

An Architecture for Gesture-Based Control of Mobile Robots

Soshi Iba, J. Michael Vande Weghe, Christiaan J. J. Paredis, and Pradeep K. Khosla

The Robotics Institute
The Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Gestures provide a rich and intuitive form of interaction for controlling robots. This paper presents an approach for controlling a mobile robot with hand gestures. The system uses Hidden Markov Models (HMMs) to spot and recognize gestures captured with a data glove. To spot gestures from a sequence of hand positions that may include non-gestures, we have introduced a “wait state” in the HMM. The system is currently capable of spotting six gestures reliably. These gestures are mapped to robot commands under two different modes of operation: local and global control. In the local control mode, the gestures are interpreted in the robot’s local frame of reference, allowing the user to accelerate, decelerate, and turn. In the global control mode, the gestures are interpreted in the world frame, allowing the robot to move to the location at which the user is pointing.

1 Introduction

An important aspect of a successful robotic system is the Human-Machine Interaction. In the early years of robotics, teach pendants were the most common mode of interaction. As software capabilities improved, the ability to do off-line programming proved to be a significant step forward. Interfaces to manipulator systems made further progress with the introduction of user friendly programming paradigms for sensor-based manipulation [7]. The current state-of-the-art in manipulator interaction is based on an iconic programming [2] and/or programming by human demonstration paradigm [4, 13]. The idea is to transfer the burden of programming manipulator systems from robot experts to task experts. These task experts have extensive knowledge and experience with respect to the task, but may only have limited expertise in robotics. To enable these new users to in-

teract with the robot, the interface needs to be intuitive and have the ability to interpret the sometimes vague specifications of the user. An example of such a system is the gesture-based programming interface developed by Voyles and Khosla [13]. The robot system observes the operator unobtrusively while he/she is demonstrating the task. The observations can be based on vision, range sensing, data gloves, and tactile sensing. The most challenging aspect of such a task is to interpret the intent of the user, rather than simply mimicking her actions. In order to capture intent, Voyles and Khosla [14] proposed a novel “port-based agent” architecture. This builds on the notion of “port-based objects” introduced in [12] and uses a multi-agent framework to determine the user’s intent. Capturing the intent of the user is an important research issue for creating new programming and interaction paradigms for intelligent systems. Our research in gesture-based programming is a step towards this long-term goal.

As robots enter the human environment and come in contact with inexperienced users, they need to be able to interact with these users in a multi-modal fashion—keyboard and mouse are no longer acceptable as the only input modalities. In this paper, we investigate the possibility of hand gestures as a rich input modality for interacting with mobile robots. However, this is only a first step towards a comprehensive multi-modal human machine interface for interaction with multiple robots. Multi-robot systems require a much richer form of interaction to allow the user to specify concisely what the goals of the robot team should be, and to provide additional feedback during the execution with respect to grouping, formation, and collaboration.

Nakamura *et al.* [9] developed a human-to-multi-robot interface system, where interaction can occur either at the level of an individual robot or a group of robots. The interaction is performed through a mouse

and keyboard, and the robots are supervised by a CCD camera mounted on the ceiling. However, the interaction is not intuitive enough to be extended to non-experts. Compared to a mouse and keyboard, hand gestures have high redundancy to convey geometric and temporal data. They are rich in vocabulary while being intuitive to users. These features make hand gestures an attractive tool to interact with robots.

We have developed a gesture spotting and recognition algorithm based on a Hidden Markov Model (HMM). HMMs are commonly used in speech and handwriting recognition, and have recently been applied to gesture recognition [5, 11]. We purposely use the term 'spotting' in addition to 'recognition' to emphasize the importance of spotting a gesture from a sequence of data containing both gestures and non-gestures. It is essential for a gesture-based robot control system to avoid sending inadvertent control signals to the robot.

Related research in the field of gesture-based human-robot interaction includes work by Boehme *et al.* [1] and Waldherr *et al.* [15]. Boehme developed a system in which the robot reliably detects gestural cues given by the user. His work focuses on improving the robustness of the interface by first identifying potential users through a multiple queue approach. Waldherr's system detects motion gestures by combining neural network and template matching approaches. Both systems are implemented on-board the robot, and use a vision-based neural network approach to recognize the gestures. In our work, we use a data glove and electro-magnetic 6DOF position sensor, because they provide more reliable measurements of the position and joints. Due to the cables attached to the glove and the position sensor, the mobility of the user is constrained. However, we envision that the data glove's current limitation on mobility will be overcome by technological advances in wearable computing and personal localization systems.

2 System Description

Figure 1 illustrates the system setup. The system consists of a mobile robot, a CyberGlove, a Polhemus 6DOF position sensor, and a geolocation system that tracks the position and orientation of the mobile robot.

The CyberGlove and the Polhemus 6DOF position sensor are both connected to the gesture server to spot and interpret gestures. The user can either wave in a direction for the robot to move, or point at the desired destination in a 'point-and-go' fashion.

The system works in either a local control mode or a global control mode. Under the local control mode,

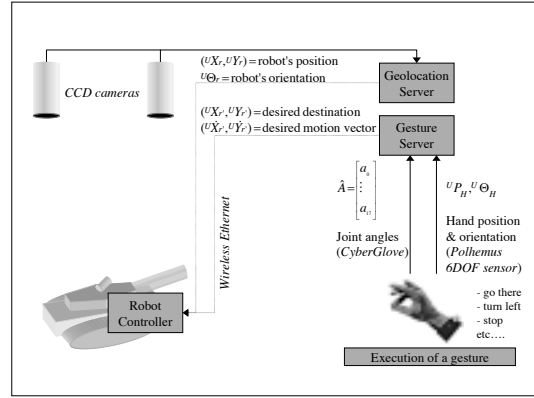


Figure 1: The system setup

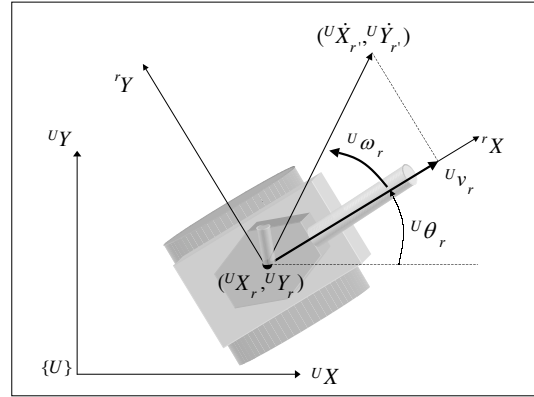


Figure 2: The conversion of velocity vectors (${}^U\dot{X}_{r,i}$, ${}^U\dot{Y}_{r,i}$) to linear and angular velocities (Uv_r , ${}^U\omega_r$)

the gesture server interprets gestures as if the user is in the robot's local coordinate frame. Under the global control mode, the user and the robot are both working in the universal coordinate frame. The gesture server spots the gesture from the data stream and the interpreter generates a desired velocity vector, (${}^U\dot{X}_{r,i}$, ${}^U\dot{Y}_{r,i}$), or a desired position, (${}^UX_{r,i}$, ${}^UY_{r,i}$). They are sent to the robot and converted to linear and angular velocities, (Uv_r , ${}^U\omega_r$), on-board. The process is shown in figure 2. When a desired position, (${}^UX_{r,i}$, ${}^UY_{r,i}$), is sent to the robot, a proportional control algorithm is used to servo to that position.

Figure 3 shows the mobile robot used in this experiment. It is equipped with 8 sonar sensors, 7 IR obstacle detectors, a black and white pan-tilt camera capable of transmitting a video signal via RF radio, stereo microphones, wireless Ethernet and an on-board PC104-based i486 PC running the Linux operating system. Both treads have wheel encoders and



Figure 3: The mobile robot used in the experiments

are independently controlled by a PID controller. The robot’s high-level control software can run on-board or off-board. For the gesture-based control experiment, a gesture spotter and a gesture interpreter are run off-board. The responsibility of the on-board PC is limited to executing motion and camera-control commands. The robot, the gesture spotter/interpreter, and the robot positioning system are all integrated within CyberRAVE, a multiple client-server architecture for distributed robots [3]. CyberRAVE provides low-level primitives and drivers to aid the users in their development of robot control programs.

A geolocation system provides the robot’s location and orientation, $({}^U X_r, {}^U Y_r, {}^U \theta_r)$, in the universal coordinates of the 3 by 6 meter workspace. A ceiling-mounted camera tracks color markers on the robot using a CognaChrome vision system. Through a calibration procedure, the image positions of the markers are mapped to the corresponding universal coordinates. The geolocation server is capable of sending position and orientation data at a rate of 10Hz to clients running the CyberRAVE interface.

The CyberGlove is a data glove that measures 18 joint angles of the fingers and wrist at 30Hz. To determine the wrist position and orientation, we use a Polhemus 6DOF positioning system that consists of the transmitter and the receiver pair. The receiver placed on the user’s hand detects the magnetic fields emitted by the fixed transmitter and inductively tracks the position and orientation.

3 Gesture Recognition

To take advantage of the temporal component of gestures, we chose to use an HMM-based recognizer. An HMM recognizer classifies time sequences of features. For each new data point in a continuous stream

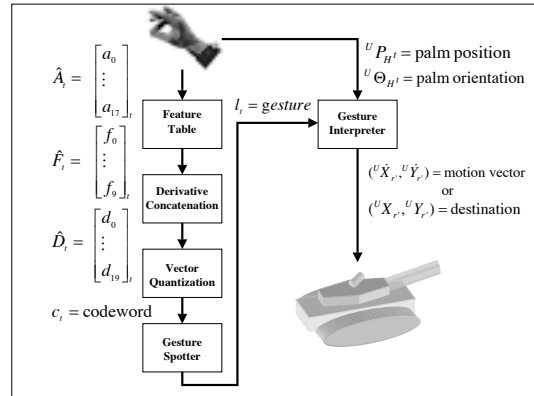


Figure 4: The data flow from the glove to the robot

of measurements, our HMM recognizer determines which gesture is currently being executed. In this section, we will present the process of data reduction and gesture spotting, as is illustrated in Figure 4. For readers unfamiliar with HMMs, we refer to Rabiner and Juang [10] for an excellent tutorial on evaluation, estimation, and decoding using HMMs as applied to the problem of speech recognition.

3.1 Data preprocessing

To improve the speed and performance of gesture classification, we pre-process the 18 joint angle measurements from the data glove in two stages. During the first stage, the 18-dimensional joint vector, \hat{A}_t , is reduced to a 10-dimensional feature vector, \hat{F}_t , through the linear combinations listed in Table 1. This feature vector is augmented with its first derivatives to produce a 20-dimensional column vector, \hat{D}_t :

$$\hat{D}_t = \begin{bmatrix} \hat{F}_t \\ (\frac{\hat{F}_t - \hat{F}_{t-1}}{\Delta t}) \end{bmatrix} \quad (1)$$

During the second stage, \hat{D}_t is reduced to a one dimensional codeword through \hat{D}_t vector quantization. The algorithm by Linde, Buzo and Gray [6] is used to generate a codebook that maps the 20-dimensional input vector to an integer codeword ranging from 0 to $N - 1$. We decided to use 32 codewords after considering the level of detail required for accurate recognition and after experimenting with a variety of values.

The codebook is trained off-line. We collected a representative sample of about 5000 measurements containing gestures and non-gestures, covering the entire space of hand movements. This subset of the entire 20-dimensional gesture space corresponds to hand

<i>Feature Description</i>	<i>Joint Angle Formula</i>
Thumb Curvature	$\angle(\text{thumb 1st} + \text{2nd})$
Index Finger Curvature	$\angle(\text{index 1st} + \text{2nd})$
Middle Finger Curvature	$\angle(\text{middle 1st} + \text{2nd})$
Ring Finger Curvature	$\angle(\text{ring 1st} + \text{2nd})$
Pinkie Curvature	$\angle(\text{pinkie 1st} + \text{2nd})$
Finger Spread	$\angle(\text{index} \leftrightarrow \text{middle} + \text{middle} \leftrightarrow \text{ring} + \text{ring} \leftrightarrow \text{pinkie})$
Thumb Angle	$\angle(\text{thumb} \leftrightarrow \text{index})$
Thumb Rotation	thumb rotation
Wrist Pitch	wrist pitch
Wrist Yaw	wrist yaw

Table 1: Definition of the feature vector.

positions and velocities that are physiologically feasible. The set $D_{0..5000}$ is then partitioned into 32 clusters. The algorithm by Linde, Buzo and Gray performs the error minimization clustering by repeatedly estimating the centroid of a cluster and splitting it into two. For each of the final 32 clusters, the associated centroid in the 20-dimensional feature space is recorded. At run time, a given feature vector is mapped to the codeword with the closest centroid (as measured in Euclidean distance). Consequently, each measurement of the hand position is reduced to a codeword, and a gesture becomes a time sequence of codewords.

3.2 Gesture spotting using an HMM

After preprocessing the data, the gesture spotter takes a sequence of codewords and determines which of the six gestures the user is performing, or “none of the above” if no gesture is being executed. The six gestures we chose to recognize consist of:

- OPENING: Moving from a closed fist to a flat open hand
- OPENED: Flat open hand
- CLOSING: Moving from a flat open hand to a closed fist
- POINTING: Moving from a flat open hand to index finger pointing, or from a closed fist to index finger pointing
- WAVING LEFT: Fingers extended, waving to the left, as if directing someone to the left
- WAVING RIGHT: Fingers extended, waving to the right

Being able to reject motions that do not correspond to any of these six gestures is very important. If every stream of motion data would be classified as one

of the six gestures listed above, almost any motion of the fingers would result in inadvertent robot actions. Including the classification “none of the above” allows the user to perform other tasks while wearing the data glove, without the HMM interpreting the hand motions as robot commands.

The algorithm described in this paper differs from the standard forward-backward technique described in Rabiner and Juang’s tutorial [10] in two important respects.

A first modification allows us to classify continuous streams of measurements. The correspondence of an HMM, λ , to an observation sequence $O = O_1, O_2, \dots, O_T$ can be quantified as $Pr(O | \lambda)$, which is the probability that the observation sequence is produced by the model. However, as the length of the sequence grows, $Pr(O_1, O_2, \dots, O_\infty | \lambda)$ decreases to zero. A solution to this problem is to limit the observation sequence to the n most recent observations, $O = O_{t-n}, \dots, O_t$. The length of this sequence needs to be determined by the user.

The second modification allows us to reject hand motions that do not match any of the six modeled gestures. In the standard HMM recognition algorithm, the gesture l with the largest confidence level $Pr(O | \lambda_l)$ is selected, where λ_l is the model corresponding to the gesture l . This means that one of six gestures will always be selected, unless a threshold is set to reject all six. However, a threshold that is large enough to reject all non-gestures, may also exclude gestures that are performed slightly differently from the training gestures. This is unacceptable, because different users tend to execute gestures differently.

To overcome these problems, we used the HMM gesture spotter illustrated in figure 5. It contains a “wait state” as the first node that transitions to all the gesture models and itself with equal probability. All these transition probabilities are all equal to make the transition independent of the observation.

When a new observation is applied to the gesture spotter, the probability of being in the state is updated for all existing states. In other words, we update all the $Pr(i_t = q_i | O, \lambda_l)$, the probability of being in state q_i at time t of the state sequence $I(= i_1 i_2 \dots i_t)$, given the observation sequence O and the model λ_l of gesture l . Next, the state probabilities are normalized so that they sum to one. If the observation sequence corresponds to any of the gestures, the probability of being in the final state of the model for this gesture will be higher than that for the other gestures. On the other hand, if the observation sequence does not belong to any gestures, the probability of being in the

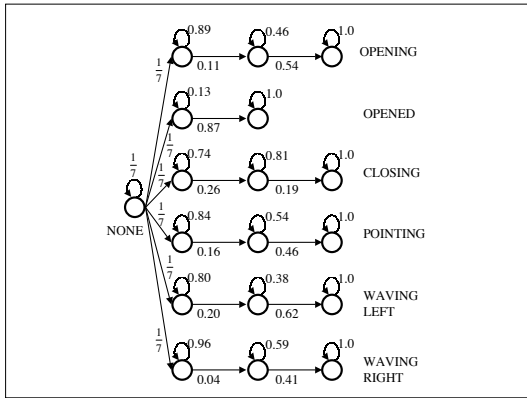


Figure 5: An HMM with a shared initial state

“wait state” will be the highest due to the normalization. If the recent sequence of observations represents a gesture, the earlier part of the sequence is trapped in the “wait state” while subsequent observations raise the correct gesture’s probability. Therefore, the gesture spotter selects the gesture that corresponds to the last state with the highest score, or no gesture if the “wait state” has the highest score.

3.3 Experimental verification of the HMM

Our implementation of the gesture spotter is based on public domain HMM C++ code written by Myers and Whitson [8]. We modified this code to calculate the probability for multiple models simultaneously, and to handle a continuous data stream.

Before an HMM gesture spotter is able to classify a sequence, it must be trained by the user. Training by multiple users is desirable to accommodate different ways in which users execute gestures. For example, one user may execute WAVING RIGHT much faster than another. For each of the six gestures, a 3-state HMM was trained separately using the Baum-Welch algorithm. Because of its static nature, the OPENED gesture has been modeled with only two states. Thirty sample sequences were used to train each gesture. Figure 6 shows the 3-state HMM used to model the POINTING gesture.

To verify the proposed HMM, the system was evaluated on a continuous stream of test data containing both gestures and non-gestures. To compare results, we compared the HMM spotter with basic left-right configurations that have variable windows of the n observations, as is illustrated in Figure 7. Unlike the HMM with “wait state”, this HMM does not have a “wait state”. HMMs from each gesture take a se-

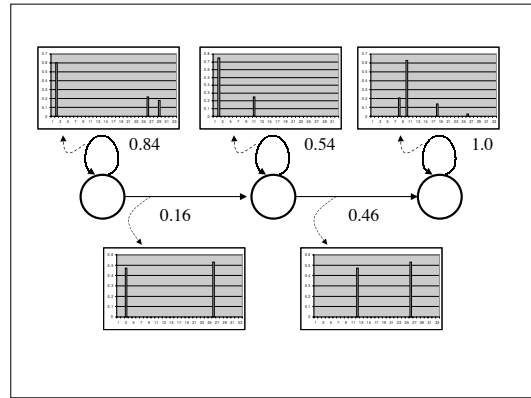


Figure 6: A sample 3-state HMM for the POINTING gesture, shown with transition probabilities and observation probabilities

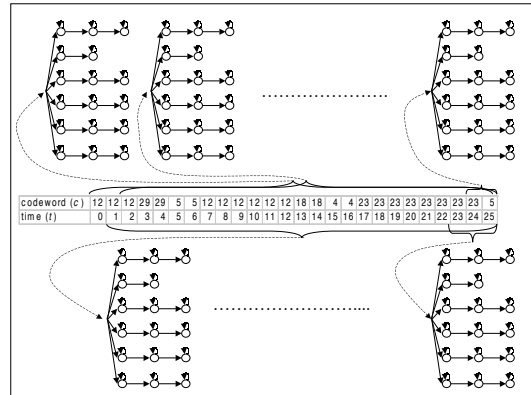


Figure 7: An HMM with a variable time window

quence of the n most recent data samples to classify a gesture. The system then evaluates $Pr(O_{t-n}..O_t | \lambda_l)$ for all n to find the gesture l that produces the maximum score. The system was set to reject a sequence as “none of the above” if $Pr(O_{t-n}..O_t | \lambda_l)$ for all gestures is less than 0.5.

On the test sequence of codewords mixed with gestures and non-gestures, the HMM with “wait state” recognized gestures with 96% accuracy (48 out of 50 gesture sequences recognized correctly), whereas the HMM with variable window recognized all gestures. However, the false-positive rate for the HMM with variable window was 20.2 observations per 1000 samples, whereas the HMM with “wait state” had a false-positive rate of only 1.6 observations per 1000. It is important to keep in mind that this number does not apply directly to the real data, since there may be other factors contributing to the errors, such as vector

quantization.

4 Gesture interpretation

Once a particular gesture has been recognized, it needs to be mapped to a corresponding robot action. We have implemented two different modes of interaction with the robot: local control and global control. In the local interaction mode, the user’s gestures are interpreted in the robot’s local frame of reference. For instance, pointing makes the robot move forward regardless of whether the robot is facing towards or away from the user. In the global interaction mode, the position of the hand is measured in the universal coordinate frame allowing us to interpret the gestures with respect to the global frame of reference.

4.1 Local robot control

In the local robot control mode, the six gestures have the following meaning:

- CLOSING: decelerates and eventually stops the robot
- OPENING, OPENED: maintains the current state of the robot
- POINTING: accelerates the robot
- WAVING LEFT/RIGHT: increase the rotational velocity to move left/right

The purpose of the local control mode is to allow the user to control the robot in a remote operation scenario. The user watches the video image transmitted from the robot’s camera and controls the robot as if he or she were sitting on it.

A CLOSING gesture is implemented by decreasing both linear and angular velocity of the robot. Continuous execution of a CLOSING gesture eventually stops the robot’s movement. The choice to make the CLOSING hand a stopping gesture was based on people’s natural reaction and the fact that it has the best accuracy in gesture spotting (a CLOSING gesture has 100% accuracy in our recognition system). A POINTING gesture accelerates the robot by increasing $^U v_r$ by a fixed value. Successive pointing gestures cause the robot to accelerate in the forward direction.

The WAVING LEFT and WAVING RIGHT gestures are implemented as simple discrete steering commands. When the gesture spotter detects WAVING LEFT (or RIGHT), the gesture server tells the robot controller to increase (or decrease) $^U \omega_r$ by a fixed value. Successive waving gestures cause the robot to turn faster.

4.2 Global robot control

In the global robot control mode, the six gestures have the following meanings:

- CLOSING: decelerates and eventually stops the robot
- OPENING, OPENED: maintains the current state of the robot
- POINTING: “go there”
- WAVING LEFT/RIGHT: directs the robot towards the direction in which the hand is waving.

The global robot control mode provides close interaction between the user and the robot. The robot is in sight of the user, and the user can point his or her finger to specify the destination for the robot or give it an additional bias by waving hands to guide the robot.

The CLOSING gesture is implemented in a manner similar to the local robot control mode. It cancels the robot’s destination if one exists, and brings $(^U \dot{X}_r, ^U \dot{Y}_r)$ to zero to stop the robot.

The POINTING gesture defines the destination for the robot as the location where the straight line extending the index finger intersects with the floor. The robot controller generates the motion vector, $(^U \dot{X}_r, ^U \dot{Y}_r)$, based on the difference between its current position and the destination.

The interpretation of the WAVING LEFT and WAVING RIGHT gestures depends also on the motion vector extracted from the Polhemus 6DOF positioning system placed on the hand. When a WAVING LEFT (or RIGHT) gesture is recognized, the robot is commanded in the direction perpendicular to the palm of the hand when the hand is waving at its maximum speed. When the robot encounters an obstacle while heading towards its destination, the user can wave in a particular direction to dodge the obstacle.

5 Summary and Future Work

In this paper, we have introduced and demonstrated an architecture for gesture based control of mobile robots. An HMM-based gesture spotter recognizes six dynamic hand-gestures from a continuous data sequence from a data glove. The HMM differs from most traditional HMM recognizers because of the inclusion of a “wait state” which allows it to effectively reject hand motions that do not correspond to any of the modeled gestures. This is very important for avoiding inadvertent robot commands while wearing the data glove. In our experiments, we verified that the HMM with a “wait state” produced an order

of magnitude fewer false positives than a traditional HMM recognizer.

We envision gesture-based control to be of particular value for interacting with teams of robots. Controlling individual robots can still be achieved using a mouse or a joystick; there is a one-to-one mapping between the joystick position and the robot velocity. Multi-robot systems require a much richer form of interaction. Instead of providing low-level control for individual robots, the user needs to provide high-level directives for the team as a whole. We will address control of multi-robot systems in our future research.

Acknowledgments

This research is funded in part by the by the Distributed Robotics program of DARPA/ETO under contract DABT63-97-1-0003, and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

References

- [1] H.-J. Boehme, A. Brakensiek, U.-D. Braumann, M. Krabbes, and H.-M. Gross, "Neural Architecture for Gesture-Based Human-Machine-Interaction," In *Proc. of the Gesture and Sign Language in Human-Computer Interaction. International Gesture Workshop*, pp. 219-232, Bielefeld, Germany, Sept. 1997.
- [2] M. W. Gertz, P. K. Khosla, "Onika: A Multilevel Human-Machine Interface for Real-Time Sensor-Based Systems," In *Proc. of the 4th Intl Conf. and Expo on Engineering, Construction and Operations in Space*, Albuquerque NM., Feb 1994.
- [3] K. Dixon, J. Dolan, W. Huang, C. Paredis, and P. Khosla, "RAVE: A Real and Virtual Environment for Multiple Mobile Robot Systems," In *Proc. Intl. Conf. on Intelligent Robots and Systems (IROS'99)*, Korea, 1999.
- [4] S. B. Kang and K. Ikeuchi, "Robot task programming by human demonstration," In *Proc. Image Understanding Workshop*, pp. 1125-1134, Monterey, CA, November 1994.
- [5] C. Lee and Y. Xu, "Online, Interactive Learning of Gestures for Human/Robot Interfaces," In *Proc. Intl. Conf. on Robotics and Automation*, vol. 4, pp. 2982-2987, 1996.
- [6] Y. Linde, A. Buzo, R.M. Grey, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. com-28, No. 1, pp. 84-95, January 1980.
- [7] J. D. Morrow and P. K. Khosla, "Manipulation Task Primitives for Composing Robot Skills," In *Proc. of the IEEE Intl. Conference on Robotics and Automation*, Albuquerque, NM, v. 4, pp. 3354-3359, 1997.
- [8] R. Myers and J. Whitson, "Hidden Markov Model for automatic speech recognition," <ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/recognition/>, University of California at Irvine, 1994.
- [9] A. Nakamura, S. Kakita, T. Arai, J. Beltran-Escávy, J. Ota, "Multiple Mobile Robot Operation by Human," In *Proc. Intl. Conf. on Robotics and Automation*, Leuven, Belgium, v. 4, pp. 2852-2857, May 1998.
- [10] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4-16, 1986.
- [11] G. Rigoll, A. Kosmala, and S. Eickeler "High Performance Real-Time Gesture Recognition Using Hidden Markov Models," In *Proc. Gesture and Sign Language in Human-Computer Interaction. International Gesture Workshop*, pp. 69-80, Bielefeld, Germany, Sept. 1997.
- [12] D. B. Stewart, R.A. Volpe, and P. K. Khosla, "Design of dynamically reconfigurable real-time software using port-based objects," *IEEE Trans. on Software Engineering*, 23(12), pp. 759-776, December 1997.
- [13] R.M. Voyles, A. Agah, P.K. Khosla and G.A. Bekey, "Tropism-Based Cognition for the Interpretation of Context-Dependent Gestures," In *Proc. Intl. Conf. on Robotics and Automation*, Albuquerque, NM, v. 4, pp. 3481-3486, Apr. 1997.
- [14] R.M. Voyles and P.K. Khosla, "Gesture-Based Programming, Part 1: A Multi-Agent Approach," in *Intelligent Engineering Systems through Artificial Neural Networks, Volume 6; Smart Engineering Systems: Neural Networks, Fuzzy Logic and Evolutionary Programming*, ASME Press, 1996.
- [15] S. Waldherr, S. Thrun, and R. Romero, "A neural-network based approach for recognition of pose and motion gestures on a mobile robot," In *Proc. of the 5th Brazilian Symposium on Neural Networks*, pp. 79-84, Belo Horizonte, Brazil, Dec. 1998.