Proceedings of the 38th
Conference on Decision & Control
Phoenix, Arizona USA • December 1999

Behavior Based Robotics Using Regularized Hybrid Automata[1]

M. Egerstedt†, K. Johansson‡, J. Lygeros‡, and S. Sastry‡

†magnuse@math.kth.se
Optimization and Systems Theory
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

‡{johans,lygeros,sastry}@eecs.berkeley.edu
Electrical Engineering and Computer Science
University of California at Berkeley
Berkeley, CA 94720, U.S.A.

Abstract

This article investigates how to model a behavior based control system for mobile robots as a hybrid automaton. We show that an automaton, with the nodes corresponding to distinct behaviors, may exhibit an infinite number of discrete transitions in finite time (a so called Zeno hybrid automaton). This can be dealt with by a regularization procedure, involving adding extra nodes to the automaton which gives a system with similar performance as a fused behavior based system. The performance aspect is also verified experimentally.

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 [3]. The major idea is that different behaviors are identified as action responses to sensory inputs. A behavior could, for instance, be obstacle avoidance in which sonar information about a close obstacle results in a movement away from that obstacle. This modular approach has the nice structural property that it allows for decentralized controllers to be used, dedicated to performing different tasks such as door traversal or obstacle avoidance. New behaviors could then easily be added to the system at the same time as the control problem can be divided into subparts, which significantly simplifies the design process.

However, no matter if the behaviors of the system are products of experimental or theoretical considerations, questions concerning action coordination needs to be addressed [9]. How should the behaviors be coordinated so that the overall robotic system behaves in a satisfactory way? There are two obvious answers to this question. One way of addressing this is to use an arbitration mechanism, where behaviors with higher priorities simply overrule other behaviors. For instance, when moving a robot toward a given goal point, not bumping into things is probably considered to be more important than actually reaching the goal point at any cost. These hard switches have the major disadvantages that they both affect the performance in a negative way, and that they increase the risk of introducing chattering into the system. The other approach is to do action fusion where different behaviors can be active simultaneously. From a performance perspective this seems preferable [3, 4, 5].

What will be investigated in this article is how a behavior based system can be modeled as a *hybrid automaton* with each of the discrete nodes corresponding to a distinct behavior. If the system where to be described by such a hybrid automaton, questions concerning safety and performance verification could be addressed in systematic way [8, 12]. Hopefully this approach would also make it possible to explain, at least some of the so called *emergent* behaviors that the complex robot system gives rise to.

On the other hand, if one chooses to work with fused behaviors, different controllers affect the system simul-

[1]The work by M. Egerstedt was sponsored by the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH. The work by K. Johansson was supported by the Swedish Foundation for International Cooperation in Research and Higher Education. The work by J. Lygeros and S. Sastry was supported by ONR under grant N00014-97-1-0946 and by DARPA under contract F33615-98-C-3614.

taneously, resulting in a possibly better performance, but the system would then neither correspond to an automaton where a node would represent many behaviors, nor a model where the discrete nodes would uniquely identify in what state the system is in. This would make the automata approach meaningless since all of the different, active nodes would affect the system simultaneously. One proposed [8] way of addressing this is to group all of the currently active behaviors into one node and thus represent them with discrete state in the automaton. This is not a road that we will follow in this article, since from our point of view, it would only hide the problems we are interested in, such as action coordination or performance verification, in the different nodes. Our idea is instead to impose hard switches on the behavior based system in such a way that we can model each behavior as a node in the automaton, at the same time as 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. This means that we want to remove the so called *Zeno*[1] properties of the system. What this corresponds to is a hybrid system that exhibits an infinite number of discrete transitions in finite time.

This last point will constitute the major contribution of this article, that is organized as follows: In Section 2, the definition of a hybrid automaton will be given together with a discussion about Zeno hybrid automata and their regularizations. In Section 3, we describe a behavior based robotic system and show how this can be modeled as a regularized hybrid automaton. We conclude, in Section 4, with some experimental results from implementing our ideas on a Nomad 200, showing that our approach works in practice as well as in theory.

2 Hybrid Automata

Even though the main focus in this article is not going to be on hybrid automata theory, we need to include some initial definitions. This is necessary in order to be able to state what we mean by a Zeno hybrid automaton as well as to capture the hybrid aspects of a behavior based robotic system.

2.1 Hybrid Automata and Executions

The following definitions are based on [10, 7].

Definition 2.1 (Hybrid Automaton) *A hybrid automaton is considered to be a collection* (Q, X, I, f, E) *where* Q *and* X *are sets of discrete and continuous variables respectively.* I *is a set of initial states, while* f *describes the continuous and* E *the discrete evolution of the states.*

A discrete state combined with the continuous dynamics connected to that state will be referred to as a *node* in the automaton. The general idea behind this construction can be seen in Figure 1.

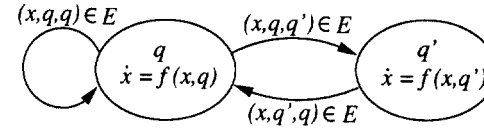


Figure 1: The basic structure of a hybrid automaton.

Definition 2.2 (Hybrid Time Trajectory) *A hybrid time trajectory* τ *is a finite or infinite sequence of intervals of the real line,* $\tau = \{I_i\}$, $i \in \mathbb{N}$, *satisfying the following conditions:*

- I_i *is closed, unless* τ *is a finite sequence and* I_i *is the last interval in which case it can be right open.*
- *Let* $I_i = [\tau_i, \tau_i']$. *Then for all* i, $\tau_i \leq \tau_i'$ *and for* $i > 0$, $\tau_i = \tau_{i-1}'$.

This should be interpreted as the times at which we arrive (τ_i) and leave (τ_i') a specific node in the automaton.

Note that hybrid time trajectories can extend to infinity if τ is an infinite sequence or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$.

Definition 2.3 (Execution) *An execution* χ *of a hybrid automaton* H *is a collection* $\chi = (\tau, q, x)$, *satisfying*

- *Initial Condition:* $(q(\tau_0), x(\tau_0)) \in I$.
- *Discrete Evolution:* $(x(\tau_{i-1}'), q(\tau_{i-1}'), q(\tau_i)) \in E$, *for all* i.
- *Continuous Evolution: for all* i *with* $\tau_i < \tau_i'$, x *and* q *are continuous over* $[\tau_i, \tau_i']$ *and for all* $t \in [\tau_i, \tau_i')$, *we have* $\frac{d}{dt}x(t) = f(q(t), x(t))$.

Furthermore, an execution $\chi = (\tau, q, x)$ is called infinite, if τ is an infinite sequence, or $\sum_i(\tau_i' - \tau_i) = \infty$. We use $\mathcal{H}_{(q_0,x_0)}$ to denote the set of all infinite executions of H with initial condition $(q_0, x_0) \in I$. An

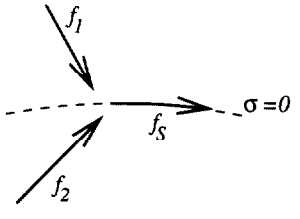
[1] The name Zeno refers to the philosopher Zeno of Elea (500–400 B.C.), whose major work consisted of a number of famous paradoxes. They were designed to explain his view that the ideas of motion and evolving time lead to contradictions. An example is Zeno's Second Paradox of Motion, in which Achilles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

execution is *admissible* if $\sum_i(\tau_i' - \tau_i) = \infty$, and it is *Zeno* if it is infinite but not admissible. For a Zeno execution $\chi = (\tau, q, x)$ we define the Zeno time as $\tau_\infty = \sum_i(\tau_i' - \tau_i) < \infty$. What this means is that the hybrid system makes an infinite number of discrete transitions in finite time, $[\tau_0, \tau_\infty)$, and we finally state the following definition.

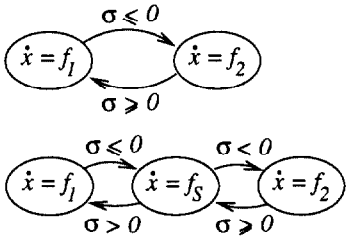
Definition 2.4 (Zeno Hybrid Automaton) *A hybrid automaton H is called Zeno, if there exists $(q_0, x_0) \in I$ such that $\mathcal{H}_{(q_0,x_0)}$ contains a Zeno execution.*

2.2 Regularization

It is clear that a Zeno hybrid automaton has the undesirable property that it blocks time. For some types of these automata, 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 [6]. They thus form a special class of Zeno hybrid automata since they, in theory, make an infinite number of transitions in zero time. The underlying, switched 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 [11]. The general idea behind this construction can be seen in Figure 2.



(a) The sliding solution.



(b) The original and the regularized automaton.

Figure 2: Regularization of a Filippov type Zeno hybrid automata.

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 the Zeno time, τ_∞. One of the most illustrative examples of this phenomenon is the bouncing ball automaton. It models an elastic ball that is bouncing off the ground, loosing a fraction, $1/c$, of its energy with each bounce.

Proposition 2.1 *The bouncing ball automaton is Zeno.*

Proof: Let x_1 denote the altitude of the ball and x_2 its vertical speed. The first bounce occurs at time $\tau_0' = \tau_1 = (-x_2(0) + \sqrt{x_2(0)^2 + 2gx_1(0)})/g$. The N^{th} bounce occurs at time $\tau_N = \tau_1 + \sum_{k=1}^{N} 2x_2(0)/(gc^{k-1})$. Since $c > 1$ the series on the right hand side converges. Thus the Zeno time of the execution with initial state $(q, x(0))$ is

$$\tau_\infty = \frac{-x_2(0) + \sqrt{x_2(0)^2 + 2gx_1(0)}}{g} + \frac{2cx_2(0)}{g(c-1)}. \quad \square \tag{1}$$

It should be noted that the bouncing ball automaton could be regularized by modeling the ground as a spring [7] and letting the spring constant be $1/\epsilon$, with $\epsilon \downarrow 0$, as seen in Figure 3. This gives (not too surprisingly) that after τ_∞, the ball just remains on the ground.

However, this last type of Zeno automata will not be of importance to the behavior based control system, where we only will have to employ Filippov solutions as our regularizations. Therefore we will not go further into detail about how to formally regularize this type of Zeno automata [7].

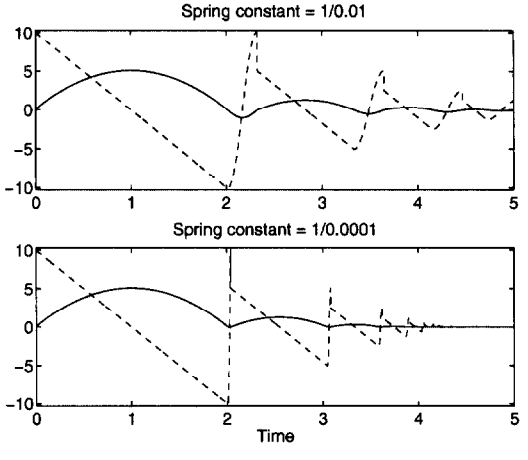


Figure 3: Regularization of the bouncing ball. The solid line corresponds to the position of the ball while the dash-dotted line corresponds to its velocity.

3 Behavior Based Robotics

For autonomous robots operating in an unknown, dynamic environment, a successful way of structuring the controllers is within a behavior based framework [3]. Different robot behaviors are identified, e.g. avoid-obstacle or move-to-goal, and their functionality is defined by a mapping from sensory data to a desired base action. These desired actions are normally fused together by an arbitration mechanism, as seen in Figure 4, where the standard way is to represent the goals, targets and obstacles by weighted attractive or repulsive potential fields. For instance, while approaching a target, an obstacle avoidance behavior has to be active for safety reasons while the performance is improved if the robot tries to approach the goal at the same time as it is avoiding obstacles [3, 2].

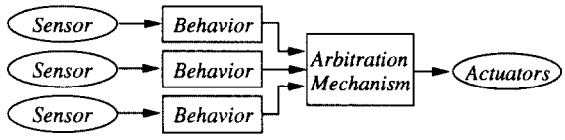


Figure 4: Block diagram of the behavior based control architecture.

3.1 The Obstacle Negotiation Problem

The specific problem that will be investigated in this article is how to move a robot between two points. This *point-to-point motion* should be done so that the detection of an obstacle results in a repulsive potential field when the robot is closer to the obstacle than a desired safety distance, d_{OA}, where the subscript OA stands for obstacle avoidance.

We assume that the robot's longitudinal velocity, v, will be kept constant and that the heading of the robot, ϕ, can be controlled directly.

The behavior that we will focus on initially is a so called *reactive* obstacle avoidance behavior. The word reactive, a commonly used one in the robotics community, is used here since it is a behavior that can be thought of as a reflex. When the robot moves too close to an obstacle, it is forced to change the motion in order to avoid hitting the obstacle. This is a reasonable safety strategy, since the robot may move around in an unstructured world, where the occurrence of unpredicted, or unmodeled obstacles is very likely.

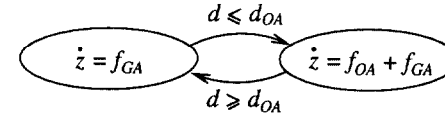
If the sonars on the robot, with center of gravity at (x, y) and heading ϕ, detect a point-obstacle at (x_{ob}, y_{ob}), closer to the robot than d_{OA}, the reactive control response will be given by a vector field acting on the robot as $\dot{z} = f_{OA}(z)$, where $z = (x, y)^T$. A standard kinematic model of the mobile robot [1] would then give us that

$$\dot{z} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} v\cos(\phi) \\ v\sin(\phi) \end{pmatrix} = f_{OA}(z) \qquad (2)$$
$$\phi = \pi + \text{atan2}(y_{ob} - y, x_{ob} - x).$$

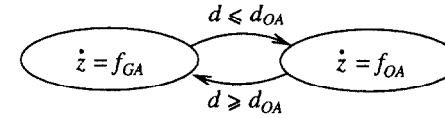
This is obviously a simplification but it still gives a model that is rich enough to capture the, from our point of view, relevant phenomena. That is, when there is chattering on the real platform, it should reveal itself here as a Zeno execution of the hybrid automaton.

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 contribute with. For a Nomad 200, that is going to be our experimental platform, this corresponds to taking the sum over 16 elements since the Nomad is equipped with 16 ultrasonic sensors.

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, depending on whether the behaviors are active simultaneously or not. (See Figure 5.) A system with hard switches is to prefer from both a design and analysis perspective, as already mentioned, since it allows us to view each behavior as a distinct node in the automaton. This will be the case for the remainder of the article, and 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 fused behaviors.



(a) Fused behaviors



(b) Hard switches

Figure 5: The two possible goal attraction and obstacle avoidance automata. Here, d_{OA} is the fixed distance from the obstacle where the obstacle avoidance behavior becomes active.

Proposition 3.1 *The hybrid automaton in Figure 5(b) can admit Zeno executions.*

This obvious fact is best illustrated by Figure 6, where the extra node that needs to be added in order to regularize the automaton can easily be identified as well. The extra node is just a node containing the sliding dynamics that is defined on the boundary between the two behaviors. In the following section we will investigate how well this approach works in practice.

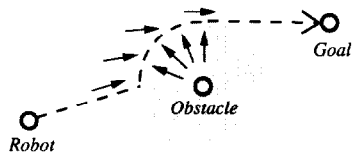


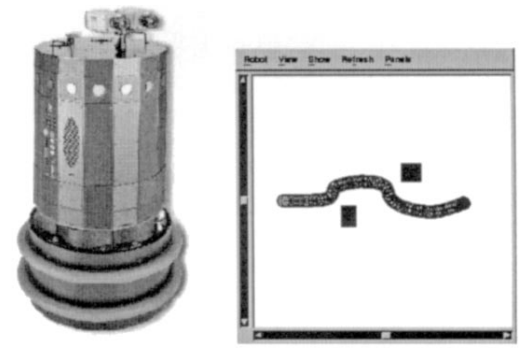
Figure 6: Goal attraction together with obstacle avoidance results in a Filippov type Zeno automaton. The grey region around the obstacle corresponds to the region where obstacle avoidance is active.

4 Experimental Results

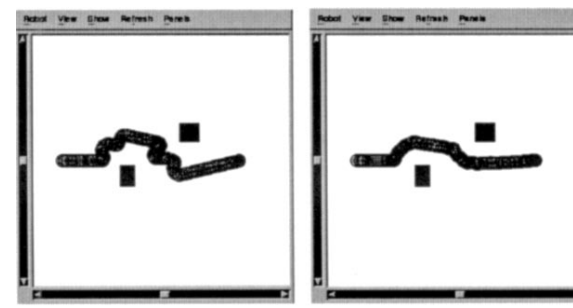
The regularized point-to-point automaton was implemented and tested on a Nomad 200 mobile robot. In Figure 7, the results from running the system on the **Nserver**, the Nomad simulation package, can be seen. The reason for choosing to display simulated results comes from the fact that we then can start with exactly the same initial conditions in all three cases. This is necessary for the evaluation of our proposed method, but the same code that is running on the **Nserver** is also running on the real Nomad in a satisfactory way.

In (7b) fused behaviors are displayed, resulting in a smooth movement around the obstacle, while the chattering solution in (7c) corresponds to hard switches. The reason why we do not have sliding in this case is due to the dynamics of the robot that was ignored in the analysis. It is still clear that from a performance perspective, (7c) is an unsuccessful design. In (7d) the results from using a regularized automaton can be seen, and even though we only have one behavior active at a time, the performance is satisfactory.

Some comments need to be made about the automatic generation of the Filippov solution in the regularized automaton. In the point-to-point motion scenario, the two behaviors send their suggested directions for the robot to follow to the arbitration mechanism. 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



(a) Nomad 200 (b) Fused behaviors



(c) Hard switches (d) Regularized hybrid automaton

Figure 7: Simulation of b) fused behaviors, c) hard switches, and d) a regularized automaton on the Nomad simulator, the **Nserver**.

sliding solution is just $f_S = \alpha f_{OA} + (1-\alpha) f_{GA}$, where $\alpha \in [0,1]$ is chosen so that $f_S \perp f_{OA}$. (See the Figures 5 and 6.) Adding this type of information about the different behaviors makes it possible to generate the extra node in the automaton automatically. It also suggests that our method would scale nicely when more that two behaviors affect the motion of the robot.

5 Conclusions

In this article, we investigated how to model a behavior based control system for mobile robots as a hybrid automaton. Based on the observation that a robotic system, where the behaviors are fused smoothly, gives a better overall performance than a system where behaviors with high priority over-rule those with low priority (hard switches), we wanted our automata approach to result in a hybrid automaton with similar performance as the fused behavior based system. The reason for introducing this automata approach in the first place is that we believe that there is much to gain from both an

analysis and a design perspective if the system could be modeled in a systematic way as a hybrid dynamic system.

Hard switches seem more appropriate as a basis for a hybrid automaton, where each node should correspond to a distinct behavior. However, this approach results in automata that exhibit an infinite number of discrete transitions in finite time, that can be dealt with by a regularization procedure, involving adding extra nodes to the automaton. This gives a system with similar performance as a fused behavior based system. The performance aspect of this approach was also verified experimentally.

References

[1] J. Ackermann. *Robust Control.* Springer-Verlag, London, 1993.

[2] M. Andersson, A. Orebäck, M. Lindström, and H.I. Christensen. *Intelligent Sensor Based Robotics.* Ch. ISR: An Intelligent Service Robot, Lecture Notes in Artificial Intelligence, Heidelberg: Springer Verlag, 1999.

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

[4] R. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23, 1986.

[5] M. Egerstedt, X. Hu, and A. Stotsky. A Hybrid Control Approach to Action Coordination for Mobile Robots. Proceedings of *IFAC'99:14th World Congress*, Beijing, China, Jul., 1999.

[6] A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides.* Kluwer Academic Publishers, 1988.

[7] K. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. Regularization of Zeno Hybrid Automata. *Systems and Control Letters*, 1999. Accepted for publication in 1999 Special Issue on Hybrid Systems.

[8] J. Košecká. *A Framework for Modeling and Verifying Visually Guided Agents: Design, Analysis and Experiments.* Dissertation, Grasp Lab, March 1996.

[9] E. Large, H.I. Christensen, and R. Bajcsy. Dynamic Robot Planning: Cooperation through Competition, in *IEEE International Conference on Robotics and Automation 1997*, Vol. 3, pp. 2306–2312, 1997.

[10] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for Reachability Specifications for Hybrid Systems. *Automatica*, Vol. 35, No. 3, March 1999.

[11] J. Malmborg. *Analysis and Design of Hybrid Control Systems.* PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, May 1998.

[12] C. Tomlin, G. Papas, J. Košecká, J. Lygeros, and S.S. Sastry. Advanced Air Traffic Automation: A Case Study in Distributed Decentralized Control, *Control Problems in Robotics*, Lecture Notes in Control and Information Sciences 230, Springer-Verlag, London, 1998.