An Automata-based Approach to Robotic Map Learning

Kenneth Basye
kbasye@black.clarku.edu
Department of Mathematics and Computer Science
Clark University, 950 Main St., Worcester, MA 01610

Abstract

We have developed a system for robotic map construction based on representing the robot's environment as a finite automaton. In this paper we first describe the automaton model, then present an algorithm used to control exploration and construct a map. We describe the robotic systems used by the algorithm as procedures, and report on experiments performed using the system in simulation and in a real environment.

Introduction

We have developed a system for robotic map construction based on representing the robot's environment as a finite automaton. This approach has been used previously by Kuipers, Levitt, Mataric and others.[Kuipers and Byun, 1988, Levitt *et al.*, 1987, Mataric, 1991, Malkin and Addanki, 1990] Essentially, the idea is to provide the robot with a set of actions that it can perform which consistently leave the robot in some set of locations in the world and with a means of sensing information about these locations. The locations reached become the states in the robot's representation, while the actions specify the transition function. To put it another way, we provide the robot with a set of actions which restrict its interaction with the world in such a way that the world/robot pair behaves like a finite state automata, and the robot's task is to infer the structure of this automaton by experimenting with it. It is natural to divide the problem of developing such a system into two parts. The first part is an algorithm for controlling exploration of the environment and constructing a map from the data gathered. The second part is the set of actions and sensing strategies that are used as procedures by this algorithm.

The first part of the problem, often called *automata inference*, has quite a long history. In his seminal paper, Moore [1956] both introduces the formulation of finite state machines that now bears his name and considers a set of problems involving inference of facts from the output of such machines. Later work provided answers to some of these problems, and introduced new variations [Gold, 1972, Angluin, 1978, Gold, 1978, Pitt and Warmuth, 1989]. Two results, one by Angluin [1987] and one by Rivest and Schapire [1989] are particularly interesting. In the first, Angluin provides a polynomial-time algorithm for inferring the smallest automaton given the ability to reset the automaton and a source of counterexamples. In this model, at any point, the robot can hypothesize an automaton and the source of counterexamples will indicate if it is correct and, if it is not, provide a sequence of inputs on which the hypothesized and actual automata generate different outputs. In the second, Rivest and Schapire show how to avoid the necessity of resetting in certain cases and how to dispense with both the reset and the source of counterexamples in others.

We have considered several variations on the problem of automata inference [Basye, 1992]. In particular, we have looked at problems that involve noisy outputs and/or noisy transition functions, but that also involve additional structure that can be exploited by the learner. Our goal has been to examine problems that reflect the realities of current robotic capabilities, and to consider what additional information might be available that will make the problem tractable.

In the remainder of this paper we look at one mapping problem and its solution. We begin by explaining the assumptions the learning algorithm makes about the environment to be learned and the capabilities of the robot. We discuss the learning algorithm and consider its application to the problem of learning simple office environments. We then look at the subsystems developed to support the algorithm in such environments, and discuss the performance to the entire system.

The Ground Rules

In this section we examine the structural properties of the environment the robot is trying to map, and the properties of the interaction between the robot and the environment. These properties are assumed by the learning algorithm introduced below to hold in any environment it is required to map. Thus they may restrict the kinds of environments the robot can function in, or dictate the level of capability the robot must have.

Most fundamentally, we assume that the robot has been provided with a set of basic actions which restrict its interaction with the world in such a way that constructing an automaton model is reasonable. Further, we assume that the environment is undirected, so that for any action which takes us from location q_1 to location q_2, there is an action which reverses this step. We assume that the robot knows how to reverse each of its actions. No other assumptions are made about the environment.

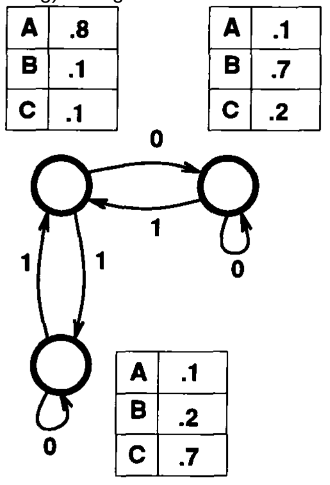
With regard to the robot's interaction with the environment, we assume that the robot is able to provide a unique label to the learner for each location reached. The label is assumed to be correct most of the time, but may occasionally be incorrect. Further, it is assumed that the robot will correctly perform the action directed by the learner most of the time. Finally, we assume that the learner is able to perceive situations in which performing some action will leave the robot in the same location. In a later section we show how the hallways of an office building, together with simple robotic behaviours for moving and sensing, comprise an environment that has the properties outlined above.

A Simple Mapping Algorithm

The algorithm uses a simple strategy to explore the graph and records, for each pair of labels, (l_i, l_k), and each action, a, the number of times that an observation of l_i followed by performing a resulted in an observation of l_k. After enough exploration, an automata can be extracted from these statistics. If l_k is the most frequently observed label after doing command a when observing label l_i, then we construct a transition on action a from state q_i to state q_k in the resulting map.

More formally, the state-transition graph can be learned in the following way:

1. Initialize L, the set of labels, to the label of the starting location.
2. For each action a, construct a two-dimensional table T_a indexed in each dimension by L.
3. For some number of steps,
   (a) Choose an action to execute by summing the rows of each table corresponding to the label of the current state. These sums represent the number of times each action has been taken following this label. Execute the least-taken action, and get the label of the resulting state.
   (b) If the label is not in L, add it, and extend each table to include a row and column for the new label. Initialize each added table entry to 0.
   (c) Increment by one the values of $T_a(l_i, l_j)$ and $T_b(l_j, l_i)$, where a is the action just taken, b is the inverse of a, l_i is the previously seen label, and l_j is the label of the just-reached location.
   (d) Increment by one the values of $T_a(l_j, l_j)$ for each action a which is perceived to leave the robot in the same location.



A	.8
B	.1
C	.1

A	.1
B	.7
C	.2

A	.1
B	.2
C	.7

T_0	A	B	C
A	12	**109**	37
B	10	**89**	55
C	56	26	**92**

T_1	A	B	C
A	35	19	**108**
B	**125**	23	17
C	**81**	54	52

Figure 1: Results of running the graph-identification algorithm in the simple environment shown for 1,000 steps.

4. Construct an automata with a transition (q_i, a, q_j) only if $T_a(l_i, l_j) > T_a(l_i, l_k)$ for all k not equal to j.

Figure 1 shows the results of a simulated run of this algorithm in a very simple environment. The small tables specify the probabilities of perceiving each label at each location; the large tables indicate, for each action, the frequency of sequential label pairs. The resulting transition matrix is encoded by the largest element in each row of each table, which is in bold-face type.

We have been deliberately vague about the requirements for this algorithm, in particular about how much noise in movement and sensing can be tolerated, and about the number of steps required. In another paper [Kaelbling *et al.*, 1992], we provide a simpler version of this algorithm and give a detailed proof of the required number of steps in terms of the probabilities of correct actions and observations, the size of the environment and other factors. Our goal in developing this algorithm was to reduce the number of steps required to a minimum, at the expense of being able construct such a proof.

Simulation and a Real Mapping System

The target environment for the algorithm was a single floor of a medium-sized building. We used doorways and the intersections of hallways as the locations, and used as actions the set $\{N, E, S, W\}$. Figure 2 shows an automata model of the 4th floor of the CIT building at Brown University. We simulated the algorithm above on this environment, controlling the number of steps made and the probability of correct observation, P. Figure 3 shows a plot of the number of successes out of 50 trials for various values of P and different numbers of steps.

These simulations were used primarily as a means

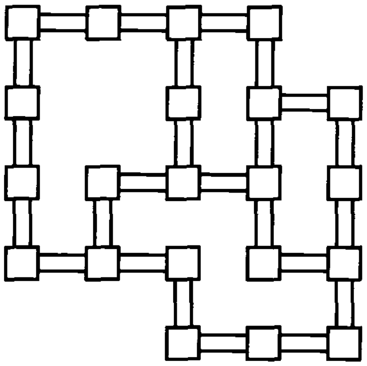


Figure 2: The CIT 4 environment

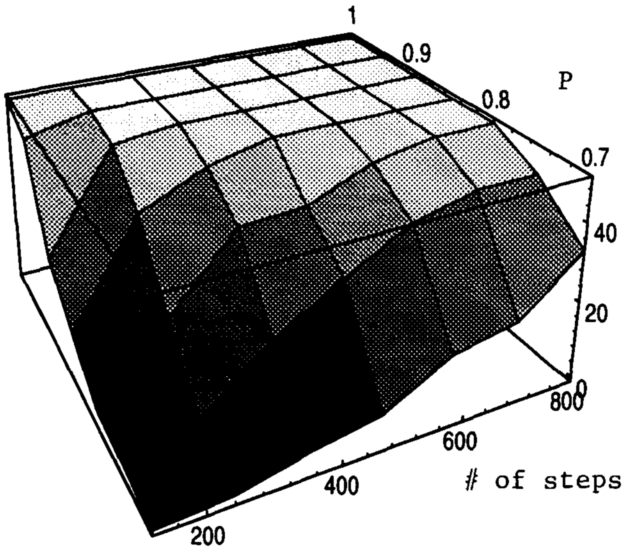


Figure 3: Results for algorithm in the CIT 4 environment

of checking that the algorithm could perform in a reasonable number of steps in the environment. Our goal was to provide the algorithm with sensors that returned correct labels at least 90% of the time, and to learn correct maps in a number of steps that was 5-10 time the number of total transitions in the environment. The simulations showed that this was a reasonable goal.

We implemented a system based on this algorithm using a small robot with simple sensors. This system consists of three layers. The top layer is basically the algorithm described above. It issues movement commands from the set $\{N, E, S, W\}$ and receives labels that consist of a unique integer index and a junction type. For example, a corner junction with hallways leading South and West will have type SW. The type of the junction is used to determine which actions would leave the robot in the same location (in this case, actions N and E).

The bottom layer is a set of low-level movement behaviours. This set includes a hallway-following behaviour, a corner-turning behaviour, and so on. Essentially, this set of behaviours is what one might assume when giving another person directions in a hallway environment, *e.g.*, "Go down this hall and turn left, then turn right at the next door." These behaviours are based on a simple set of sensors. They consist of several *controllers*, and a single decision module that, at any given point, determines which controller to use to best accomplish the behaviour. For example, the hallway-following behaviour has a *centerline* controller it uses for moving down a hallway when it is near the center of the hallway and aligned correctly, and several others that can be used to align the robot in the hallway and move it toward the center.

Each controller is a simple proportional integral derivative (PID) control scheme. An error signal is generated from the sensors by some simple means, and attenuated by a proportional factor. Other correction factors are derived by differentiating and integrating past signals and are added to the proportional signal. The resulting output signal is used to command the drive and steering motors on the robot base. In the case of the centerline controller, the error signal is the difference between the left and right sonar readings. This signal is lowest when the robot is near the middle of the hall, and is sufficient to keep the robot there while moving down the hall. Other controllers are used to put the robot in a state where this controller is useful.

The decision module for a behaviour is responsible for deciding which controller is appropriate in order to achieve the desired behaviour. It is also responsible for determining when a behaviour has been completed or has failed completely. This is implemented using a finite state machine with controllers as states. Transitions between states are made on the basis of outputs from the controllers; these are available after each cycle.

The middle layer has two functions. First, it translates action commands from the top layer into invocations of the appropriate behaviours. Second, it generates the labels used by the algorithm. It uses both the robot's odometers and sonars to achieve this goal. The odometers are used to keep track of the position of the robot and this position is used to provide the part of the label that makes it unique. At each location, the robot calculates its position from the past position and new odometer readings. The robot also takes several sonar readings as it reaches the center of the junction, and uses these to classify the junction as a corner, T, etc. This is combined with the robot's current orientation to provide the oriented junction type of the location. It is then necessary to determine whether the current location corresponds to one that has been visited before. This is done by considering those labels that match the junction type of the current label and finding the stored label with the minimum Euclidean distance between its position and the position of the current label. If this distance is below a certain threshold, the two locations are deemed to be identical, and the label of the stored location is used as the current label. The position portion of the stored label is updated using the position of the current label; specifically, the position in the stored label is a running average of all positions which have matched that label. If no position is close enough, a new label is generated and tagged with the current position.

Conclusion

Finite state automata can be used by robots to represent their environments when the capabilities of the robot and the structure of the environment together limit the robots interactions with the environment. Such representations are compact, easy to understand, and natural. In this paper we have presented an algorithm that directs exploration of such an environment and builds a representation of that environment. The algorithm is designed to tolerate noise in both movement and sensing, whether due to error or changes in the environment.

In future work, we are interested in reducing the amount of data required by the algorithm by using constraints on the final result as a bias in interpreting the data collected by the algorithm. The idea is to consider the most likely automaton given only the data, and check for violations of additional constraints, for example, undirectedness. If the most likely inference violates a constraint, we look for ways to modify the inference to satisfy the constraints and use standard statistical tests to determine which modification best fit the original data.

References

[Angluin, 1978] Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[Angluin, 1987] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[Basye, 1992] Kenneth J. Basye. *A Framework for Map Construction.* PhD thesis, Brown University Department of Computer Science, Providence, Rhode Island, 1992.

[Gold, 1972] E. Mark Gold. System identification via state characterization. *Automatica*, 8:621–636, 1972.

[Gold, 1978] E. Mark Gold. Complexity of automaton identification from given sets. *Information and Control*, 37:302–320, 1978.

[Kaelbling *et al.*, 1992] Leslie Kaelbling, Kenneth Basye, Thomas Dean, Evangelos Kokkevis, and Oded Maron. Robot map-learning as learning labeled graphs from noisy data. Technical Report CS-92-15, Brown University Department of Computer Science, 1992.

[Kuipers and Byun, 1988] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 774–779. American Association for Artificial Intelligence, 1988.

[Levitt *et al.*, 1987] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, and Philip C. Nelson. Qualitative landmark-based path planning and following. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 689–694. American Association for Artificial Intelligence, 1987.

[Malkin and Addanki, 1990] Peter K. Malkin and Sanjaya Addanki. Lognets: A hybrid graph spatial representation for robot navigation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1045–1050. American Association for Artificial Intelligence, 1990.

[Mataric, 1991] Maja J. Mataric. A distributed model of mobile robot environment-learning and navigation. Technical Report Technical Report 1228, MIT Artificial Intelligence Laboratory, 1991.

[Moore, 1956] Edward F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton University Press, Princeton, New Jersey, 1956.

[Pitt and Warmuth, 1989] Leonard Pitt and Manfred K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*, pages 421–432, 1989.

[Rivest and Schapire, 1989] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*, pages 411–420, 1989.