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

Performance Evaluation of Sensorimotor Primitives Using Eigenvector Learning Method

Michael S. Sutton†, Amy Larson‡ and Richard Voyles‡

† Trinity University, San Antonio TX 78212
‡ Department of Computer Science, University of Minnesota, Minneapolis MN 55455

Abstract

We present a method to evaluate the performance of an eigenvector learned sensorimotor primitive for mobile robots. At runtime, the learning system projects sensor data onto the eigenspace using eigenvectors determined in training. The result of the projection is a set of sensor values and actuator values. We developed an error metric based on comparing the projected values with the actual sensor values. When the system performs closely to how it was trained, the difference between projected and actual sensors is small and hence the error metric is small. The error increases as the performance degrades. This method is not task specific and can be used for any eigenvector learned primitive.

Two example applications of the error metric are shown using wall following skills for a mobile robot. First, the metric is used as a transition cue for multi-primitive sequential tasks. Second, the error metric is used to create an adaptive system that chooses the best performing skill.

1 Introduction

Programming robots can be a complicated and arduous task requiring expert programmers. The costs of setting up and programming a robot to automate a task can outweigh the benefits of the automation; a problem amplified as product lifespan and batch size decrease. Robotic systems that could learn a task rather than being explicitely programmed would mitigate the problems of automation. Additionally, end users who are skilled at the task could guide learning (thus programming), requiring less of the limited resources of expert programmers.

There has been considerable success with algorithms that learn low level encapsulated capabilities[3, 8, 1]. These capabilities, or *primitives*[4], characterize an intelligent sensor-to-actuator mapping. One notable example of primitive learning is with artificial neural networks. ALVINN [5], an artificial neural network for autonomous driving, observes a human driver for a few minutes and then takes control of the vehicle. Another primitive learning technique uses eigenvector extraction to determine an underlying representation of the input-output relationship [1, 8]. The eigenvector method is able to learn linear relationships between sensors and actuators. Advantages of this learning method include transparency (unlike neural networks) and primitives that are easily combined. Because the method is linear, new primitives can be programmed simply by forming linear combinations.

Once a sensorimotor primitive is learned, the system may need to fine-tune the primitive or make minor adjustments to successfully perform in novel environments. Without an error measure, it would be very difficult to determine when such refinements are necessary. An error measure is particularly useful when the system is expected to work in a variety of situations and/or dangerous environments. The robot can be self-monitoring and alert users to its inability to successfully perform its task before doing irreparable damage.

In this paper, we present a task-independent performance metric for eigenvector learned sensorimotor primitives. In addition to its usefulness solely as an error measure, we have found it applicable to forming an adaptive system and to the sequencing of primitives to perform a complex task. Specifically, it can be used to optimize performance by selecting the best primitive for the current task and to identify transition points in a task of sequenced primitives. In related work, the authors of [6, 7] identified transition points using Hidden Markov Models. In both works, primitives were hand coded then a sequential task was demonstrated, requiring the system to automatically find component primitives. Our work uses the information already obtained during programming by demonstration of the primitives, then applies it during on-line execution of the sequential task.

2 Eigenvector Learning

We follow Hancock[1] and Voyles[8] in their approach to eigenvector learning. In the training process, the

robot learns a relationship between sensor data and actuator data. A training matrix is created from a collection of training vectors. Training vectors consist of actuator data appended to the end of the sensor data. For example, a training vector might look like, $v = [s_1, s_2, ..., s_n, c_1, c_2]$, where s_n is the range returned from sonar sensor n and c_1 and c_2 are the velocities for two motors. The training matrix Z is:

$$Z = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

where there are m training vectors. The next step is to find the average value of Z, where the average $a = (1/m)\sum v_i$. An offset matrix, T is then generated by subtracting the average vector from each row in Z. Given the offset matrix T, the covariance matrix is formed $C = T^t\, T$. Now the eigenvectors, e_i and eigenvalues, λ_i of C are found. The extracted eigenvectors provide a representation for the data in the training matrix. Generally, only the eigenvectors with the largest eigenvalues dominate the system. A good eigenvector will have large components corresponding to important sensors and actuators.

After the system has been trained, it is possible to generate actuator commands given sonar data. Given an eigenvector or a set of eigenvectors, the average vector, and the sensor values x, the sensor data is projected onto the eigenspace formed by the eigenvectors. The projection is performed with the following calculation:

$$v = a + \sum_{i=1}^{n} \left[(x - \text{sensor}(a)) \cdot \text{sensor}(e_i) \right] e_i$$

where a is the average of Z, x is the runtime vector of sensor values, sensor(a) represents just the sensor values of the vector a, and i is the number of eigenvectors used to reconstruct the data. The resulting vector v is a reconstruction of the sensor data and motor data. The motor data can then be given as the next motor command.

This method will learn a sensor-actuator mapping as long as the relationships between columns are linear. This may seem prohibitive, however many nonlinear relationships can be approximated with linear or piecewise linear relationships.

3 Evaluating Performance

Once a primitive is learned, it is important to be able to evaluate the performance level of the learned task. There are many reasons a robot may encounter difficulties, particulary if the robot is to function in a dynamic environment. For example, if a sensorimotor primitive for wall following was learned in an environment with few doors and completely flat walls, the robot may have difficulty when encountering many open doorways or walls with many variations (such as the walls in our hallway). To address these concerns, we developed a task-independent performance metric for use with sensorimotor primitives learned with the method as described in section 2.

The eigenvector method regenerates the entire training vector, v, which consists of both projected sensor and motor data. Generally the sensor data of the regeneration is ignored during runtime. However, this information can be compared to the actual sensor data, x, to develop a performance metric. When the system is performing well, there is very little difference between $sensor(v)$ and x, but as system performance degrades, the difference increases. Just as the sensors all have varying importance in the sensor/actuator mapping, each sensor will have varying weight in the performance evaluation. A large variation between the projected and the actual sensor values may indicate poor performance for some sensors but give no useful information for other sensors. To resolve this problem, the eigenvector can be used to determine the relative weighting of the sensors because an important sensor will have a relatively large absolute value in the corresponding component of the eigenvector.

The following error metric exploits these characteristics of the eigenvector learning algorithm.

$$E = |x - sensor(v)| \cdot |sensor(e)|$$

The error is found by matrix multiplying the absolute value of the difference of the projected sensors and the actual sensors with the sensor components of the eigenvector. Currently, the error metric has only been used with systems that are represented by one eigenvector. E equals zero when the projected sensors are the same as the runtime sensor values, indicating that the system is performing as well as it can. The error is in proportion to the size of E.

An error metric is useful for numerous tasks. Following, we give some example applications of the error metric. First we show how it is used to transition through a sequential task consisting of several primitives. Then we look at using the error metric to create an adaptive wall following robot that is able to select the best performing primitive from a collection of primitives.

4 Experiments

Several experiments were conducted using a Nomadic XR4000 holonomic mobile robot. The global rotation

of the robot was kept fixed throughout all experiments and just the x and y velocities of the robot were varied. The XR4000 was taught a collection of wall following primitives including left, right, front and back. Left wall-following is characterized by orienting the robot so that it moves along the wall at a fixed distance keeping the wall on its lefthand side.

For left wall following[1], the robot learned to vary the x velocity in order to keep the distance from the wall fixed, while the y velocity was kept constant. Wall following in this manner is a very straightforward linear problem and was easily learned and represented by one eigenvector. To ensure that the robot learned the correct primitive, it was trained on a wall that had several variations. Right, front, and back wall following were taught in a similar manner.

Four sonar measurements, representing the front, back, right, and left side of the robot, were used for each primitive. Training data was collected while a user demonstrated the task through teleoperation. For these experiments, key sonar were hand-selected but in a companion paper [2], we showed that these key sensors could be automatically selected. By analyzing training data for a consistent pattern, we found the sensors that were key to the execution of the primitive without prior knowledge. A consistent pattern in the training data signified the characteristic behavior which was subsequently used to filter the data. Filtering simplified the collection of a complete set of training data. Application of the work presented in [2] to these experiments is straight-forward.

4.1 Sequential Task

The power of sensorimotor primitives is that they can be sequenced to create more complex behavior. One of the difficulties of performing a sequenced task is determining when to transition from the primitive currently being used. We used the error metric as a transition cue for switching between the above described basic wall following primitives.

Figure 1 shows an experiment where the robot transitioned from right wall following to back wall following in order to negotiate the corner. The map was hand measured and the robot's path was recorded using onboard odometry. Figure 1.b shows the error metrics during this transition. As the robot moved along the right wall, the right wall error was low and the back wall error was high. Then as the robot reached the corner,

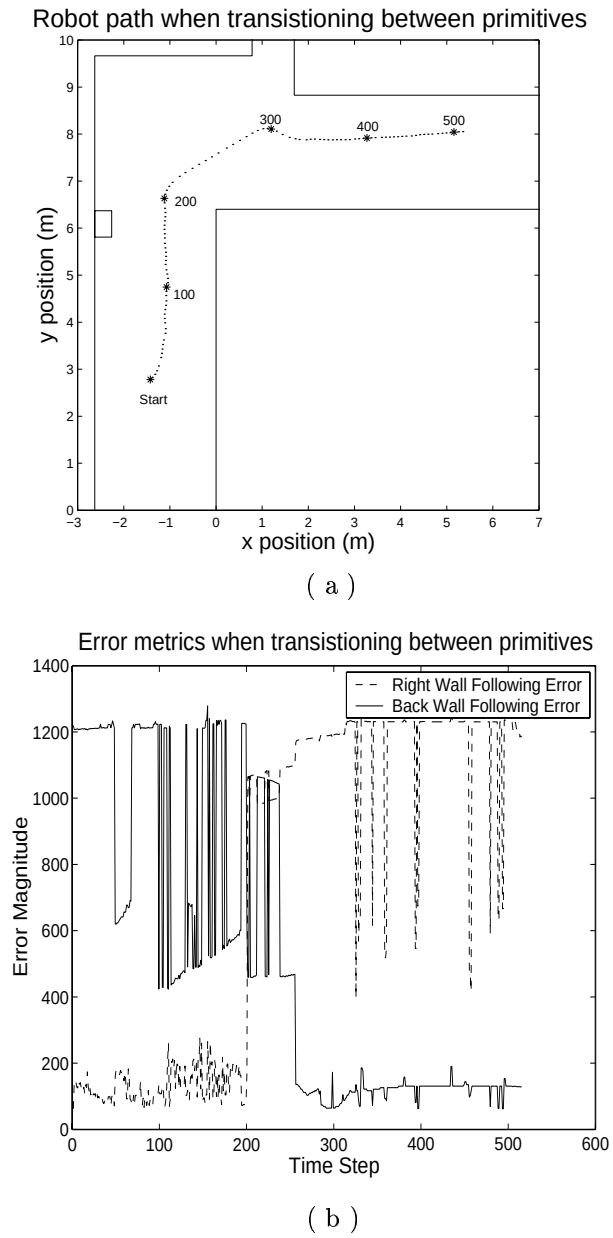


Figure 1: (a) *Map of robot's path as it transitioned from right wall following to back wall following* (b) *Error metrics for the transition*

the right wall error suddenly increased because of the large measurements from the right sonar. As the back sonar began to receive a shorter measurement, the back error became much smaller. The error metric provides a clear indication of when to switch primitives. Due to the large separation between the error metrics, sensor noise did not cause a false transition.

A similar experiment was conducted whereby the robot progressed through several states using the error

[1] Left wall following is relative to the robot's global axis. Since the robot is never rotated, left wall following is always the wall parallel to the y axis and on the negative side of the x axis. Likewise, back wall following is on the negative side of the y axis and parallel to the x axis.

metric to determine transition points. Figure 2 shows a map of the series of hallways in which the experiment was conducted. Each hallway encountered during operation was considered a new state and is designated as such in the figure. The robot successfully navigated through the building without any difficulty in recognizing the state transitions.

The programming to accomplish the sequenced task is straightforward and could easily be performed by a technician rather than an expert programmer. The programming mainly consists of: "Perform primitive A then perform primitive B." The error metric gives enough information for the system to transition between the two primitives.

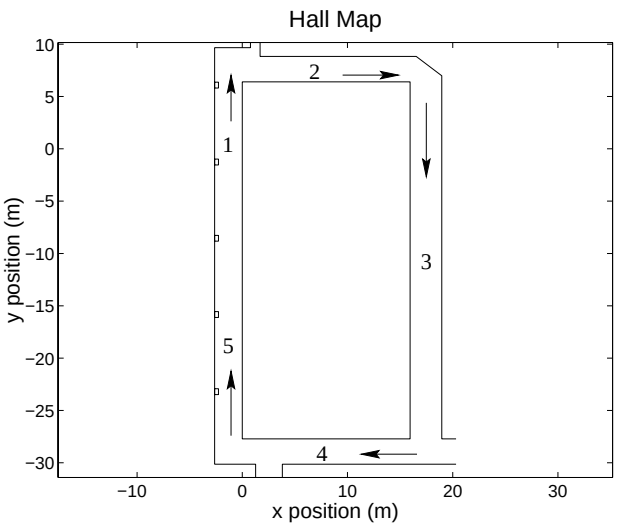
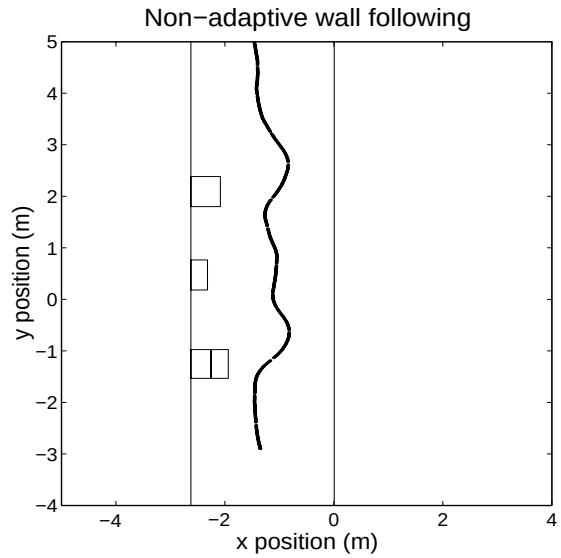


Figure 2: *Map of hall used for sequential task*

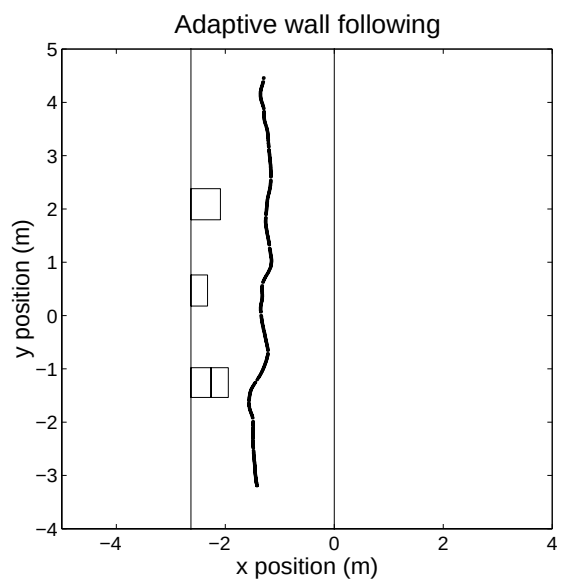
4.2 Adaptive Wall Following

Mobile robotic systems that operate in the real world must contend with dynamic environments. It may be possible to explicitely program when one skill has an advantage over another but it would be much simpler if the robot could automatically determine this. We have used the performance metric as a means of adaptation. The scenario presented here is a simplified problem, but it does capture the essence of an adaptive system.

Consider the problem of navigating a hallway. If items are placed along the wall or the hallway has a varied width (such as our hallway) the robot will have to zigzag to maintain the fixed distance it was trained to keep. Obviously, that would not be an efficient means of hallway navigation. Instead, the robot should adapt to the changing conditions so that it can maintain a straightline path. This is what our system achieved by



(a)



(b)

Figure 3: (a) *Wall following using one left wall following primitive that maintains a distance of 1 meter from the wall* (b) *Adaptive wall following using two primitives*

using the error measure as a means of primitive selection.

Figure 3 shows the two cases of wall following. In Figure 3a, a standard left wall following primitive is used whereby the robot learned to maintain a fixed distance from the wall. As expected, the path varies about the obstacles as the robot moves down the hallway. In Figure 3b, the robot uses the error metric to adaptively select from two left wall following primitives. The primitive used in Figure 3a is used as well as a primitive trained to keep a smaller fixed distance. As the robot moves, a recent history of error measures for both primitives is maintained. If one primitive outperforms the other for a fixed time, it takes over control of the robot. (Buffering in this manner prevents the system from oscillating between primitives if it encounters small environmental changes or momentary sensor noise.) As can be seen in Figure 3b, the path along the same hallway with the same obstacles is much smoother than the path in Figure 3a.

5 Conclusion

Performance evaluation can help refine sensorimotor primitives so that they can be used successfully in a variety of situations. The performance evaluation presented here can be used in conjunction with programming by demonstration using an eigenvector learning method. This error measure is not task dependent. We used this error measure as a cue to transition between states for a multi-primitive sequenced task. Additionally, we showed how it could be used to adaptively select the best primitive to optimize performance. Our examples are only a very small collection of the possible applications. Future work will involve exploring other applications of this method.

Generally the only difference when adaptively using several similar type primitives is the average vector described in (Section 2). For example, in Section 4.2, where two primitives are trained to maintain two different fixed distances from the wall, the only essential difference is the average component of the sonar measurement. Instead of switching between several primitives, it is seemingly possible to modify the primitive by modifying the average vector to attain the desired results. Future research plans include looking at ways of using the error metric to modify the average vector to increase adaptability of primitives.

Acknowledgements

This work sponsored by Air Force Research Lab under contract F30602-96-2-0240 and the NSF REU summer program at the Department of Electrical and Computer Engineering at the University of Minnesota.

References

[1] J. A. Hancock and C. E. Thorpe. ELVIS: Eigenvectors for land vehicle image system. Technical Report CMU-RI-TR-94-43, The Robotics Institue Carnegie Mellon University, 1994.

[2] A. Larson and R. M. Voyles. Automatic training data selection for sensorimotor primitives. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2001.

[3] K. F. MacDorman. Responding to affordances: Learning and projecting a sensorimotor mapping. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3253–3260, San Francisco, CA, April 2000.

[4] J. D. Morrow and P. K. Khosla. Sensorimotor primitives for robotic assembly skills. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1894–1899, May 1995.

[5] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1992.

[6] P. Pook and D. Ballard. Recognizing teleoperated manipulations. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 578–585, 1993.

[7] P. Rybski and R. M. Voyles. Interactive task training of a mobile robot through human gesture recognition. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 163–166, 1999.

[8] R. M. Voyles and J. D. Morrow. Gesture-based programming, Part 2: Primordial learning. *Intelligent Engineering Systems Through Artificial Neural Networks*, 6:305–310, 1996.