Control of Autonomous Mobile Robots

Magnus Egerstedt

School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A., magnus@ece.gatech.edu

1 Background

An example of a reactive, mobile, embedded control system that has received considerable attention during the last decade is the *autonomous mobile robot*. The flurry of research activities in this area can be directly traced to the many exciting current and future applications where robotic systems are safer/cheaper/more effective than their human counterparts. Examples of current applications include

- Domestic service robots, such as autonomous vacuum cleaners, lawn mowers, and pool cleaners;
- Planetary exploration robots, such as the NASA Mars rovers Spirit and Opportunity;
- Autonomous robots for military applications, including surveillance and search-and-destroy robots; and
- Robots for monitoring, exploring, and securing unsafe environments, such as bomb sniffers, disaster site robots, and mine sweepers.

Notably absent from this list are the many robots employed in industrial settings. Such industrial robots are not considered here since they typically operate in highly structured environments, where the maneuvers can be planned in advance, resulting in challenging, yet standard, tracking problems. In contrast to this, autonomous mobile robots operate in partially or completely unknown environments, where the occurrences of unmodeled obstacles are commonplace. What this means is that the complexity of the control task is increased due to the complexity of the environment in which the system operates, which imposes a number of challenges on the control design. In this chapter we will cover modeling and architectural design issues for such systems. We will moreover discuss some implementation aspects as well as outline a collection of major challenges that still remain to be solved.

2 Multi-Modal Control

For mobile, autonomous robots the ability to function in and interact with a dynamic, changing environment is of key importance. As such, they fall under a class

of reactive, mobile systems where environmental changes trigger changes in what objectives the control system must meet. The standard way of structuring the control system in order to deal with this problem is within a *multi-modal* control framework, sometimes referred to in the robotics literature as the *behavior-based* robotics framework [1, 4, 11]. The main idea is to identify different controllers, responses to sensory inputs, with desired robot behaviors. This way of structuring the control system into separate behaviors, dedicated to performing certain tasks, has gained significant momentum within the robotics community. This momentum stems from the fact that a modular design both simplifies the design process and also makes it possible to add new behaviors to the system without causing any major increase in complexity.

Once a collection of behaviors has been designed, different options present themselves at the supervisory level. For instance, one can let different behaviors run concurrently in the sense that they all can have an effect on the low-level motor commands according to some coordination rule. This construction with concurrent behaviors makes it relatively straightforward to stress robustness issues explicitly, since, for example, an "avoidance behavior" can just be given a higher priority or weight than a "reach target behavior." However, as multiple behaviors are allowed to affect the system simultaneously, a number of theoretical as well as practical issues present themselves. For example, given a collection of individual behaviors, how should these be coordinated in order to achieve a satisfactory, global behavior?

An alternate route to take is to insist on letting only one behavior affect the system at each time instant. This somewhat simplifies the analysis since the resulting system is a hybrid system for whose analysis a number of tools are available. But, as we will see, hard switches between behaviors may cause the performance of the system to degrade if care is not taken when dealing with chattering and other issues.

2.1 Behaviors

If we let the autonomous robot be modeled at the kinematic level as a unicycle

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

where (x, y) denotes the position of the robot, and ϕ denotes its orientation, a behavior is characterized by the way sensory data is mapped to the control inputs v and ω, corresponding to translational and angular velocities, respectively. Now, relative to this robot model, a straightforward way [1] of specifying the effect of individual behaviors is to let the behavior define a vector

$$\mathcal{B} = r_{\mathcal{B}} \begin{pmatrix} \cos(\phi_{\mathcal{B}}) \\ \sin(\phi_{\mathcal{B}}) \end{pmatrix},$$

where $r_{\mathcal{B}}$ is the "magnitude" of the behavior vector, and $\phi_{\mathcal{B}}$ is its orientation. This vector formalism allows us to map behavior vectors to control values using some appropriate map $F(\mathcal{B}) = (v, \omega)^T$. For example, one can let

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = F(\mathcal{B}) = \begin{pmatrix} \min\{v_0, 1/r_{\mathcal{B}}\} \\ C(\phi_{\mathcal{B}} - \phi) \end{pmatrix}.$$

Here, the translational velocity achieves its nominal value $v_0 > 0$ when the magnitude of the behavior vector is small, but is reduced as this magnitude grows. Furthermore, the angular velocity is simply given by a proportional error feedback law, with $C > 0$ being the gain. Note that it is also quite standard to let the gain vary as a function of $r_{\mathcal{B}}$.

Now, if we are given $\mathcal{B}_1$ and $\mathcal{B}_2$, i.e., two vectors corresponding to two different behaviors, they can be combined directly using a vector addition operation $\mathcal{B}_1 + \mathcal{B}_2$ in order to produce a new behavior, and this semigroup property is why the vector notation is particularly appealing. Here the coordination mechanism is thus explicitly given. Moreover, the magnitude of the behavior vector, $r_{\mathcal{B}}$, is what determines how much weight that particular behavior is given in the summation. As we will see in the next few paragraphs, avoidance behaviors should increase in magnitude, typically according to an inverse square law, as the robot draws closer to the obstacles.

To make matters more concrete, let us in consider an obstacle-avoidance behavior (denoted OA in what follows) in some detail. Most mobile robots are equipped with a collection of k range sensors, such as ultrasonic sonars or infra red sensors, and a standard sonar ring typically consists of 8 or 16 sensors. Each of these sensors measures the distance to the closest obstacle along a particular, fixed relative orientation; we let d_j denote the distance to the closest obstacle detected by sensor j, and we let ϕ_j be the corresponding angle. We can then define the obstacle avoidance behavior, $\mathcal{B}_{OA}$, through the vector summation

$$\mathcal{B}_{OA} = \mathcal{B}_{OA,1} + \mathcal{B}_{OA,2} + \ldots + \mathcal{B}_{OA,k},$$

where the behavior vectors are given by

$$\begin{aligned} r_{\mathcal{B}_{OA,j}} &= \begin{cases} 0 & \text{if } d_j > D \\ C_{OA}\frac{(D-d_j)}{d_j^3} & \text{otherwise} \end{cases} \\ \phi_{\mathcal{B}_{OA,j}} &= \pi + \phi_j, \end{aligned}$$

where $C_{OA} > 0$, and D is the safety distance at which the obstacle-avoidance behavior starts affecting the system.

In a similar manner, we can define an "approach target behavior," $\mathcal{B}_{AT}$, as

$$\begin{aligned} r_{\mathcal{B}_{AT}} &= C_{AT} \\ \phi_{\mathcal{B}_{AT}} &= \arctan((y_g - y)/(x_g - x)), \end{aligned}$$

where $C_{AT} > 0$ is the constant magnitude, and the goal is located at (x_g, y_g), as shown in Fig. 1.

2.2 Regularizations

However, it may not always be desirable to let different behaviors affect the system simultaneously, even though such an approach results both in notational convenience as well as an intuitively appealing mechanism for combining multiple control objectives. Unfortunately, such an approach ruins the modularity that comes with a switched control strategy in the sense that if a new behavior is introduced, its impact on the system is almost impossible to characterize analytically. This lack of analytical characterization tools is one of the main reasons why *emergent behaviors*, i.e., unpredictable global behaviors obtained through local rules, have received

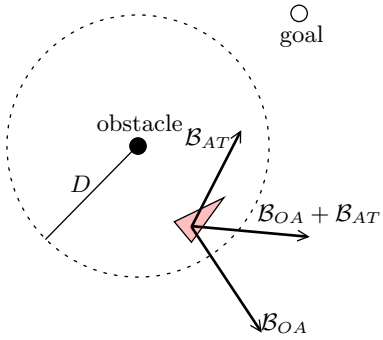


Fig. 1. Vector addition of "obstacle-avoidance" and "approach target" behaviors

considerable attention in the literature. Moreover, if an obstacle-avoidance behavior has been designed so that the robot is guaranteed not to hit static obstacles, by combining this behavior with other behaviors, this guarantee no longer holds.

A remedy to this problem is to let the control system switch between different behaviors. Unfortunately, such an approach may have a negative impact on the performance of the system since it increases the risk of introducing chattering into the system. Chattering is a phenomenon that occurs when two vector fields, corresponding to two different behaviors, both point in toward the switching surface that dictates when the robot should switch between the behaviors. In other words, if we switch from mode 1, where $\dot{x} = f_1(x)$, to mode 2, where $\dot{x} = f_2(x)$, when x leaves the region $g(x) < 0$, where g is a smooth map from $\mathbb{R}^n$ to $\mathbb{R}$, then chattering occurs if

$$\frac{\partial g(x)}{\partial x} f_1(x) > 0 \quad \text{and} \quad \frac{\partial g(x)}{\partial x} f_2(x) < 0$$

on the boundary $g(x) = 0$.

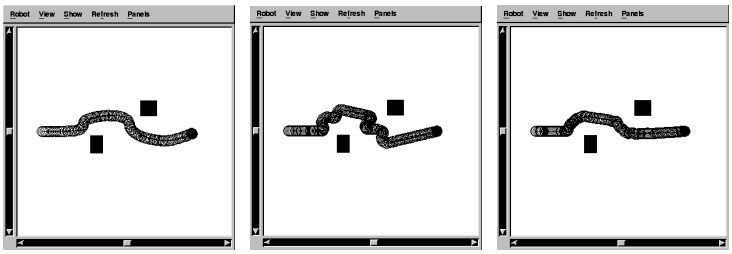
The standard way out of this problem is to regularize the system so that sliding is allowed to occur. For example, assume that we have access to instantaneous heading control in our control laws. When an obstacle is closer to the robot than D, the obstacle-avoidance behavior is active. Since the repulsive potential field from that behavior will be orthogonal to the surface on which the behavior becomes active, the sliding solution is given by

$$\mathcal{B}_S = \alpha \mathcal{B}_{OA} + (1 - \alpha)\mathcal{B}_{AT},$$

for some $\alpha \in [0, 1]$ such that $\mathcal{B}_S \perp \mathcal{B}_{OA}$.

Some results from applying this regularization approach to the chattering problem are shown in Fig. 2, where Fig. 2(a) shows a situation when vector summation is used. Fig. 2(b) corresponds to switches between the behaviors, and it is clear that a chattering-like behavior is produced. In Fig. 2(c), the regularized solution is shown. Even though we only have one behavior active at a time, the performance is clearly satisfactory in that case.

By incorporating this type of information about the different behaviors, it is possible to generate the sliding modes automatically. It furthermore suggests that this method would scale when more than two behaviors affect the motion of the



(a) (b) (c)

Fig. 2. (a) Combined behaviors using vector summation, (b) switches between the different behaviors, and (c) a regularized solution. These results were obtained on the Nomad simulator, the Nserver.

robot, as long as an automatic procedure for designing the sliding solutions can be identified. Hence we assume throughout the remainder of this chapter that only one behavior affects the robot at each time instant, and that, when appropriate, a sliding mode may be induced from the system dynamics. In the next section we will consider this issue to be settled and instead focus on the question of how one should model and specify multi-modal control procedures for robotics tasks.

3 Task Specifications

As an example, consider the scenario that is played out in Fig. 3. The problem the robot is instructed to solve is to find the door while avoiding obstacles. Once the door is found, the robot should go though the door and then start following the wall.

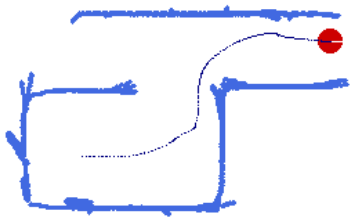


Fig. 3. The robot finds the door and then moves into a corridor, where it follows the wall. The reason why the sonar readings seem rather inaccurate is due both to the odometric drift and the coarseness of the sonar resolution.

3.1 Hybrid automata

There are, of course, a number of ways in which this particular navigation problem can be solved. If we let our solution lay within the multi-modal framework, we first note that the behaviors needed to solve this problem could be

- Wander;
- Avoid obstacle;
- Go through door; and
- Follow wall.

Moreover, it is clear that by designing the individual behaviors, a complete, executable multi-modal control procedure has not yet been produced in the sense that we cannot run it directly on the robot. What is missing is the set of rules for switching between the different behaviors. For instance, if we let the dynamics of the system be

$$\begin{aligned} \dot{x} &= f(x,u),\ x \in \mathbb{R}^n,\ u \in U \subset \mathbb{R}^m \\ y &= h(x),\ y \in Y \subset \mathbb{R}^p, \end{aligned}$$

then the different behaviors correspond to particular feedback laws $u_{\mathcal{B}} : Y \to U$. If we let u_1 and u_2 be two such behaviors, then, in mode 1, we have $\dot{x} = f(x, u_1(h(x)))$ and, in mode 2, $\dot{x} = f(x, u_2(h(x)))$. We moreover choose to switch between these two modes as a certain property, defined on the sensory inputs, is satisfied. We can thus define a Boolean map $\xi : Y \to \{0,1\}$ and use mode 1 as long as $\xi(y) = 0$, and switch to mode 2 when $\xi(y) = 1$. Such a mapping is referred to as a *guard* in the hybrid systems literature, and in Fig. 4, a *hybrid automaton* is shown that generates the desired switched behavior. The interpretation here is that the system starts at the initial condition $x = x_0$ in mode 1, and switches to mode 2 as the guard condition is satisfied.

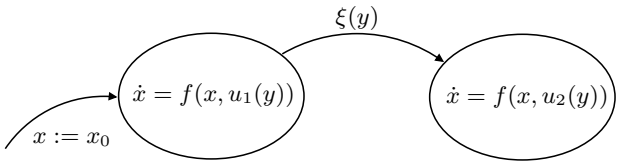


Fig. 4. A simple hybrid automaton

If we now return to the problem of having the robot find the door and then go through it, a hybrid automaton that would implement a solution to this problem is shown in Fig. 5, where u_W corresponds to the "wander" behavior, u_O corresponds to "obstacle avoidance," u_D corresponds to "go through door," and u_F to "follow wall." Moreover, $\xi_O = 1$ if an obstacle is too close, while $\xi_S = 1$ when this is no longer the case. $\xi_D = 1$ when the door has been detected, and $\xi_T = 1$ when the robot has passed through the door.

The hybrid automaton in Fig. 5 captures the switched aspects of a multi-modal control procedure in a very natural way. As such, it has been the main model used when analyzing the performance of autonomous mobile robots governed by reactive,

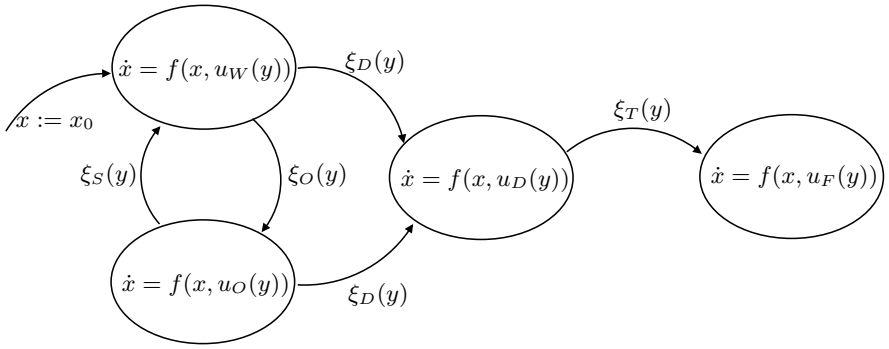


Fig. 5. A hybrid automaton for the scenario in Fig. 3

multi-modal control strategies. Formally, a hybrid automaton can be defined as $H = (Q, X, f, E, \Xi, x_0, q_0)$, where $Q = \{q_1, q_2, \ldots, q_N\}$ is the set of discrete states (or modes). X is the state space on which the continuous state is evolving, e.g., $X = \mathbb{R}^n$, and $f : X \times Q \to TX$ is the mode-dependent differential equation that dictates the evolution of the continuous state. Moreover, $E \subset Q \times Q$ is the edge set, and Ξ is the set of guards, with $\Xi \ni \xi : X \times E \to \{0, 1\}$. (We will suppress ξ's dependence on $e \in E$ whenever this dependence is clear from the context.) Finally, q_0 and x_0 denote, respectively, the initial discrete mode and continuous state [8].

Due to its expressiveness, the hybrid automaton is a useful modeling and analysis tool. However, from a specification and implementation point of view, we note that it may be more beneficial to use a more compact notation, which leads us to the idea of a formal language. (See, for example, [9] for an introduction to this subject.)

3.2 Motion description languages

We are given a finite automaton $A = (Q, \Gamma, \delta, q_0)$, where Q (finite) is the state space, Γ (finite) is the event set, q_0 is the initial condition, and $\delta : Q \times \Gamma \to Q$ is the transition function. We say that A generates the *language* $L(A)$, where

$$L(A) = \{s \in \Gamma^\star \mid \delta(q_0, s) \text{ is defined } \}.$$

Here $\Gamma^\star$ is the free monoid over Γ, i.e., the set of all finite length words over Γ (including the empty word ε). In other words, $\Gamma^\star$ is the set of all finite length concatenations of events, and if $s = \gamma_1 \cdot \gamma_2 \cdots \gamma_N$ then $\delta(q_0, s) = \delta(\cdots \delta(\delta(q_0, \gamma_1), \gamma_2), \cdots, \gamma_N)$.

Consider, for example, the case where we ignore the differential equations in the automaton in Fig. 5, and use ξ and u as shorthand for the event $\xi(y) = 1$ and the mode corresponding to the behavior $\dot{x} = f(x, u)$, respectively. Then $Q = \{u_W, u_O, u_D, u_F\}$, $\Gamma = \{\xi_O, \xi_S, \xi_D, \xi_T\}$, $q_0 = u_W$, and $\delta(u, \xi)$ is given by the discrete transitions in the automaton. In this case, we the generated language becomes

$$L(A) = \overline{\overline{\{(\xi_O \cdot \xi_S)^\star\}} \cdot \{\xi_D \cdot \xi_T\}},$$

where, given a language $L \subset \Gamma^\star$, $\overline{L} = \{s \in \Gamma^\star \mid l = s \cdot t \in L \text{ for some } t \in \Gamma^\star\}$ is the *prefix closure* of the language L, the language *concatenation* $L_1 \cdot L_2 = \{s \in \Gamma^\star \mid s =$

$l_1 \cdot l_2$ for some $l_1 \in L_1,\ l_2 \in L_2\}$, and, given $s \in \Gamma^\star$, $s^\star = \{\varepsilon, s, s \cdot s, s \cdot s \cdot s, \ldots\}$, where ε is the empty word.

Now, given A and a string $s \in L(A)$, this string completely determines the evolution of A, as long as A is deterministic. However, it does not capture the structure of A itself. In other words, from $L(A)$ we can analyze the evolution of A, but as a tool for specifying and implementing finite automata it falls short. And, since we in this section are interested in finding compact yet expressive ways for implementing and specifying multi-modal control procedures, where each discrete state corresponds to a particular mode, the specification language must include a description of what behaviors to use as well as what guard relations should characterize the mode transitions.

One such language that has appeared in the literature is the *motion description language*. Given a finite set of behaviors B (or more precisely, symbols that correspond to behaviors) and interrupts Ξ, we can let a motion description language be a subset of the set $(B \times \Xi)^\star$, i.e., a multi-modal control procedure contains strings of behavior-guard pairs, where the behavior specifies what control law to use, while the guard (or interrupt) dictates under which conditions this behavior should terminate. Using these ideas (and using the convention that u is a symbol corresponding to the map $u : Y \to U$), the multi-modal control procedure implemented in Fig. 5 is

$$(((u_W, \xi_O) \cdot (u_O, \xi_S))^\star, \xi_D) \cdot (u_D, \xi_T) \cdot (u_F, \xi_\varepsilon),$$

where $\xi_\varepsilon(y) = 0,\ \forall y \in Y$. The interpretation here is that the "meta-behavior" $(u_W, \xi_O) \cdot (u_O, \xi_S)$ is repeated until $\xi_D(y) = 1$, followed by the string $(u_D, \xi_T) \cdot (u_F, \xi_\varepsilon)$.

3.3 Complexity issues

Given that motion description languages are used for specifying and implementing multi-modal control procedures on embedded robotics systems, one has access to a formalism in which control procedures can be thought of as having an information theoretic content. In other words, they need to be coded using a certain number of bits. This is facilitated by the fact that a behavior-interrupt pair corresponds to a particular symbol in a finite set. Each such symbol thus represents a different control actions that, when applied to a specific machine, define a particular segment of motion [3, 6, 10, 12].

Now, assume that we are given a string of modes $s \in (B \times \Xi)^\star$. The number of bits needed for describing s uniquely is given by the *description length*:

$$\mathcal{D}(s, B \times \Xi) = |s| \log_2(card(B \times \Xi)),$$

where $|s|$ denotes the length of s, i.e., the number of modes in the string, and $card(\cdot)$ denotes cardinality. The description length thus tells us how complicated s is, i.e., how many bits we need for describing it.

Now, since Y corresponds to the set in which the sensor readings take on their values, and since every physical sensor has a finite range and resolution, we can assume that Y has finite cardinality. A similar argument can be applied on the actuator side as well, and hence we can assume that both Y and U are finite sets, which is the case in the emerging area of *quantized control* as well [5].

Moreover, since

$$card(B \times \Xi) = card(U)^{card(Y)} 2^{card(Y)},$$

we see directly that a higher resolution measurement results in a larger Y (and hence in a potentially higher description length) than a lower resolution measurement does. A better sensor might thus make the control procedures significantly more complicated while only providing marginally better performance. This trade-off between complexity and performance is something that can be capitalized on when designing control laws. The idea is simply (in theory) to pick $(U, Y, s \in (B \times \Xi)^\star)$ in such a way as to make the robot behave satisfactorily, while minimizing the description lengths of the control procedures.

For example, in [6], a quantitative analysis was given, showing that the availability of feedback can reduce the length of the shortest description of the multi-modal control procedures. In particular, it was shown that the length of the description can be reduced by a factor that depends on the ratio of the size of the entire state space to the size of the set of states for which feedback is locally effective, i.e., where convergent observers can be constructed. In other words, in some situations it is better to rely on sensors than to work solely with map-based open-loop instructions if one wants to minimize the description length of the multi-modal control procedure.

This result holds the promise of further generalizations since it can be thought of as a special case of the *sensor selection problem* (what sensors are needed to carry out the navigation task successfully?), as it tells us whether or not sensors should be used at all. Furthermore, the many visible and successful applications of feedback mechanisms at work testify to their effectiveness and, over the years, various arguments have been advanced showing why, in particular settings, feedback is useful. The models commonly used bring to the fore considerations of sensitivity, uncertainty, e.t.c., and specific formalizations include

1. H.S. Black's argument for reducing the effect of drift in a high-gain amplifier by the use of a relatively constant, but low gain, feedback term;
2. The stochastic disturbance argument for using measurements to reduce the effect of probabilistic uncertainty; and
3. The game theoretic argument in which a saddle point condition is enforced by feedback. (H_∞ control can be thought of this way.)

To this list a fourth item has thus been added that can be cast in terms of the effect feedback has on reducing the description length of the control procedures for robot navigation tasks:

4. The complexity argument, showing that feedback can shorten the description of the control procedure if reliable sensory information is available.

Some extensions to this work were undertaken in [7] based on the observation that goals are seldom final goals. More often they tend to be intermediary goals in a grander scheme, which, for instance, is the case when mobile robots are navigating using sequences of landmarks. A collection of results was derived that describe how the complexity of the input signals decrease if the robot is navigating in an environment populated by many, easily detectable landmarks.

However, the description length does not tell the whole story. If we assume that we have been able to establish a probability distribution over $B \times \Xi$, we can use optimal coding schemes, such as the Huffman code, for finding the shortest expected number of bits $l^\star(B \times \Xi)$ needed for coding an element drawn at random

from $B \times \Xi$. Shannon's classic source coding theorem tells us that $\mathcal{H}(B \times \Xi) \leq l^\star(B \times \Xi) < \mathcal{H}(B \times \Xi) + 1$, where the *entropy* $\mathcal{H}(B \times \Xi)$ is given by

$$\mathcal{H}(B \times \Xi) = - \sum_{i=1}^{card(B \times \Xi)} p_i \log_2 p_i.$$

Here the interpretation is that the behavior-interrupt pair $s_i \in B \times \Xi$ occurs with probability p_i, and it should be noted that a probability distribution over $B \times \Xi$ corresponds to a specification of which modes are potentially useful. But, to establish such a probability distribution over a structured set, such as the set of modes, is not a trivial task, and only initial work has been conducted along these lines [2].

4 Final Remarks

The main focus of this chapter has been on the modeling and specification issues that arise when dealing with autonomous mobile robots. In particular, a multi-modal control framework has been promoted as a way to decompose the control system into a collection of building blocks. The resulting hybrid control system is particularly suited to the reactive character of the navigation system that complex and unknown environments inevitably give rise to. However, there are a number of challenging research issues that remain largely unsolved and need to be addressed. To conclude this chapter, we present a partial list over such areas of particular importance in robotics research.

- *From local rules to global behaviors:* Given a multi-modal control procedure (or a class of such procedures), what can be said about the resulting system in terms of stability, robustness, expressiveness, and task completion?
- *From global behaviors to local rules:* Given a desired overall behavior, what local behaviors, or modes, are needed in order to achieve the global task?
- *Adaptive multi-modal control:* Given a collection of behaviors and guards, can they be adaptively varied over time in order to achieve a better overall performance?
- *Computational complexity versus real time:* In robotics, a distinction is made between *deliberative* and *reactive* behaviors, where the former relies on internal representations of the environment, which facilitates the use of planning of optimal paths, etc. Given constraints on the reaction time, can a trade-off be achieved between the complexity of the control algorithm and the time in which the algorithm has to terminate with an answer?

References

1. R. C. Arkin. *Behavior Based Robotics.* The MIT Press, Cambridge, MA, 1998.
2. A. Austin and M. Egerstedt. Mode reconstruction for source coding and multi-modal control. *Hybrid Systems: Computation and Control*, Springer-Verlag, Prague, The Czech Republic, April 2003.

3. R. W. Brockett. On the computer control of movement. In the *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.
4. R. Brooks. A robust layered control system for mobile robots. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14–23, 1986.
5. D. F. Delchamps. The stabilization of linear systems with quantized feedback. *Proceedings of the IEEE Conference on Decision and Control*, Austin, TX, Dec. 1988.
6. M. Egerstedt and R. W. Brockett. Feedback can reduce the specification complexity of motor programs. *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, pp. 213–223, Feb. 2003.
7. M. Egerstedt. Some complexity aspects of the control of mobile robots. *American Control Conference*, Anchorage, AK, May 2002.
8. T. A. Henzinger. The theory of hybrid automata. *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pp. 278–292, IEEE Computer Society Press, Washington D.C., 1996.
9. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, 2nd edition. Addison Wesley, Reading, MA, 2000.
10. D. Hristu-Varsakelis and S. Andersson. Directed graphs and motion description languages for robot navigation and control. *Proceedings of the IEEE Conference on Robotics and Automation*, May 2002.
11. D. Kortenkamp, R. P. Bonasso, and R. Murphy, Eds. *Artificial Intelligence and Mobile Robots*. The MIT Press, Cambridge, MA, 1998.
12. V. Manikonda, P. S. Krishnaprasad, and J. Hendler. Languages, behaviors, hybrid architectures and motion control. In *Mathematical Control Theory*, Eds. Willems and Baillieul, pp. 199–226, Springer-Verlag, Berlin, 1998.