Smooth Path Planning and Control for Mobile Robots

Shangming Wei and Miloš Žefran
Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607
{swei3,mzefran}@uic.edu

***Abstract*— In this article a new complete sensor based path planning algorithm for autonomous mobile robot is presented. The algorithm guarantees global convergence to the target while at the same time the generated path is smooth, a key requirement for implementation on real robots. The completeness of the proposed scheme is proved using techniques from hybrid control.**

I. Introduction

We are studying the path planning problem of an autonomous robot operating in a 2-dimensional surface with obstacles. The objective is to find a collision-free path from a given initial position to a predefined target point. A complete path planning algorithm should guarantee that the robot can reach the target if possible, or prove that the target can not be reached. There is also a very crucial performance consideration, i.e., the robot should move smoothly. This smoothness objective is of key importance especially for practical service robots, considering the physical limits of robot actuators, safety issues and that some service robots are required to carry sensitive loads.

Many researchers have addressed this problem. Many authors have considered a model with complete information [1]–[6], where the robot has perfect knowledge about the obstacles. The drawback of these approaches is that under many practical circumstances robot does not have access to complete information about the environment. In [7]–[9], different Bug algorithms were proposed. They are complete (the solution will be found if it exists, otherwise the algorithm indicates there is no solution), but in terms of smoothness their performance is not satisfactory: when the robot switches between different segments of the path it moves abruptly. An alternative to these algorithms is behavior based control architecture [10], [11]. The drawback of this approach is that the performance is difficult to quantify formally. In [12], a hybrid control approach was used to combine behaviors for mobile robots. The resulting robot response is satisfactory from both safety and performance perspectives. However, the approach is not complete and there are no guarantees that the robot can reach the target when the target is reachable. Therefore, to be used in practical robots, all these existing approaches need improvements. In this work, we combine ideas from [9] and [12] and propose an approach which guarantees global convergence and satisfactory performance. We also prove the global convergence of the algorithm using a hybrid control approach proposed in [13].

This research is partially supported by NSF grants IIS-0093581 and CCR-0330342.

II. Model

The environment in which the robot is operating is a 2-dimensional manifold. It has one starting point S and one target point T. It also has a finite number of static obstacles. We assume that the boundaries of the obstacles are smooth curves.

The robot is considered to be a point. It knows the coordinates of its current position C and the target T. It is equipped with limited range sensors, which provide readings 360° around it. Therefore, the sensor range is a disc of radius R centered at C. The robot can measure the distance to the closest obstacles which are within the sensor range. We assume that we can control the robot's translational and rotational velocities, v and ω. The equations describing the robot are thus:

$$\begin{aligned}\dot{x} &= v\cos\phi,\\ \dot{y} &= v\sin\phi,\\ \dot{\phi} &= \omega,\end{aligned}$$

where (x, y) is the position of the robot, and ϕ is its orientation. The setup corresponds to a path planning problem with incomplete information. The robot doesn't know the location and shapes of the obstacles until they are within the sensor range. Local sensory information is used for feedback control of the robot's motion. This

0-7803-8812-7/05/$20.00

model is attractive because many practical robots operate in unknown and changing environments.

III. Algorithm

A. Overview of TangentBug

The TangentBug algorithm was proposed by Kamon et al. in 1995 [9]. Here we restate the core part of the approach that will be needed by our algorithm.

TangentBug includes two basic behaviors, which are governed by the function $d(x, T)$, the distance from the robot's position x to the target T. The two behaviors are *moving-towards-target* and *boundary following*. In *moving-towards-target*, $d(x, T)$ decreases monotonically. And *boundary following* attempts to escape from a local minimum of $d(x, T)$. The robot constructs a local tangent graph (LTG) based on its sensors' immediate readings. The LTG is constantly updated and it is used by the robot to decide the next motion.

The steps of the TangentBug algorithm are:

1) Move along the locally optimal direction (the direction along the shortest path to the target according to current LTG) towards the target T until one of the following occurs:
   a) T is reached. Stop.
   b) It is found that moving in the locally optimal direction will drive the robot into a local minimum of $d(x, T)$. Go to Step 2.
2) Choose a boundary following direction. Move around the boundary using the LTG while recording $d_{followed}(T)$, the minimal distance along the followed obstacle's boundary to T and $d_{reach}(T)$, the minimal distance within the visible environment to T, until one of the following occurs:
   a) T is reached. Stop.
   b) The leaving condition $d_{reach}(T) < d_{followed}(T)$ holds. Go to Step 1.
   c) The robot completes a loop around the obstacle. T is unreachable. Stop.

The TangentBug algorithm is complete and generates paths that are short compared to other sensor-based motion planning algorithms. But the robot moves abruptly when it switches between the two behaviors (see example below), which is not desirable.

B. Smooth Path

In order to control a mobile robot to move smoothly, we need a method for producing and tracking low curvature paths. Such a method was proposed in [12]. There, cubic splines are generated to connect the robot and the target in *approach-target* behavior, and the robot and the obstacle in *approach-obstacle* behavior. Cubic splines have two desired features. First, they are paths of minimal curvature. And second, they are easy to generate online.

In [12] the following tracking algorithm was proposed. Let the general reference path, parameterized by s, be given by

$$\left.\begin{aligned} x_d &= p(s), \\ y_d &= q(s), \end{aligned}\right\} \qquad 0 \leqslant s \leqslant s_f. \tag{1}$$

The control objectives are

$$\begin{aligned} &\limsup_{t\to\infty} \rho(t) \leqslant \varepsilon_\rho, \\ &\limsup_{t\to\infty} |\phi(t) - \phi_d(t)| \leqslant \varepsilon_\phi \end{aligned} \tag{2}$$

where ε_ρ and ε_ϕ are positive numbers that can be arbitrarily small, $\rho(t) = \sqrt{(x_d - x)^2 + (y_d - y)^2}$, where (x, y) is the actual position of the robot, and ϕ and ϕ_d are actual and desired robot orientations.

The control algorithm proposed in [12] is:

$$\begin{aligned} v &= \gamma\rho\cos(e_\phi), \\ \omega &= ke_\phi + \dot{\phi}_d, \ k > 0, \end{aligned} \tag{3}$$

where both γ and k are positive, $e_\phi = \phi_d - \phi$ and $\phi_d = \arctan 2(y_d - y, x_d - x)$. We will directly use these expressions in our approach.

C. Our approach

The idea of our approach is to modify the TangentBug algorithm so that the generated paths are smooth. The

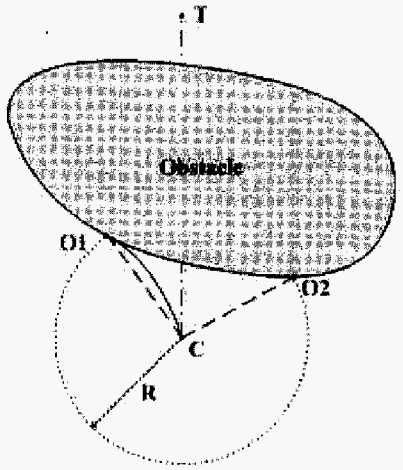

Fig. 1. When a blocking obstacle is detected, a cubic spline is generated to connect the robot's current position C and an LTG node O_1.

basic procedures of our algorithm are the same as those of TangentBug but we will replace abrupt, sudden motions in TangentBug with smooth ones. Such abrupt, sudden motions are generated after the robot decides to make shortcuts, that is, the robot needs to change its heading discontinuously.

During *moving-towards-target*, the robot's motions include both motion between obstacles and sliding along obstacle boundaries. Before an obstacle is detected, the robot moves along a straight line directly towards T. And on obstacle boundaries, the robot's path is also smooth (because of the assumption that all the obstacles' boundaries are smooth). Therefore, the only shortcut happens when a blocking obstacle is detected and the robot searches the LTG nodes to decide the locally optimal direction (see Figure 1). For the TangentBug algorithm, the robot's heading will change to the locally optimal direction (towards the node O_1 in Figure 1) of the LTG immediately.

In our approach, the robot does not change its direction suddenly. Instead, we use a cubic spline to connect the two points C and O_1 and let the robot follow the curve. While generating the cubic spline, we should control the robot's velocity so that the curve does not goes out of the area enclosed by line segments CO_1, CO_2 and the boundary arc O_1O_2 (Figure 1). In this way the generated path traverses the same key point (O_1 and C in this case) as TangentBug's path.

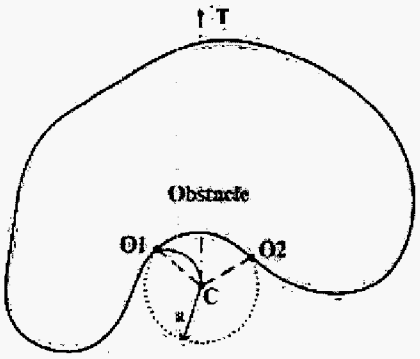

Fig. 2. When the robot switches to *boundary following* behavior, a cubic spline is generated to connect its current position C and an LTG node O_1.

In terms of convergence, our approach is the same as TangentBug. However, our path is smooth, which is a nice property that TangentBug does not have. We should note an extreme case here, i.e., the line segment CO_1 lies so close to another obstacle's boundary that to avoid that obstacle, the generated spline approximates the line segment CO_1. It means that in the worst case, where a smooth path is not possible, the path generated by our algorithm is the same as that by TangentBug.

When the robot finds that it will be trapped in the basin of attraction of a local minimum, it will switch from *moving-towards-target* behavior to *boundary following* behavior (see Figure 2). Here another shortcut takes place. In the same way as above, we use a cubic spline to connect C and O_1 and the robot will move along it.

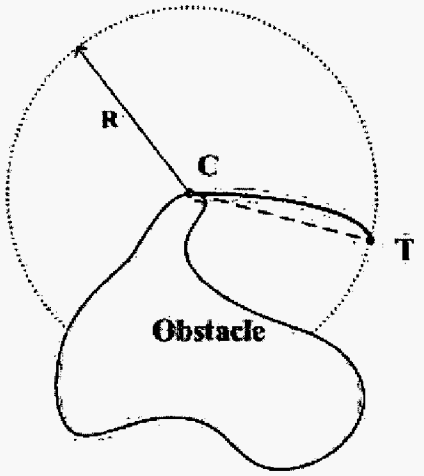

Fig. 3. When the robot leaves an obstacle's boundary, a cubic spline is generated to connect its current position C and the target T.

In *boundary following* behavior the robot makes use of the LTG to plan shortcuts relative to the boundary and for testing the leaving condition. Because we have assumed that all the obstacles' boundaries are smooth, there is no shortcut relative to the boundary. Therefore, the only shortcut in this behavior happens when the leaving condition (see Section III-A) is satisfied, the robot can leave the obstacle's boundary and switch to *moving-towards-target* behavior (see Figure 3). Again, the robot moves along the generated cubic spline to the target T.

Having dealt with all the sudden abrupt shortcuts, our approach can achieve satisfactory performance and at the same time retain guaranteed global convergence.

IV. Example

An example of our algorithm, compared to the TangentBug algorithm, is showed in Figure 4). The path planned by TangentBug is showed with dashed line. And the path planned by our algorithm is showed with solid

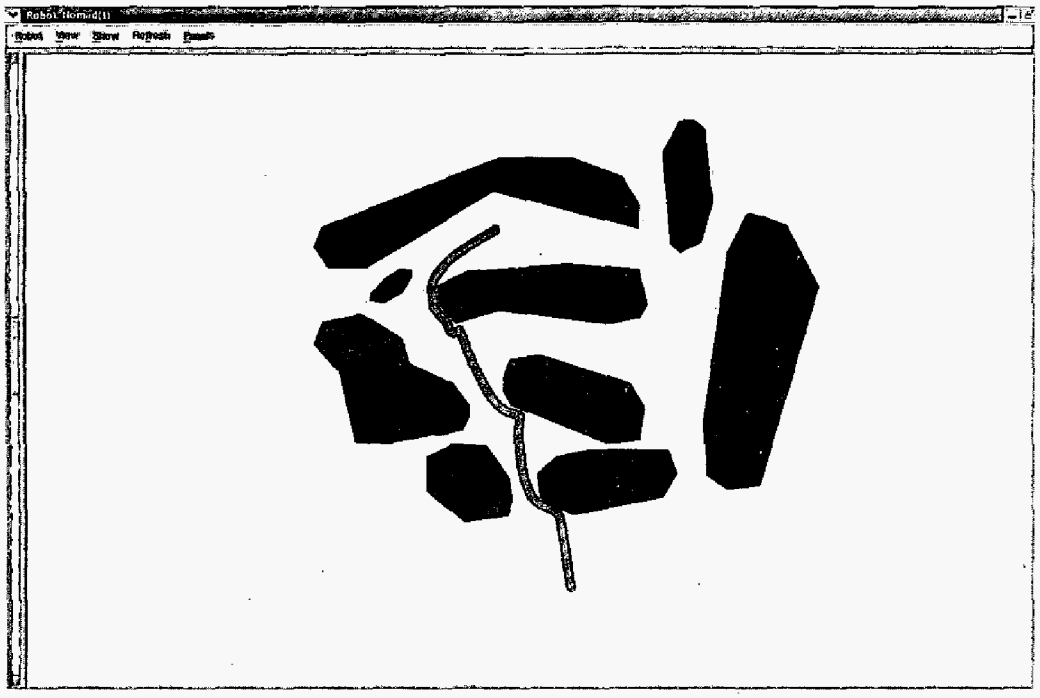

Fig. 5. Simulation result of the algorithm in Environment 1.

line. It can be seen from the figure that under the control of our algorithm, the target can be reached by a smooth path, instead the path planned by TangentBug has some abrupt switches.

A. Simulation

We tested our algorithm using Super Scout simulation software from Nomadic Technologies. Simulation results show that the algorithm generates smooth, optimal path for the robot and the robot always successfully reaches the target. Figure 5 and Figure 6 are two examples of simulation results. From the results we can see that the performance of the algorithm is satisfying.

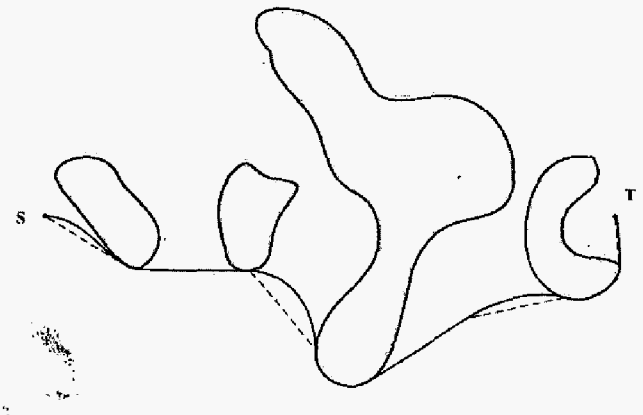

Fig. 4. An example comparing the paths generated by TangentBug and our algorithm.

V. Convergence Proof Using Partial Order

In this section we will prove the completeness of our algorithm using a hybrid control approach proposed in [13]. The robot switches between two behaviors, *moving-towards-target* and *boundary following*, so it can be modeled as a hybrid system. It was shown in [13] that convergence to a desired state for a hybrid system can be realized by imposing a partial order hierarchy different behaviors. If switches between the controllers are consistent with the partial order, system convergence towards the desired equilibrium set follows. The details can be found in [13]. We just restate the required proposition here:

Proposition 1: Let $\Sigma = (\Xi, \mathcal{X}, \mathcal{U}, \Upsilon, \mathcal{F}, \Phi)$ be a hybrid system, where Ξ is a (finite) set of discrete states, $\mathcal{X} = \{X_i\}_{i \in \Xi}$ where X_i are the continuous state spaces where different behaviors are defined, $\mathcal{U} \subset \mathbb{R}^m$ is the set of continuous inputs, $\Upsilon \subset \mathbb{Z}$ is the set of discrete inputs, $\mathcal{F} = \{f_i\}_{i \in \Xi}$ is a set of ($\mathcal{C}^1$) vector fields describing different behaviors, and $\Phi : \Xi \times M \times \Gamma \times \mathcal{U} \rightarrow \Xi$ is a function describing the discrete evolution of the system. The discrete input is selected by a discrete controller $\mathcal{S}$

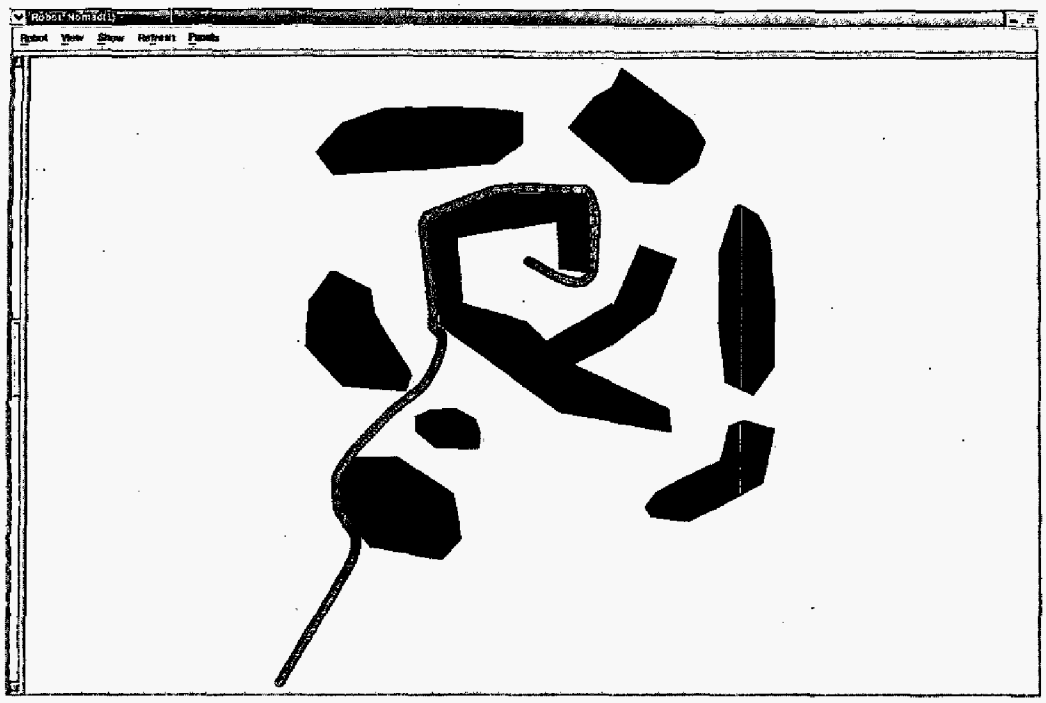

Fig. 6. Simulation result of the algorithm in Environment 2.

(also called *a switching scheme*) that at each state (i, x) selects a behavior (a discrete state) that should be active next. The switching scheme $\mathcal{S}$ formally defines a relation $\sigma(\Xi)$,

$$\sigma(\Xi) = \{(i, j) \mid \exists x \in X_i \text{ s.t. } \mathcal{S}(i, x) = j\} \quad (4)$$

where $\sigma(i, j)$ when it is possible to switch from manifold X_i (behavior i) to manifold X_j (behavior j). Assume we can construct a Lyapunov function V_n for behavior n. Take $\sigma^{\text{Trans}}(\Xi)$ and let $\preceq$ be a partial order within $\sigma^{\text{Trans}}(\Xi)$ that has n for the smallest element. Assume the switching scheme $\mathcal{S}$ has the following properties:

(1) There exists $L > 0$ such that $\mathcal{S}(n, x) = n$ for every $x \in B(E_n, L) \cap X_n$.

(2) There exists $\Delta > 0$ such that if $x(t)$ is a trajectory of Σ and X_i, $i \neq n$ is a manifold on which $x(t)$ evolves for an infinite amount of time, then for every T we can find $\tau > T$ so that $\mathcal{S}(v(t), x(t)) \prec i$ for every $t \in [\tau, \tau + \Delta]$.

(3) If a system switched from behavior n to some other behavior at time t_{off} and if t_{on} is the time when the system switches again to behavior n, then $V_n(t_{\text{off}}) \geq V_n(t_{\text{on}})$.

Then the equilibrium set E_n of the behavior n is globally attractive.

First we will prove global convergence of the TangentBug algorithm. And then we will show that our algorithm is the same as TangentBug with respect to convergence.

For the model studied in this paper, the equilibrium set E_n is the target T. The Lyapunov function V_n is the robot's distance to the target $d(x, T)$. The TangentBug algorithm includes two basic motion behaviors, *moving-towards-target* and *boundary-following*. Thus a partial order depicted in Figure 7 is prescribed. The switching scheme $\mathcal{S}$ is defined by the switching conditions

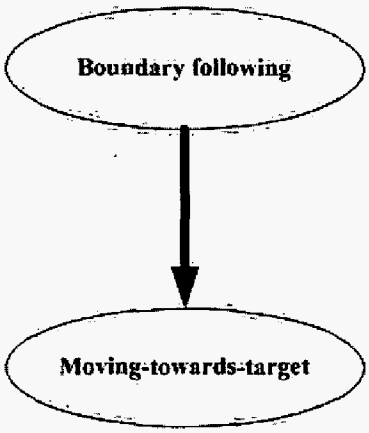

Fig. 7. Partial order between two behaviors.

described in section III-C. Let's assume that the target T is reachable. Now we show that all the three conditions in the proposition are satisfied. Condition (1) means that in the neighbor of T, the robot always switches to *moving-towards-target* behavior. This is assured by step 2 of TangentBug. Because we can always find a neighbor of T in which the leaving condition is satisfied and the robot moves towards the target. Condition (2) requires that after the robot stays in *boundary following* behavior for some time, it can always switch back to *moving-towards-target* behavior. This is satisfied because if the robot completes a loop in step2 and does not leave the boundary, the target is unreachable, which is a contradiction to the assumption. Condition (3) means that each time the robot switches to *moving-towards-target*, $d(x, T)$ is smaller than when the system last switched off the behavior. It is also true because the leaving condition of Step 2 guarantees this. So all the conditions of the proposition are guaranteed by the TangentBug algorithm. Under its control the target is globally attractive.

If the target is globally attractive under TangentBug, then its convergence is also guaranteed by our algorithm. This is because the only difference between these two algorithms is that TangentBug uses straight lines to connect key points, while our algorithm uses cubic splines. For any such straight line segment in TangentBug, we can always find a cubic spline to replace it. In the worst case, our algorithm's path is the same as TangentBug's. Both algorithms have the same steps, pass the same key points, make behavior switches at the same points. Therefore, our algorithm also guarantees that the robot can reach the target if the target is reachable.

VI. Conclusions

In this paper we proposed a smooth version of the TangentBug algorithm proposed in [9]. The algorithm has the same convergence property as TangentBug and replaces paths with abrupt switches with low curvature paths. Furthermore, we use the smooth control laws proposed in [12] to follow the generated paths. Smoothness of the robot path is especially important to practical mobile robots. Simulation results show that the proposed algorithm achieves satisfying performance. The correctness of the proposed algorithms is proved using a result from hybrid control described in [13].

One of the assumptions in the paper was that the obstacles' boundaries are smooth. Actually our approach can be modified to deal with any shape obstacles. The only modification that needs to be performed is that we have to detect the corners between smooth boundary segments and then use cubic splines to go around such corners.

References

[1] N. Nilsson, *Problem solving methods in artificial intelligence*. New York, NY: McGraw-Hill, 1971.

[2] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[3] R. A. Brooks, "Solving the find-path problem by representing free space as generalized cones," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 3, pp. 190–197, 1983.

[4] C. Yap, *Algorithmic motion planning*, ser. Advances in Robotics. Hillsdale, NJ: Lawrence Erlbaum Associates, 1987, pp. 95–143.

[5] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[6] J.-C. Latombe, *Robot Motion Planning*. New York, NY: Kluwer Academic Publishers, 1990.

[7] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica (New York)*, vol. 2, no. 4, pp. 403–430, 1987.

[8] V. J. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 5, pp. 1058–1069, 1990.

[9] I. Kamon, E. Rivlin, and E. Rimon, *A new range-sensor based globally convergent navigation algorithm for mobile robots*, ser. Proceedings - IEEE International Conference on Robotics and Automation. Minneapolis, MN, USA: IEEE, Piscataway, NJ, USA, 1996, vol. 1, pp. 429–435.

[10] R. C. Arkin, *Behavior-based robotics*. Cambridge, MA: MIT Press, 1998.

[11] D. Kortenkamp, R. P. Bonasso, and R. Murphy, Eds., *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. Cambridge, MA: MIT Press, 1998.

[12] M. Egerstedt and X. Hu, "A hybrid control approach to action coordination for mobile robots," *Automatica*, vol. 38, no. 1, pp. 125–130, 2002.

[13] M. Žefran and J. W. Burdick, "Stabilization of systems with changing dynamics by means of switching," in *IEEE Conf. on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 1090–1095.