PERGAMON

Automatica 38 (2002) 125–130

automatica

www.elsevier.com/locate/automatica

Brief Paper

A hybrid control approach to action coordination for mobile robots☆

Magnus Egerstedt[a], Xiaoming Hu[b,*]

[a]Division of Engineering and Applied Science, Harvard University, Cambridge, MA 02138, USA
[b]Division of Optimization and Systems Theory, Royal Institute of Technology, SE-100 44 Stockholm, Sweden

Received 12 October 1998; revised 28 May 2001; received in final form 19 June 2001

Abstract

In this paper, the problem concerning how to coordinate the contributions from concurrent controllers, when controlling mobile robots, is investigated. It is shown how a behavior based control system for autonomous robots can be modeled as a hybrid automaton, where each node corresponds to a distinct robot behavior. This type of construction gives rise to chattering executions, but it is shown how regularized automata can be used to solve this problem. As an illustration, the obstacle-negotiation problem is solved by using a combination of a robust path-following behavior and a reactive obstacle-avoidance behavior that move the robot around a given obstacle at a predefined safety distance. © 2001 Published by Elsevier Science Ltd.

Keywords: Mobile robots; Hybrid dynamic systems; Motion control; Path planning

1. Introduction

For mobile, autonomous robots the ability to function in, and interact with, a dynamic, changing environment is of key importance. A successful way of structuring the control system in order to deal with this problem is within a *behavior based* control architecture. (See for example, Arkin, 1998; Kortenkamp, Bonasso, & Murphy, 1998). The main idea is to identify different controllers, responses to sensory inputs, with desired robot behaviors. A behavior could, for instance, be *obstacle-avoidance* in which sonar information about a close obstacle should result in a movement away from that obstacle. This way of structuring the control system into separate behaviors, dedicated to performing certain tasks has the major advantage that it makes the system modular, which both simplifies the design process as well as offers a possibility to add new behaviors to the system without causing any major increase in complexity, as pointed out in Brooks (1986).

However, within this framework, a number of design issues need to be addressed. Those range from questions concerning the design of the individual behaviors to coordination issues. For instance, given a reactive obstacle-avoidance behavior, modeled as a repulsive potential field surrounding the obstacle, how should an approach-target behavior be designed so that it takes advantage of the fact that it is going to run in parallel with an obstacle-avoidance behavior? Furthermore, how should these behaviors be combined? Addressing this last question will be the main point of this paper. In other words, via a case study where the robot is negotiating obstacles, we will try to answer the question: *How should the different behaviors be combined so that the resulting robot response is satisfactory from both a safety and a performance perspective?*

2. Path following and obstacle negotiation

The specific problem that will be used to illustrate our point about action coordination is how to move a robot between two points in the plane. This *point-to-point motion* should be done so that the detection of an obstacle results in a repulsive potential field, acting on the platform when the robot is closer to the obstacle than a desired safety distance, d_{OA}, where the subscript stands for obstacle-avoidance. This behavior is an example of

☆This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Jurek Z. Sasiadek under the direction of Editor Mituhiko Araki. This work was sponsored in part by the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH and in part by TFR.

* Corresponding author.
E-mail addresses: magnuse@hrl.harvard.edu (M. Egerstedt), hu@math.kth.se (X. Hu).

0005-1098/02/$ - see front matter © 2001 Published by Elsevier Science Ltd.
PII: S0005-1098(01)00185-6

a so-called *reactive* obstacle-avoidance behavior. The word reactive, a commonly used one in the robotics community, is used here since the behavior can be thought of as a reflex. This is a reasonable safety strategy since the robot may be moving around in a highly unstructured world, where the occurrences of unpredicted, or unmodeled obstacles are very likely.

We assume that we can control the robot's translational and rotational velocities. Naturally, for platforms that do not give direct control over the velocities, one needs to design an actuator control so that these velocity controls are realized. The implementation could be just a static mapping, as in the car case, or a dynamic regulator. In those cases, the velocity controllers can be viewed as higher-level controllers.

An example of systems that give direct velocity control is the Nomadic 200 mobile platform, which is the experimental platform used in this work. For such systems, a unicycle model (see for example Ackermann, 1993) is quite standard:

$$\begin{aligned} \dot{x} &= v\cos\phi, \\ \dot{y} &= v\sin\phi, \\ \dot{\phi} &= \omega, \end{aligned} \tag{1}$$

where $(x, y)\in\mathbb{R}^2$ is the position of the robot, ϕ is its heading, and v and ω are the controlled translational and rotational velocities, respectively. Furthermore, if the sonars on the robot detect a point-obstacle[1] at $(x_{\mathrm{ob}}, y_{\mathrm{ob}})$ that is closer to the robot than the predefined safety distance d_{OA}, a standard, reactive control response is

$$\omega = C_{\mathrm{OA}} W_{\mathrm{OA}}(d)\,(\tilde{\phi} - \phi), \tag{2}$$

where $d = \sqrt{(x - x_{\mathrm{ob}})^2 + (y - y_{\mathrm{ob}})^2}$, $W_{\mathrm{OA}}(d) = 1/d^2$ if $d < d_{\mathrm{OA}}$ and 0 otherwise, and $\tilde{\phi} = \pi + \mathrm{atan2}(y_{\mathrm{ob}} - y, x_{\mathrm{ob}} - x)$. Here, C_{OA} is a constant weight, and the choice of C_{OA} should reflect the hardware constraint on the maximal angular velocity. The choice of d_{OA} should reflect the sensitivity and accuracy of the sensors.

3. A hybrid control approach

When adding a goal attraction behavior, defined in the same way as the obstacle-avoidance behavior except that we now have an attractive instead of a repulsive field, we get two different possible hybrid automata (see for example Lygeros, Tomlin, & Sastry, 1999) for describing the situation. This depends on whether the two behaviors are active simultaneously or not, as seen in Fig. 1, where $z = (x, y, \phi)$ and where we let $f_{\mathrm{OA}}(z)$ and $f_{\mathrm{GA}}(z)$ denote the full state, obstacle-avoidance and goal-attraction behaviors, respectively. Initially we let v be constant and we postpone the discussion about how to design v to Section 4.2.

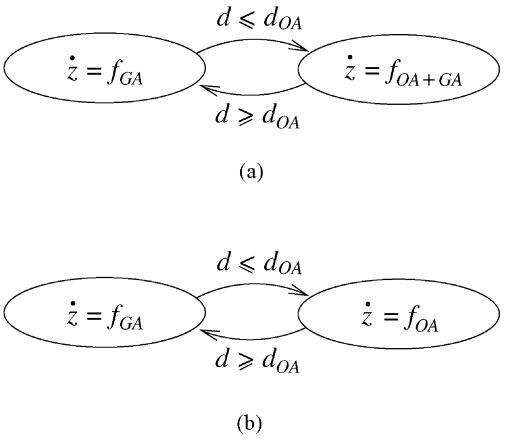

Fig. 1. The two possible goal-attraction and obstacle-avoidance automata. (a) Concurrent behaviors, (b) Hard switches.

If one chooses to work with concurrently active behaviors, different controllers affect the system simultaneously, resulting in a smooth overall performance, as shown in Košecká (1996). However, in this case the analysis of the system is becoming significantly more difficult as new behaviors are added.

The other possible solution to the coordination problem, corresponding to hard switches between the different behaviors, has the major disadvantage that it both affects the performance in a negative way, not allowing for the smooth performance that concurrently active behaviors produce, and that it increases the risk of introducing chattering into the system. Therefore our idea is to impose hard switches on the behavior based system in such a way that we can model each behavior as a node in an automaton. The reason for using hard switches is due to the fact that we want each individual behavior to correspond to a node in the automaton, due to scalability issues. This construction also enables us to theoretically analyze the performance of each behavior individually. At the same time, we want to avoid the negative, chattering effects that such an approach could potentially give rise to. This will be done by adding nodes to the automaton as a way of regularizing it. In what follows, we will show that even though we introduce hard switches, the performance is not affected much when using a regularized automaton instead of concurrently active behaviors. In other words, what we want to do is to eliminate the

[1] Since a real, extended obstacle cannot be considered to be a point, in the actual implementation of the avoidance behavior, the desired heading needs to be calculated as the orientation of the sum of the weighted vectors that each individual sonar reading contributes with. For a Nomadic 200 this corresponds to taking the sum over 16 elements since that platform is equipped with 16 ultrasonic sensors.

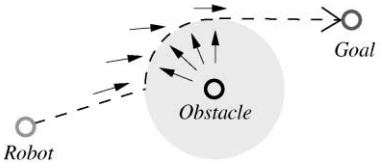

Fig. 2. Goal attraction together with obstacle-avoidance results in a Filippov type Zeno automaton. The grey region around the obstacle is the region where obstacle-avoidance is active. The arrows correspond to the different vector fields that are acting on the robot.

possibility of the so-called *Zeno*[2] phenomena in the system. What this corresponds to is a hybrid system that exhibits an infinite number of discrete transitions in finite time (Johansson, Egerstedt, Lygeros, & Sastry, 1999).

For the type of automata that we encounter here, the infinite number of discrete transitions, made in finite time, is caused by the fact that the underlying system that the automaton tries to model is a switched system that exhibits sliding in the sense of Filippov (1988). They thus form a special class of Zeno hybrid automata since they, theoretically, make an infinite number of transitions in zero time.[3] Such sliding modes arise when the systems have continuous flows that point toward the switching surface, resulting in a new, induced flow on that surface. In these cases, the automaton can be regularized by the introduction of a new node with the continuous flow given by the Filippov solution (Malmborg, 1998).

If we now assume that C_{OA} in (2) is large enough so that the heading of the robot can be considered to be more or less instantaneously driven to its desired configuration, the hybrid automaton in Fig. 1(b) can admit Zeno executions. This obvious fact is best illustrated by Fig. 2, where the extra node that needs to be added in order to regularize the automaton can easily be identified as well. This extra node is just a node containing the sliding dynamics that is defined on the boundary between the two behaviors.

When an obstacle is closer to the robot than d_{OA}, the obstacle-avoidance behavior becomes active. Since the repulsive potential field from that behavior is orthogonal to the surface on which the behavior becomes active, the sliding solution is just

$$f_S = \alpha f_{\mathrm{OA}} + (1-\alpha) f_{\mathrm{GA}}, \tag{3}$$

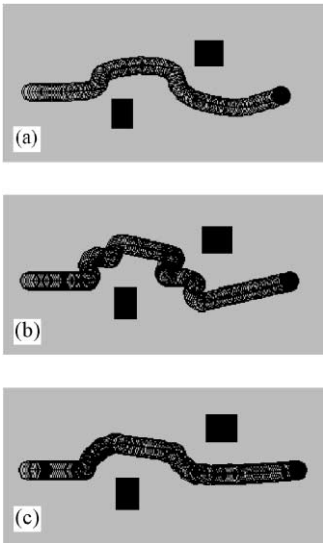

Fig. 3. Simulation of (a) concurrently active behaviors, (b) hard switches, and (c) a regularized automaton on the Nomadic simulator, the `Nserver`.

where $\alpha \in [0,1]$ is chosen so that f_S is orthogonal to f_{OA}. Adding this type of information about the different behaviors makes it possible to generate the extra node in the automaton automatically. It furthermore suggests that our method would scale when more than two behaviors affect the motion of the robot, as long as an automatic procedure for designing the sliding solutions could be identified for the new behaviors as well.[4]

The assumption about instantaneous heading control is obviously a simplification but it still gives a model that is rich enough to capture the, from our point of view, relevant phenomena. In fact, in real life we have a possibility of chattering that here reveals itself as a Zeno execution.

The regularized point-to-point automaton was implemented and tested on the Nomadic 200 mobile robot. In Fig. 3, the results from running the system on the `Nserver`, the Nomadic simulation package, can be seen. In Fig. 3(a), concurrently active behaviors are displayed, resulting in a smooth movement around the obstacles, while the chattering solution in Fig. 3(b) corresponds to hard switches. The reason why we do not have sliding in this case is due to the part of the dynamics of the robot that was ignored in the analysis. It is still clear that from a performance perspective, Fig. 3(b) is an unsuccessful design. In Fig. 3(c) the result from using a regularized hybrid automaton can be seen, and even though we only

[2] The name Zeno refers to the philosopher Zeno of Elea (500–400 BC), whose major work consisted of a number of famous paradoxes. They were designed to explain his view that the ideas of motion and evolving time lead to contradictions. An example is Zeno's Second Paradox of Motion, in which Achilles is racing against a tortoise.

[3] The other class of Zeno automata has a slightly more complex dynamics. Here the automaton changes nodes faster and faster, with the jump times converging to a finite, so-called *Zeno time* (Johansson et al., 1999).

[4] This typically depends on whether we have access to a geometric description of the switching surface or not.

have one behavior active at a time, the performance is satisfactory.

4. Path planning and tracking

Given the reactive obstacle-avoidance behavior from the previous section, the main question that we want to address here is: *How do we construct an appropriate approach-target behavior*? Obviously, we can do better than to just use an attractive potential field, and it will turn out that our regularized automata approach allows us to answer this question. What we want to do is to produce a robot behavior that satisfies the safety specifications at the same time as the solution is close to optimal with respect to a given performance cost function. A first formulation, inspired by Tomlin, Papas, Košecká, Lygeros, and Sastry (1998), of what we want to accomplish could be the following: If we let our admissible controls be $u \in U$, and assume that we only encounter one point obstacle (x_{ob}, y_{ob}), we can define a safety cost function

$$J_s(u) = \min_{t \geqslant 0} \{(x(t) - x_{ob})^2 + (y(t) - y_{ob})^2\}, \tag{4}$$

where the dependence on the control, u, is given implicitly by the controlled system dynamics from the previous section. The set of controls, $U_s(C)$, that make the robot move at least a distance C away from the obstacle can thus be defined as $U_s(C) = \{u \in U\colon J_s(u) \geqslant C\}$.[5]

The next step is to define another cost function that penalizes high curvature of the chosen path. This is a reasonable performance criterion since it penalizes paths that make the robot move in sudden, abrupt ways. Furthermore, this smoothness objective gives a trajectory that a robot has good chances of following when it is governed by physical limits on what signals the actuators can actually track.

The idea now is to choose the control candidates for minimizing this new performance cost function from the set of safe controls, $U_s(C_s)$, where C_s is our preferred safety margin. Unfortunately, it turns out that this is a very hard problem to solve numerically (not to mention analytically), as pointed out in Tomlin et al. (1998). It is thus not suitable in situations where on-line computations are necessary. However, the underlying approach could suggest a way for producing a solution to the obstacle negotiation problem that is both safe, computationally feasible, and makes the system behave in a satisfactory way with respect to keeping the curvature of the produced path small.

[5] It should be mentioned that both J_s and U_s depend on the robot's initial position, but for the sake of notational simplicity we leave that out from the definitions.

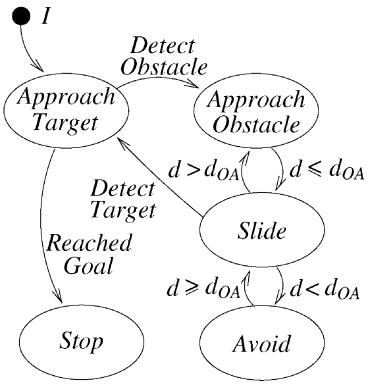

Fig. 4. A heuristic, near-optimal high level control design is shown.

The main idea is that instead of focusing on the hard optimal control problem, we should concentrate on producing suboptimal (but close to optimal) geometric paths that lead around the obstacles. This way we do not have to deal with the actual kinematics of the robot in the optimization formulation. Instead, we design our tracking controller in such a way that the robot can even track reasonably well those paths which do not always meet the kinematical constraints. This means that we cannot be sure that we actually find the optimal controller, but rather that we find one that is reasonably close to the optimal one as long as we have a good enough trajectory tracker.

The desired overall behavior that these heuristics give rise to (under the assumption of perfect tracking), together with the corresponding automaton, is depicted in Fig. 4, where an optimal path is planned and followed by an approach-target behavior until an obstacle is detected. Then an approach-obstacle behavior follows another path to the region where the regularized sliding behavior becomes active. When the target can be reached by an optimal path, not intersecting the safety region around the obstacle (called detect-target in the figure), approach-target becomes active again.

4.1. Path planning

One first observation is that for a path produced by a scalar function $y_d = f(x_d)$, the curvature is given by

$$\kappa(x_d) = \frac{f''(x_d)}{(1 + f'(x_d)^2)^{3/2}}, \tag{5}$$

where the subscript d stands for the desired robot position, and $f'(\cdot)$ and $f''(\cdot)$ denote first and second derivatives, respectively.

Thus, if we minimize $f''(x_d)^2$ instead of $\kappa(x_d)^2$ we make $\kappa(x_d)^2$ small automatically, which is a desired feature, as seen in the previous paragraph.

Since we, by following this proposed route, minimize the L^2-norm of the second derivative, the resulting curve will be a *cubic spline*. This is a fortunate fact since it means that we will not be forced to relay on extensive world information or to do any heavy computations on-line which tends to be the case when more sophisticated planning algorithms are used (Krogh & Thorpe, 1986; Latombe, 1991; Stenz, 1994). It is thus an almost trivial task to generate the splines that connect the robot and the target in the approach-target behavior, and the robot and the obstacle in the approach-obstacle behavior, as seen in Fig. 4.

4.2. Tracking

We now have an on-line method for producing low curvature paths around detected obstacles, and hence our next task is to find a good tracking algorithm so that the robot follows the proposed path robustly.

We let the general reference path, parameterized by s, be given by

$$\begin{aligned} x_d &= p(s), \\ y_d &= q(s), \end{aligned} \quad 0 \leqslant s \leqslant s_f, \tag{6}$$

where the idea is to let the motion of the reference point be governed by a differential equation containing error feedback. It can be viewed as a combination of the conventional trajectory tracking, where the reference trajectory is parameterized in time, and a dynamic path following approach (Sarkar, Yun, & Kumar, 1993), where the criterion is to stay close to the geometric path, but not necessarily close to an a priori specified point at a given time. This approach makes our algorithm robust to measurement errors and external disturbances since, if both the tracking errors and disturbances are within certain bounds, the reference point moves along the reference trajectory while the robot follows it within a prespecified look-ahead distance. Otherwise, the reference point should slow down and "wait" for the robot, as suggested in Egerstedt, Hu, and Stotsky (1998)

Our control objectives are

$$\begin{aligned} &\limsup_{t\to\infty} \rho(t) \leqslant \varepsilon_\rho \\ &\limsup_{t\to\infty} |\phi(t) - \phi_d(t)| \leqslant \varepsilon_\phi, \end{aligned} \tag{7}$$

where ε_ρ and ε_ϕ are positive numbers that can be made 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.

From (6) we directly get that $\dot{x}_d = p'(s)\dot{s}$, $\dot{y}_d = q'(s)\dot{s}$, which implies that if the robot would track the path perfectly, we would have

$$\dot{s} = \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y} \tag{8}$$

since this corresponds to $\dot{x} = \dot{x}_d$ and $\dot{y} = \dot{y}_d$. On the other hand, (8) does not contain any position error feedback, which is important for the robustness. Therefore we propose our dynamics for the reference point as follows:

$$\dot{s} = \frac{c e^{-\alpha\rho} v_0}{\sqrt{p'^2(s) + q'^2(s)}}, \tag{9}$$

where v_0 is the desired speed at which one wants the vehicle to track the path, and α and c are appropriate, positive numbers.

We now let our control algorithm be as follows:

$$\begin{aligned} v &= \gamma\rho\cos(e_\phi) \\ \omega &= k e_\phi + \dot{\phi}_d, \quad k > 0, \end{aligned} \tag{10}$$

where both γ and k are positive, $e_\phi = \phi_d - \phi$, and $\phi_d = \arctan 2(y_d - y, x_d - x)$.

Remark. We should point out that e_ϕ is not defined at $\rho = 0$ since ϕ_d is not defined. In implementation, one can replace ϕ_d by

$$\tilde{\phi}_d = \begin{cases} \phi_d \\ \quad \text{if } \rho > \varepsilon, \\ \dfrac{\phi_d(-2\rho^3 + 3\varepsilon\rho^2) + \theta_r(-2(\varepsilon - \rho)^3 + 3\varepsilon(\varepsilon - \rho)^2)}{\varepsilon^3} \\ \quad \text{if } \rho \leqslant \varepsilon, \end{cases} \tag{11}$$

where ε is a small positive number. It is easy to see that $\tilde{\phi}_d$ is well defined at $\rho = 0$ since $\lim_{\rho\to 0} \phi_d(-2\rho^3 + 3\varepsilon\rho^2) = 0$.

An intuitive interpretation for this control algorithm is that the robot is steered toward the reference point on the desired path with a speed proportional to the tracking error and whenever it reaches the steady tracking error (the desired looking-ahead distance) it will track the path with an almost constant speed. It is easy to see that even if a path is planned beyond the dynamic or kinematic constraints of the system, this control algorithm can still be used to track the path the best it can.

In Egerstedt et al. (1998), it was shown that for the platform model (1), governed by the control (10), the steady state tracking error, ρ, can be made as small as one wants while ϕ tends to ϕ_d exponentially. Furthermore, in steady state we have that $v \approx v_0$. Thus we, by using the control law (10), meet the control objectives defined in (7).

We thus have a way of both producing and tracking paths, and we now combine these two together into the

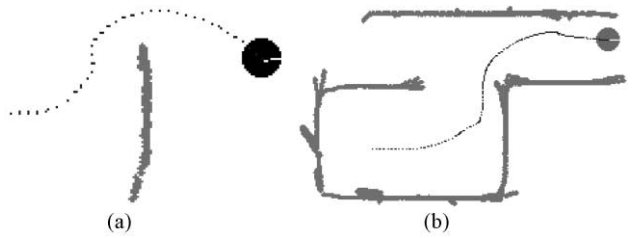

Fig. 5. Implementation of the control approach from Fig. 4. The robot tries to reach a goal point straight ahead of it but is forced to negotiate unmodeled obstacles and walls.

path following behavior that moves the robot safely around the obstacles at the same time as its executed trajectories are not too far from optimal with respect to curvature. As seen in Fig. 5, where real experimental data are displayed, the method seems to work well.

5. Conclusions

In this paper, it is shown that a behavior based control system can be modeled as a hybrid automaton, where each node corresponds to a distinct robot behavior. In order to achieve this, we have to impose hard switches on the transitions between the different behaviors, resulting in a potentially chattering overall behavior. We furthermore show how regularization techniques can be used to solve this problem by adding extra nodes to the automaton. Those extra nodes correspond to the sliding dynamics on the boundary between the different behaviors. The performance aspect of this approach is verified experimentally on a Nomadic 200 mobile platform.

We also propose a method for designing reach-target behaviors in such a way that questions concerning safety and curvature minimization can be addressed explicitly. Our proposed, suboptimal method is based on a combination of path planning and trajectory tracking techniques, placing it in the deliberative part of the behavior based control architecture spectrum. Furthermore, we show that this approach works well in practice on our experimental platform.

References

Ackermann, J. (1993). *Robust control*. London: Springer.

Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge, MA: The MIT Press.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Egerstedt, M., Hu, X., & Stotsky, A. (1998). Control of a car-like robot using a virtual vehicle approach. *Proceedings of the 37th IEEE conference on decision and control*, Tampa, FL, USA, December (pp. 1502–1507).

Filippov, A. F. (1988). *Differential equations with discontinuous righthand sides*. Dordrecht: Kluwer Academic Publishers.

Johansson, K., Egerstedt, M., Lygeros, J., & Sastry, S. (1999). Regularization of Zeno hybrid automata. *Systems and Control Letters*, *38*, 141–150.

Kortenkamp, D., Bonasso, R.P., & Murphy, R. (Eds). (1998). *Artificial intelligence and mobile robots*. Cambridge, MA The MIT Press.

Košecká, J. (1996). *A framework for modeling and verifying visually guided agents: Design, analysis and experiments*. Dissertation, Grasp Lab, March.

Krogh, B., & Thorpe, C. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA (pp. 1664–1669).

Latombe, J. C. (1991). *Robot motion planning*. Dordrecht: Kluwer Academic Publishers.

Lygeros, J., Tomlin, C., & Sastry, S. (1999). Controllers for reachability specifications for hybrid systems. *Automatica*, *35*(3), 349–370.

Malmborg, J. (1998). *Analysis and design of hybrid control systems*. Ph.D. thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, May.

Sarkar, N., Yun, X., & Kumar, V. (1993). Dynamic path following: A new control algorithm for mobile robots. *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, December.

Stenz, A. (1994). Optimal and efficient path planning for partially known environments, Proceedings of the *1994 IEEE International Conference on Robotics and Automation*, San Diego, California.

Tomlin, C., Papas, G., Košecká, J., Lygeros, J., & Sastry, S.S. (1998). Advanced air traffic automation: a case study in distributed decentralized control. *Control problems in robotics, Lecture Notes in Control and Information Sciences,* Vol. 230. London: Springer.

Magnus B. Egerstedt was born in Stockholm, Sweden, in 1971. He received the M.S. degree in Engineering Physics and the Ph.D. degree in Applied Mathematics from the Royal Institute of Technology, Stockholm, in 1996 and 2000, respectively. He also received a B.A. degree in Philosophy from Stockholm University in 1996. He is an Assistant Professor in Electrical and Computer Engineering at the Georgia Institute of Technology.

He spent 2000–2001 as a Postdoctoral Fellow at the Division of Engineering and Applied Science at Harvard University and during 1998 he was a Visiting Scholar at the Robotics Laboratory at the University of California, Berkeley. Dr. Egerstedt's research interests include optimal control as well as modeling and analysis of hybrid systems, with emphasis on motion planning and control of mobile robots.

Xiaoming Hu was born in Chengdu, China, in 1961. He received the B.S. degree from University of Science and Technology of China in 1983. He received the M.S. and Ph.D. degrees from Arizona State University in 1986 and 1989 respectively. From 1989 to 1990 he was a Gustafsson Postdoctoral Fellow at the Royal Institute of Technology, Stockholm, where he is currently an associate professor. His main research interests are in nonlinear feedback stabilization, nonlinear observer design, sensing and active perception, motion planning and control of mobile robots, and mobile manipulation.