Interactive Multi-Modal Robot Programming

Soshi Iba[1], Christiaan J.J. Paredis[3], and Pradeep K. Khosla[1, 2]

[1)] *The Robotics Institute, Carnegie Mellon University*

[2)] *Electrical and Computer Engineering, Carnegie Mellon University*

Pittsburgh, Pennsylvania 15213-3890

[3)] *Systems Realization Laboratory*

G. W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology

Atlanta, Georgia 30332-0405

iba@ri.cmu.edu, chris.paredis@me.gatech.edu, pkk@ece.cmu.edu

December 17, 2003

Abstract

As robots enter the human environment and come in contact with inexperienced users, they need to be able to interact with users in a multi-modal fashion—keyboard and mouse are no longer acceptable as the only input modalities. This paper introduces a novel approach for programming robots interactively through a multi-modal interface. The key characteristic of this approach is that the user can provide feedback interactively at any time—during both the programming and the execution phase. The framework takes a three-step approach to the problem: multi-modal recognition, intention interpretation, and prioritized task execution. The multi-modal recognition module translates hand gestures and spontaneous speech into a structured symbolic data stream without abstracting away the user's intent. The intention interpretation module selects the appropriate primitives to generate a task based on the user's input, the system's current state, and robot sensor data. Finally, the prioritized task execution module selects and executes skill primitives based on the system's current state, sensor inputs, and prior tasks. The framework is demonstrated by interactively controlling and programming a vacuum-cleaning robot. The demonstrations are used to exemplify the interactive programming and the plan recognition aspect of the research.

Nomenclature

Φ^p = p-th robot program described in sequence of discrete actions

N_p = $\|\Phi^p\|$ = number of actions in Φ^p

T_p = last time index in Φ^p

$\phi^{p,a}$ = a-th action in Φ^p

P = number of programs in the library

o_t^p = robot position and orientation at time t for $\Phi^p = (x_t, y_t, \theta_t)^p$ [m, m, rad]

O^p = set of all o_t^p during the execution of $\Phi^p = \{o_1^p, o_2^p \ldots o_{T_p}^p\} = \{O^{p,0}, O^{p,1}, \cdots, O^{p,N_p}\}$

τ^p = set of all time $t = \{1 \ldots T_p\}$ during the execution of Φ^p [sec]

$\tau^{p,a}$ = set of all time t during the execution of $\phi^{p,a}$, before $\phi^{p,a+1} = (\tau^{p,a_{\text{ON}}}, \tau^{p,a_{\text{OFF}}})$ [sec]

$O^{p,a}$ = set of all o_t^p during the execution of $\phi^{p,a}$ and before $\phi^{p,a+1} = (O^{p,a_{\text{ON}}}, O^{p,a_{\text{OFF}}})$

$O^{p,a_{\text{ON}}}$ = set of all o_t^p during the execution of $\phi^{p,a} = O^p_{\tau^{p,a_{\text{ON}}}}$

$O^{p,a_{\text{OFF}}}$ = set of all o_t^p after the execution of $\phi^{p,a}$ and before $\phi^{p,a+1} = O^p_{\tau^{p,a_{\text{OFF}}}}$

λ^p = Continuous Density HMM description of Φ^p

a_{ij} = state transition probability from state i to state j

b_{ij} = observation probability from state i to state j = $N(\bar{\mu}_{ij}, \Sigma_{ij}) \bullet WN(\mu_{\theta ij}, S_{\theta ij})$

$\bar{\mu}_{ij}$ = mean robot position (x,y) from state i to state j [m, m]

Σ_{ij} = covariance matrix of robot position from state i to state j

$\mu_{\theta ij}$ = mean robot orientation θ from state i to state j [rad]

$S_{\theta ij}$ = angular variance (between 0 and 1) of robot orientation from state i to state j

ψ_{ij} = value of b_{ij} at a distance of 3σ from the mean

$r_{ij}(t)$ = Mahalanobis distance between o_t and $\{\bar{\mu}_{ij}, \mu_{\theta ij}\}$

$\delta_t(i)$ = token value of state i at time t

P_i, S_i, T_i = i-th robot position (x,y) in the set of P, S, or T [m, m]

1. Introduction

An important aspect of a successful robotic system is the human-machine interaction. As robots enter the human environment and come in contact with inexperienced users, they need to be able to interact with users in a multi-modal fashion—keyboard and mouse are no longer acceptable as the only input modalities. Humans should be able to communicate with robots using methods as similar as possible to the concise, rich, and diverse means they use to communicate with one another. This paper introduces a novel approach for programming a robot interactively through a multi-modal interface. The key elements behind this novice-friendly system are intuitive interfaces based on speech and hand gesture recognition, and interaction capabilities that allow the user to take over the control of the robot at any given time. Such interaction capabilities give a sense of assurance to the users and help them in dealing with loosely calibrated position sensors by including a human in the control loop. Users are able to initiate a programming phase through voice commands and move the robot to any desired location. The sequence of commands turns into a sequential robot program. The user can then initiate an execution phase and execute the program while taking control at any given time. The multi-modal human-robot interaction described above is similar to the WYSIWYG (*what you see is what you get*) interface introduced in the human-computer interaction domain. Instead of off-line robot programming, this on-line robot programming method lets the user see what to expect from the program execution.

Industrial Robotics: In the early years of robotics, a hand-held control box called a teach pendant connected to a robot controller used to direct and program a robot was 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 [27]. The current state-of-the-art in manipulator interaction is based on iconic programming [9] and/or programming by human demonstration [13,42]. The goal of these paradigms is to translate the burden of programming manipulator systems from robot experts to task experts. Task experts have extensive knowledge and experience with respect to the task, but may only have limited

expertise in robotics. To enable these novice users to interact with the robot, the interface needs to be intuitive and have the ability to interpret the vague specifications of the user. An example of such a system is the gesture-based programming interface developed by Voyles and Khosla [42]. The robot system observes the operator unobtrusively while she is demonstrating the task. The observations can be based on vision, range sensing, data gloves, or tactile sensing.

Personal Robotics: Due to the growing field of personal robotics, we come in contact with more robots than ever before. Examples of such robots include pet robots [8], tour-guiding robots [40], entertainment robots [14], intelligent wheelchairs [21,25], and mobile vacuuming robots [28]. Traditionally, mobile robots are controlled via a joystick or mouse, but increasingly, voice or gestures are included as input modalities [5]. In this paper, we explore the task of interactively controlling and programming a vacuum-cleaning robot called Cye [4]. This task requires both interactive multi-modal control and a certain degree of autonomy. To accommodate novice users, the programming framework is based on multi-modal interaction (hand gestures and voice commands) and encompasses preemptive interaction during both programming and execution.

Multi-modal Interface: From the perspective of multi-modal interfaces, (*e.g.* gestures, speech) the interaction between the user and the robot systems has many advantages over conventional interaction modes, such as teach-pendants or joysticks. Hand gestures have an advantage in specifying geometric objects and spatial (three-dimensional) data, and are more intuitive for conveying information to robots that exist in the three-dimensional world [34,37]. The advantage is even more obvious when interacting with a team of robots, where complicated maneuvers and grouping commands can be executed by gesturing a set of points, a region of interest, or a group formation [33]. Hand gestures are convenient for specifying parametric and 3D information, but not for symbolic gestures. For symbolic information and commands, speech input is a natural choice. In comparison to the GUI used in personal computers, hand gesture can be a superset of a mouse, and speech can be a superset of a keyboard.

Intention Interpretation: The most challenging aspect of interactive robot programming is to interpret the intent of the users, rather than simply mimic their actions. Intent is the purpose or goal the user has in mind. User input can be vague, inaccurate,

and often contradicting. An intention aware system can be used to reduce unnecessary and often redundant instructions by being aware of what the user really wants. Intention interpretation can be thought of as a search for the mapping from the user input and robot sensory data to the correct set of robot actions. To accomplish such interpretation, the user needs an intuitive mode of interaction with the robot, while letting the system collect additional data leading to the correct intention interpretations.

The term intent is often loosely defined since it is very task dependent. In our framework, intention refers to a set of goal-directed robot actions resulting in a sequential robot program that the user would like to execute or modify, and the system needs to determine from inputs given by the user if such a robot program exists in the system's database. In other words, the user's intent is captured in the form of a sequential robot program, and the flexibility given to the user through real-time interaction and the framework's intuitive interface allows the captured intent to be closer to the user's true intent. Previous work on intention-aware systems such as [2,42] lacks this flexibility, and our system is more robust by being aware of a user's intent and incorporating real-time alterations based on this information.

Task level programming: The multi-modal interactive programming framework has several distinct advantages over conventional methods. From the robot programming perspective, on-line interaction adds a new flavor to the robot programming problem.

- It enables novice users to program robots,
- It enables interactive composition of primitives to create robot programs,
- It enables task model adaptation through continuous interaction.

To some degree, other paradigms such as iconic programming [9], and programming by demonstration [13] succeed in shifting the burden of robot programming from robot experts to task experts. However, due to the current lack of understanding of intention interpretation and of the robotic task itself, such off-line programming methods are very fragile. The task expert may demonstrate the task to the robot, but the task expert has no idea how the robot has interpreted his skill, or whether the robot has a sufficient set of actions to perform the demonstrated task. In contrast, our framework allows the task expert to "coach" the robot and to make adjustments on-line as it performs the new task.

2. Related Work

The area of human-robot interaction is a rich and diverse field of study. In order to understand the work of controlling and programming robots through a multi-modal interface, this section is divided into two subsections: multi-modal robot control, and robot programming.

2.1. Multi-Modal Robot Control

The area of robot control refers to the problem of efficiently conveying control signals to the robot system. Every robot system must have a device through which its user can control the behavior of the robot. The control signal can be in various forms ranging from low-level joint motor torque to high-level symbolic skill representations. For both mobile robots and industrial manipulators, the basic level of control is in the joint space, where the user input often comes from a teach-pendant or a joystick. At the higher level of abstraction, the control specifications are symbolic and come from either a graphical user interface or a natural user interface such as eye gaze tracking, finger pointing, or natural language interpretation.

Several researches have implemented a variety of natural interface to control mobile robots. The *GestureDriver* and *HapticDriver* systems by Fong [6] provide a teleoperation interface through symbolic hand gestures and force feedback through a haptic device. Other mobile robot interactions systems are capable of receiving symbolic gesture commands through an on-board camera [5,19,44]. Kuno et al. [21] have developed a wheelchair robot controlled by detecting hand gestures with a camera. This system is capable of dealing with unknown gestures by considering all periodic hand motions as potential gestures. Another example is Matsumoto's wheelchair robot [26], which can detect the user's gaze and facial direction to navigate.

To move a step closer to the human-human interaction, researchers are currently exploring multi-modal interaction scenarios. The advantage of working with multi-modal input mainly lies in its redundancy. For example, the system developed by Perzanowski et al. [34] combines natural language and hand gestures to interpret both complete and fragmental commands. The multi-modal interface system by Ghidary et al. [10] makes use of speech, posture, and object recognition to navigate a mobile robot to an object of

interest. Human-robot interaction can become more intuitive as the level of flexibility in the human interface increases. However, to achieve a higher level of human-robot interaction, the human interface and robot programming modules must work together.

A multi-modal interface combines multiple input modalities such as natural speech, pen-based input, hand gestures, facial gestures, eye gaze, body language, or tactile input. In the past, before robust multi-modal approaches were available, skeptics believed that a multi-modal interface incorporating two error-prone recognition technologies would compound errors and yield even greater unreliability. However, recent data shows that fusing two or more information sources can effectively reduce recognition uncertainty, thereby improving robustness [32]. The multi-modal mobile robot interface by Perzanowski et al. [34] is an example of a successful multi-modal interface system.

Hand gesture recognition is a popular field due to its broad applicability. Many successful gesture recognition methods are derived from algorithms in natural language recognition. They are roughly divided into three approaches: template-based, stochastic, and neural net based approaches. Nishimura and Oka [29] used template based continuous dynamic time warping (DTW) for spotting continuous visual gestures. The mobile robot interaction system by Kuno et al. [21] also used a gesture-spotting strategy based on DTW. Starner [39] applied Hidden Markov Models (HMM; often used to model doubly stochastic processes) to visual hand recognition of dynamic American Sign Language (ASL). Lee and Xu [22] used a similar HMM based method to recognize static ASL alphabets with a data glove as an input device. Kortenkamp et al. [19] developed a model-based method which models different parts of the body as a set of proximity spaces and defines pose gestures by examining the angles between the links that connect these proximity spaces. Waldherr et al. [44] combined a neural net approach for static pose gestures with a temporal template matching approach for motion gestures. They all differ in their assumptions, implementations (vision vs. magnetic spatial sensor, controlled lighting/background condition vs. mobile robot's on-board camera), and capabilities (pose vs. motion gesture, recognition rate), and it is important to keep in mind that their advantages and disadvantages are task dependent.

2.2. Robot Programming

In addition to using gesture-based interaction for direct control of robots, it can also be used for robot programming. Position and path-based applications such as arc welding and machine loading typically employ walk-through or lead-through teaching [41]. For walk-through teaching the user specifies intermediate points with a teach pendant. For lead-through teaching, the user performs the required motions manually while holding some device (the manipulator itself, or a replica) to record the path. While these forms of teaching are useful for non-contact applications; other methods are needed for applications that involve contact. Kang and Ikeuchi's [15] learning-from-observation system models human behavior as transitions of contact states, by observing a human demonstration. The system is able to model high-level task specifications but not the sensor feedback during contact. Voyles et al. [43] proposed a gesture-based programming paradigm where the system is assumed to have a set of basic skills (also referred to as *a priori* control policies [18], or sensori-motor primitives [27]) from which the system can compose programs. Human demonstration is observed through gesture recognition and interpretation agents, and the correct skills are selected based on the votes from the agents. A similar skill-based approach is used in the telerobotics system by Onda et al. [31] that combines geometric modeling, teaching by demonstration in a virtual environment, and execution based on manipulation skills.

To achieve robot interaction at elevated conceptual levels, robot programs can be composed from primitive behaviors. Such composition of skills can either be prepared in advance or learned from observation. Asada's human-robot interaction system uses Petri-nets to model the interaction between the robot and the human, but a plan has to be prepared in advance by a programmer [24]. Kimura and Ikeuchi [17] model human-robot cooperation tasks by observing both parties and placing pre- and post-conditions into a stack to compose a program. For its humanoid application, Kawamura's DBAM architecture [16] captures similar pre- and post- conditions into a look-up table.

3. Framework

The programming approach introduced in this paper offers an intuitive interface for the user and the ability to provide interactive feedback to coach the robot throughout

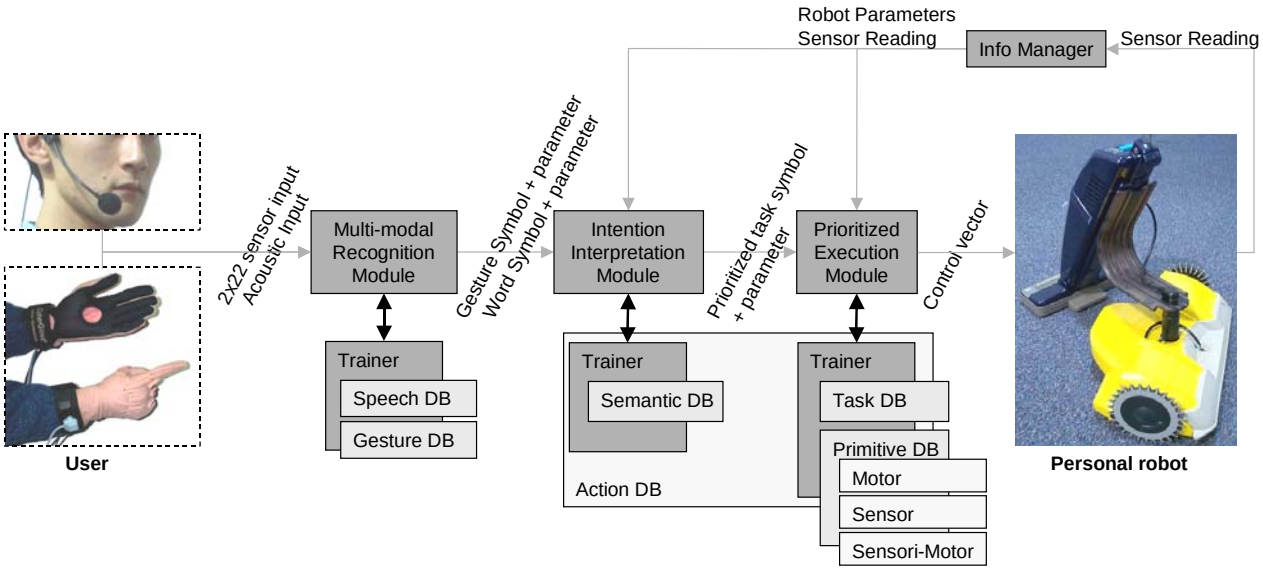


Figure 1: Framework for a Multi-Modal Interface

the programming process. The approach addresses shortcomings apparent in previous approaches, which are an unfriendly user interface preventing a novice user from using a service robot, and an inability to teach and program a robot on-the-fly. As input modalities, we support hand gestures and spontaneous speech. We selected hand gestures as a modality to convey parametric information such as speed, angles or positions, and spontaneous speech is selected as a modality to convey symbolic information such as names, confirmations, or program statements. The selection is made based on intuitiveness of the modality.

The framework is composed of three functional modules as illustrated in Figure 1. The first module (multi-modal recognition) translates hand gestures and spontaneous speech into a structured symbolic data stream without abstracting away the user's intent. The second module (intention interpretation) selects the appropriate set of primitives based on the user input, current state, and robot sensor data. Finally, the third module (prioritized task execution) selects and executes primitives based on the current state, sensor inputs, and the task given by the previous step. Each module includes two modes of operation: a learning and an execution mode. Depending on the mode of operation, the overall system can provide interactive robot control, adjustment of primitives, or composition of robot programs.

There are three main reasons for implementing the system in a modular fashion as described in Figure 1. First, the implementation follows a functional decomposition of the problem: recognition, interpretation, and execution. Second, in a modular architecture, one can easily replace the implementations of individual modules. For example, if we were to program an industrial manipulator instead of a vacuum cleaning mobile robot, the task execution module can be replaced by another implementation. Finally, because the first and last module can be implemented as slight modification of existing software and hardware products, a modular implementation allows us to work independently on the intention interpretation module, which is the main focus of this research.

3.1. Multi-Modal Recognition Module

The function of the multi-modal recognition module (the first block in Figure 1) is to translate hand gestures and spontaneous speech into a structured symbolic data stream without abstracting away the user's intent. The symbols could be gestures, words, or both. Abstraction of intent can be avoided by ensuring that the robot can cover the entire configuration space by using the multi-modal interface, since intention is defined as a set of goal-directed robot actions. We consider two sub-functions. First, the module needs to translate incoming audio and gesture signals into a structured stream of word and gesture unit symbols with appropriate parameters. Second, the module needs to be able to adapt to new users by reinforcing recognition models using new incoming data during recognition.

The recognition module generates a parameterized output stream. Examples of such parameters are the direction and velocity of the hand for a *waiving* gesture, or the designated x-y coordinates on the floor for a *pointing* gesture. The types of input modalities discussed throughout the paper are human voice and hand gestures parameterized by two 22-sensor CyberGlove. Other modalities can replace or be added to the current recognition module.

The second function of the module is to adapt to the data from new users by reinforcing symbols during recognition. Online adaptation of the recognition model to the data from new users contributes to a better recognition rate than that achievable without adaptation. The multi-modal recognition module is implemented using a Hidden Markov

Model [35], a stochastic method in which on-line model adaptation and reinforcement are very common. The Maximum Likelihood Linear Regression technique in the Hidden Markov Model Toolkit [47] is used to estimate a set of linear transformations for the mean and variance parameters of a Gaussian mixture HMM system that reduces the mismatch between the current model set and the adaptation data. The technique is used in both supervised and unsupervised mode. The supervised mode uses adaptation data of the new user from a known gesture sequence transcript. The unsupervised mode uses an estimated transcription based on the recognition result to adapt model parameters.

In our implementation, spontaneous speech is translated into words using SPHINX-II, an off-the-shelf speech recognition package [11]. For hand gestures, we implemented a word spotting technique using the Hidden Markov Model Toolkit. The current system works only with a basic set of words and gestures and does not include interactive learning of new gestures. Table 1 lists some of the initial candidate gestures and words that such a basic vocabulary could include. For example, the user is able to point at a certain position on the floor using a hand gesture coupled with the "Go There" command to the "Yellow Robot" via voice.

In the future, we plan to rely on cross-modal analysis to implement on-line learning. There are already attempts to automatically discover new gestures [45] from single-modal data; however, it is easier to rely on a redundant input mode to manage the learning process. For instance, speech could be used to signal the beginning and end of the learning process for gestures, and vice versa.

3.2. Intention interpretation Module

The intention interpretation module (The second block in Figure 1) has three

Input	*Candidate Symbols*
One-Handed Gestures	Point, Waive, Open, Grasp, Turn, Power Grasp, Precision Grasp
Two-Handed Gestures	Relative Position Specification (*e.g.* two Points)
Speech Vocabulary	Go, Move, Goto, Stop, Turn, Forward, Backward, Right, Left Deictics (This, That, There) Attributes (Yellow, Green), Names (Robot, Cye) Numbers (One, Two, Three, etc.) Sweep, Vacuum, On, Off, Program, Execute, Complete

Table 1: Initial Gesture and Speech Vocabularies

functions. The first is to recognize and select the appropriate task based on the current context. The second is to attach priorities to the task to handle multiple task requests. The third is to adapt the task representation used for task recognition and selection to the most current observation.

The problem of intention interpretation can be considered as a mapping problem from the stream of user inputs, the current state of the system, and the robot sensor data, to the correct robot task. The user input is an incoming stream of structured symbolic data (with parameters) from the multi-modal recognition module. The robot sensor data is an abstracted version of the robot's sensor stream. For a mobile robot, the robot sensor data could include range sensor data, distance to the closest obstacle, the robot's global position and current velocity. For manipulators, the robot sensor data includes the end-effector's position and velocity in the joint space or Cartesian space, and contact data if force, torque, or tactile sensors are available.

The output of the intention interpretation module is a task symbol representing a configuration of robot primitives. The usage and detailed definition of the terms *primitive* and *task* are discussed in the next section. In short, a primitive is an encapsulation of a low level robot behavior; that is, a policy $\pi(\boldsymbol{x},t,\alpha)=\boldsymbol{u}$ that maps the state $\boldsymbol{x}$ of a system and its environment into an appropriate action $\boldsymbol{u}$ at a particular time t, with additional parameters α. The task is a robot program composed of various primitives. The semantics database (Table 2) is implemented as a look-up table of candidate task symbols and their priorities from input symbols from the multi-modal recognition module. Initially, the semantics database contains tasks that are composed of single primitives. The primitives (Table 3) in our vacuum-cleaning robot scenario are single-purpose controllers. Priorities in the semantics database are assigned in a way such that critical and smaller tasks receive higher priorities. For example, critical tasks such as *Stop*() receive highest priority, single primitive tasks such as *Goto*(*P*), *Move*(*v*), etc. receive medium priorities, and finally tasks composed of multiple primitives and those associated with program composition receive low priorities. Tasks with identical priorities are arbitrated on a first-come first-serve basis (Section 3.3).

Primitive	*Parameter*	*Action*
Goto	*Position P*	Move to the position, w/path-planning
Vacuum	*On/Off*	Toggle the state of the vacuum cleaner
AreaCoverage	Rectangular Area (P_0, P_1)	Traverse the area specified
GoHome	*N/A*	Move the robot back to the home position
Move	*Velocity(v)*	Apply additional velocity *v* to the robot
Turn	*Angular Velocity(ω)*	Apply additional angular velocity ω to the robot
Stop	*N/A*	Stops the robot motion

Table 3: Primitives Database

Instead of merely mapping the sequence of multi-modal recognition results to the set of actions using the semantics database, the intention aware system should suggest which task (set of primitives) the user may want to execute based on an incomplete sequence of primitives executed by the user. This recognition ability is similar to the auto-completion ability in a text editing program [38]. It is especially helpful when there are a large number of programs, and explicitly searching for any particular program may be time-consuming.

In order to perform such recognition in the real world, it is necessary to represent tasks in a probabilistic framework rather than as a discrete sequence of commands such as {*Goto*(P_1), *Vacuum(vacOn), AreaCoverage*(P_2,P_3), *Vacuum(vacOff), GoHome()*}, where the P_i's describe robot positions in terms of (x, y). A Hidden Markov Model (HMM) provides a way to model the task in a probabilistic framework, where both state

Input Symbol ("voice" and 'gesture')	*Candidate Task*	*Priority*
"Stop" or two 'Closed' fists	Stop()	High
"Go" + "This", "That", *etc* + 'Point'	Goto(*P*)	Medium
"Go" or "Move" + Direction ("Right", "Forward", "Left", "Back")	Move(*v*)	Medium
'Waive' or "Go" + 'Waive'	Move(*v*)	Medium
"Go Home"	GoHome()	Medium
"Vacuum" + "On" or "Off"	Vacuum(vacOn/vacOff)	Medium
"Turn" or "Turn" + direction ("Right", "Left")	Turn(ω)	Medium
"Cover Area" + two 'Point's	AreaCoverage(P_1,P_2)	Medium
"Program" *p*	Program a task *p*	Low
"Complete" *p*	End of program	Low
"Execute Program" *p*	Execute a task *p*	Low

Table 2: Semantics Database

transitions and observations can be expressed stochastically. Since no branching or looping is allowed in tasks, each task can be described as a left-right (Bakis) HMM using an observation sequence collected at the time of programming. Tasks represented in HMMs are organized and compared to the current observation sequence to detect which task, if any, the user may want to execute. Other work, such as the human intention recognition by Yamada et al. [46] and the online point-based hand writing recognition by Bahlmann [3], employs similar strategies. However, our method has an advantage since it is capable of disregarding non-task (garbage) sequences through a garbage collector, without prior training of a garbage model. In this work, the garbage sequence refers to a sequence of observations that are not previously modeled. Such a sequence could therefore occur when the user guides the robot to a new position, and thus needs to be disregarded in the system's task suggestion.

A second important function of the intention interpretation module is to prioritize tasks. Not all tasks are of equal importance. For example, the gesture or word that corresponds to an emergency stop has a very high priority, and should be executed even if the robot is already engaged in another task. Similarly, a high-level task, like navigating to a point (x,y), may require assistance from the user to avoid obstacles and dead ends. The task, therefore, has a lower priority than the tasks for interactive user assistance.

A third function of the intention interpretation module is to perform online modification and adjustment, which are essential since it is unreasonable to expect the system to have prior knowledge of every intended task. The system must be capable of adjusting and adding primitives to the program with ease. The system supports these adjustments by letting the user interrupt the task while it is running, and by registering the interrupts as additional primitives in the task.

Adaptation of models is necessary to reflect changes in the environment and modifications to a program made by the user while interacting with the system. Also, observation sequences collected from subsequent executions of the same task can be combined to improve stochastic parameters used in the HMM representation of the task.

In the remainder of this section, we explain how HMM representations of tasks are constructed and used for intention recognition, and how they can be updated in real-time during execution.

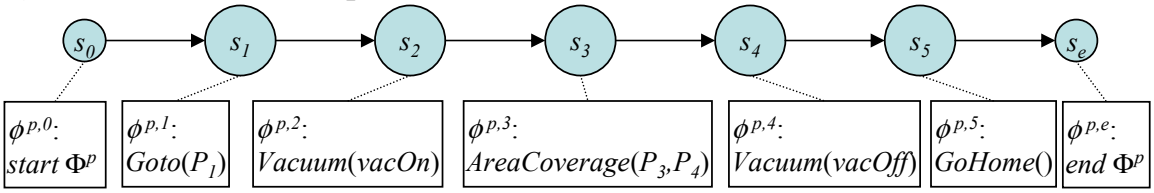
3.2.1. Construction

The aim in this section is to create λ_p, a Continuous-Density HMM (CDHMM) description of a program p, from Φ^p, a command sequence of the program and to combine the λ_p's into a network CDHMM, λ_{net}, that can be used for real-time program recognition. This process is illustrated in Figures 2 and 3.

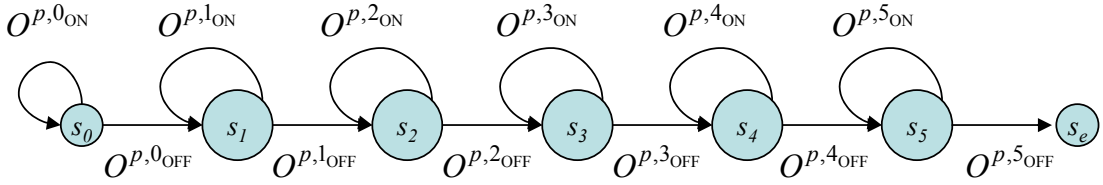
Robot Program Φ^p:

$\Phi^p = \{Goto(P_1),\ Vacuum(vacOn),\ AreaCoverage(P_3,\ P_4),\ Vacuum(vacOff),\ GoHome()\}$
$= \{\ \phi^{p,1}\ ,\ \phi^{p,2}\ ,\ \phi^{p,3}\ ,\ \phi^{p,4}\ ,\ \phi^{p,5}\ \}$

1) Markov Chain Description of Φ^p:



2) Association of Observations Generated by Φ^p with CDHMM:



$O^p = \{O^{p,0}, O^{p,1}, \cdots, O^{p,N_p}\}$

$O^{p,a} = (O^{p,a_{ON}}, O^{p,a_{OFF}}) = (O^p_{\tau^{p,a_{ON}}}, O^p_{\tau^{p,a_{OFF}}})$

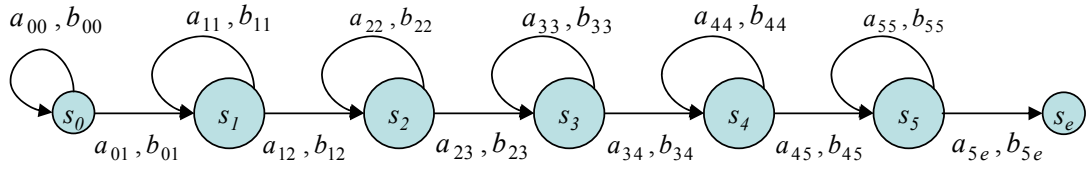
$O^p_t = (x_t, y_t, \theta_t)^p = (x, y, \theta)$ of program Φ^p at time index t

$\tau^{p,a_{ON}} = \{t \in \tau^p$: robot at time t executing an action $\phi^{p,a}\}$

$\tau^{p,a_{OFF}} = \{t \in \tau^p$: robot at time t after execution of an action $\phi^{p,a}$, before executing $\phi^{p,a+1}\}$

$\tau^p = \{$set of all time t during execution of program $\Phi^p\}$

3) Generated CDHMM description λ_p of Φ^p:



$\lambda_p = \{\pi_p, A_p, B_p\}$, where A_p is the set of a's, B_p the set of b's
π_p = initial state probability
a_{ij} = state transition probability from s_i to s_j
b_{ij} = observation probability during transition from s_i to s_j
$= N(\bar{\mu}_{ij}, \Sigma_{ij}) \cdot WN(\mu_{\theta ij}, S_{\theta ij})$

Figure 2: Conversion of a sample program Φ^p to Continuous Density HMM λ_p

HMM Network for Task Recognition:

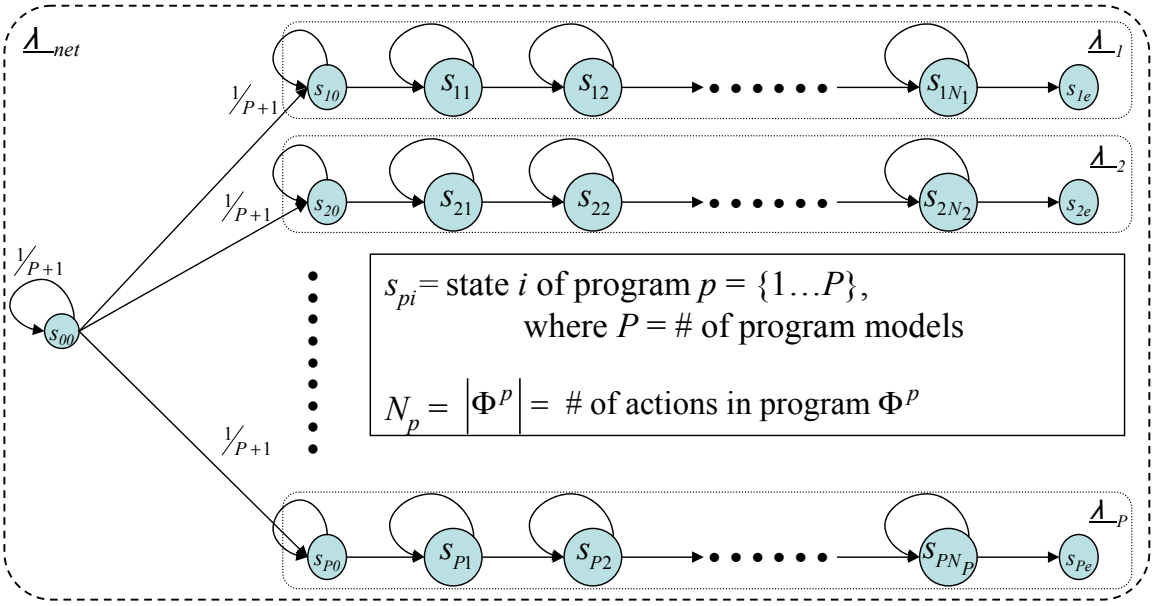


Figure 3: HMM Network with Shared Initial State

A program Φ^p consists of a set of sequential program actions $\phi^{p,0}$, $\phi^{p,1}$,…, $\phi^{p,n}$. When a robot is programmed interactively, the system collects an observation sequence $O^p = \{o_1^p, o_2^p \ldots o_{T_p}^p\}$ where $o_t^p = (x_t, y_t, \theta_t)^p$ correspond to the robot position and orientation at time t for program p. The sequence O^p of the program Φ^p is the collection of all observations $O^{p,a}$ resulting from program actions $\phi^{p,0}$, $\phi^{p,1}$,… $\phi^{p,a}$. The robot program Φ^p is then converted into a CDHMM λ_p through the process illustrated in Figure 2. The program is first converted into a Markov chain (top of Figure 2) whose states correspond to each programmed action. The number of states in the chain is the number of actions in the program plus two (the start and end states). Observations of the robot's position (x, y, θ) are collected during the task execution and are associated with each of the arcs in the state transition diagram. For each arc, the observation sequence is encoded in an observation density function, b_{ij}.

The observation density function, b_{ij}, is an expression for the likelihood of observing a given robot position $o_t^p = (x_t, y_t, \theta_t)^p$ given that the robot is moving from state i to state j. b_{ij} (Equation 1) is modeled as a combination of a normal distribution, $N(\bar{\mu}_{ij}, \Sigma_{ij})$ (Equation 4) and a wrapped normal distribution $WN(\mu_{\theta ij}, S_{\theta ij})$ (Equation 9).

$$b_{ij} = N(\bar{\mu}_{ij}, \Sigma_{ij}) \cdot WN(\mu_{\theta ij}, S_{\theta ij}) \tag{1}$$

$$\bar{\mu}_{ij} = \frac{1}{T_{ij}} \sum_t^{T_{ij}} \bar{x}_t \tag{2}$$

$$\Sigma_{ij} = \frac{1}{T_{ij}} \sum_t^{T_{ij}} [\bar{x}_t - \bar{\mu}_{ij}][\bar{x}_t - \bar{\mu}_{ij}]^T \tag{3}$$

$$N(\bar{\mu}_{ij}, \Sigma_{ij}) = \frac{1}{2\pi\sqrt{|\Sigma_{ij}|}} \exp\left(-\frac{[\bar{x} - \bar{\mu}_{ij}]^T \Sigma_{ij}^{-1} [\bar{x} - \bar{\mu}_{ij}]}{2}\right) \tag{4}$$

$$\mu_{\theta ij} = \arctan\left(\sum_t^{T_{ij}} \sin\theta_t \Big/ \sum_t^{T_{ij}} \cos\theta_t \right) \tag{5}$$

$$S_{\theta ij} = 1 - \frac{\sqrt{\sum_t^{T_{ij}} \cos\theta_t + \sum_t^{T_{ij}} \sin\theta_t}}{T_{ij}} \tag{6}$$

$$\sigma_{\theta ij}^2 = -2\log(1 - S_{\theta ij}) \tag{7}$$

$$WN(\mu_{\theta ij}, S_{\theta ij}) = \frac{1}{\sqrt{2\pi\sigma_{\theta ij}^2}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{1}{2} \frac{(\theta_t + 2\pi k)^2}{\sigma_{\theta ij}^2} \right) \tag{8}$$

$$\cong \begin{cases} \dfrac{1}{\sqrt{2\pi\sigma_{\theta ij}^2}} \exp\left(-\dfrac{1}{2} \dfrac{\theta_t^2}{\sigma_{\theta ij}^2} \right) & , \text{for } \sigma_{\theta ij}^2 \le \pi \\ \left(1 + 2\sum_{p=1}^{3} e^{-\frac{1}{2}\sigma_{\theta ij}^2 p^2} \cos p\theta_t \right) \Big/ 2\pi & , \text{for } \sigma_{\theta ij}^2 > \pi \end{cases} \tag{9}$$

Observed positions are assumed to be correlated but orientations are assumed to be independent from the positions (*i.e.* $WN(\mu_{\theta ij}, S_{\theta ij})$ is independent from $N(\bar{\mu}_{ij}, \Sigma_{ij})$) since the robot motion tends to be either unidirectional or equally varying throughout the configuration space. We should point out that extra care is needed to calculate the sample mean and variance of orientation data, which are expressed in circular rather than Cartesian coordinates. A wrapped normal distribution $WN(\mu_{\theta ij}, S_{\theta ij})$ is approximated in one of two methods (Equation 9) depending on its angular variance, $\sigma_{\theta ij}^2$. Refer to Mardi's text [23] for justification of the approximation.

In addition to the observation probabilities, b_{ij}, the CDHMM for program λ_p is also characterized by state transition probabilities, a_{ij}. The state transition probability a_{ij} is determined by taking the ratio of the number of observations used at recurring and transition arcs. $O^{p,a} = (O^{p,a_{\mathrm{ON}}}, O^{p,a_{\mathrm{OFF}}}) = (O^{p}_{\tau^{p,a_{\mathrm{ON}}}}, O^{p}_{\tau^{p,a_{\mathrm{OFF}}}})$ is an observation sequence for program action $\phi^{p,a}$, where $\tau^{p,a_{\mathrm{ON}}}$ is a set of time index t while action $\phi^{p,a}$ is being executed, and $\tau^{p,a_{\mathrm{OFF}}}$ is a set of time index t after execution of $\phi^{p,a}$ and before execution of $\phi^{p,a+1}$. The state transition probability a_{ij} is defined as follows:

$$a_{ij} = \begin{cases} \dfrac{\left|\tau^{p,a_{\mathrm{ON}}}\right|}{\left|\tau^{p,a_{\mathrm{ON}}}\right| + \left|\tau^{p,a_{\mathrm{OFF}}}\right|} & \text{for } i = j \\ \dfrac{\left|\tau^{p,a_{\mathrm{OFF}}}\right|}{\left|\tau^{p,a_{\mathrm{ON}}}\right| + \left|\tau^{p,a_{\mathrm{OFF}}}\right|} & \text{for } i \neq j \end{cases} \tag{10}$$

After converting all programs Φ^p to λ_p for $p = 1 \ldots P$, where P is the number of program models, the λ_p are combined into one λ_{net} for recognition purposes. Figure 3 describes how λ_{net} is constructed from the CDHMMs λ_1, λ_2 … λ_P. All the transition probabilities from s_{00} ($a_{s_{00}s_{00}}, a_{s_{00}s_{10}}, \cdots a_{s_{00}s_{P0}}$) are assumed to be equal with a value of $1/(P+1)$. Observation probabilities for these arcs $b_{s_{00}s_{00}}, b_{s_{00}s_{10}}, \cdots b_{s_{00}s_{P0}}$ are undetermined at this point. They are assigned dynamically inside the recognition algorithm described in the following section.

3.2.2. Recognition

During recognition, the current sequence of position observations is evaluated and compared to all the robot program CDHMMs stored in the network λ_{net}. It is necessary not only to detect in real-time which program the user may be interested in, but also to reject observations that are not part of any existing program. The goal here is to find the most likely state q_t at the current time t, given observations up to time t, and the CDHMMs λ_1, λ_2 … λ_P, constructed from P robot programs organized into λ_{net} as described in Figure 3.

Initialization:

Assign a token with value of 1 to the initial shared state s_{00}.

Assign a token with value of 0 to all other states.

For all arcs not originating from state s_{00}, compute and store the value ψ_{ij}.

Algorithm:

for each time t **do**

for each state $i \neq s_{00}$ **do**

Compute and store the Mahalanobis distance between o_t and μ_{ij};

Pass a copy of the token in state i to each connecting state j, multiplying its value by $a_{ij}b_{ij}(o_t)$.

If the new token value underflows to 0, let the value be ε;

end;

Pass a copy of the token in state s_{00} to each connecting state j, multiplying its value by $a_{ij} \bullet \psi_{mn}$. Choose the ψ_{mn} for which the Mahalanobis distance between μ_{mn} and o_t is the smallest;

Discard the original tokens;

for each state i **do**

Find the token in state i with the largest value and discard the others;

end;

Normalize all tokens such that their sum equals 1;

Find the state q_t with the largest token value;

end;

Figure 4: Viterbi Algorithm with Dynamic Garbage Collection

To find the single most likely state q_t out of all states in the shared CDHMM network for the current observation sequence, we use a modified Viterbi Algorithm described in Figure 4. The Viterbi algorithm, based on the Token Passing paradigm [9], has been modified by adding dynamic garbage collection, that is, recognizing the state, s_{00}, in which none of the programs is being executed. The modification involves the dynamic computation of the observation probabilities for state s_{00}, as is illustrated in Figure 5.

Consider a sample CDHMM network (Figure 5.a) constructed from a shared garbage state s_{00}, and two member CDHMMs each with only one state, s_{10} and s_{20}. The member CDHMMs have observation density functions $b_{s_{10}s_{10}}$ and $b_{s_{20}s_{20}}$ (Figure 5.b/c). If the observation o_t is close to either $\mu_{s_{10}s_{10}}$ or $\mu_{s_{20}s_{20}}$, the algorithm will consider the observation to indicate that the corresponding state s_{10} or s_{20}, respectively, should be promoted. However, if the observation is far removed from both $\mu_{s_{10}s_{10}}$ and $\mu_{s_{20}s_{20}}$, then the shared garbage state, s_{00}, should be promoted. This is achieved by introducing a

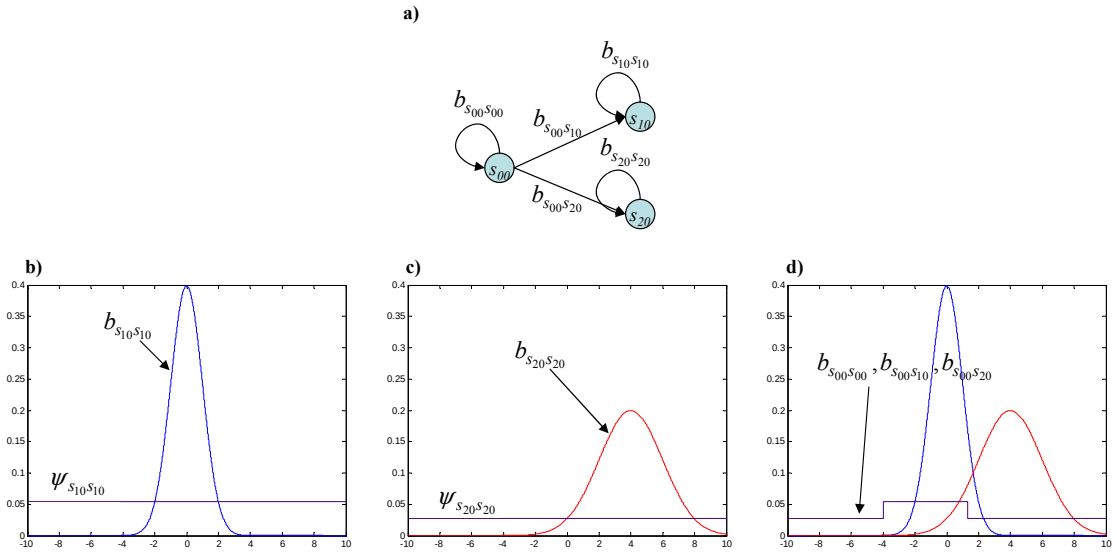


Figure 5: (a) sample CDHMM network (b) pdf for b_{s10s10} (c) pdf for b_{s20s20} (d) dynamically generated b_{s00s00}, b_{s00s10}, b_{s00s20}

threshold ψ_{ij} for each b_{ij} below which the garbage state will be given preference. The value of ψ_{ij} is chosen as the value of b_{ij} at a distance of 3σ from the mean, that is:

$$\psi_{ij} = \frac{e^{-3^2/2}}{2\pi\sqrt{|\Sigma_{ij}|}} = \frac{e^{-4.5}}{2\pi\sqrt{|\Sigma_{ij}|}} \tag{11}$$

Since this value is different for every b_{ij}, the algorithm uses the value for the state that is closest to the current observation according to the Mahalanobis metric [7]:

$$r_{ij}(t) = \sqrt{[o_t - \mu_{ij}]^T \Sigma_{ij}^{-1} [o_t - \mu_{ij}]} \tag{12}$$

This is illustrated in Figure 5.d. The lines for $b_{s_{00}s_{00}}$, $b_{s_{00}s_{10}}$, and $b_{s_{00}s_{20}}$ can be thought of as classification boundaries below which the algorithm gives preference to the garbage state.

The advantage of the above algorithm is that, unlike the garbage models used in large vocabulary state-spotting systems, this algorithm requires no previous training or batch processing of the garbage model. It also requires little additional computation during the recognition phase because the ψ values only need to be computed once, during initialization, and the Mahalanobis distances, used to select the appropriate ψ values, are calculated and stored when the observation density functions are evaluated for a particular observation.

Since we are only interested in the most likely state, we only need to keep track of the CDHMM trellis of token scores as a starting point for the processing of additional observations. Each token value corresponds to the likelihood of being in the particular state after going through the most likely state sequence. Based on the assumption that the model λ_{net} fully explains all observation sequences, the entire trellis is normalized to 1.0 for every observation. The initial "garbage" state becomes the most likely state if the observation sequence can not be explained by any of the other models $\lambda_1 \ldots \lambda_P$. After finding the current CDHMM node, the system can determine the action that should be taken according to the most probable robot program.

3.2.3. Model Update

Online seamless adjustments of the statistics (a_{ij}, b_{ij}) that describe the robot program are essential for keeping the system healthy. For example, an additional obstacle on the path between via-points can change the trajectory of the mobile robot, requiring that the program description be adjusted. Parameter adaptation can be used to improve the CDHMM parameters over multiple executions of the same task. This can be done by first partitioning the observation sequence and merging statistics derived from new samples with the old statistics. n_{add} additional samples with mean vector, μ_{add}, and covariance matrix, Σ_{add}, can be merged with n_{old} old samples with statistics μ_{old} and Σ_{old} to derive the combined statistics, n_{new}, μ_{new}, Σ_{new} as follows: (Appendix A) [20]:

$$n_{new} = n_{add} + n_{old} \tag{13}$$

$$\mu_{new} = (n_{add}\mu_{add} + n_{old}\mu_{old}) / n_{new} \tag{14}$$

$$\begin{aligned} \Sigma_{new} &= (A + B + C + D) \ / \ (n_{new} - 1) \\ &\begin{cases} A = (n_{add} - 1)\Sigma_{add} \\ B = n_{add}(\mu_{add} - \mu_{new})(\mu_{add} - \mu_{new})^T \\ C = (n_{old} - 1)\Sigma_{old} \\ D = n_{old}(\mu_{old} - \mu_{new})(\mu_{old} - \mu_{new})^T \end{cases} \end{aligned} \tag{15}$$

Using the above equations, one can compute the statistics for adapted observation probabilities without having to keep the entire observation history. For implementation purposes, we always set $n_{old} = n_{new}$, so that the effects from old samples will eventually decay with additional adaptation cycles.

3.3. Prioritized Task Execution Module

The prioritized task execution module (the third block in Figure 1) has two functions. The first is to arbitrate and execute primitives based on the current state, the sensor inputs, and the prioritized task given by the previous module. The second is to generate a robot program (task) by configuring primitives.

Before going into the details of each function, we distinguish tasks from primitives based on their level of abstraction. Primitives are encapsulations of low-level robot behaviors and serve as building blocks of high-level behaviors. They consist of motor (M), sensor (S), or sensori-motor (SM) primitives. Motor primitives generate open loop behaviors that do not depend on sensor feedback. For mobile robots, motor primitives include sensor independent acceleration, stop, turn, beep, and directional motions. Sensor primitives provide the system with observable sensor signals, such as the current robot position, range sensor data, and bumper switch data. Sensori-motor primitives generate closed loop behaviors, such as wall following or navigation towards a particular destination. The sensori-motor primitives can be thought of as pre-tailored configurations of motor and sensor primitives.

A task is a configuration of primitives—either a sequence of primitives, or a single primitive. Tasks are stored in a database in the form of a state buffer [17], Markov chain [36] or finite state machine [27]. Because the intention interpretation module requires access to some of the task data also, it shares the semantic, primitive, and task databases with the task execution module, as is shown in Figure 1.

The first important function of the task execution module is task arbitration. As is explained in the section on intention interpretation, not all tasks are equally important. When tasks with different priorities are passed to the prioritized task execution module, the module orders the tasks and executes them according to their priority. The scheme can be described as event driven preemption, where the event (a request from the intention interpretation module to execute a task) triggers an active switch from the running task with lower priority to another with higher priority. This allows the user to handle situations such as making an emergency stop or avoiding an obstacle during the execution of other tasks. Figure 6 describes the first function, where the task T2 with low priority (3) is preempted by the task T1 with medium priority (5) during the execution of

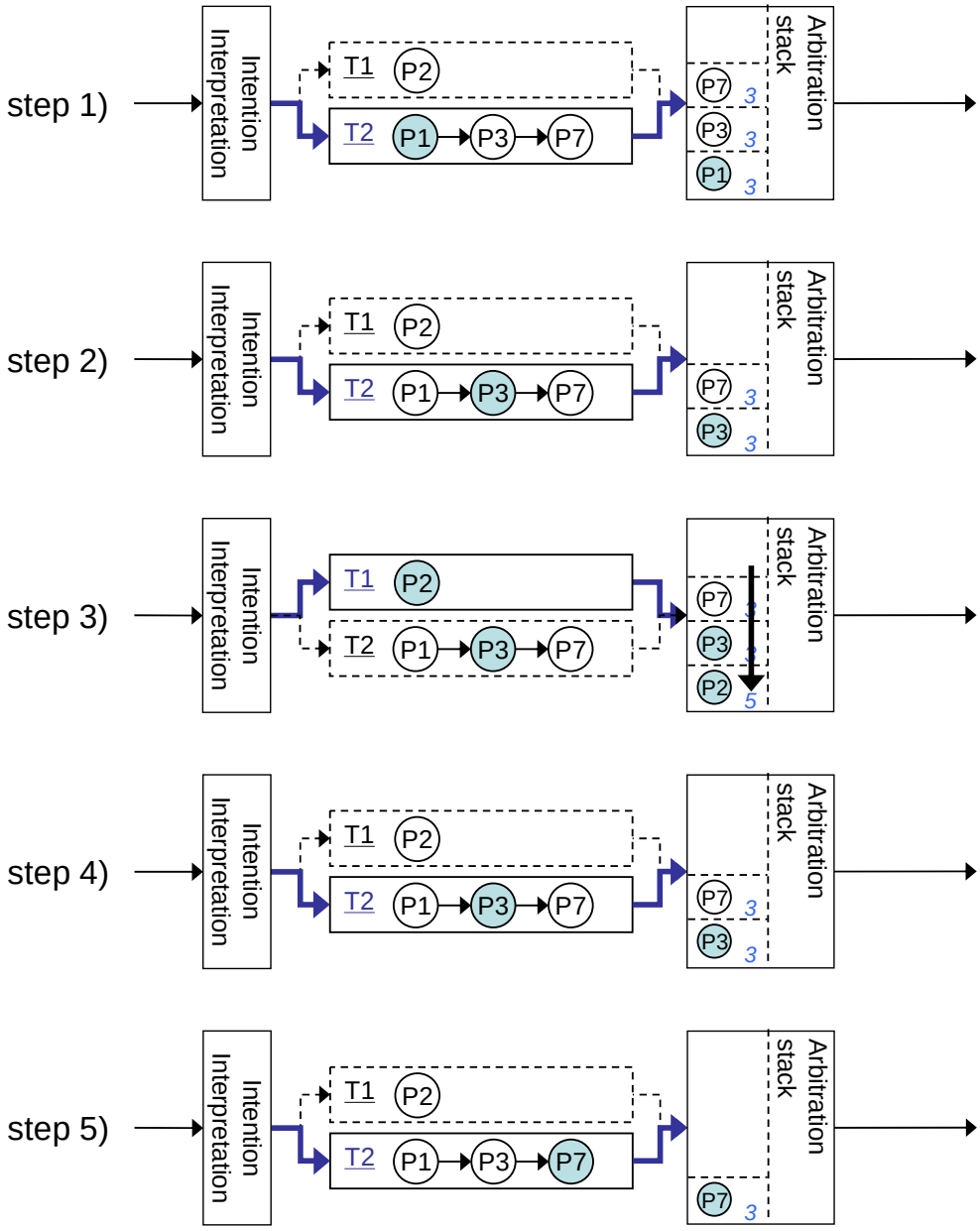


Figure 6: HMM Arbitration Based on Task Priority with T1 (priority=5), T2 (priority=3)

primitive P3. The task T2 can be thought of as a program (task) that consists of primitives P1, P3, and P7, while the task T1 is a single primitive task.

The second function of the task execution module is to generate a robot program (task) interactively. The basic approach is to take a coaching strategy using a redundant input mode. The user sets the module to a learning mode and executes primitives

sequentially; the system remembers the sequence as a task. There are two obvious problems with this approach. The first problem is that the robot programs include conditional branching and looping. Forcing the user to remember a special gesture command to indicate branching and looping conditions would make the system counter intuitive. A tool to convey a program structure and an intuitive interface to edit the program are necessary, unless the system can infer such conditions from multiple examples. In the current implementation, iconic programming is used to convey and edit non-sequential program structures.

The second problem is the lack of generality. The task would be useless if it would only work for the particular parameter value for which it was trained. Somehow, the user must let the module know that some attributes can be generalized while others need to be retained as important features of the task. For the point and navigate task, the goal coordinates should be variable, while the sequence of primitives used to navigate needs to be retained. Making this distinction is the subject of future research.

Table 4 summarizes the functions offered by each of the three modules in the framework. The three modules work synchronously in a continuous flow of data for providing intuitive multi-modal interaction and programming of robots.

Module	*Input*	*Function (Execution and/or Learning mode)*	*Output*
Multi-modal Recognition	CyberGlove-R Polhemus-R CyberGlove-L Polhemus-L Acoustic (8bit-16KHz)	• Translate incoming audio and gesture signals into a structured stream of word and gesture unit symbols with appropriate parameters. (*E*) • Reinforce models during recognition (exec. & learn)	Gesture Symbol-R + param. Gesture Symbol-L + param. Word Symbol + param
Intention Interpretation	Gesture Symbol-R + param Gesture Symbol-L + param Word Symbol + param Robot Data Robot Position Robot Velocity Sensor Readings Knowledge of its current state	• Select the appropriate primitives based on the user input, current state, and robot sensor data. (*E*) • Prioritization of tasks, according to the database (*E*) • Adapt task model used for selections using the most current observations (exec. and learn)	Task symbol + priority + param
Prioritized Task Execution	Robot Status Sensor Readings Task symbol + priority + param	• Arbitrate and execute primitives based on current state, sensor input, and the prioritized task given by the previous module. (*E*) • Generate a robot program (task) by configuring primitives. (*E* & *L*)	Control vector

Table 4: Functional Summary

4. Demonstration

We have conducted two demonstrations to exemplify the interactive programming and the plan recognition aspect of the research. The first demonstration is to verify the operation of overall system through sequential programming and adjustment of a mobile vacuum cleaning robot. The second is to demonstrate intention awareness by letting the system detect the most likely program the user wants to execute, and have the intention model adapt to the current observations.

4.1. Sequential programming and adjustment

The first demonstration was conducted to verify the connections between all three modules and to illustrate the overall operation of the framework with a basic interactive programming example. The framework is implemented using a Cye vacuum cleaning robot [4], two 22-sensor CyberGloves [1], and a microphone. We modified the graphical user interface provided with the vacuum cleaning robot, to accept hand gestures and speech input, while retaining its original functionality: mapping, iconic programming, and path-planning. As a result, Cye can be controlled via mouse, speech, and hand gestures.

The multi-modal recognition module is implemented using the *Sphinx-II* speech recognition engine [11] and the Hidden Markov Toolkit (*HTK*) [47] that has been customized to recognize gestures at 60Hz with 92% recognition accuracy. A discussion of the gesture recognition methodology is outside the scope of this article; however, the method is similar to the one in [30], where parameters of Hidden Markov models for each gesture are obtained from known strings of gesture examples. Each gesture consists of gesture phonemes that take into account finger-joint positions, joint velocities, and the hand's Cartesian position and velocity. The vocabulary of gestures is listed in Table 1. The on-line addition of vocabulary is not implemented at this point although the system is capable of adapting model parameters for new users with very few additional training samples, using capabilities offered by HTK.

The intention interpretation module is implemented with a semantic database (Table 2). The semantic database connects inputs such as gesture and speech symbols, the robot's sensor readings, and the current state to the most likely robot task. A task, which

can be considered as a robot program, is a set of one or more primitives. Each task has predefined priorities attached to specify the importance of the task over the others in the event of preemption. At this point, the semantic database is fixed and does not support the on-line addition of entries.

The prioritized task execution module ensures that the primitives are executed according to their assigned priorities. The primitives used in the current scenarios are listed in Table 3. Primitives such as *GoHome* and *AreaCoverage* provide high-level navigation, whereas primitives such as *Move*, *Turn*, and *Vacuum* give low-level control of the robot. Primitives are executed in order of arrival except when a high-priority task is introduced; such tasks pre-empt the current task and execute immediately.

For the current implementation, we have considered two interactive programming scenarios. The first scenario is to have a user register numerous via-points to which the robot should navigate using its path planning capability. The second scenario is to use a two-handed gesture to specify an area that the robot should vacuum; the robot then vacuums the area using its area coverage primitive. In both scenarios, the robot can accept the user's preemptive speech and hand gesture commands to deal with unforeseen events. Figure 7 and Figure 8 illustrate the sequences of the first and second scenario. Each figure contains a sequence of camera snapshots with the corresponding conceptual illustrations of the framework, and the cropped images of the GUI. Refer to Multimedia Extension 1 for the video of the programming phase and Extension 2 for the execution phase of the first scenario.

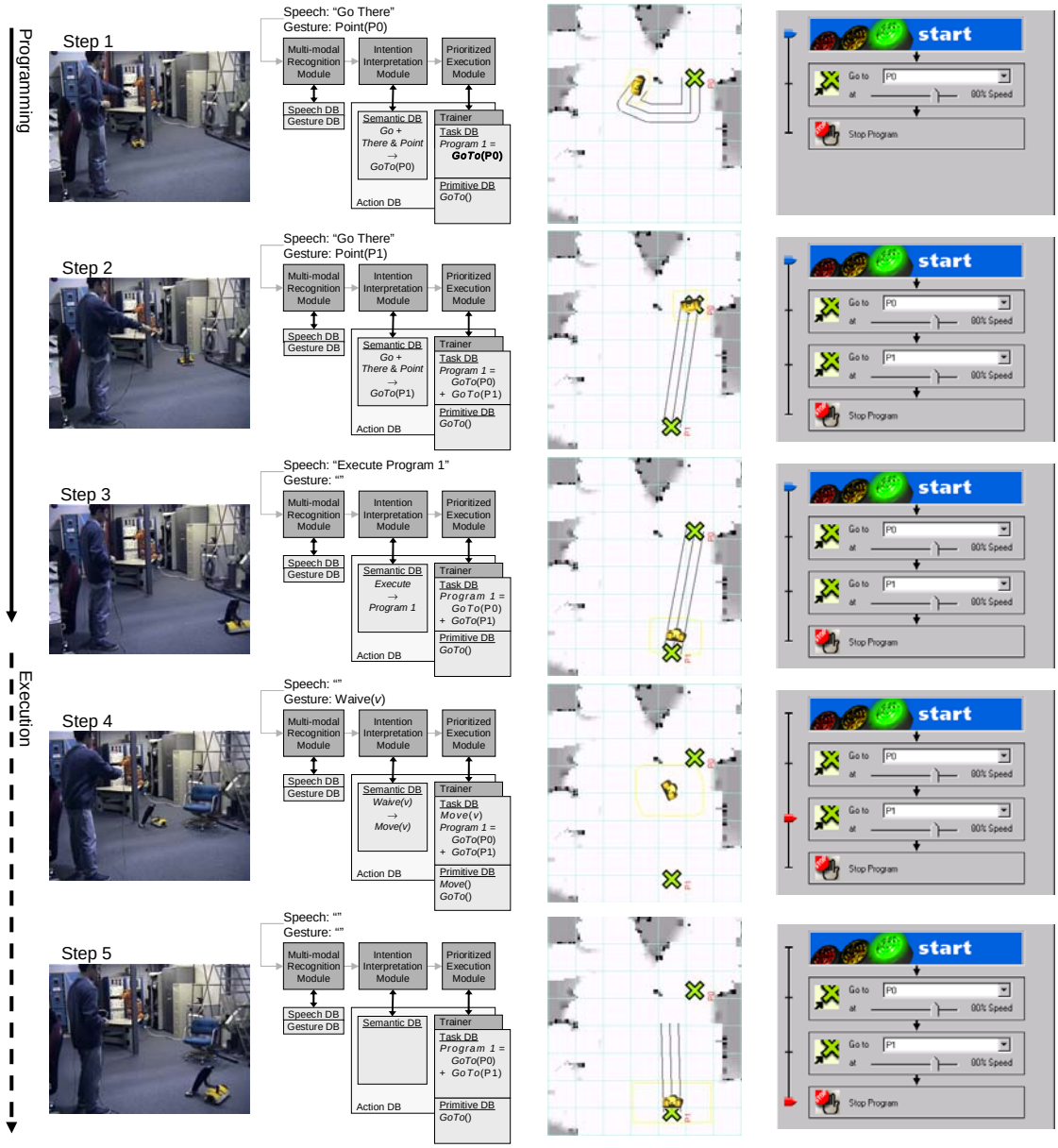


Figure 7: Demonstration Scenario 1

In the first scenario, the user first verbally commands that the subsequent actions be stored as "Program One". The user then executes the *Goto* primitive by combining the voice command "Go There" with the gestural command 'Point' to indicate the destination. In general, deictic terms such as "This", "That", and "There" must be accompanied by a referential gesture to specify the corresponding task parameters. For the *Goto* primitive, the Cartesian coordinates are extracted from the intersection between the extension of the index finger and the ground [12]. In step 2, the user enters another *Goto* primitive but

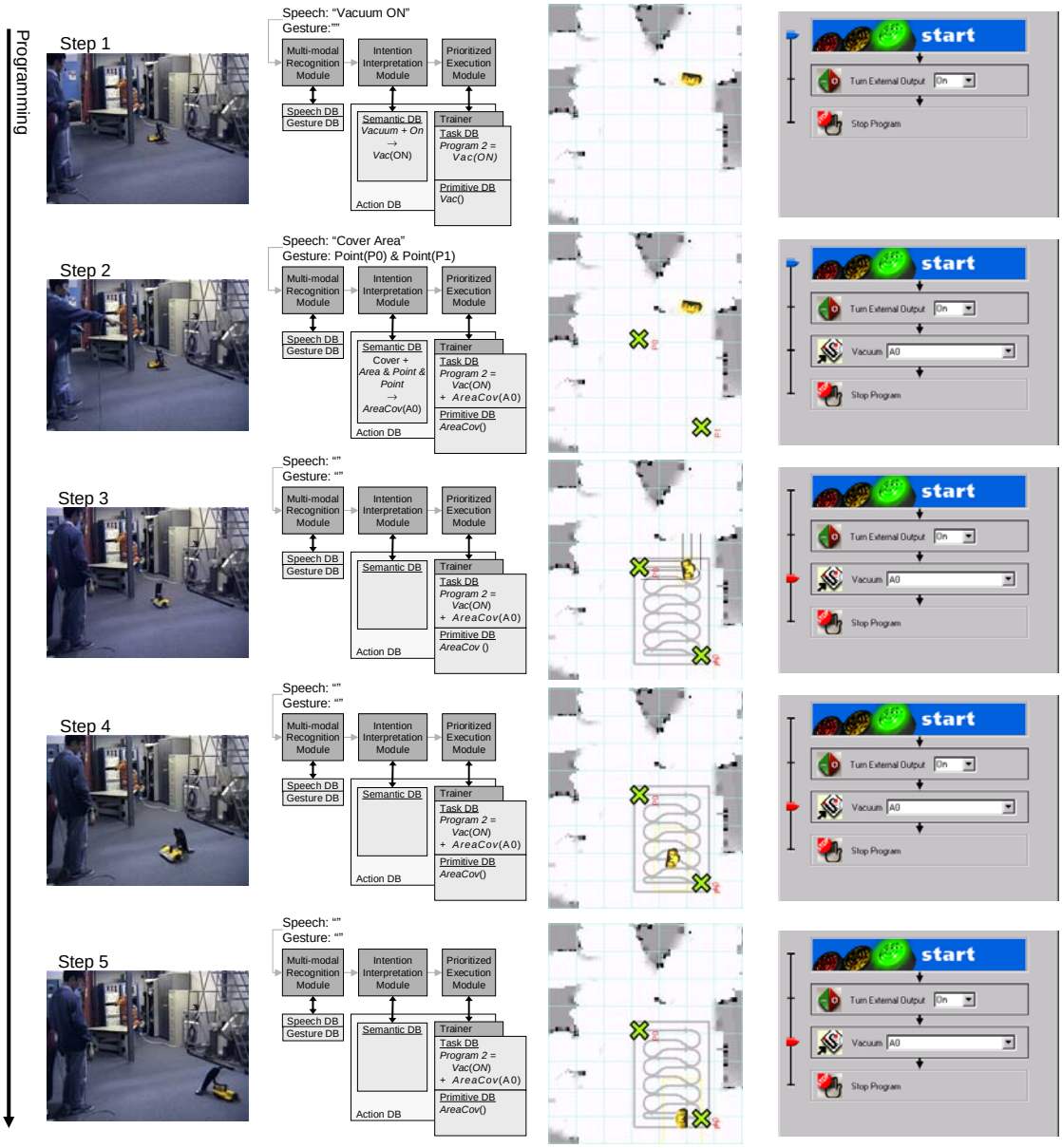


Figure 8: Demonstration Scenario 2

with a different end-position. After having saved these two primitives in "Program One" with the "Complete" command, the user can re-execute the program through with the voice command "Execute Program One". However, in step 4, when the robot navigates to the second position from the first, it encounters an unknown obstacle. At this point, the user gestures the 'Waive' command, which has a higher task priority and can be used to control the robot around the obstacle. When the obstacle has been cleared and the user stops waiving, the robot returns to the execution of "Program One" (Step 5).

In the second scenario, illustrated in Figure 8, the user defines the task "Program Two." After turning on the vacuum attachment with the voice command "Vacuum On" (step 1), the user issues the *AreaCoverage* command with one two-handed gesture; each hand performs a 'Point' gestures to specify the diagonally opposite corners of the area (with the direction aligned along the axes of the GUI). Steps 3 to 5 show the execution of the *AreaCoverage* command. As in the first scenario, at any point can the user re-execute "Program Two", interrupt the execution, or interactively adjust the execution with higher-priority commands.

4.2. Plan recognition

The second demonstration was conducted to verify the system's intention awareness. Assume that the database of robot programs contains three test programs:

Φ^1 = {Goto(P_1), Vacuum(vacOn), AreaCoverage(P_2, P_3), Vacuum(vacOff), GoHome()}

Φ^2 = {Vacuum(vacOn), Goto(S_1), Goto(S_2), AreaCoverage(S_3, S_4), GoHome()}

Φ^3 = {Goto(T_1), Goto(T_2), Goto(T_3)}

where P_i, S_i, T_i all represent positions on the map in (x, y), as described on Figure

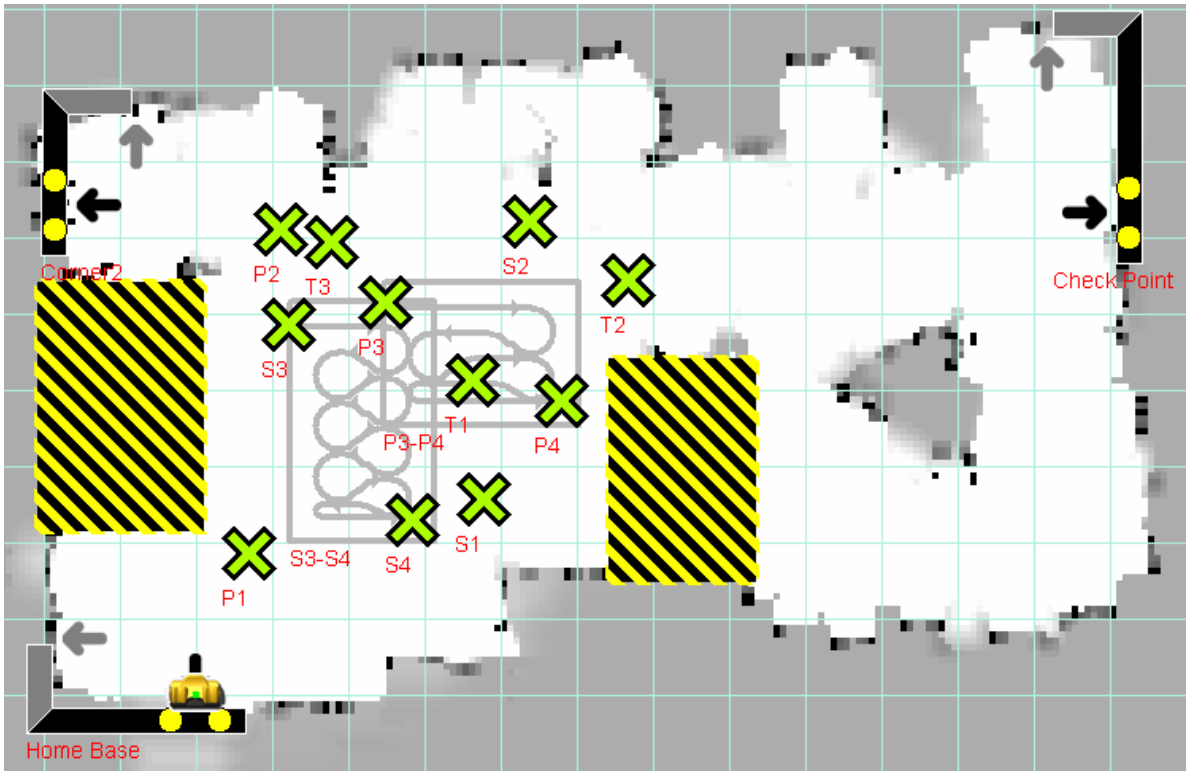


Figure 9: Positions used for the test programs λ_1, λ_2, λ_3

9.

For each test program, a CDHMM representation was created through the method described in section 3.2.1. The trajectories and the resulting observation probability densities are illustrated in Figure 10. Each program was executed four times while collecting observation sequences with a 5Hz sample frequency. There was variability in the path in λ_1 and λ_2 to test adaptation.

Recognition was performed on the constructed CDHMM network λ_{net} using three test observation sequences, illustrated in Figure 11 shows three test sequence in different columns, and each row correspond to the observations sequence, $\log(\delta_t(i))$ to enhance small scores, and $\delta_t(i)$ to show which state i had the best score in each time step t.

The first test sequence is one of the training sequences used for the first program λ_1. Its $\delta_t(i)$ image (3rd row, 1st column) shows that the algorithm follows the states in λ_1 that are between s_{10} and s_{1e}. The second test sequence used Goto() commands to move in the order of $\{Home, S_1, S_2, T_3, T_1, T_2, T_3\}$. The test result shows what is expected: the recognition algorithm starts by selecting the states in the program λ_2 and jumps to the states in program λ_3. In the third test sequence, the robot follows a random trajectory that

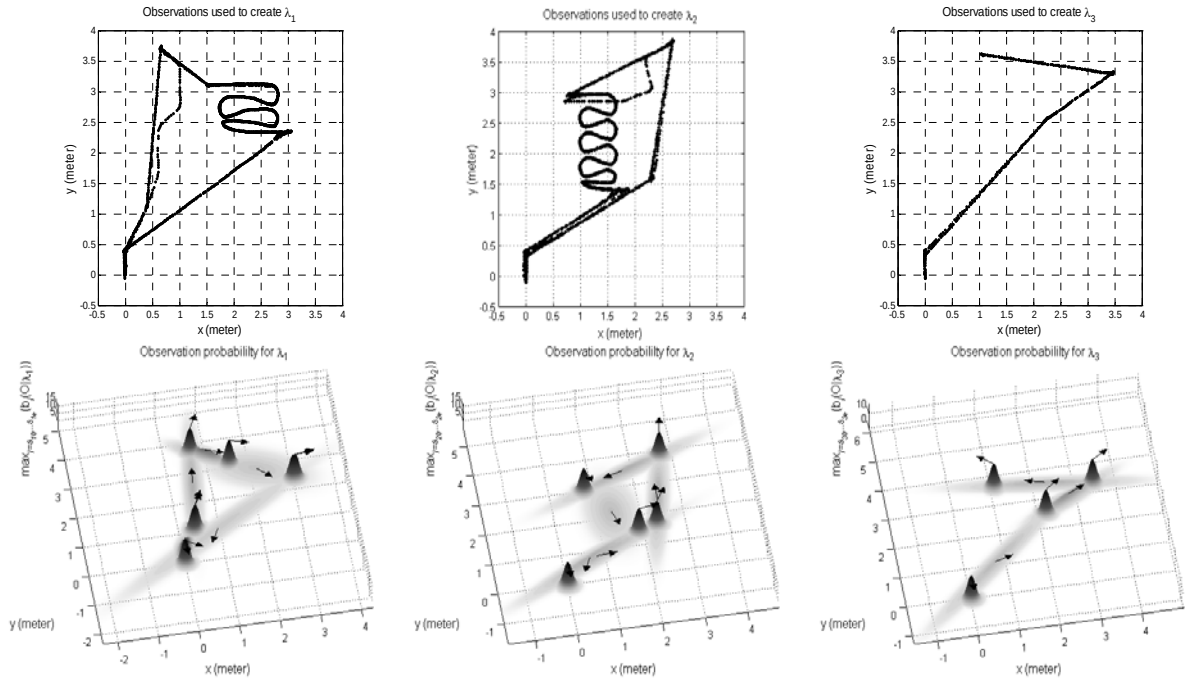


Figure 10: Observations used to construct the programs λ_1, λ_2, λ_3 (from left to right) and the resulting observation probability densities (arrow = mean orientation)

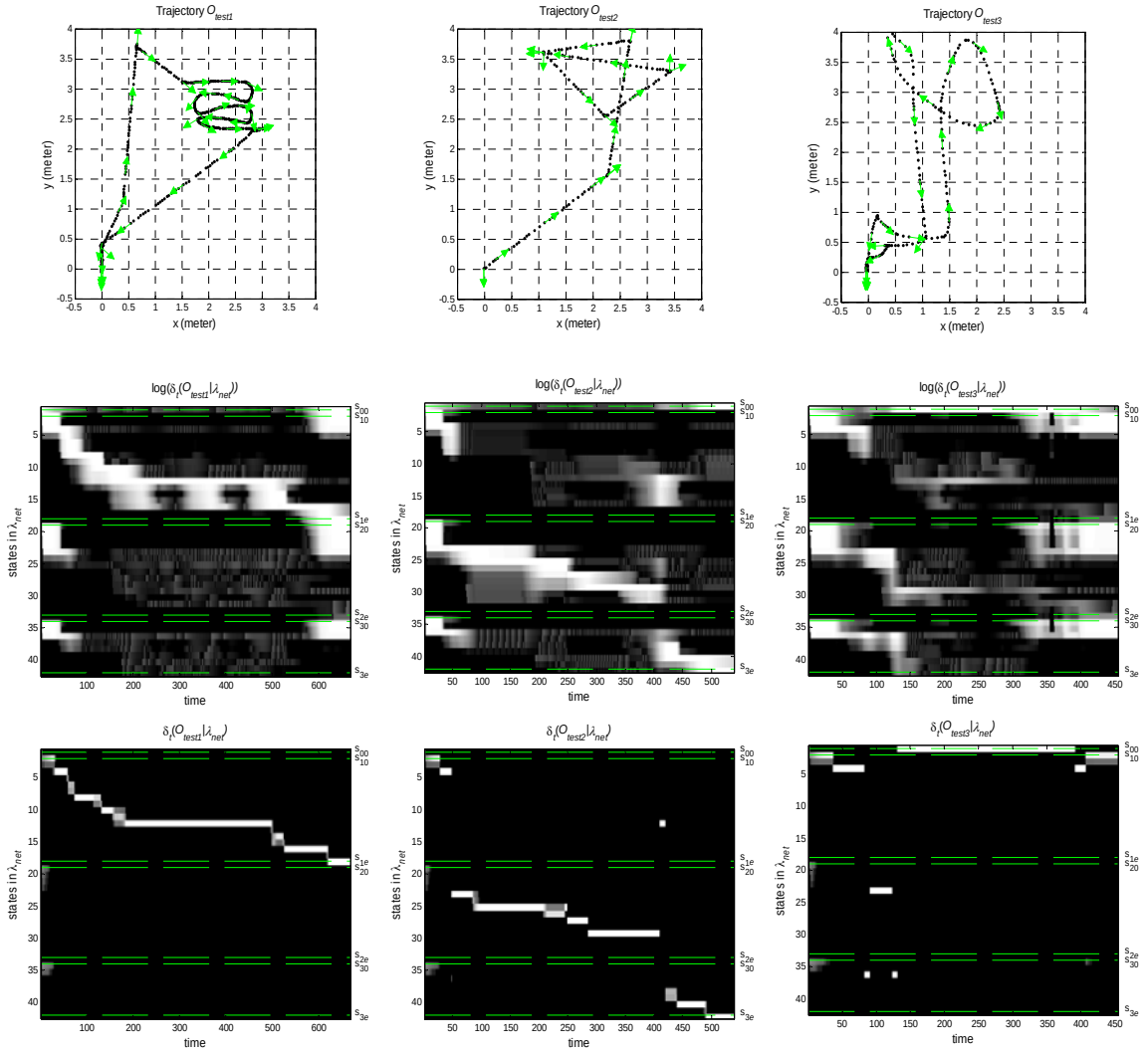


Figure 11: observation sequences O_{test1}, O_{test2}, O_{test3}, with their corresponding $\log(\delta_t(i))$ and $\delta_t(i)$

does not include any of the three programs in the database. Since this trajectory does not resemble any of the pre-defined programs, the recognition algorithm selects the shared garbage state almost the entire time. This is shown in the right-most graphs in Figure 11 where the top row of the figure is white, indicating that state zero has the highest probability value.

5. Summary

Human-Robot interaction needs to be intuitive, interactive, and intention aware. In this paper, we have described the overall framework for interactive multi-modal robot programming and have illustrated the framework using two demonstrations. The

programming approach offers, through an intuitive interface using hand-gestures and speech recognition, the ability to provide interactive feedback to the robot to coach it throughout the programming and execution phases. The user's intent is captured in the form of a sequential robot program, and the flexibility given to the user by the framework through real-time interaction and intuitive interface allows the captured intent to be closer to the user's true intent.

The framework is composed of three functional modules. The first module (multi-modal recognition) translates hand gestures and spontaneous speech into a structured symbolic data stream without abstracting away the user's intent. The second module (intention interpretation) selects the appropriate set of primitives based on the user input, current state, and robot sensor data. Finally, the third module (prioritized execution) selects and executes primitives based on the current state, sensor inputs, and the task given by the previous step.

The first demonstration verified interactive multi-modal programming and execution of two sequential programming scenarios: point-to-point navigation and area coverage, which clearly illustrates the usefulness of multi-modal interaction, including the capability to interrupt commands preemptively. The second demonstrated that the system can determine the most likely high-level goal the user is trying to achieve, given a limited, initial sequence of task primitives. A set of user intentions, expressed as a robot program, was converted to HMM representations, and was used to recognize the most likely action that could be suggested to the user. Furthermore, we suggested a way to incorporate new observations to adapt the statistical model to previously unknown situations.

To obtain a comprehensive multi-modal interactive robot programming system, several elements still need to be added in the future. Although the programs generated by the current system can be re-executed, they are limited to fixed task sequences. To expand the generality of the paradigm, we need to add the ability to re-configure the task parameters interactively and define non-sequential flow structures such as conditional branching and looping. Lastly, the system performance needs to be quantitatively evaluated through user studies, to determine the benefits provided by multi-modal programming, interactivity, and intention interpretation.

6. Acknowledgements

This research was funded in part by DARPA under contract DAAD19-02-1-0389 and ABB under contract 1010068. Additional support was provided by the Robotics Institute at Carnegie Mellon University.

Appendix A: Merging Sampled Statistics without Prior Samples

The following is the derivation of the method used to merge p-dimensional sampled statistics of set X and set Y without using samples themselves. In other words, solve for $\underset{\sim}{\bar{Z}}$ and $S_Z{}^2$ from n_X, $\underset{\sim}{\bar{X}}$, $S_X{}^2$, n_Y, $\underset{\sim}{\bar{Y}}$, and $S_Y{}^2$ (without X and Y).

Sample Set	Size	Mean	Variance
$X = \left(\underset{\sim}{X}_1, \cdots, \underset{\sim}{X}_{n_X} \right)$	n_X	$\underset{\sim}{\bar{X}}$	$S_X{}^2$
$Y = \left(\underset{\sim}{Y}_1, \cdots, \underset{\sim}{Y}_{n_Y} \right)$	n_Y	$\underset{\sim}{\bar{Y}}$	$S_Y{}^2$
$Z = \left(\underset{\sim}{X}_1, \cdots, \underset{\sim}{X}_{n_X}, \underset{\sim}{Y}_1, \cdots, \underset{\sim}{Y}_{n_Y} \right)$	$n_Z = n_X + n_Y$	$\underset{\sim}{\bar{Z}}$	$S_Z{}^2$

Given samples:

$$\underset{(p \times n_X)}{X} = \left(\underset{\sim}{X}_1 \cdots \underset{\sim}{X}_{n_X} \right) = \begin{bmatrix} X_{11} & \cdots & X_{n_X 1} \\ \vdots & \ddots & \vdots \\ X_{1p} & \cdots & X_{n_X p} \end{bmatrix}$$

$$\underset{(p \times n_Y)}{Y} = \left(\underset{\sim}{Y}_1 \cdots \underset{\sim}{Y}_{n_Y} \right) = \begin{bmatrix} Y_{11} & \cdots & Y_{n_Y 1} \\ \vdots & \ddots & \vdots \\ Y_{1p} & \cdots & Y_{n_Y p} \end{bmatrix}$$

where $\underset{\sim}{X}_1 = \begin{bmatrix} X_{11} \\ \vdots \\ X_{1p} \end{bmatrix} \cdots \underset{\sim}{X}_{n_X} = \begin{bmatrix} X_{n_X 1} \\ \vdots \\ X_{n_X p} \end{bmatrix}$ and $\underset{\sim}{Y}_1 = \begin{bmatrix} Y_{11} \\ \vdots \\ Y_{1p} \end{bmatrix} \cdots \underset{\sim}{Y}_{n_Y} = \begin{bmatrix} Y_{n_Y 1} \\ \vdots \\ Y_{n_Y p} \end{bmatrix}$

Merged sample mean:

$$\underset{(p \times 1)}{\underset{\sim}{\bar{Z}}} = \frac{n_X \underset{\sim}{\bar{X}} + n_Y \underset{\sim}{\bar{Y}}}{n_Z}$$

where $\underset{(p \times 1)}{\underset{\sim}{\bar{X}}} = \begin{bmatrix} \bar{X}_1 \\ \vdots \\ \bar{X}_p \end{bmatrix}$ and $\underset{(p \times 1)}{\underset{\sim}{\bar{Y}}} = \begin{bmatrix} \bar{Y}_1 \\ \vdots \\ \bar{Y}_p \end{bmatrix}$

Merged sample variance:

$$\underset{(p \times p)}{S_Z{}^2} = \frac{1}{n_Z - 1} \left\{ (n_X - 1) S_X{}^2 + n_X (\underset{\sim}{\bar{X}} - \underset{\sim}{\bar{Z}})(\underset{\sim}{\bar{X}} - \underset{\sim}{\bar{Z}})^T \right.$$

$$+(n_Y-1)S_Y{}^2+n_Y(\underset{\sim}{\bar{Y}}-\underset{\sim}{\bar{Z}})(\underset{\sim}{\bar{Y}}-\underset{\sim}{\bar{Z}})^T\Big\}$$

given

$$\underset{(p\times p)}{S_X{}^2}=\frac{1}{n_X-1}W_XW_X{}^T$$

where $\underset{(p\times n_X)}{W_X}=X-\underset{\sim}{\bar{X}}\cdot\underset{\sim}{1}^T=\begin{bmatrix}X_{11}&\cdots&X_{n_X1}\\ \vdots&\ddots&\vdots\\ X_{1p}&\cdots&X_{n_Xp}\end{bmatrix}-\begin{bmatrix}\bar{X}_1&\cdots&\bar{X}_1\\ \vdots&\ddots&\vdots\\ \bar{X}_p&\cdots&\bar{X}_p\end{bmatrix}$

and

$$\underset{(p\times p)}{S_Y{}^2}=\frac{1}{n_Y-1}W_YW_Y{}^T$$

proof

$$\underset{(p\times p)}{S_Z{}^2}=\frac{1}{n_X+n_Y-1}W_ZW_Z{}^T$$

where $\underset{(p\times n_Z)}{W_Z}=\begin{bmatrix}X_{11}&\cdots&X_{n_X1}&Y_{11}&\cdots&Y_{n_Y1}\\ \vdots&\ddots&\vdots&\vdots&\ddots&\vdots\\ X_{1p}&\cdots&X_{n_Xp}&Y_{1p}&\cdots&Y_{n_Yp}\end{bmatrix}-\underbrace{\begin{bmatrix}\bar{Z}_1&\cdots&\bar{Z}_1\\ \vdots&\cdots&\vdots\\ \bar{Z}_p&\cdots&\bar{Z}_p\end{bmatrix}}_{n_Z\text{ columns}}$

$$=\frac{1}{n_Z-1}\left\{\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{X}_i-\underset{\sim}{\bar{Z}})(\underset{\sim}{X}_i-\underset{\sim}{\bar{Z}})^T\Big\}+\sum_{j=1}^{n_Y}\Big\{(\underset{\sim}{Y}_j-\underset{\sim}{\bar{Z}})(\underset{\sim}{Y}_j-\underset{\sim}{\bar{Z}})^T\Big\}\right\}$$

whose $\displaystyle\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{X}_i-\underset{\sim}{\bar{Z}})(\underset{\sim}{X}_i-\underset{\sim}{\bar{Z}})^T\Big\}$

$$=\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}}+\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}}+\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})^T\Big\}$$

$$=\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}})(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}})^T\Big\}+\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})(\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})^T\Big\}$$
$$+\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}})(\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})^T\Big\}+\sum_{i=1}^{n_X}\Big\{(\underset{\sim}{\bar{X}}-\underset{\sim}{\bar{Z}})(\underset{\sim}{X}_i-\underset{\sim}{\bar{X}})^T\Big\}$$

$$= \sum_{i=1}^{n_X} \left\{ (\underset{\sim}{X}_i - \bar{\underset{\sim}{X}})(\underset{\sim}{X}_i - \bar{\underset{\sim}{X}})^T \right\} + n_X (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T$$

$$+ \underbrace{\sum_{i=1}^{n_X} \left\{ (\underset{\sim}{X}_i - \bar{\underset{\sim}{X}}) \right\}}_{=0} (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T + (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}}) \underbrace{\sum_{i=1}^{n_X} (\underset{\sim}{X}_i - \bar{\underset{\sim}{X}})^T}_{=0}$$

$$= \sum_{i=1}^{n_X} \left\{ (\underset{\sim}{X}_i - \bar{\underset{\sim}{X}})(\underset{\sim}{X}_i - \bar{\underset{\sim}{X}})^T \right\} + n_X (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T$$

$$= W_X W_X{}^T + n_X (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T$$

$$= (n_X - 1) S_X{}^2 + n_X (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T$$

similarly,

$$\sum_{j=1}^{n_Y} \left\{ (\underset{\sim}{Y}_j - \bar{\underset{\sim}{Z}})(\underset{\sim}{Y}_j - \bar{\underset{\sim}{Z}})^T \right\} = (n_Y - 1) S_Y{}^2 + n_Y (\bar{\underset{\sim}{Y}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{Y}} - \bar{\underset{\sim}{Z}})^T$$

$$\therefore S_Z{}^2 = \frac{1}{n_Z - 1} \left\{ (n_X - 1) S_X{}^2 + n_X (\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{X}} - \bar{\underset{\sim}{Z}})^T \right.$$

$$\left. + (n_Y - 1) S_Y{}^2 + n_Y (\bar{\underset{\sim}{Y}} - \bar{\underset{\sim}{Z}})(\bar{\underset{\sim}{Y}} - \bar{\underset{\sim}{Z}})^T \right\}$$

References

[1] "CyberGlove Reference Manual." Virtual Technologies Inc., 1998.

[2] Agah, A. and Tanie, K., "Human-Machine Interaction Through an Intelligent User Interface Based on Contention Architecture," *IEEE International Workshop on Robot and Human Communication RO-MAN'96*, Tsukuba, Japan, pp. 537-42, 1996.

[3] Bahlmann, C. and Burkhardt, H., "Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition," *Sixth International Conference on Document Analysis and Recognition*, Seattle, WA, USA, pp. 406-11, 2001.

[4] Batavia, P. H. and Nourbakhsh, I., "Path planning for the Cye personal robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 15-20, 2000.

[5] Boehme, H. J., et al., "Neural architecture for gesture-based human-machine-interaction," *Gesture and Sign Language in Human-Computer Interaction*, Bielefeld, Germany, pp. 219-32, 1997.

[6] Fong, T., Conti, F., Grange, S., and Baur, C., "Novel Interfaces for Remote Driving: Gesture, Haptic and PDA," *SPIE Telemanipulator and Telepresence Technologies VII*, Boston, MA, 2000.

[7] Forsyth, D. and Ponce, J., "Computer Vision : A Modern Approach." Upper Saddle River, N.J.: Prentice Hall, 2003.

[8] Fujita, M. and Kitano, H., "Development of an autonomous quadruped robot for robot entertainment," *Autonomous Robots 5*, no. 1, pp. 7-18, 1998.

[9] Gertz, M., Stewart, D., and Khosla, P. K., "A software architecture-based human-machine interface for reconfigurable sensor-based control systems," *IEEE International Symposium on Intelligent Control*, Chicago, IL, USA, pp. 75-80, 1993.

[10] Ghidary, S. S., et al., "Multi-Modal Human Robot Interaction for Map Generation," *International Conference on Intelligent Robot and Systems*, Maui, Hawaii, USA, pp. 2246-51, 2001.

[11] Huang, X., et al., "The SPHINX-II speech recognition system: an overview," *Computer Speech and Language*, vol. 7, no. 2, pp. 137-48, 1993.

[12] Iba, S., Vande Weghe, J. M., Paredis, C. J. J., and Khosla, P. K., "An Architecture for Gesture-Based Control of Mobile Robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Korea, pp. 851-57, 1999.

[13] Ikeuchi, K. and Suehiro, T., "Toward an Assembly Plan from Observation, Part I: Task Recognition with Polyhedral Objects," *IEEE Transactions Robotics and Automation*, vol. 10, no. 3, pp. 368-85, 1994.

[14] Ishida, T., et al., "Motion entertainment by a small humanoid robot based on OPEN-R," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Piscataway, NJ, pp. 1079-86, 2001.

[15] Kang, S. B. and Ikeuchi, K., "Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 1, pp. 81-95, 1997.

[16] Kawamura, K., Alford, A., Hambuchen, K., and Wilkes, M., "Towards a Unified Framework for Human-Humanoid Interaction," *First IEEE-RAS International Conference on Humanoid Robots*, Boston, MA, 2000.

[17] Kimura, H., Horiuchi, T., and Ikeuchi, K., "Task-Model Based Human Robot Cooperation Using Vision," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Korea, pp. 701-06, 1999.
[18] Kortenkamp, D., Bonasso, R. P., and Subramanian, D., "Distributed, Autonomous Control of Space Habitats," *IEEE Aerospace Conference*, Piscataway, NJ, USA, pp. 2751-62, 2001.
[19] Kortenkamp, D., Huber, E., and Bonasso, R. P., "Recognizing and interpreting gestures on a mobile robot," *National Conference on Artificial Intelligence*, Portland, OR, USA, pp. 915-21, 1996.
[20] Koyama, T., "On Combining Sampled Statistics without Prior Samples," Personal Communication to Author, September 27, 2002.
[21] Kuno, Y., Murashima, T., Shimada, N., and Shirai, Y., "Interactive gesture interface for intelligent wheelchairs," *International Conference on Multimedia and Expo*, New York, NY, USA, pp. 789-92, 2000.
[22] Lee, C. and Xu., Y., "Online, Interactive Learning of Gestures for Human/Robot Interfaces," *IEEE International Conference on Robotics and Automation*, Minneapolis, MN, pp. 2982-87, 1996.
[23] Mardia, K. V., "Statistics of directional data." London and New York: Academic Press, 1972.
[24] Mascaro, S. and Asada, H. H., "Hand-in-glove human-machine interface and interactive control: task process modeling using dual Petri nets," *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 1289-95, 1998.
[25] Matsumoto, Y., Ino, T., and Ogasawara, T., "Development of Intelligent Wheelchair System with Face and Gaze Based Interface," *10th IEEE International Workshop on RObot and Human Commnunication (ROMAN 2001)*, Bordeaux-Paris, France, pp. 262-67, 2001.
[26] Matsumoto, Y. and Zelinsky, A., "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement," *Fourth International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, pp. 499-504, 2000.
[27] Morrow, J. D. and Khosla, P. K., "Manipulation task primitives for composing robot skills," *IEEE International Conference on Robotics and Automation*, Albuquerque, NM, USA, pp. 3354-9, 1997.
[28] Musser, G., "Robots That Suck," *Scientific American*, vol. 288, no. 2, pp. 84-6, 2003.
[29] Nishimura, T., Mukai, T., Nozaki, S., and Oka, R., "Adaptation to gesture performers by an on-line teaching system for spotting recognition of gestures from a time- varying image," *Transactions of the Institute of Electronics, Information and Communication Engineers D-II*, vol. J81D-II, no. 8, pp. 1822-30, 1998.
[30] Ogawara, K., et al., "Acquiring hand-action models in task and behavior levels by a learning robot through observing human demonstrations," *IEEE-RAS International Conference on Humanoid Robots*, Boston, 2000.
[31] Onda, H., et al., "A Telerobotics System using Planning Functions Based on Manipulation Skills and Teaching-by-Demonstration Technique in VR," *Journal of the Robotics Society of Japan*, vol. 18, no. 7, pp. 979-94, 2000.

[32] Oviatt, S., "Taming recognition errors with a multimodal interface," *Communications of the ACM*, vol. 43, no. 9, pp. 45-51, 2000.
[33] Perzanowski, D., et al., "Communicating with Teams of Cooperative Robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata*, A. C. Schultz and L. E. Parker, Eds. The Netherlands: Kluwer, 2002, pp. 185-93.
[34] Perzanowski, D., et al., "Building a multimodal human-robot interface," *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 16-21, 2001.
[35] Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-86, 1989.
[36] Rybski, P. E. and Voyles, R. M., "Interactive task training of a mobile robot through human gesture recognition," *IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, pp. 664-9, 1999.
[37] Skubic, M., Perzanowski, D., Schultz, A., and Adams, W., "Using Spatial Language in a Human-Robot Dialog," *International Conference on Robotics and Automation*, Washington, DC, pp. 4143-48, 2002.
[38] Stallman, R. M., "Emacs: The extensible, customizable, selfdocumenting display editor," in *Interactive programming environments*, D. R. Barstow, H. E. Shrobe, and E. Sandewall, Eds. New York: McGraw-Hill, 1984, pp. 300-25.
[39] Starner, T. and Pentland, A., "Real-time American Sign Language recognition from video," *IEEE International Symposium on Computer Vision*, pp. 265-70, 1995.
[40] Thrun, S., et al., "MINERVA: a second-generation museum tour-guide robot," *IEEE International Conference on Robotics and Automation*, Piscataway, NJ, USA, pp. 1999-2005, 1999.
[41] Todd, D. J., "Fundamentals of robot technology : an introduction to industrial robots, teleoperators, and robot vehicles." New York: Wiley, 1986.
[42] Voyles, R. M., Agah, A., Khosla, P. K., and Bekey, G. A., "Tropism-Based Cognition for the Interpretation of Context-Dependent Gestures," *IEEE International Conference on Robotics and Automation*, Albuquerque, NM, USA, pp. 3481-6, 1997.
[43] Voyles, R. M., Morrow, J. D., and Khosla, P. K., "Gesture-based programming for robotics: human-augmented software adaptation," *IEEE Intelligent Systems*, vol. 14, no. 6, pp. 22-31, 1999.
[44] Waldherr, S., Romero, R., and Thrun, S., "A gesture based interface for human-robot interaction," *Autonomous Robots*, vol. 9, no. 2, pp. 151-73, 2000.
[45] Wren, C. R., Clarkson, B. P., and Pentland, A. P., "Understanding purposeful human motion," *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 378 -83, 2000.
[46] Yamada, Y., Morizono, T., Umetani, Y., and Yamamoto, T., "Human error recovery for a human/robot parts conveyance system," *International Conf. on Robotics and Automation*, Washington, DC, USA, pp. 2004-9, 2002.
[47] Young, S. J., et al., "HTK: Hidden Markov Model Toolkit V3.0." Redmond, Washington, USA: Microsoft Corporation, 2000.