated.

5.3 Structure of the Solution

In what concerns the MTTR problem, the main issue resides in the fact that the control function must be defined for the unbounded domain $\mathbb{R} \times [-\pi, \pi]$. This could pose a problem, since it is not possible to numerically compute the value function for an unbounded domain. However, it can be observed that the optimal control for $|d| \gg u/r_{\max}$ is independent of d: if the vehicle is very far from the path, the optimal action is to turn such that the vehicle becomes perpendicular to the path and then travel that way until it reaches a certain distance of the target, *i.e.*, until $|d| \leq d_{\pi/2}$ where $d_{\pi/2}$ is some positive constant. Moreover, analysis of (11) shows that the control must be of bang–bang type. Thus, the optimal control for $|d| > d_{\pi/2}$ is given by the following expression:

$$r = \begin{cases} 0, & |\psi_r| = \pi/2 \\ \text{sign}(d) r_{\max}, & |\psi_r| > \pi/2 \\ -\text{sign}(d) r_{\max}, & |\psi_r| < \pi/2 \end{cases} \tag{24}$$

where ψ_r is assumed to be normalized to the interval $[-\pi, \pi]$.

Assuming that the target is the origin, the vehicle will have to leave the perpendicular direction at

$$d_{\pi/2} = (u + c_{v,\max}\pi/2)/r_{\max} \tag{25}$$

This is the minimum radius of curvature, $u/r_{\max}$, expanded to account the drift imposed by the worst case disturbance.

Further analysis shows that the analytical solution can also be obtained for $|d| \leq d_{\pi/2}$. The solution is also of bang–bang type, with two switching surfaces. The switching surface for $|\psi_r| \leq \pi/2$ is defined by the following expression:

$$d = -\text{sign}(\psi_r)\frac{u}{r_{\max}}((1 - \cos(\psi_r)) + c_{v,\max}|\psi_r|) \tag{26}$$

In this case, ψ_r is normalized to the interval $[-\pi/2, 3\pi/2]$; the switching surface for the remaining range of ψ_r can be defined as in (26) by considering $\psi'_r = \psi_r - \pi$ instead of ψ_r.

It must be remarked that this analytical solution assumes that the target is the origin. In the current problem, the desired target is the set $\mathcal{S}$. In this case, it is much harder or even impossible to find the exact analytical solution. However, (24) will still be valid for some threshold that can be inspected from the numerical solution and (25)

can be used as a guideline for defining the computational space. Moreover, this analysis shows that the optimal control for the MTTR problem must be either $-r_{\max}$ or $r_{\max}$ except for those singularities where $V_{\mathrm{mt}}(x)$ reaches a minimum with respect to ψ_r (on those cases, the optimal control is set to 0).

6. Numerical Example

6.1 Model Data

The following model data is assumed: constant surge $u = 1\,\mathrm{m/s}$, maximum angular velocity $r_{\max} = 0.26\,\mathrm{rad\,s^{-1}}$, $c_{v,\max} = 0.25\,\mathrm{m\,s^{-1}}$ and $r_v = 0\,\mathrm{rad\,s^{-1}}$ (straight line following). The maximum cross-track error is assumed to be 2 m; therefore, $\mathcal{R} = [-2, 2] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$. These parameters were chosen to mimic the motion of the autonomous underwater vehicle (AUV) described in [21] operating at constant depth. Regulation of the roll and pitch angles is assumed (*e.g.*, by PID controllers). Under these assumptions, the dynamics of the AUV may be approximated by the following equations:

$$\dot{v} = -1.90v - 1.05v|v| - 0.11r + 0.004r|r| + 0.57\delta_r \quad (27)$$

$$\dot{r} = -3.41v - 1.93v|v| - 4.56r - 1.93r|r| - 3.67\delta_r \quad (28)$$

These equations will be used to evaluate the impact of neglecting the sway (v) and yaw (r) dynamics in the DP controller (22). To accommodate both models, (19) and (27) and (28), the output of the DP controller will be denoted as r_d. In what concerns model (19), $r = r_d$; for models (27) and (28), r_d feeds the following regulator, thus closing the loop:

$$\delta_r = 0.166 \arctan(25(r_d - r)) \quad (29)$$

To perform the local optimization of (9) and (10), discrete inputs are assumed. In what concerns input r_d, this is in line with what happens in a digital implementation. For the computation of $V_S(x)$ and $f_S(x)$, the set U_a is composed of 31 equally spaced values ranging from $-r_{\max}$ to $r_{\max}$. For $V_{\mathrm{mt}}(x)$ and $f_{\mathrm{mt}}(x)$, $U_a = \{-r_{\max}, 0, r_{\max}\}$. On both cases, $U_b = \{-c_{v,\max}, c_{v,\max}\}$.

It must be remarked that the method is completely general and these parameters can be easily modified.

6.2 Computation of the Value Functions

All computations were performed on a Intel Core 2 Duo T7250 based system with 2 GB of 667 MHz DDR2 RAM. The numerical solver used two threads of computation (half of the data for each thread). The executable was generated using the GNU C compiler optimizations for the *core2* family of processors.

For the computation of $V_S(x)$ and $\mathcal{S}$, a 161×601 grid was used to sample $\mathcal{R}$. The time-step was 10 ms. The computation of $V_S(x)$ took 143 s of wall clock time (WCT) and 264 s of cumulative CPU time (CCT). Figure 2 represents $\mathcal{S}$ as derived from $V_S(x)$.

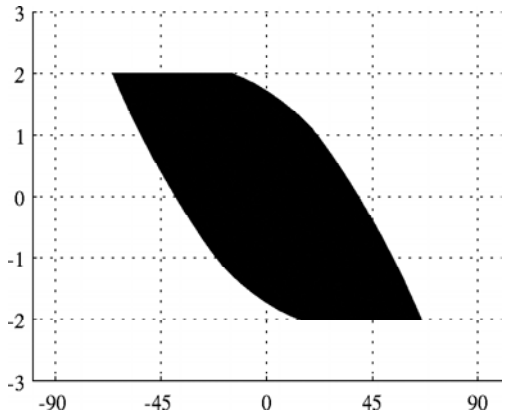

Figure 2. Maximal invariant set for $|d(t)| < 2$. Vertical axis is for state d, in metres, and horizontal axis is for state ψ_r, in degrees.

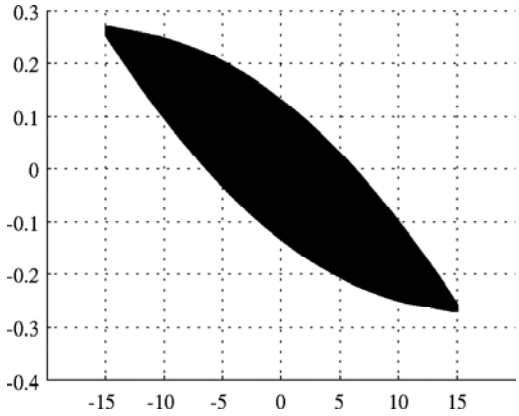

Figure 3. Minimal value of d for constrained operation. The invariant set collapses to the empty set if a lower value is chosen as a constraint for d. Vertical axis is for state d, in metres, and horizontal axis is for state ψ_r, in degrees.

For the considered numerical data, the minimal assured cross-track error, when departing from the origin, is approximately 0.26 m (see Fig. 3). Therefore, the considered 2 m give a good margin of tolerance.

For the computation of $V_{\mathrm{mt}}(x)$, the computational domain for the d state variable was set as $[-20, 20]$. This is much more than needed but it was chosen this way to clearly show that, above a certain value of $|d|$, the optimal control is independent of d. The target set was defined as square of 3×3 grid nodes centred at the origin. Note that the vehicle is just required to reach some set in the interior of the invariant set. For a grid of 161×121 and time-step of 100 ms, the WCT was 1 s WCT and the CCT was 2 s. The image of the minimal time control law, computed at the grid nodes using (10), is represented on Fig. 4. Inspection of Fig. 4 shows that for $|d| > 5.4$ the optimal control is in fact independent of d. Therefore, in this case, $d_{\pi/2} = 5.4$.

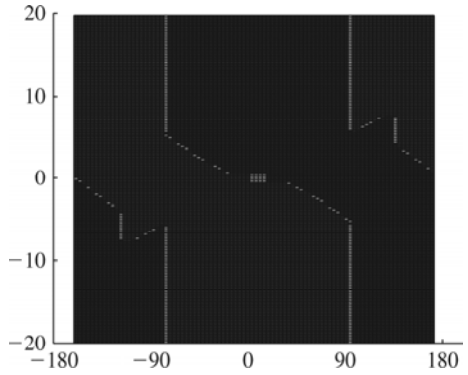

Figure 4. Control law to reach $\mathcal{T}$ in minimum time. Vertical axis is for state x, in metres, and horizontal axis is for state ψ_r, in degrees. For instance, if $d > 5.4$ and $|\psi_r| < 90$ then $f_{\text{mt}}(x) = r_{\max}$.

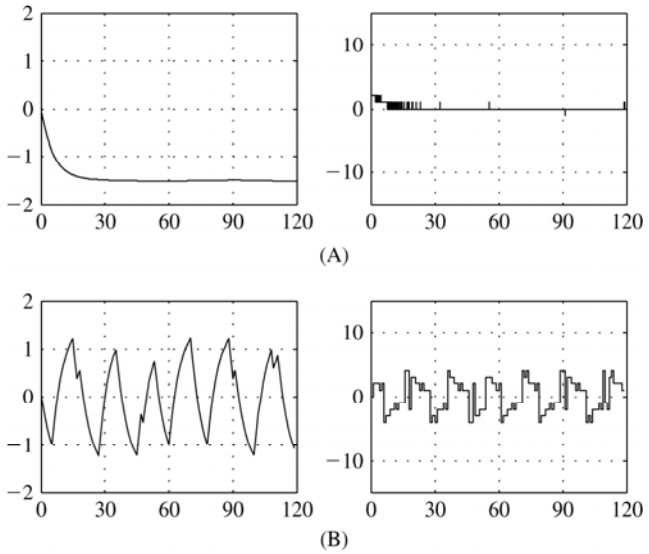

Figure 5. Evolution of $d(t)$ (metres) and $r_d(t)$ (deg s^{-1}) with $K_r = 1000$. The results at 100 Hz are similar to those obtained at 10 Hz. Horizontal axis is time in seconds. (A) 10 Hz control rate and (B) 1 Hz control rate.

6.3 Simulation Results

The simulation results concern only the operation inside $\mathcal{S}$ since this will be the most frequent mode of operation. The value function $V_S(x)$ was computed for two different values of K_r. With $K_r = 1000$, the cost function strongly penalizes the actuation effort. With $K_r = 0$, the cost function penalizes only the distance to the origin (see (21)). Three different control rates were considered: 100, 10 and 1 Hz. Figures 5 and 6 show the evolution of the cross-track error, and the corresponding input $r_d(t)$ for the kinematic model (19). Little or no difference is observed between the 100 and 10 Hz control rate (and therefore only the 10 Hz case is represented). For the 1 Hz sampling rate, the adversarial controller concludes that it may maximize the accumulated running cost by alternating c_v between $-c_{v,\max}$ and $c_{v,\max}$ at key points. This shows that at 1 Hz

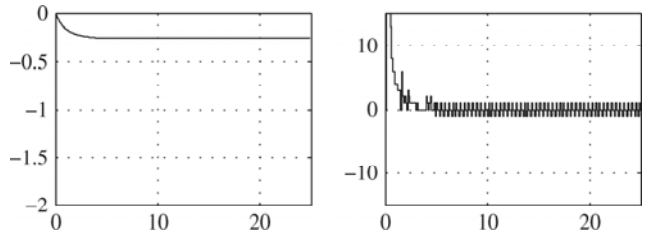

Figure 6. Evolution of $d(t)$ (metres) and $r_d(t)$ (deg s^{-1}) for the kinematic model with $K_r = 0$. The control rate is 10 Hz (similar performance is observed at 100 Hz). Horizontal axis is time in seconds.

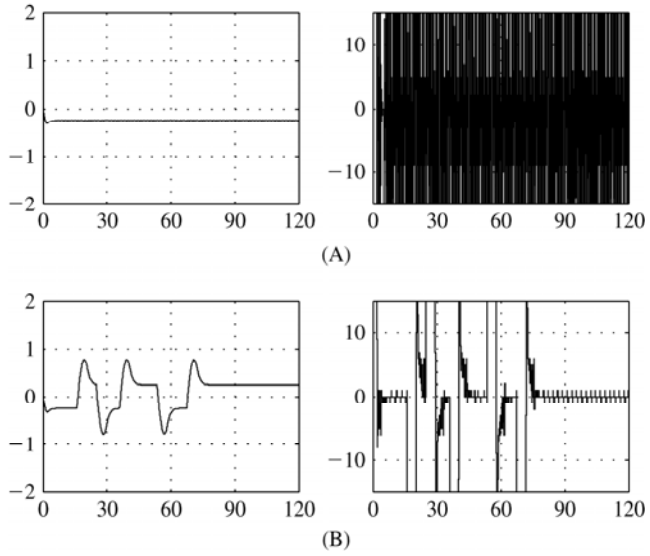

Figure 7. Evolution of $d(t)$ (metres) and $r_d(t)$ (deg s^{-1}) for the AUV dynamical model with $K_r = 0$. Horizontal axis is time in seconds. (A) 100 Hz control rate and (B) 10 Hz control rate.

control rate, the controller already departs from its ideal behaviour. Therefore, in what follows, only the 100 and 10 Hz control rates are considered.

Note that input c_v is always chosen using (10) in these simulations. Therefore these results represent worst case scenarios (under the model assumptions). For instance, a change of 0.5 m s^{-1} in the water velocity in a single control cycle implies a very turbulent environment.

The same control laws are employed for the simulation of the AUV dynamical model defined by (15), (16), (27)–(29). With $K_r = 1000$, no noticeable differences are observed. This is because the controller does not demand large variations for r_d (compare the graphs of $r_d(t)$ on Fig. 5(a) and Fig. 6). With $K_r = 0$ and a control rate of 100 Hz, the controller is still able to replicate the behaviour of the kinematic model (see Fig. 7(a)). With a 10 Hz control rate, the controller seems to be on the verge of being able to replicate the behaviour of the kinematic model (see Fig. 7(b)).

Finally, the performance of the system when subject to measurement noise is analysed. This measurement

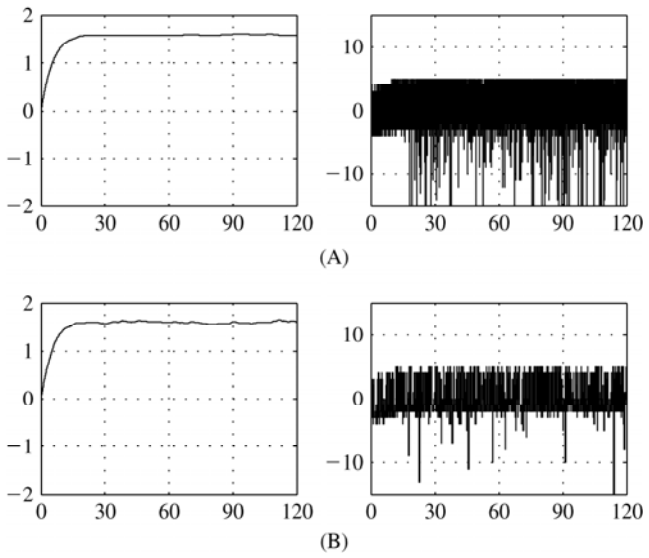

Figure 8. Evolution of $d(t)$ (metres) and $r_d(t)$ ($\deg\,\mathrm{s}^{-1}$) for the kinematic model with $K_r = 1000$. Measurement noise is drawn from the uniform distribution $[-0.25, 0.25] \times [-3, 3]$ (metres $\times$ degrees). Horizontal axis is time in seconds. (A) 100 Hz control rate and (B) 10 Hz control rate.

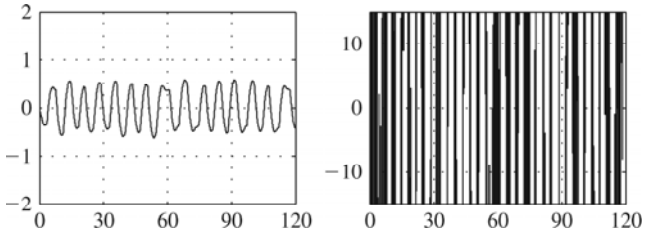

Figure 9. Evolution of $d(t)$ (metres) and $r_d(t)$ ($\deg\,\mathrm{s}^{-1}$) for the kinematic model with $K_r = 0$. Measurement noise is drawn from the uniform distribution $[-0.25, 0.25] \times [-3, 3]$ (metres $\times$ degrees). The control rate is 10 Hz (similar performance is observed at 100 Hz). Horizontal axis is time in seconds.

noise is always present even if only due to sensor quantization. The considered measurement noise is drawn from the uniform probability distribution on the interval $[-0.25, 0.25] \times [-3, 3]$. For $K_r = 1000$, the controller is able to compensate the noise by demanding higher values for r_d (Fig. 8). For $K_r = 0$, the controller drives the system to regions where the adversarial finds it profitable to change its input, leading to a limit-cycle (Fig. 9); not surprisingly, in this case the AUV dynamic model performs even worse, due to the increased demands on the r dynamics. However, in every case the required maximum cross-track error is never exceed. A less pessimistic scenario was considered, assuming a constant disturbance of $0.25\,\mathrm{m\,s}^{-1}$. In that scenario, the closed loop system does not amplify the measurement noise. However, the input $r_d(t)$ consist of high frequency chattering between the maximum and minimum allowed values. Similar results are obtained for the AUV model. In the absence of the limit-cycle, the AUV dynamics seem to be well approximated by the kinematic model.

The numerical results for this nonlinear system show what is well known for linear systems: emphasis on error control (higher penalization of deviations from the origin) leads to a system more sensitive to measurement errors and requiring higher control rates.

7. Conclusion

The numerical experiments show that the proposed approach provides an efficient solution for the considered problem. The control design is based on a simple kinematic model; this model is characterized by two easy to determine parameters (vehicle nominal velocity and maximum angular velocity). This way, dependencies on hard to determine model parameters are avoided and the computation time is kept small. It was shown, on a realistic scenario with worst case disturbances, that the derived controller was able to keep the cross-track error under the required tolerance even when the full dynamics of the vehicle were considered and the system was subject to measurement errors.

The controller can be automatically synthesized with respect to different model parameters and optimization criteria in few minutes, using common computers. The computing requirements for the real time execution of the controllers in the target system are much less demanding and are feasible for most current computational platforms. The main requirement is the extra storage space for the table of optimal controls; the actual computation consists of a bilinear interpolation at each control cycle.

The considered model assumes constant vehicle velocity. Operation at different cruise speeds can be tackled by generating a different controller (22) for each desired speed. Obviously, this leads to increased storage requirements in the target system.

As future work, the authors plan to investigate what can be gained by considering the full dynamical model in the computation of the value functions.

Acknowledgment

The first author was supported by a FCT grant under the PROTEC program. This work was partially funded by Fundação da Ciência e Tecnologia (FCT) under the NetV project and by the EU under the Control for Coordination project.

References

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

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

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

[4] D. Nelson, D. Barber, T. McLain, and R. Beard, Vector field path following for miniature air vehicles, *IEEE Transactions on Robotics*, *23*(3), 2007, 519–529.
[5] G. Indiveri, M. Aicardi, and G. Casalino, Nonlinear time-invariant feedback control of an underactuated marine vehicle along a straight course, *Proc. 5th IFAC Conference on Manoeuvring and Control of Marine Crafts, MCMC 2000*, Aalborg, Denmark, August 2000, 221–226.
[6] L. Lapierre and B. Jouvencel, Robust nonlinear path-following control of an auv, *IEEE Journal of Oceanic Engineering*, *33*(2), 2008, 89–102.
[7] M. Aicardi, G. Casalino, G. Indiveri, A. Aguiar, P. Encarnação, and A. Pascoal, A planar path following controller for underactuated marine vehicles, *Proc. 9th Mediterranean Conference on Control and Automation*, Dubrovnik, Croatia, 2001.
[8] J. Slotine and W. Li, *Applied nonlinear control* (Prentice-Hall: Englewood Cliffs, New Jersey, 1991).
[9] U. Nunes and L. Bento, Data fusion and path-following controllers comparison for autonomous vehicles, *Nonlinear Dynamics*, *49*, 2007, 445–462.
[10] N. H. H. M. Hanif and T. Z. Yaw, Modeling of high speed vehicle for path following and obstacles avoidance, *The 18th IASTED Int. Conf. Modelling and Simulation*, Anaheim, CA, USA, ACTA Press, 2007, 434–439.
[11] V. Cadanat, P. Soueres, and T. Hamel, A reactive path-following controller to guarantee obstacle avoidance during the transient phase, *International Journal of Robotics and Automation*, *21*, 2006, 256–265.
[12] R. Bellman, *Dynamic programming* (Princeton University Press: Princeton, 1957).
[13] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations* (Birkhauser: Boston, 1997).
[14] R. Isaacs, *Differential games; a mathematical theory with applications to warfare and pursuit, control and optimization* (John Wiley & Sons: New York, 1965).
[15] N. Krasovskii and A. Subbotin, *Game-theoretical control problems* (Springer-Verlag: New York, 1988).
[16] W.H. Fleming and H.M. Soner, *Controlled Markov processes and viscosity solutions* (Springer: New York, 2006).
[17] F. Clarke, Y. Ledyaev, R. Stern, and P. Wolenski, *Nonsmooth analysis and control theory*, ser. Graduate Texts in Mathematics vol. 178 (Springer-Verlag: New York, 1998).
[18] F. Blanchini and S. Miani, *Set-theoretic methods in control* (Birkhauser: Boston, 2008).
[19] E. Cristiani and M. Falcone, Fully-discrete schemes for the value function of pursuit-evasion games with state constraints, *Annals of International Society of Dynamic Games*, Birkauser, Boston, *10*, 2009, 178–205.
[20] B. Gough, *GNU scientific library reference manual, 2nd edition* (Network Theory Ltd., 2003).
[21] J.E. da Silva, B. Terra, R. Martins, and J.B. de Sousa, Modeling and simulation of the lauv autonomous underwater vehicle, *13th IEEE IFAC Int. Conf. Methods and Models in Automation and Robotics*, Szczecin, Poland, August 2007.

Biographies

Jorge Estrela da Silva received the M.Sc. degree in electrical and computer engineering in 2002. He is currently a Ph.D. student at the Electrical and Computer Engineering Department from Porto University in Portugal. He has been a lecturer at the Electrical Engineering Department from the Institute of Engineering of Porto since 2002. His research interests include optimal and robust control of nonlinear systems, numerical methods for dynamic programming and applications to unmanned vehicle systems.

João Borges de Sousa is a lecturer at the Electrical and Computer Engineering Department from Porto University in Portugal and the head of the Underwater Systems and Technologies (LSTS) Laboratory at Porto University. His research interests include unmanned vehicle systems, networked control, control and coordination of multiple dynamic systems, hybrid systems, systems engineering and control architectures for multi-vehicle systems. He has authored over 200 publications.