Continuous Categories For a Mobile Robot

Michael T. Rosenstein and Paul R. Cohen
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{mtr, cohen}@cs.umass.edu

Abstract

Autonomous agents make frequent use of knowledge in the form of categories — categories of objects, human gestures, web pages, and so on. This paper describes a way for agents to learn such categories for themselves through interaction with the environment. In particular, the learning algorithm transforms raw sensor readings into clusters of time series that have predictive value to the agent. We address several issues related to the use of an uninterpreted sensory apparatus and show specific examples where a Pioneer 1 mobile robot interacts with objects in a cluttered laboratory setting.

Introduction

"There is nothing more basic than categorization to our thought, perception, action, and speech" (Lakoff 1987). For autonomous agents, categories often appear as abstractions of raw sensor readings that provide a means for recognizing circumstances and predicting effects of actions. For example, such categories play an important role for a mobile robot that navigates around obstacles (Tani 1996), for a machine-vision system that recognizes hand gestures (Darrell, Essa, & Pentland 1996), for a simulated agent that maneuvers along a highway (McCallum 1996), and for a human-computer interface that automates repetitive tasks (Das, Caglayan, & Gonsalves 1998). Like Pierce and Kuipers (1997), Ram and Santamaria (1997) and others, e.g., (Iba 1991; Thrun 1999), we believe that sensorimotor agents can discover categories for themselves. Thus, the focus of this paper is an unsupervised method by which a mobile robot deduces meaningful categories from uninterpreted sensor readings.

Previously, we demonstrated a technique for extracting sensory concepts from time series data (Rosenstein & Cohen 1998). Our results were from a simple pursuit/avoidance game where two simulated players followed one of several deterministic movement strategies. The simulator recorded the distance between players throughout many games, and the resulting time series were transformed by an unsupervised learning algorithm into clusters of points. In effect, the algorithm found categories of sensorimotor experience, i.e., clusters of time series with similar patterns. This paper shows that a cluster-based approach to learning such categories scales to a more complicated robot domain with diverse kinds of sensors and real-world effects such as measurement noise, wheel slippage, and sonar reflections.

The learning algorithm, which we describe in detail in the next section, returns a *prototype* (Rosch & Lloyd 1978), or best example, for each category. For this work, prototypes are time series computed by averaging the members of a category or cluster. For example, Figure 1 shows a prototype based on seven instances of a Pioneer 1 robot bumping into a wall. By recognizing that its current situation is a match to the time series in Figure 1, the robot can predict that its bump sensor will go off a short time later. Below we provide evidence that sensory categories of this sort allow an agent to carve its world in some meaningful way. Since our robot refers to its prototypes with arbitrary symbols — not words like WALL or CONTACT — the *meaning* from such categories comes from the predictions it can make about sensor readings.

From Sensors to Categories

For a mobile robot operating in an environment of even modest complexity, sensory categories supply a needed level of abstraction away from raw sensor readings (Mahadevan, Theocharous, & Khaleeli 1998; Michaud & Mataric 1998; Pierce & Kuipers 1997; Ram & Santamaria 1997). Since our objective is that agents discover such categories for themselves — without supervision — we make use of clustering techniques that offer a general, unsupervised framework for categorizing data. However, the following subproblems exist, and this section outlines our solution to each one: event detection, time series comparison, sensor comparison, and sensor weighting.

Event Detection

Agents in continuous-time settings typically generate tremendous amounts of sensor data. Temporal abstraction is needed to focus a learning algorithm on the most

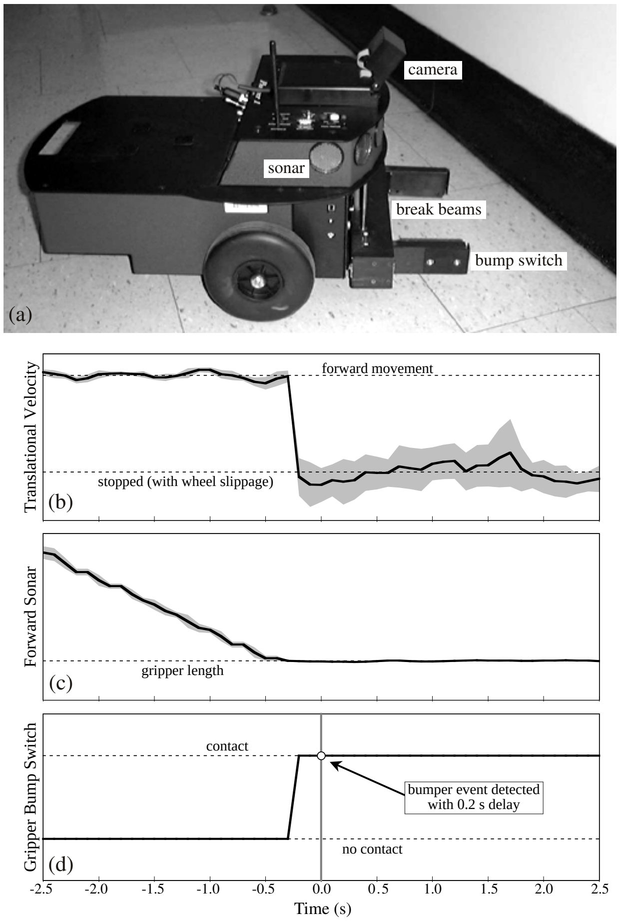
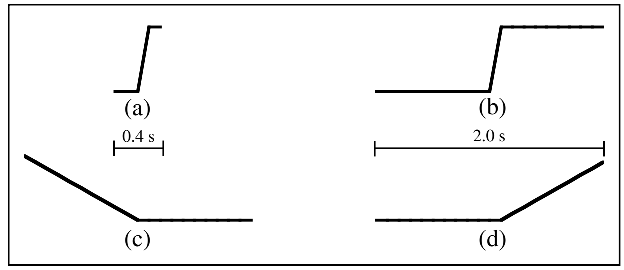


Figure 1: Prototype for seven instances of the Pioneer 1 mobile robot (a) bumping into a wall. Component time series are (b) translational velocity in mm/s, (c) forward sonar in mm, and (d) bump sensor (on or off). Gray regions indicate the level of prototype variability (one standard deviation from the mean).

pertinent parts of a robot's lifetime. For instance, a finite state machine representation can isolate key times when a robot encounters a landmark (Kuipers & Byun 1991; Mataric 1992) or branch point (Tani 1996). Another way to emphasize the most relevant parts of a long time series is to apply a suitable amount of compression and expansion along the time axis. For instance, Darrell *et al.* (1996) used a dynamic time warping (DTW) algorithm to perform this very sort of temporal abstraction when categorizing human gestures. Schmill *et al.* (1999) also utilized DTW to learn categories and operator models for a mobile robot. Dynamic time warping algorithms have the advantage of classifying time series in a velocity-independent fashion, although DTW represents a costly preprocessing step for clustering algorithms (Keogh 1997).

The alternative used here involves the real-time detection of events, i.e., key points in the sensor history.



		Template			
		a	b	c	d
Sensor	velocity	-0.99	-0.98	0.72	-0.70
	sonar	-0.67	-0.71	0.97	-0.54
	bump	1.00	1.00	-0.74	0.79

Figure 2: Event detection templates and sample correlations. The templates are (a) sharp edge, (b) long sharp edge, (c) slope-then-plateau, and (d) plateau-then-slope. The correlation values are for the data in Figure 1 where the templates were centered at the time of the rising edge for the bump switch. Shaded cells indicate values that are strong enough to trigger an event.

Our premise is that sensorimotor agents, such as infants and mobile robots, possess the innate ability to detect unexpected changes in sensor readings. A similar approach was taken by Das *et al.* (1998), who employed "triggers" as a way to isolate time series segments for a human-computer interface. In their application, a trigger such as the "Select All" command in a word processing program splits a prototype into two pieces: a prefix pattern for recognizing context and a suffix for predicting the user's next action. Our approach differs from theirs in that events act as signals for cluster analysis, rather than explicit decision points pulled out of existing clusters.

To recognize events, our learning algorithm makes use of simple rules that detect simple, conspicuous patterns such as the rising edge from a bump switch, the sudden change in wheel velocity when a robot stalls, or the jump in vision readings when an object suddenly disappears. These rules were implemented by computing the correlation of the most recent sensor readings with one of four templates (a short time series pattern) shown in Figure 2. Whenever one of the correlation values exceeds a threshold, an event signal triggers the learning algorithm to grab a five-second window of sensor readings centered on the event, and this multivariate time series then becomes a new instance for cluster analysis.

Time Series Comparison

History-based categories alleviate the real-world difficulties associated with hidden state, i.e., partially observable environments (McCallum 1996; Michaud & Mataric 1998; Ram & Santamaria 1997; Rosenstein &

Cohen 1998). One way to build such categories is to perform clustering of measurement *sequences*, although clustering algorithms originally designed for individual feature vectors must be extended to handle finite sensor histories. In other words, one must devise a means for time series comparison.

Every clustering algorithm, of which there are many (Everitt 1980), requires a measure of instance similarity, or dissimilarity, to guide its decisions about cluster membership. When designing a measure of dissimilarity for time series, one might take into account many different criteria, such as amplitude scaling, time-axis scaling, or linear drift (Keogh & Pazzani 1998). Our choice yields a very fast and simple algorithm, where we consider two time series, $\mathbf{X} = \{x_1, x_2, ..., x_m\}$ and $\mathbf{Y} = \{y_1, y_2, ..., y_m\}$, as vectors and quantify dissimilarity as the Euclidean distance[1] between them:

$$\text{dissimilarity} = \|\mathbf{X}^T - \mathbf{Y}^T\| = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \qquad (1)$$

Our choice of Eq. (1) was motivated by our previous work with a dynamics-based simulator (Rosenstein & Cohen 1998) and with the *method of delays*, a technique based on theory about dynamics. (See (Schreiber & Schmitz 1997) and references therein for other ways to classify time series by dynamics.) The method of delays transforms part of a time series to a point in *delay-coordinate space*, where delay coordinates are just successive measurements taken at a suitable time interval (Rosenstein, Collins, & De Luca 1994). Takens (1981) proved that delay coordinates preserve certain geometric properties of a dynamical system, even when access to the underlying state space is limited to a low-dimensional projection. The relevance for cluster analysis is that nearest neighbors in state space remain near one another in delay-coordinate space.

Sensor Comparison

One difficulty in working with robots is the variety of sensors. For instance, the Pioneer 1 mobile robot used in this study has sonars that measure distance in millimeters, infrared break beams that return one of two discrete values, and a vision system that computes an object's size, among other things, in square-pixels. How should a clustering algorithm weigh the contributions of *uninterpreted* sensors with different units and different ranges of values? Furthermore, how should the algorithm deal with sensors that are both continuous and categorical, such as the sonars which normally give real-valued measurements but also return a large default value when no objects are present?

We propose a two-step solution: (1) Cluster individual sensor histories as described above, thereby creating a small alphabet of patterns specific to each sensor. (2) Construct a unit- and scale-independent *signature* that stores the pattern of similarity between a newly observed time series and each member of the alphabet. For robot navigation and exploration, Kuipers and Byun (1991) defined the signature of a "distinctive place" as a subset of feature values that are maximized at the place. In general, signatures can be built for sensory categories, which may or may not involve physical locations. Moreover, the feature set, i.e., the alphabet patterns, need not be specified in advance, but rather can be learned by the agent from its raw sensor readings. For instance, Thrun (1999) used artificial neural networks and Bayesian statistics to extract features from a robot's sensor/action histories.

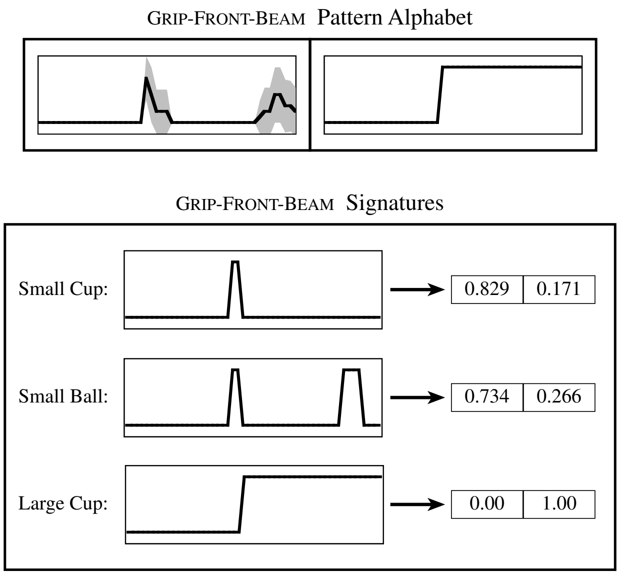


Figure 3: Pattern alphabet and representative signatures from seven interactions with each of three objects. Gray regions indicate the level of pattern variability (one standard deviation from the mean). The interaction with the small cup best matches the first alphabet pattern, which accounts for 82.9% of the aggregate similarity. Comparatively, the ball exhibits an improved match with both patterns, yet the net effect is an increased emphasis in the second alphabet pattern (from 17.1% to 26.6%). Unlike the large cup, both the ball and the small cup trip the front break beam momentarily before reaching the back of the robot's gripper. The small cup and ball differ in that the ball rolls away once the robot comes to a stop (passing through the front break beam a second time).

As an example, Figure 3 shows the alphabet of patterns for the sensor GRIP-FRONT-BEAM (one of two infrared break beams between the robot's gripper paddles). With the alphabet size set to two, the first two patterns encountered make up the initial alphabet, with each subsequent pattern forcing one iteration of an agglomerative clustering algorithm (Ward 1963)

[1] Our implementation actually makes use of the *squared* distance, which gives the same results using just m multiplications and $2m - 1$ additions or subtractions for each dissimilarity computation.

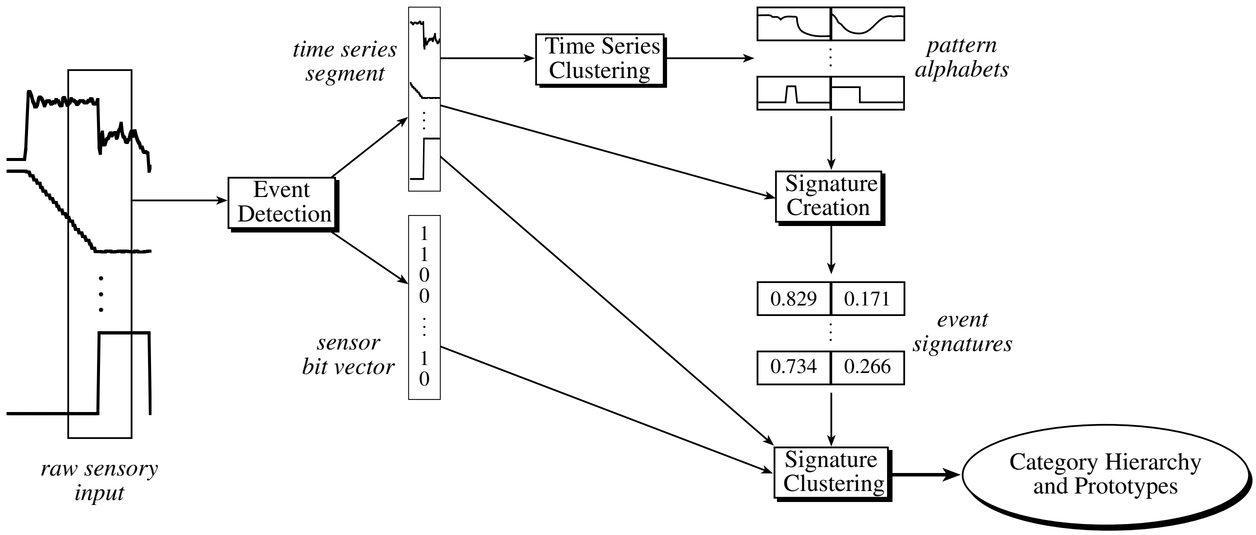


Figure 4: Schematic of the algorithm for learning time series categories from uninterpreted sensors. The first step, event detection, isolates a new segment of sensor readings and forms the input for the remaining algorithm steps. The purpose of time series clustering and signature creation is to convert each segment of sensor readings into a vector of unit- and scale-independent signatures. The final step forms clusters of signature vectors and supplies not only a category hierarchy but also a means for averaging the raw time series into a prototype for each category.

and thereby updating the alphabet to reflect the contribution of the new pattern. The signatures in Figure 3 are the result of several interactions with objects that fit the Pioneer's gripper. In each case, the slots in the signature were filled by computing the similarity (the reciprocal of dissimilarity) between the corresponding alphabet pattern and the recent history of GRIP-FRONT-BEAM. The actual values were also normalized by the total similarity for the signature. Thus, a signature is much like a unit vector in the space of alphabet patterns, with the projection onto each axis indicating the degree of match with the corresponding pattern. Notice in the figure that the small cup and the ball have similar (though consistently distinct) signatures which are vastly different from the large cup's signature. One could recognize the objects in Figure 3 based solely on the GRIP-FRONT-BEAM signature, although one must account for other sensors in more complex examples.

One limitation of the current algorithm is the need to specify the alphabet size in advance. Moreover, the same alphabet size is used for simple types of sensors (such as the break beams which show simple rising and falling edges) as for rich types of sensors (such as the sonars which respond to arbitrary movement patterns of the robot and its environment). One obvious way around this limitation is to customize each alphabet size to match the capabilities of the sensor, much like the approach taken with the event detectors. However, our previous results for a pursuit/avoidance game (Rosenstein & Cohen 1998) lead us to speculate about another alternative. We found that prediction of the game outcome was adversely affected when the number of clusters, i.e., the alphabet size, was too small, whereas little benefit was gained by increasing the number of clusters beyond a certain point. Thus, one could initialize the alphabet size to a large, conservative value, wait until the alphabet patterns stabilize, and then gradually shrink the alphabet by combining patterns until some performance criterion degrades to an unsatisfactory level.

Sensor Weighting

For any given event, only a small subset of a robot's sensors may contribute to the time series patterns that characterize the event. More generally, a learning algorithm should weigh the importance of each sensor when deciding if two patterns belong to the same category. For example, when grasping a small object a robot should place the greatest emphasis on its gripper, with little or no attention paid to battery level. Mahadevan *et al.* (1998) solved this problem with feedforward neural networks and supervised learning, whereas Schmill *et al.* (1999) handpicked the sensors that receive the same non-zero weight before utilizing an unsupervised clustering algorithm.

The final step in our approach to learning sensory categories applies another stage of clustering, but this time with *weighted* signatures as the input rather than raw time series. Specifically, this second pass of cluster analysis computes the dissimilarity between the ith and jth event patterns by taking a weighted average of the individual signature dissimilarities:

$$\text{dissimilarity}_{ij} = \frac{\sum_{k=1}^{N}(w_{ik}+w_{jk})\cdot\|\mathbf{S}_{ik}-\mathbf{S}_{jk}\|}{\sum_{k=1}^{N}(w_{ik}+w_{jk})}, \quad (2)$$

where N is the number of sensors and, $\mathbf{S}_{ik}$ is the ith signature for the kth sensor, with weight w_{ik}. Each

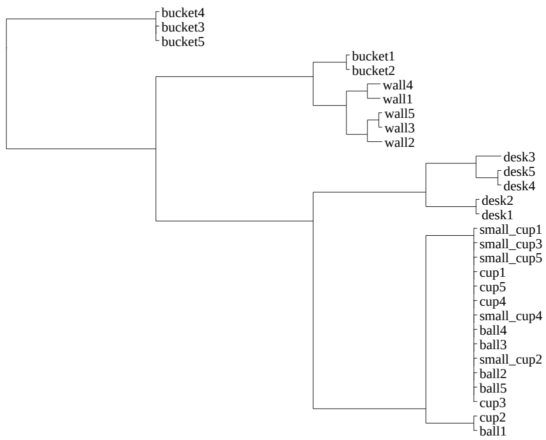


Figure 5: Category hierarchy for five interactions with each of six objects. The signature alphabet size was set to one for each sensor.

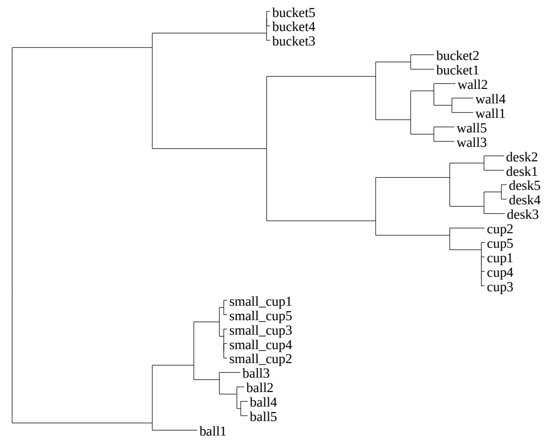


Figure 6: Category hierarchy for five interactions with each of six objects. The signature alphabet size was set to two for each sensor.

merge operation by the clustering algorithm creates another node in a cluster hierarchy, with new signatures and weights computed as an average of the constituents (adjusted for the sizes of the merged clusters). In a parallel operation, the raw time series used to create the signatures are also merged to give a prototype as in Figure 1. However, these time series play no direct role in the computation of Eq. (2).

To initialize the weights when a cluster has just one member (a new instance of sensor signatures) the learning algorithm makes use of the event detectors described previously. In particular, all the sensors that exhibit sudden changes within a 800 ms window are considered to be part of the same event and their weights are set to one; all other weights are set to zero. Essentially, the initial weights form a bit vector where 1 and 0 indicate, respectively, activity and no activity for the corresponding sensor. For example, bumping into a wall as in Figure 1 causes several sensors, like the forward sonars, the bump switch, and the velocity encoders, to have initial weights of one, but others, like the battery level and the gripper break beams, to have initial weights of zero. Although we found encouraging results with this straightforward approach that adds little computational cost, we imagine that some situations may require a more sophisticated weight initialization procedure. For instance, we make no attempt to adjust for correlated sensors such as the robot's five forward sonars. (The sonars carry, in effect, five times the influence of the bump switch which also returns information about frontward objects.)

Summary

Figure 4 is a schematic of the entire learning algorithm which runs both incrementally and in real time as the robot interacts with its environment (although the results in this paper are for post-processed data sets). On the left, raw sensor readings are made available to the robot at a rate of 10 Hz, with the event detectors continuously monitoring the time series for abrupt changes. When an event occurs, the algorithm collects a five-second segment of sensor readings and constructs a bit vector that indicates the active sensors. These data are passed to the clustering algorithms for additional analysis. First, the new time series are used to update the pattern alphabet for each sensor. Next, these same time series are converted to a set of signatures for subsequent clustering (with the bit vector acting as the initial sensor weights in Eq. (2)). On the right, the final output is a hierarchy of sensory categories with each category represented by a prototype like the one in Figure 1.

Interaction With Objects

We tested the algorithm depicted in Figure 4 by recording sensor data while the Pioneer robot interacted randomly with various objects, such as a ball, a cup, and a bucket. To control the robot we designed a simple SEEK-AND-APPROACH behavior, where the Pioneer turns a random amount, approaches the object closest to its line of sight, stops moving shortly after making contact, and then reverses direction for a randomly chosen time. Objects were recognized with the help of the robot's "blob" vision system that detects patches of red pixels in its image plane. Each object was given an otherwise indistinguishable red mark, so sensory categories were based on the nature of the interaction, not features from a detailed analysis of the visual scene. We ran the SEEK-AND-APPROACH controller repeatedly until the robot interacted at least five times with each of six objects.

Figures 5 and 6 are representative cluster hierarchies that summarize the output of the learning algorithm.

Notice that sensory experiences with the same object tend to cluster together at the lowest levels of the binary tree. Further up the hierarchy, the nodes represent abstractions of these individual experiences. For example, in Figure 5 all the graspable objects (the ball, the cups, and the leg of a desk) fall in the same branch of the tree and all the ungraspable, immovable objects (the wall) fall in another branch. Recall that labels such as "graspable" and "immovable" are meaningful to ourselves but may as well be arbitrary symbols to the robot. They symbolize prototypes, i.e., average time series.

Figures 5 and 6 differ in the size of the pattern alphabet used to construct the signatures. In Figure 5 the alphabet size was one, forcing each event's list of signatures to be equivalent to the corresponding bit vector constructed by detecting unexpected changes in sensor readings. Notice that the event detectors alone are capable of discriminating several categories of experiences. However, each bit must be expanded to a signature with at least two slots — as in Figure 6 — in order to tease apart some interactions such as those for the ball and the cups (which trigger both break beams but no other sensors). A small pattern alphabet always sufficed in our experiments, although we expect more complicated environments to require larger alphabets.

Prototypes serve not only as representatives for sensory categories, but also as state abstractions or goals. At any given time, the robot can determine its state by running a simple nearest neighbor algorithm that finds the best match between its recent sensor history and each of the prototypes. More generally, one can view the distance to the nearest neighbor as an indication of progress toward a goal state.

For example, suppose the robot's goal is to locate a small ball. Figure 7 shows the *relative* distance from the most recent sensor readings to several prototypes as the Pioneer approaches a ball. Initially, no progress is made toward each of the prospective goal states because they all involve activation of the break beams which are quiescent until about 0.5 s before the event is first detected. Moreover, the best match for the quiescent beams is the small cup since its prototype has the shortest activation time for GRIP-FRONT-BEAM. Similarly, the large cup is the worst match since its prototype has the longest activation times for both GRIP-FRONT-BEAM and GRIP-REAR-BEAM. Once the first beam breaks progress is made toward each prototype (which hurts the *relative* distance to the small cup as shown in Figure 7). At about 1.8 s after the initial event the rear break beam deactivates as the ball rolls away from the robot. Until this time, the robot's impoverished sensory apparatus is unable to distinguish the ball from the superordinate category that also includes the small cup.

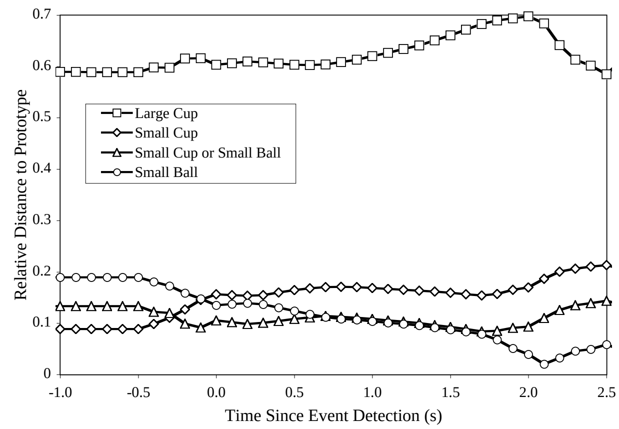


Figure 7: Relative distance to several prototypes. Shown are averages for five instances of the Pioneer robot approaching a small ball and then stopping a short time after making contact.

Discussion

Sensory categories and their prototypes not only act as states that support recognition and prediction, but also serve as operator models. In this paper we took the former view and focused on prototypes as state abstractions and goals, although several researchers augmented sensory prototypes for control purposes as well. For example, Schmill *et al.* (1999) recorded a mobile robot's current activity as part of the prototype data structure. The result was an operator model for a STRIPS-like planner where the prototype was split into two parts that correspond to pre- and post-conditions for the stored activity. Similarly, Ram and Santamaria (1997) used a case-based reasoning approach to control a mobile robot for a navigation task. Their system made use of continuous cases that store time series of sensor inputs and control outputs; given a desired sensory state, a case library can be queried for the sequence of control commands most likely to achieve that state.

Whether we view them as state abstractions or operator models, sensory categories may provide the foundation upon which to form abstract, propositional concepts. Mandler postulated that to build such a foundation, humans make use of an innate mechanism for sensory analysis that searches for regularities and preserves the continuous nature of perception (Mandler 1992). The unsupervised approach proposed here performs a similar form of sensory analysis for mobile robots. In particular, our implementation finds clusters of similar time series patterns and builds prototypes that retain the characteristics of the original sensor traces. We have yet to show a path from sensory categories to highly abstract concepts, although autonomous agents can still accomplish a great deal with prototypes.

Interestingly, prototypes and categories play a crucial role in human intelligence yet the act of categorization is often automatic and unconscious (Lakoff 1987). We regularly take categories for granted until forced to reason about them explicitly, such as when designing a feature set that helps a mobile robot navigate a cluttered office environment. Then we realize how difficult it can be to list the properties, *from the robot's perspective*, of simple categories like corridors, doorways, desks, and waste buckets. Supervised learning offers one common solution, where a *person* classifies the instances for subsequent category induction by a machine learning algorithm. This research is part of an effort to push the classification process inside the machine, freeing scientists and engineers from much of the tedious work when designing autonomous agents.

Acknowledgments

This research is supported by the National Defense Science and Engineering Graduate Fellowship and by DARPA under contracts F30602-97-1-0289 and F49620-97-1-0485. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation herein. The views and conclusions contained herein are those of the authors and should not be interpreted as the official policies or endorsements, expressed or implied, of DARPA or the U.S. Government.

References

Darrell, T. J.; Essa, I. A.; and Pentland, A. P. 1996. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(12):1236–1242.

Das, S.; Caglayan, A.; and Gonsalves, P. 1998. Increasing agent autonomy in dynamic environments. In *Proceedings of the Second International Conference on Autonomous Agents*. New York: ACM Press.

Everitt, B. S. 1980. *Cluster Analysis*. New York: John Wiley & Sons, Inc.

Iba, W. 1991. Learning to classify observed motor behavior. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 732–738. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Keogh, E. J., and Pazzani, M. J. 1998. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Presented at the 1998 AAAI Workshop on Predicting The Future: AI Approaches to Time Series Analysis.

Keogh, E. J. 1997. Fast similarity search in the presence of longitudinal scaling in time series databases. In *Proceedings of the Ninth International Conference on Tools with Artificial Intelligence*, 578–584. IEEE Press.

Kuipers, B., and Byun, Y.-T. 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems* 8:47–63.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things*. Chicago: University of Chicago Press.

Mahadevan, S.; Theocharous, G.; and Khaleeli, N. 1998. Rapid concept learning for mobile robots. *Machine Learning* 31(1–3):7–27. Also published in Autonomous Robots, Volume 5, Nos. 3/4.

Mandler, J. M. 1992. How to build a baby: II. Conceptual primitives. *Psychological Review* 99(4):587–604.

Mataric, M. J. 1992. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation* 8(3):304–312.

McCallum, A. K. 1996. Learning to use selective attention and short-term memory in sequential tasks. In *From Animals to Animats: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 315–324. Cambridge: MIT Press.

Michaud, F., and Mataric, M. J. 1998. Learning from history for behavior-based mobile robots in nonstationary conditions. *Machine Learning* 31(1–3):141–167. Also published in Autonomous Robots, Volume 5, Nos. 3/4.

Pierce, D., and Kuipers, B. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* 92:169–227.

Ram, A., and Santamaria, J. C. 1997. Continuous case-based reasoning. *Artificial Intelligence* 90:25–77.

Rosch, E., and Lloyd, B. B. 1978. *Cognition and Categorization*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rosenstein, M. T., and Cohen, P. R. 1998. Concepts from time series. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 739–745. AAAI Press.

Rosenstein, M. T.; Collins, J. J.; and De Luca, C. J. 1994. Reconstruction expansion as a geometry-based framework for choosing proper delay times. *Physica D* 73:82–98.

Schmill, M. D.; Oates, T.; and Cohen, P. R. 1999. Learned models for continuous planning. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 278–282. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Schreiber, T., and Schmitz, A. 1997. Classification of time series data with nonlinear similarity measures. *Physical Review Letters* 79(8):1475–1478.

Takens, F. 1981. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics* 898:366–381.

Tani, J. 1996. Model-based learning for mobile robot navigation from the dynamical systems perspective. *IEEE Transactions on Systems, Man, and Cybernetics Part B* 26(3):421–436.

Thrun, S. 1999. Bayesian landmark learning for mobile robot localization. *Machine Learning*. To appear.

Ward, J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* 58(301):236–244.