Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Optimal Switching Surfaces in Behavior-Based Robotics

Henrik Axelsson, Magnus Egerstedt, and Yorai Wardi

Abstract— **In this paper an optimal solution is presented for the problem of avoiding obstacles while progressing towards a goal for a single robot. In particular, the solution is obtained by allowing the robot to switch between a fixed number of behaviors and optimizing over what behaviors to use and when to switch between them. It is moreover shown that the structure of the switching law only depends on the distance between the obstacle and the goal. Hence, once initial simulations are done, a guard can be generated with a fixed structure, and, given that the robot knows the distance between the obstacle and the goal, it knows when to switch in order to execute the pre-computed (optimal) solution. Therefore the solution lends itself nicely to real-time implementations. Experiments moreover verify that the proposed methods transitions well onto a real robotic platform.**

I. Introduction

In the literature on robot navigation, two distinctly different approaches have emerged. The first approach that we will denote by the *reactive* approach (following the terminology in [1]) consists of designing a collection of behaviors, or modes of operations, such as avoid-obstacle or approach goal. These different behaviors are defined through a particular control law, dedicated to performing a specific task, and the robot switches between different behaviors as obstacles, landmarks, etc. are encountered in the environment. This way of structuring the navigation system has the major advantage that it simplifies the design task. Each controller is designed with only a limited set of objectives under consideration and no elaborate world maps are needed. Unfortunately, very little can be said analytically about such systems, and we contrast them with the second approach under consideration here, namely the *deliberative* approach. Here the motion is carefully planned out in advance and care can be taken to minimize energy consumption and so on. This plan-based approach has proved very useful in structured environments, e.g. in industrial settings, while unstructured environments pose a challenge. This is due to the fact that there is normally a hefty computational burden associated with path planning and optimal control. And, even if one is willing to pay this cost once, as soon as unmodeled obstacles are encountered, the cost will be incurred again.

In this paper we stay within the reactive navigation architecture but argue that optimality might still be relevant. Assuming that a number of control modes, or behaviors, have been designed, the question remains when to switch between them. This problem can be referred to as the guard design problem for hybrid systems, where a guard enables the transition between different modes of operation. Our approach is thus similar in spirit to the program developed in [2], where the guards were derived based on game theory to ensure safety in a multi-aircraft scenario. Formally, the state of the system evolves in mode i as $\dot{x} = f_i(x)$ until the guard $G_{ij}(x) = TRUE$, at which point the mode changes from i to j.

The particular problem that we will investigate in this paper is the problem of switching between *go-to-goal* and *avoid-obstacle* in an optimal manner. Previously proposed guards typically involve a safety distance Δ so that $\dot{x} = f_g(x)$ (subscript g denotes go-to-goal) as long as $||x - x_{ob}|| > \Delta$, where x_{ob} is the location of the obstacle ([3]-[5]). If $||x - x_{ob}|| \leq \Delta$ then $\dot{x} = f_o(x)$ (subscript o denotes avoid-obstacle) and hence the guard is defined through a circle centered at x_{ob} with radius Δ. One can thus view the optimal control problem as a problem of determining the optimal radius Δ. More generally, one can also optimize a parameterized surface $g_\alpha(x) = 0$, with respect to α (see [6] for a discussion about this topic).

Unfortunately, no guarantee can be given that we are optimizing over the right surface class (i.e. did we choose the right g_α), and in this paper we take an alternate route and view the optimization problem as a free timing control problem, and the starting point for this paper can be found in [7], where we consider a similar problem but in a simpler setting. In [7] we assumed that the robot was governed by single integrator dynamics and that the robot only encountered a single point-obstacle while moving to the goal point. Both these assumptions are clearly unreasonable when employing our switching law in any realistic settings, and we will not make these assumptions in this paper.

It should be noted, already at this point, that even though the solution to the guard generation problem is obtained by optimizing over a well-defined and known environment, the resulting navigation strategy will be transitioned onto a real robotic platform operating in an unknown environment. As such, it may no longer be optimal but rather correspond to a performance enhancing design strategy, as will be shown experimentally.

II. Behavior Based Robotics

In order to formally characterize the main design challenges associated with the coordination of a set of behaviors under consideration here, some comments about the basic behavioral building blocks must be made. Assume that the

This work was supported by the National Science Foundation through grant #0509064.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 U.S.A., {henrik,magnus,ywardi}@ece.gatech.edu

1-4244-0171-2/06/$20.00 ©2006 IEEE.

robot dynamics are given by

$$\dot{x} = f(x, u), \tag{1}$$

where $x \in \mathbb{X}$ is the robot state and $u \in \mathbb{U}$ is the control input. We identify individual behaviors with feedback laws defined with respect to a particular task, data source, or operating point. In other words, the set of behaviors available to us are given by the set $\{\kappa_1, \ldots, \kappa_k\}$, where each κ_i is a feedback mapping from $\mathbb{X}$ to $\mathbb{U}$.

As an example consider the unicycle dynamics

$$\begin{cases} \dot{x}_1 = & v\cos(x_3), \\ \dot{x}_2 = & v\sin(x_3), \\ \dot{x}_3 = & u, \end{cases} \tag{2}$$

where (x_1, x_2) is the position of the robot and x_3 is its heading. Assume that the translational velocity v is constant, and the angular velocity u is our control variable. In this paper we consider the following three behaviors

$$u_g = \kappa_g(x) = c_g(\phi_g - x_3), \tag{3}$$

$$u_{\circlearrowleft} = \kappa_{\circlearrowleft}(x, x_{ob}) = c_{ob}(\phi_{ob} - \frac{\pi}{2} - x_3), \tag{4}$$

$$u_{\circlearrowright} = \kappa_{\circlearrowright}(x, x_{ob}) = c_{ob}(\phi_{ob} + \frac{\pi}{2} - x_3), \tag{5}$$

where u_g is a standard "approach-goal" behavior, $u_{\circlearrowright}$ and $u_{\circlearrowleft}$ are "avoid-obstacle" behaviors that makes the robot move in a circle around the closest obstacle in the given direction (clockwise and counter-clockwise respectively). Furthermore, c_g and c_{ob} are the gains associated with each behavior and ϕ_g and ϕ_{ob} are the angles to the goal and nearest obstacle respectively. Both of these angles are measured with respect to the x-axis and can be expressed as

$$\phi_g = \arctan\left(\frac{x_{g_2} - x_2}{x_{g_1} - x_1}\right), \tag{6}$$

$$\phi_{ob} = \arctan\left(\frac{x_{ob_2} - x_2}{x_{ob_1} - x_1}\right), \tag{7}$$

where (x_{g_1}, x_{g_2}) and (x_{ob_1}, x_{ob_2}) are the Cartesian coordinates of the goal and the nearest obstacle respectively.

Having designed a set of behaviors, the initial task that we want the robot to achieve is to reach a given goal location $x_g \in \mathbb{R}^2$, while staying clear of a point-obstacle located at $x_{ob} \in \mathbb{R}^2$. Even though the point-obstacle assumption is clearly unrealistic in a real environment, it is straightforward to extending these results to the multi-obstacle case.

In order for the robot not to collide with any obstacles while moving towards the goal, the instantaneous cost $L : \mathbb{R}^2 \to \mathbb{R}^+$ includes a term that ensures that the robot goes towards the goal and one term that incurs a cost whenever the robot is close to an obstacle. To this end,

$$L(x(t)) = \rho||x_g - x(t)||^2 + \alpha e^{-\frac{||x_{ob} - x(t)||^2}{\beta}}, \tag{8}$$

where ρ is the gain of the goal attraction term, α is the gain of the obstacle avoidance term, and β is the shaping parameter for the effective range of the obstacle avoidance term. In this paper, the instantaneous cost is only a function of x_1 and x_2 but for notational convenience, we will still write $x - x_g$ when we mean $(x_{g_1} - x_1, x_{g_2} - x_2)^T$ whenever the dimensions are clear from the context. Different instantaneous costs can be imagined but, as the main focus of the paper is to show that feasible robotic controllers can be provided in real-time without giving up on optimality, we do not elaborate on that here.

The total cost associated with a particular trajectory then becomes

$$J = \int_0^T L(x(t))dt, \tag{9}$$

where T is the total time of the run.

As noted in the introduction, we will minimize the total cost by finding the best sequence of behaviors (given a bound on the number of switches) and optimal times when to switch between these behaviors. To formalize this, we let $\mathcal{K} = \{g, \circlearrowright, \circlearrowleft\}$ and let $\mathcal{K}^*$ denote the set of all finite length strings over $\mathcal{K}$. Hence, $\mathcal{K}$ is the index set of behaviors. Furthermore, assuming that we switch N times, we denote by τ_i, $i \in \{1, \ldots, N\}$ the time of the i'th switch and let $\bar{\tau} = (\tau_1, \ldots, \tau_N)^T$ be the vector of switching-times. Then the switching-time problem, denoted by P, can be cast as

$$\begin{aligned} P: \qquad & \min_{B,\bar{\tau}} J = \int_0^T L(x(t))dt \\ & \textit{s.t.} \quad \dot{x} = \begin{cases} f(x, u_{B_1}), & 0 \le t < \tau_1, \\ \vdots & \vdots \\ f(x, u_{B_N}), & \tau_{N-1} \le t < T, \end{cases} \\ & x(0) = x_0, \\ & 0 \le \tau_1 \le \ldots \le \tau_{N-1} \le T, \end{aligned} \tag{10}$$

where the dimension of $\bar{\tau}$ is induced by $B = (B_1, \ldots, B_N)$, $B \in \mathcal{K}^*$, and the robot starts at x_0.

III. Optimal Control Derivation

In order to obtain a (locally optimal) solution to Problem P we need to both find a good sequence of behaviors and the optimal times when to switch between them. To this end, we will approach the Problem P by answering the following two questions:

1) What is a good sequence of behaviors?
2) Given a sequence of behaviors B, how do we find the optimal switching-times?

In fact, the answer to Question 2 above was recently solved in [9], and a first attempt to answer Question 1 was provided in [10], both written by the authors. However, for the sake of completeness, we recall the major results in the following paragraphs.

We will start by assuming that the sequence of behaviors is fixed and present a solution to Question 2 above. In this case, we have that the cost is only a function of the switching-times, hence $J(\bar{\tau})$. To this end, an expression for the gradient of the cost with respect to the switching

vector, $\nabla J(\overline{\tau})$, is presented together with an algorithm that finds a locally optimal switching-time vector. An expression for $\nabla J(\overline{\tau})$ was derived in [9]. Recall that τ_i is the time when the robot switches between behavior i to behavior $i+1$. Assuming that B consists of a string of N behaviors and by defining $f_i = f(x, u_{B_i}),\ i = 1, \ldots, N$, the following assertion characterizes the derivatives $\frac{dJ}{d\tau_i}$, and hence the gradient $\nabla J(\overline{\tau}) = (\frac{dJ}{d\tau_1}, \ldots, \frac{dJ}{d\tau_N})$:

Proposition 3.1: [9] For every $i \in (1, \ldots, N-1)$ the following equation is in force,

$$\frac{dJ}{d\tau_i} = p(\tau_i)^T \left(f_i(x(\tau_i)) - f_{i+1}(x(\tau_i))\right), \qquad (11)$$

where the costate is given by the following backwards differential equation

$$\dot{p}(t) = -\left(\frac{df_i}{dx}(x(t))\right)^T p(t) - \left(\frac{dL}{dx}(x(t))\right)^T, \qquad (12)$$

for all $t \in [\tau_{i-1}, \tau_i)$ and for every $i \in \{1, \ \ldots, \ N\}$, with the given final condition $p(T) = 0$.

Once we have the gradient formula we will use it in the following gradient descent algorithm in order to get an optimal switching-time vector:

Algorithm 3.1: Gradient descent algorithm.
Given: A sequence of N behaviors.
Initialize: Choose a feasible initial point $\bar{\tau}_0 \in \mathbb{R}^{N-1}$. Set $i = 0$.
Step 1: Compute $x(t)$ forward from time 0 to T and $p(t)$ backward from time T to 0 through (12). Calculate the gradient through (11).
Step 2. Set $\bar{\tau}_{i+1} := \bar{\tau}_i - \gamma_i \nabla J(\bar{\tau}_i)^T$, set $i = i+1$, and go to Step 1.

Note that the choice of the step-size γ_i can be critical for the algorithm to converge. An efficient method among others is to choose the step-size according to the Armijo procedure [11]. This algorithm solves the problem posed in Question 2. Next, it will be shown how the gradient formula (11) can be used in order to find a good sequence of behaviors.

The problem of finding a good sequence of behaviors is a well studied subject [13], [14], [12], [15], [16]. The approach we will take to solve this problem is different from the above references in that we will use local information, i.e. the gradient formula (11), in order to find a good sequence of behaviors. This approach was presented in [17] and was further developed in [9] by the authors. In fact, given a sequence of behaviors, we will try to reduce the cost by searching to see if it is beneficial to insert a new behavior for a short period of time. To this end, we need to know how the cost changes when we insert a new behavior. Inserting a new behavior $u_b \in \{u_g, u_{\circlearrowleft}, u_{\circlearrowright}\}$ centered at time $t \in [0, T]$ for a temporal duration of λ seconds corresponds to adding two new switches, one at time $t - \lambda/2$ and another one at $t - \lambda/2$, and the new behavior between them. Denoting the length of the interval we are inserting by λ, we will view the cost as a function λ, $J(\lambda)_{b,t}$ where subscript b denotes the behavior we are inserting and t is the time we are inserting the behavior at. Assuming that x evolves according to u_g at time t an expression for the derivative of the cost with respect to length of the interval we are inserting was given in [9], with the conclusion that

$$\lim_{\lambda \downarrow 0} \frac{dJ_{b,t}}{d\lambda} = p(t)^T \left(f(x(t), u_b) - f(x(t), u_g)\right). \qquad (13)$$

If $\lim_{\lambda\downarrow 0} \frac{dJ_{b,t}}{d\lambda} < 0$ we add the two switching points, corresponding to inserting b at time t, and optimize over the switching-times. We are then guaranteed a descent in the cost J. This way, we can optimize the sequence of behaviors and an answer to Question 1 presented earlier have been presented.

Having presented expressions for $\nabla J(\overline{\tau})$ and the one-sided derivative we are now in position to derive the locally optimal solution to the go-to-goal, obstacle-avoidance problem. To this end, Algorithm 3.2 is presented:

Algorithm 3.2: Sequencing Algorithm.
Given: Problem P.
Initialize: Set $B = (g)$.
Step 1: Compute $x(t)$ and $p(t)$. Let

$$(b, t) := argmin_{b,t} \left\{ p(t)^T \left(f(x(t), u_b) - f(x(t), u_g)\right) \mid b \in \{g, \circlearrowleft, \circlearrowright\}, t \in [0, T] \right\}.$$

Step 2. If $\lim_{\lambda\downarrow 0} \frac{dJ_{b,t}}{d\lambda} = 0$, then $STOP$. Else, insert two new switching-times at t and update B and $\bar{\tau}$ to $B = (g, b, g)$ and $\bar{\tau} = (0, t, t, T)$.
Step 3. Use Algorithm 3.1 to optimize over $\bar{\tau}$. Go to Step 1.

Note that Algorithm 3.2 can insert several new behaviors if it enters Step 2 more than once. It turns out that with our particular choice of instantaneous cost and behaviors, we do not get any significant descent by performing a second insertion after we have optimized over $\bar{\tau}$ given by the first insertion. Therefore, we only consider performing one insertion.

The solution to P presented above is indeed locally optimal but it is not applicable to real-time robotics problems since we need to calculate $x(t)$ and $p(t)$ for each iteration in Algorithm 3.1, and this is time consuming. We would like to obtain a suboptimal solution where the optimal switches between the different behaviors are given by a geometric guard defined around the obstacle. Moreover, the structure of the guard should only depend on the distance between the obstacle and the goal. Hence, independent of where the robot starts, the guard should be the same.

In order to arrive at this result, it first needs to be proven that the solution is invariant along trajectories, given that the final time T is big enough. Invariance along trajectories means that the optimal solution starting at x_0 switches at the same point in the state space as the optimal solution

starting along the trajectory of the solution starting at x_0. To this end, Assumption 1 and Lemma 3.1 are presented.

Assumption 1: Given Problem P, assume that the instantaneous cost L and the dynamic representation $f(x,u)$, associated with the different behaviors, are continuously differentiable and that L is bounded from above. Furthermore, assume that there exists a finite time $t_1 < T$ such that the robot evolves according to a behavior κ_b, after time t_1, where $\frac{\partial f(x,u_b)}{\partial x}$ is negative definite.

Note that the statements in Assumption 1 are reasonable since we can assume that the goal and the obstacle are separated for the problem to be meaningful. Furthermore, we need to choose T big enough to guarantee that the robot reaches the goal, hence the robot will evolve according to κ_g after some time t_1. The assumption that the robot evolves according to a behavior b that satisfies the constraint that $\frac{\partial f(x,u_b)}{\partial x}$ is negative definite, should be seen as if we are requiring the robot to converge towards the goal position after a certain time. We are now in position to present Lemma 3.1:

Lemma 3.1: Under Assumption 1: For an initial state x_0, denote by $x(t)$ and by $p(t)$ the state and the costate trajectories obtained when $x(t)$ evolves according to (10) for a given switching vector $\bar{\tau}^1$. Denote by $\overline{x}(t)$ and by $\overline{p}(t)$ the state and the costate trajectories associated with the same system, but with an initial condition along the trajectory of $x(t)$, i.e. $\overline{x}(0) = x(\Delta)$ for some $\Delta \in (0,T)$. Furthermore, assume that the switching vector for the second system $\bar{\tau}^2$ is identical to $\bar{\tau}^1$ but with all switches increased by Δ. Then the state and the costate trajectories satisfy the following two equations

$$\overline{x}(t-\Delta) = x(t), \quad t \in [\Delta, T], \tag{14}$$

$$\lim_{T\to\infty} \overline{p}(t-\Delta) = p(t), \quad t \in [\Delta, T] \tag{15}$$

Proof: See [7]. ∎

Lemma 3.1 provides us with the result needed in order to show that Problem P satisfies the property of invariance along trajectories. To this end, assume that the robot starts at $x_0 \in \mathbb{R}^3$ and evolves according to κ_g. We denote the trajectory corresponding to this by $x^1(t)$. Likewise, let $x^2(t)$ be the state trajectory when we start at $x^1(\Delta)$ for some time $\Delta \in (0,T)$, but evolve according to the same behavior. From Lemma 3.1 we know that (14) and (15) are in force and hence $x^1(t) = x^2(t-\Delta)$ and $p^1(t) = p^2(t-\Delta)$ for all finite times $t \in [\Delta, T]$ as $T \to \infty$. Denote the cost associated with $x^1(t)$ by J^1, and the cost associated with $x^2(t)$ by J^2. We define $\overline{\Delta}$ to be the vector with the same dimension as $\overline{\tau}$ with each element equal to Δ. Since $\nabla J(\overline{\tau})$ and $\lim_{\lambda\downarrow 0} \frac{dJ}{d\lambda}$ only depend on the state $x(t)$ and the costate $p(t)$ it follows that

$$||\nabla J^1(\overline{\tau}) - \nabla J^2(\overline{\tau} - \overline{\Delta})|| = 0, \tag{16}$$

$$\left| \lim_{\lambda\downarrow 0} \frac{dJ^1_{b,t}}{d\lambda} - \lim_{\lambda\downarrow 0} \frac{dJ^2_{b,t-\Delta}}{d\lambda^+} \right| = 0, \tag{17}$$

are close to zero for all finite times $t \in (\Delta, T)$ and for all behaviors $b \in \{g, \circlearrowright, \circlearrowleft\}$, as $T \to \infty$.

The implication of (16) and (17) is that $\frac{dJ^1_{b,t}}{d\lambda}$ will be minimized at the same time $\overline{t} \in (\Delta, T)$ and with the same behavior b as $\frac{dJ^2_{b,t-\Delta}}{d\lambda}$. Hence, given that $\frac{dJ^1_{b,t}}{d\lambda} < 0$, the insertion of a new behavior will occur at the same point in the state space for both systems. After the insertions, $\bar{\tau}$ consists of two switching-times but we still have $\nabla J^1(\overline{\tau}) = \nabla J^2(\overline{\tau} - \Delta)$. Hence Algorithm 3.1 will terminate at two distinct switching vectors: $\overline{\tau}^1$ associated with $x^1(t)$ and $\overline{\tau}^2$ associated with $x^2(t)$ such that $\overline{\tau}^1 = \overline{\tau}^2 - \overline{\Delta}$. Moreover, the switches occur at the same point in the state space.

Thus, we have shown that the solution to problem P is invariant along trajectories, e.g. if we start along a trajectory it is optimal to switch at the same points in the state space independently of where on the trajectory we start. This is exactly the result needed in order to generate the guards for when to switch from the *go-to-goal* behavior to the *obstacle-avoidance* behavior.

IV. Guard Generation

In order to be able to generate the guard, we need to make sure that Assumption 1 is satisfied. In particular, we need to ensure that the robot evolves according to a behavior b that satisfies the constraint that $\frac{\partial f(x,u_b)}{\partial x}$ is negative definite when it arrives at the goal. Unfortunately, u_g does not satisfy the above constraint due to the fact that it has a constant translational velocity v.

Therefore, and in order for the robot to arrive smoothly at the goal, a new behavior is introduced. The behavior is active whenever the robot is closer to the goal than one meter, and the behavior is such that its translational velocity is decreasing proportional to the distance to the goal, until zero velocity at the goal. Furthermore, it is assumed that the robot is heading straight towards the goal meaning that $x_3 = \phi_g$, where ϕ_g is the angle to the goal, as defined in (7), and is assumed to be constant. This is a reasonable assumption since we assume that the goal and the obstacle are far apart, hence we can assume that the robot will be heading towards the goal once it is within a distance of one meter from the goal. The behavior is defined by

$$\begin{cases} \dot{x}_1 = & ||y - x_g|| \cos(\phi_g), \\ \dot{x}_2 = & ||y - x_g|| \sin(\phi_g), \\ \dot{x}_3 = & 0, \end{cases}$$

where $y = (x_1, x_2)^T$. Here the control variable is the translational velocity v instead of u, a departure from the setting of Section II, where v was assumed to be constant. Denoting the system above by $\dot{x} = f(x, u_q)$ and taking the derivative with respect to x, we get that

$$\frac{\partial f(x,u_q)}{\partial x} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{18}$$

where we see that $\frac{\partial f(x,u_q)}{\partial x}$ is only negative semi-definite. However, L does not depend on x_3 , hence $\frac{\partial L}{\partial x} = (\cdot, \cdot, 0)$,

has a zero as its third component. This guarantees that (15) holds even though $\frac{\partial f(x,u_q)}{\partial x}$ is not negative definite.

Having defined $f(x, u_q)$, we have shown that all the claims made in Assumption 1 can be fulfilled for our system. Left to do is to mention that the property of invariance along trajectories, for our problem, is still true for all practical purposes even though the final time T is finite.

We can now proceed to derive the suboptimal geometric guard for problem P. In order to get the data needed to generate the guard for a given distance between the obstacle and the goal we execute the following Algorithm:

Algorithm 4.1:

Init: Given Problem P, select a finite set of representative initial states X_0.

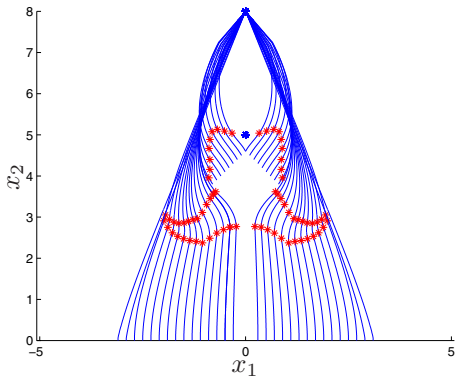
Step 1: If $X_0 = \emptyset$ STOP. Else, select an initial state $x_0 \in X_0$ and execute Algorithm 3.2 for one insertion, starting from this state. Save the switching positions obtained for the optimal trajectory. Remove x_0 from X_0. Go to Step 1.

Note that we assume that Algorithm 3.2 only performs one insertion in Algorithm 4.1.

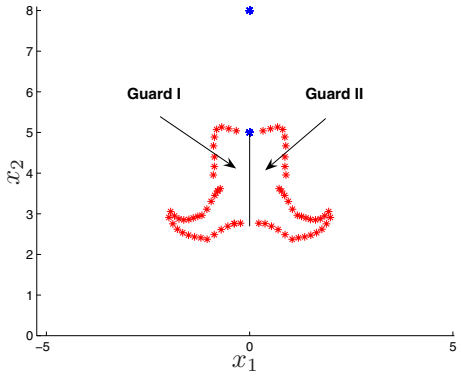
An example of the guard obtained when executing Algorithm 4.1 is shown in Figure 1. There, $x_{ob} = (0, 5)^T$, $x_g = (0, 8)^T$, and the values for ρ, α and β are as before. As can be seen, the guard lies in $\mathbb{R}^2$, corresponding to the robots position, even though $x \in \mathbb{R}^3$. Hence, the robots direction is implicit in Figure 1(a). The justification for this is that we assume that the robot is directed towards the goal when it encounters the obstacle. It should be noted that a guard in $\mathbb{R}^3$ can be generated, but for simplicity of our exposition, we do not consider that in this paper.

By examining Figure 1(a), we see that whenever the robot is inside the region denoted by Guard I in Figure 1(b), it is optimal to let the robot evolve according to $\kappa_{\circlearrowleft}$. Likewise, if the robot is inside the region denoted by Guard II in Figure 1(b), it is optimal to let the robot evolve according to $\kappa_{\circlearrowright}$. Everywhere else it is optimal to use κ_g. The regions where it is beneficial to evolve according to $\kappa_{\circlearrowleft}$ or $\kappa_{\circlearrowright}$ can approximately be described by a set of linear matrix inequalities together with some additional logic (the additional logic is needed since the guards are not convex). To this end, we let $A_{\circlearrowright,d} \cdot y \leq b_{\circlearrowright,d}$, where $y = (x_1, x_2)^T$, denote the linear matrix inequality corresponding to Guard I in Figure 1(b) and similar for Guard II.

At this point it should be noted that given our instantaneous cost and our behaviors, the structure of the guards depends *only* on the distance between the goal and the obstacle, denoted by d. As we change d it might be conceivable that we get a big change in the A and b matrices but simulation shows that this is not the case for our range of distances. Hence, we denote by $A_{\circlearrowright,d}$ and $b_{\circlearrowright,d}$ the linear matrix inequality associated with Guard I in Figure 1(b) where subscript d denoted the distance between the goal and the obstacle. Then, under the assumption that the obstacle



(a)



(b)

Fig. 1. State trajectories and associated guard structure: In the top figure the result for executing Algorithm 4.1 for a set of initial states is shown. In the bottom figure, an approximation of the associated guards is depict. Guard I corresponds to the region inside the stars and to the left of the vertical line between the goal and the obstacle, similar for Guard II.

and the goal lie on the x_2-axis, the guards associated with $\kappa_{\circlearrowleft}$ and $\kappa_{\circlearrowright}$, denoted by $G_{\circlearrowleft,d}$ and $G_{\circlearrowright,d}$ can be expressed as functions of $A_{\circlearrowleft,d}$, $b_{\circlearrowleft,d}$ and $A_{\circlearrowright,d}$, $b_{\circlearrowright,d}$ respectively. If the goal and obstacle do not line up, a simple rotation and translation is needed.

From the assumption that x_{ob} and x_g are far enough apart, we argued earlier that is was enough to consider the following three sequences of behaviors $B = (g)$, $B = (g, \circlearrowleft, g)$ and $B = (g, \circlearrowright, g)$. From this it follows that we never switch between $\kappa_{\circlearrowleft}$ and $\kappa_{\circlearrowright}$. Therefore the optimal solution is given in terms of the guards $G_{\circlearrowleft,d}$ and $G_{\circlearrowright,d}$ and the optimal solution can be cast on the form of Figure 2.

Once we have calculated $G_{\circlearrowleft,d}$ and $G_{\circlearrowright,d}$ for a range of distances d, this solution is suitable for realtime applications since the guards are easily stored and evaluated.

V. Implementation

In order to verify that the proposed navigation strategy perform well when implemented on a real robotics platform, it was tested on the Magellan Pro platform from iRobot with the setup shown in Figure 3, and we compared the

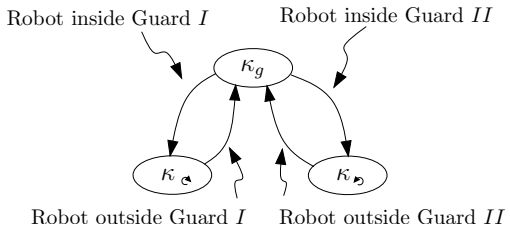


Fig. 2. The optimal solution to problem P given in terms of guards associated with each point obstacle encountered in the robots path.

Fig. 3. The experimental setup for testing our real-time reactive navigation method.

performance of the optimal strategy with a situation where the switching surface was given by a semi-circle, with a given radius, generated around the obstacle.

The results of these tests are shown in Figure 4. The resulting trajectories as well as the detected obstacles are plotted using the odometry and sensor readings from the robot. Figure 4 shows the resulting trajectory using a switching controller with our optimal switching surface and the standard semi-circle switching surface, with a radius of 0.5 meters.

The costs for each trajectory computed according to (9) were 16.64 and 10.64 for the semi-circle approach and our optimal guard approach respectively. Hence, we see that the optimal method gives a lower cost than the standard counterpart, as should be expected.

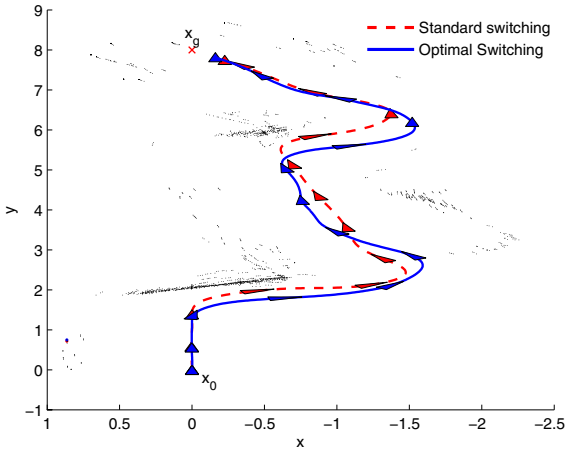


Fig. 4. The trajectories are plotted together with the sensor readings from the Magellan Pro for our Optimal Transitions System as well as standard circular guard switch-rules.

REFERENCES

[1] R. Arkin, *Behavior-Based Robotics*, MIT Press, Cambridge, Massachusetts, 1998.

[2] J. Lygeros, C. Tomlin and S. Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, **35**, pp. 349-370.

[3] J.H. Reif and H. Wang, "Social Potential fields: A distributed behavioral control for autonomous robots", *Robotics and Autonomous Systems*, **27**, 171-194, 1999.

[4] E.W. Large, H.I. Christensen and R. Bajcsy, "Dynamic Robot Planning: Cooperation through Competition", *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Vol. 35, pp. 2306-2312, 1997.

[5] D. Kortenkamp, R.P Bonasso and R Murphy, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, AAAI Press, Cambridge, Massachusetts, 1998.

[6] Y. Wardi, M. Egerstedt, M. Boccadoro and E. Verriest, "Optimal Control of Switching Surfaces", *in 43rd IEEE Conference on Decision and Control*, Atlantis, Bahamas, 2004.

[7] H. Axelsson, M. Egerstedt, and Y. Wardi, "Reactive Robot Navigation Using Optimal Timing Control", *Proceedings of the 2005 American Control Conference*, Portland, Oregon, 2005.

[8] M. Egerstedt. *Behavior Based Robotics Using Hybrid Automata*. Lecture Notes in Computer Science: Hybrid Systems III: Computation and Control, pp. 103-116, Pittsburgh, PA, Springer-Verlag, March 2000.

[9] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-Time Optimization for Switched Systems", *IEEE Transactions on Automatic Control*, **51**, no. 1, January 2006.

[10] H. Axelsson, Y. Wardi, and M. Egerstedt, "Transition-Time Optimization for Switched Systems", *IFAC World Congress*, Prague, The Czech Republic, July 2005.

[11] L. Armijo, "Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives", *Pacific Journal of Mathematics*, **16**, pp. 1-3, 1966.

[12] A. Guia, C. Seatzu, and C. Van der Mee. "Optimal Control of Switched Autonomous Linear Systems", In *Proceedings of the 40th Conference on Decision and Control*, pp. 1816-1821, Phoenix, Arizona, December 1999.

[13] M.S. Shaikh and P.E. Caines. "On the Optimal Control of Hybrid Systems: Optimization of Trajectories, Switching Times and Location Schedules", in *Proceedings of the 6th International Workshop on Hybrid Systems: Computation and Control*, Prague, The Czech Republic, 2003.

[14] M. Alamir and S.A. Attia, On Solving Optimal Control Problems for Switched Nonlinear Systems by Strong Variations Algorithms. *NOLCOS*, 2004.

[15] A. Rantzer and M. Johansson. "Piecewise Linear Quadratic Optimal Conrol", *Proceedings of the America Control Conference*, 1997.

[16] A. Bemporad, A. Giua, and C. Seatzu, "A master-slave algorithm for the optimal control of continuous-time switched affine systems", in *Proc. 41th IEEE Conf. on Decision and Control*, 2002, pp. 1976-1981.

[17] M. Egerstedt, Y. Wardi, and F. Delmotte. Optimal Control of Switching Times in Switched Dynamical Systems. In the *Proceedings of the IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[18] J.E. Hopcroft and G. Wilfong, "Motion of Objects in Contact", *The International Journal of Robotics Research*, **4**, no. 4, pp 32-46, 1986.