Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 - Nov. 03, 2001

Automatic Training Data Selection for Sensorimotor Primitives

Amy Larson and Richard Voyles
Department of Computer Science and Engineering
University of Minnesota
4-192 EE/CS Bldg, 200 Union Street SE
Minneapolis, MN 55455
{larson,voyles}@cs.umn.edu

Abstract

Sequencing sensorimotor primitives to achieve complex behaviors can simplify programming of robotic systems. Using programming by demonstration to code the component primitives can further simplify the process. Learning methods employed in programming by demonstration require comprehensive data sets, which place a significant burden on the user during demonstration. We present a generalized method whereby training sets can be automatically filtered, freeing the user from knowledge of the underlying learning method. We achieve this by first capturing the characteristic behavior for a demonstrated task, then determining a measure of distance from that behavior. With this information, data sets can be analyzed to determine whether a particular moment of demonstration is "good" and should be included in the final training set. Results from programming by demonstration of left wall-following on a mobile platform are presented. Additionally, we present a method for on-line performance analysis that takes advantage of the characteristic behavior identified in the filtering process.

1 Introduction

Building a set of base sensorimotor primitives, which can be sequenced to create a variety of behaviors or to carry out a complex task, can simplify the programming of robotic systems [9, 5, 11]. The hand-coding of primitives demands expertise due to the complexity of robotic systems and warrants further simplification. An ideal approach to programming sensorimotor primitives would capitalize on the expertise of the user while limiting the resource requirements of the expert programmer. Programming by demonstration is such an approach.

To create primitives, programming by demonstration can be implemented with a supervised learning technique such as artificial neural networks (ANN) or principal component analysis to learn a sensor-motor mapping. The key to successful training lies in the quality of the training data, which must be comprehensive (for robustness) and concise (for efficient training). Comprehensive data sets are particularly important when training robots, due to their mobility which increases the probability of interaction with novel environments. Therefore, good training data should include corrective behavior for a variety of situations.

Consider learning to drive. Training data should include examples of how to return to the road if the car inadvertently strays on to the shoulder. To demonstrate the corrective behavior the car must first be positioned on the shoulder, but unless training is explicitly turned off while the driver veers off the road to position the car to demonstrate the correction, the system receives contradictory information. Granted that training could be explicitly turned off and on, but if a multitude of situations is required, this places a large burden on the user.

In [6], a neural network is trained to autonomously operate a vehicle. The author addresses this issue of data collection by simulating corrective behavior. The system collects sensory (visual) and actuator (steering and velocity) data while a human demonstrates driving in the lane. The sensory data is then modified to simulate a variety of situations and the corresponding actuator data is modified to simulate corrective behavior for that situation. This approach requires task domain knowledge, precluding it from a generalized solution.

Similar to the work presented here, in [2] the authors develop a generalized approach to data filtration as applied to manipulators. Training data contained excessive human error preventing the system from efficiently performing the demonstrated task (specifically, peg-in-hole and opening a door). In contrast, our approach addresses efficient and convenient data collection for the construction of robust sensorimotor primitives for a mobile robot. Other previous work in the area of programming by demonstration includes [6, 4, 1, 8] for

a mobile system and includes [7, 3, 4] for stationary manipulator control.

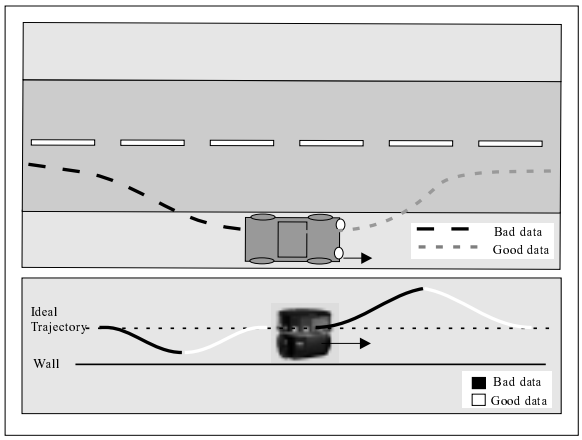

Figure 1: Top:*Depicts repositioning of car to demonstrate corrective behavior. The goal of automatic selection is to extract the "good" data.* Bottom:*Analogous to driving, this depicts right wall following where the task is to keep the robot a constant distance from the wall.*

2 Filtering

If, while demonstrating a sensorimotor primitive, a user is free to reposition the robot without regards to turning training off, then the observation data must be filtered to extract the "good" data. Without filtration, contradictory information exists within the training data to a point where the learning method cannot extract the inherent sensor-actuator mapping. Filtering can be achieved by determining a general rule of behavior and extracting the data adhering to that rule. We can assume that the user is following a certain rule of motor response to sensory information when demonstrating the task, but the difficulty lies in automatically capturing it in the presence of contradictory information.

Returning to the driving example, we can consider positioning the car between the shoulder and the yellow line as the general rule. Additionally, we could state that the ideal or characteristic behavior of driving is moving forward while centered in the lane. When filtering, however, data corresponding to the car not in this characteristic pose cannot simply be eliminated. Figure 1 demonstrates situations where the car is not in the lane, yet the corresponding data is essential to training because it demonstrates corrective behavior. Note that it is not the position of the car at a particular moment that determines the usability of the data, rather it is the direction of movement. Wall-following for mobile robots is analogous to driving and is also shown in Figure 1.

Moving the robot in a manner that brings it closer to its characteristic pose can be thought of as the underlying rule of any sensorimotor primitive. Thus when filtering, we must find precisely that data which corresponds to better positioning of the robot. Of course this can only be achieved when both the characteristic pose and a means of measuring distance from that ideal has been determined. Once again the difficulty lies in capturing this information from a data set containing contradictory information.

To address this, we consider those sensor readings containing only minor fluctuations throughout a demonstration as representative of the characteristic pose of the robot for that task. Projecting these values of consistent sensor readings onto vector space provides a measure of distance. With this information, we can evaluate an individual time-step of training in the context of surrounding data to determine whether it is moving closer to or farther from the characteristic behavior.

Returning to our driving example, we would first quantify the behavior of staying within the lane. When analyzing the data, if at a particular time-step the car is moving away from the lane (possibly due to the user positioning the car for a demonstration of corrective behavior), the corresponding data would be filtered out. Conversely, if at a time-step the car is moving towards the lane, it would be automatically selected.

2.1 Characteristic Vector

Data from a demonstrated task consists of sensor and actuator vectors. Each one of these vectors contains sensor readings and actuator values for a single time-step. The collection of sensor and actuator vectors across time comprises training data from which the learning method extracts the inherent sensor-actuator mapping.

To begin the filtering process, we first calculate the standard deviation of each sensor across time, thus obtaining a measure of consistency. Each sensor whose standard deviation falls below a threshold is labeled a key sensor. *Key sensors* are used to determine the characteristic behavior for the demonstrated primitive. Figure 2 depicts this process for the case of left wall-following whereby the robot must move forward, keeping its lefthand-side to the wall.

For each key sensor, a histogram of readings across time is constructed. The average of readings in the fullest bin is the characteristic reading for that sensor.

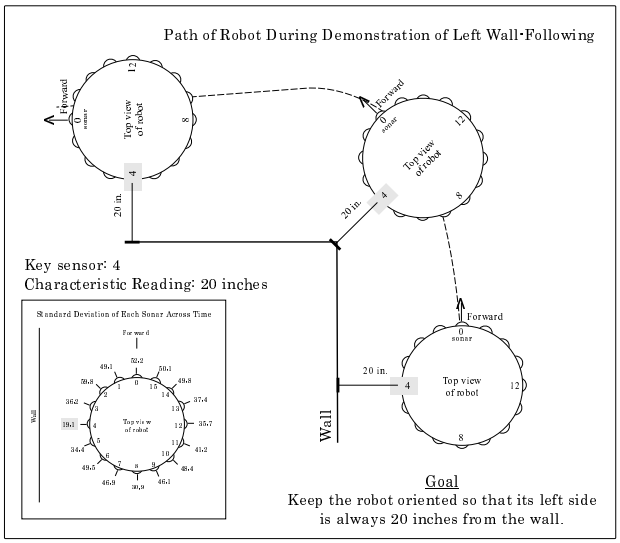

Figure 2: *Depicts a potential trajectory along which the demonstrator would move the robot during demonstration of left wall-following. The inset shows standard deviations obtained from a training set. Sonar 4 was the only one falling below the threshold and was labeled as a key sensor.*

The collection of characteristic readings for all key sensor is the *characteristic vector*. This vector quantifies the ideal pose for a specific task and can be used to evaluate the training data.

2.2 Data Filtering

At each time-step, the vector of key sensor readings is extracted from the training data, then the vector difference

$$v(t) = \sqrt{\sum (k(i) - s_t(i))^2}$$

is calculated, where $k(i)$ is the key sensor reading for sensor i and $s_t(i)$ is the reading for sensor i at time t. The resulting series of vector differences is smoothed over time. Smoothing insures that the local trend of the data can be established without getting lost in the minute changes that exist from one time-step to the next.

To analyze the slope at each data point of the training set, a window is slid along the smoothed vector differences, always centered at the point in question. The slope of this point is equated to the slope of the interpolated line of points contained within the sliding window. A positive slope indicates the vector difference is increasing, thus the robot is moving away from the characteristic behavior. This data point is then assumed "bad" and is removed.

3 Experiments

3.1 Wall-Following

3.1.1 Methods

We will present results from demonstrations of left wall-following using Nomad's Super Scout. Data sets were collected during three separate demonstrations. In Matlab, data was processed and filtered, then used as training data for a back-propagation artificial neural network. Although there were 16 sensor readings, every other sensor, in addition to the key sensors, was used for training. This was done to speed up training.

All networks were trained for a total of 100 epochs. Each network had an input layer of 9 units, a hidden layer of 6 units, and an output layer of 2 units. 5 separate initial weight configurations were chosen at random. Each of the 3 training sets, as described above, was used both as-is (non-filtered) and filtered, for a total of 6 training sets. Each of these was used to train each of the 5 networks for a total of 30 networks.

Each of the 30 networks was tested by running the robot along a straight wall with no obstacles. If the robot ran into the wall, the robot would undergo corrective measures to back away from the wall and appropriately re-orient to continue progress. On each run, sensor data was collected and used for performance metrics as described below. Some networks failed in that the robot did not make forward progress. These failed networks were not used in performance analysis.

For comparison, in one run, training data was marked as it was being collected and then was hand-filtered. The same networks were trained using the hand-filtered data and their performance was evaluated.

3.1.2 Performance Metrics

During testing, sonar and positional data was collected and used for three performance metrics which analyzed the filtering method. First we looked at the mean vector difference of the characteristic vector and the key sensors across time. The mean vector difference is a measure of the average distance from the ideal pose. It is subject to sensor noise. The vector difference is as described in the above formula.

Second, we analyzed positional error, which is also a measure of average distance from the ideal pose. Positional error was determined using the robot's onboard odometric system. In our necessarily constrained testing environment, the robot should move in a straight line, therefore we simply evaluated the y-value, which ideally should always be 0. Unfortunately, this value is subject to odometric error and to the initial position-

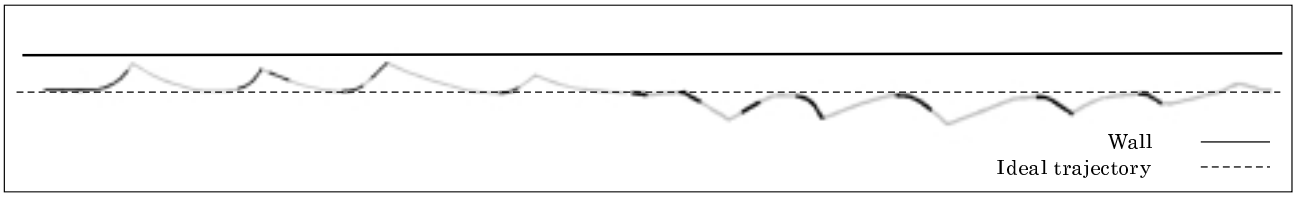

Figure 3: *Robot trajectory during demonstration of wall-following. The darkened sections indicate filtered sections of the training data set.*

ing of the robot. Therefore, the mean vector difference makes for a stronger performance metric.

Finally, the standard deviation of the key sensors was evaluated. The standard deviation measures variability in performance. For all three metrics, quantity and performance are inversely related. All values are indicated in inches.

3.1.3 Results

Figure 3 depicts the robot trajectory during a demonstration of left wall-following. The robot is under human control via a joystick during this phase. The *ideal trajectory* is the desired performance of the system once training has completed. (This is analogous to drawing a trajectory down the middle of the lane for driving.) The ideal trajectory is the characterstic behavior, therefore when the robot is steered away from this ideal, the filtering process should eliminate the corresponding training data. Figure 3 also depicts which of the data (the ligher sections) has been automatically selected for use in training.

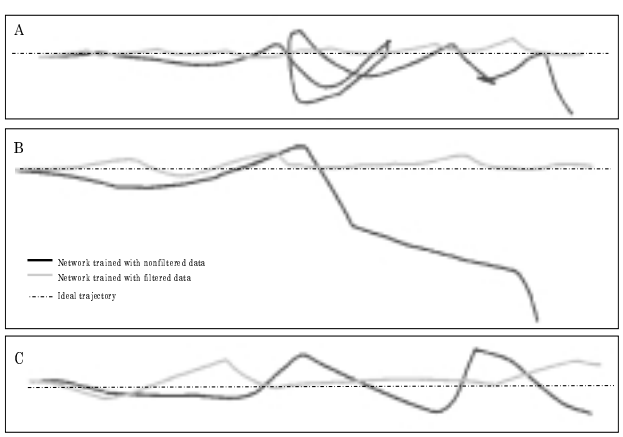

Figure 4: *Robot trajectories during testing. The three images correspond to three separate neural networks.* Table 1 *contains the error measures for all these trials.*

Figure 4 depicts robot trajectories during testing of left wall-following. The robot is now moving autonomously using a neural network trained on the data

		nonfiltered	filtered
A	mean vector difference	46.40	6.96
	standard deviation	62.94	20.97
	positional error	18.50	6.16
B	mean vector difference	66.51	10.78
	standard deviation	61.64	24.79
	positional error	17.84	8.31
C	mean vector difference	63.33	7.39
	standard deviation	60.22	22.97
	positional error	51.77	5.90

Table 1: *Performance results for networks A, B and C shown in Figure 4.*

	non-filtered	filtered	hand filtered
mean vector difference	30.91	7.58	2.45
standard deviation	42.21	31.81	13.17
positional difference	15.92	10.55	10.45
failures	2	2	1

Table 2: *Average performance of all trained networks.*

obtained during human demonstrations. The two trajectories compare a network trained first with a data set, unfiltered, then with the same data set, filtered. Each of the three figures corresponds to one of the three data sets. While the robot behavior that corresponds to training with filtered data is not perfect, it clearly outperforms the confused behavior of the other. Table 1 displays the performance error for these specific networks.

Performance was analyzed for all 30 networks using the metrics described above. Table 2 summarizes those results. One set of hand-filtered data was used for comparison. Hand-filtering is analogous to turning data collection on and off during repositioning.

It should be noted that collection of hand-filtered data took twice as long and required the user to pay

very close attention to the data collection process. This is precisely what we were trying to avoid in developing our filtering method.

3.2 Midline Traversal

The assumption that there is a consistency in sensor readings may seem strong, but extending the concept of a sensor, relaxes this assumption. To demonstrate this idea, we applied our filtering method to data collected during traversal of a hallway, in which the robot was shown to move down the center of a hallway.

Similar to wall-following, a midline traversal of a hallway requires the robot to move forward keeping a certain distance from the wall. Unlike wall-following, if the robot encounters hallways of varying width, the distance between the robot and the wall will vary relative to the hallway width. In this case, we will no longer have consistency in sonar readings. However, we are still looking for a consistent behavior, namely keeping the distance between the robot and the wall equal on both sides.

For this task, our sensors measured the difference in readings between sonar pairs. Each pair combination of every other sonar was considered for a total of 28 sensors. The characteristic vector was determined and filtering was carried out as described above for wall-following. The filtered data is shown in Figure 5. The dashed line represents the center line down the hallway. Movement away from the line is "bad" and should be filtered out.

Figure 5: *Robot trajectory during demonstration of midline traversal of a hallway. The dark sections indicate where data was filtered. The dashed line indicates the ideal trajectory.*

4 On-line Performance Analysis for Enhanced Training

Having a representation of the characteristic behavior has an additional advantage. It can be used as a performance measure during operation. Due to the randomness and plethora of tweak-able parameters of neural networks, it is difficult to find just the right combination of structure and training for success. With a measure of performance (i.e. the vector difference) we can identify points of difficulty while performing the task. Specific system weaknesses, which can be due to deficiencies in the training data, may be remedied by demonstrating the task within the environment that the system is having trouble.

To find points of performance difficulty, a recent history of sonar readings is maintained. If the vector difference of the characteristic vector and key sonar readings continually stays above a threshold, the slope of the vector differences within the history window is analyzed. If the slope is positive, then we alert the system that its performance is inadequate. By looking at the slope instead of exclusively at the vector difference, we can determine whether the robot's performance is continuing to degrade. This gives the system an opportunity to correct itself.

Figure 6 shows a trajectory of the robot under autonomous control during testing of left wall-following. The vertical lines indicate points at which the robot would be alerted to its degraded performance. For the purposes of this paper, the performance analysis was conducted off-line. However, the algorithm is designed explicitely for on-line use.

Once the system has insight into when its performance is inadequate, steps can be taken to improve. The system could either request immediate assistance in the form of more demonstrations or it could use a recent sonar history to identify environmental conditions that caused performance degradation.

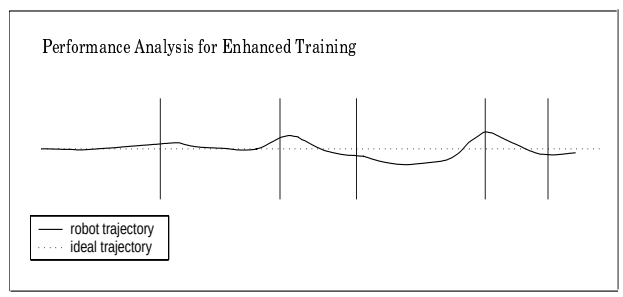

Figure 6: *Off-line analysis of performance during operation. Vertical lines indicate points at which the system would have been initially alerted to inadequate performance. Perfect performance would have been a* robot trajectory *equal to the* ideal trajectory

5 Conclusions and Future Work

Difficulties with programming inherently complex robotic systems motivated the development of automatic training data selection for sensorimotor primitives. Particularly troublesome is collecting a comprehensive training set due to the repositioning of the robot for demonstration of corrective behavior in a variety of situations. Similar to [2], we made use of an assumption that corrective behavior is generally com-

prised of movement towards a characteristic pose of the robot relative to its environment.

Without task knowledge, this method identified and quantified that characteristic pose. Additionally, we showed a means of determining at a single time-step (in the context of the surrounding data) whether the robot was moving towards or away from the characteristic behavior. This allowed for the successful extraction of good training data, as evidenced by the experiments.

Our method relies on the assumption that key sensors are those with continuity across time. If we extend the notion of a sensor to include such things as an interaction among sensor readings or as a rate of change of sensor readings, then it is reasonable to assume that the characteristic behavior of a sensorimotor primitive can be represented by "sensors" with a stable pattern of behavior across time. This was demonstrated in the hallway traversal experiment. Further exploration is required to determine the limits of this assumption.

The ultimate goal of this work is to create a robotic system capable of learning sequential tasks from human demonstration. Correctly parsing a task into its component primitives and identifying transition points when performing the task are the most challenging aspects of creating such a system. The characteristic vector and key sensors could be used for both. If we have captured a measure of good performance for the component primitives, then we should be able to match portions of the demonstration to specific primitives. Similarly, performance degradation may be used as an indicator of transition. In a companion paper [10], we developed an error measure for eigenvector learning, then used it as a transition cue between states and to select among primitives to optimize performance. The error measure developed in this paper can be similarly applied to artificial neural network learned behavior.

Acknowledgements

This work sponsored by Air Force Research Lab under contract F30602-96-2-0240.

References

[1] J. Hancock and C. Thorpe. Elvis: Eigenvectors for land vehicle image system. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 35–40, 1995.

[2] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, pages 2700–2705, 1996.

[3] S. B. Kang and K. Ikeuchi. Robot task programming by human demonstration: Mapping human grasps to manipulator grasps. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 97–104, 1994.

[4] Y. Kuniyoshi. Vision-based behaviors for multi-robot cooperation. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, volume 2, pages 925–932, 1994.

[5] P. Michelman and P. Allen. Forming complex dextrous manipulations from task primitives. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, pages 3383–3388, 1994.

[6] D. Pomerleau. *Neural network perception for mobile robot guidance.* PhD thesis, Carnegie Mellon University, 1992.

[7] P. Pook and D. Ballard. Recognizing teleoperated manipulations. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, volume 2, pages 578–585, 1993.

[8] P. Rybski and R. M. Voyles. Interactive task training of a mobile robot through human gesture recognition. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, pages 163–166, 1999.

[9] T. H. Speeter. Primitive-based control of the utah/mit dextrous hand. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, pages 866–877, 1991.

[10] M. Sutton, A. Larson, and R. M. Voyles. Performance evaluation of sensorimotor primitives using eigenvector learning method. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2001.

[11] R. M. Voyles, J. Morrow, and P. Khosla. Towards gesture-based programming: Shape from motion primordial learning of sensorimotor primitives. *IEEE Journal of Robotics and Automation*, 22(3-4):361–375, 1997.