Interactive Multi-Modal Robot Programming

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

[1] The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[2] Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[3] Systems Realization Laboratory, G. W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

Abstract. The goal of the Interactive Multi-Modal Robot Programming system is a comprehensive human-machine interface that allows non-experts to compose robot programs conveniently. Two key characteristics of this novel programming approach are that the user can provide feedback interactively at any time through an intuitive interface and that the system infers the user's intent to support interaction. The framework takes a three-step approach to the problem: multi-modal recognition, intention interpretation, and prioritized task execution. The system is demonstrated by interactively controlling and programming a mobile vacuum cleaning robot. The demonstrations are used to exemplify the interactive programming and plan recognition aspects of the research.

1 Introduction

The number of vacuum cleaning robots being manufactured and sold to household consumers is rising as they become increasingly popular as technical gadgets [1]. Depending on the price range, these robots can navigate, avoid obstacles, localize themselves, and cover an area autonomously or through manual control by the user. However, none of them offers a novice-friendly interface to control and program a robot for particular tasks. 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 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, intention modeling and 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 users and help them in dealing with a robot by including a human in the control loop.

Designing and building such a system involves multiple problems. The system needs to infer underlying robot commands from a sequence of multi-modal user inputs to formulate a robot program. The system also needs to allow preemptive interaction between the user and the robot, which involves suspension, arbitration,

and resumption of a robot task. Furthermore, it is desirable to reduce user interaction with a graphical user interface in order to allow direct interaction between the user and the robot.

There have been many approaches to improving task automation. The current state-of-the-art in user-friendly task automation is based on iconic programming [2] and/or programming by human demonstration [3]. The goal of these paradigms is to translate the burden of programming robotic systems from robot experts to task experts. The problem arises in these approaches when the programmed system is diverted from its intended task, and the user is required to re-execute, or even worse, to reprogram the task all over again. Our approach adds an intuitive multi-modal interface and interactive programming and execution capability so that the user can take over and correct the system with ease. The use of multi-modal interfaces in human-robot interaction is an active field with numerous applications such as teleoperation [4] and navigation [5]. We use voice and hand gestures to convey a combination of symbolic and parametric information to the robot.

2 System Design

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. The second module

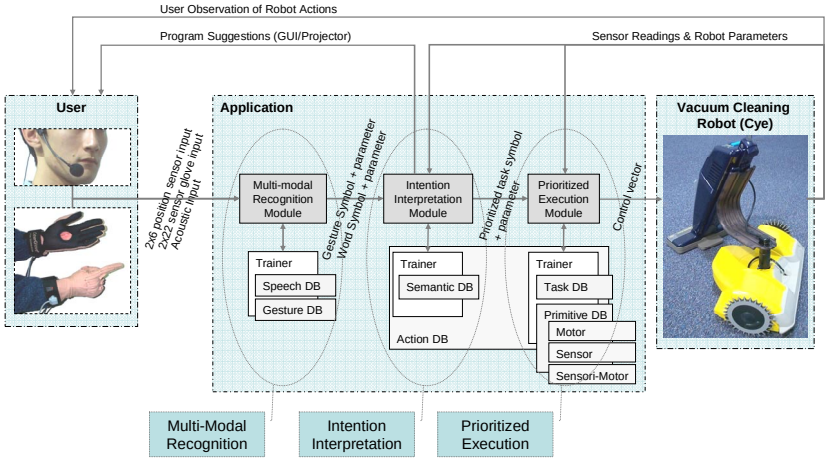

Figure 1. The framework is roughly divided into three modules: multi-modal recognition, intention interpretation, and prioritized execution, and each roughly corresponds to providing an intuitive interface, robot programming and suggestion, and an interactive control capability of the overall system.

(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 vacuum cleaning robot used in our experiments is Cye, a 10" by 16" two-wheeled robot which carries a portable vacuum cleaner on its tail. It uses dead reckoning to localize itself with respect to the starting position, and is subject to cumulative error as it navigates. The robot comes with a graphical user interface that allows users to control Cye using a mouse, and to program it using an iconic programming framework. We added the three modules on top of the graphical user interface to provide features such as: (1) A hand gesture and spontaneous speech recognition interface; (2) A robot simulator for users to control and execute a program in the virtual environment; (3) Capability to preemptively interrupt a set of instructions; (4) Capability to create and adjust the program on-the-fly; (5) Capability to suggest a program that the user may want to execute based on the partial sequence of robot trajectory.

3 Multi-Modal Recognition

The system is capable of recognizing two different modalities: hand-gestures and spontaneous speech. The primary motivation for multi-modality is that no single mode provides a highly competent human-robot interface. Verbal cues are most appropriate when either party needs to convey symbolic information with an unambiguous context, such as "stop", "move forward", "turn right", etc. Motion cues are an essential supplement when deictic elements are involved, as in the verbal commands "go there" and "move this way". Such instructions are ambiguous without accompanying gestures, which are inherently more suitable to express position and geometry.

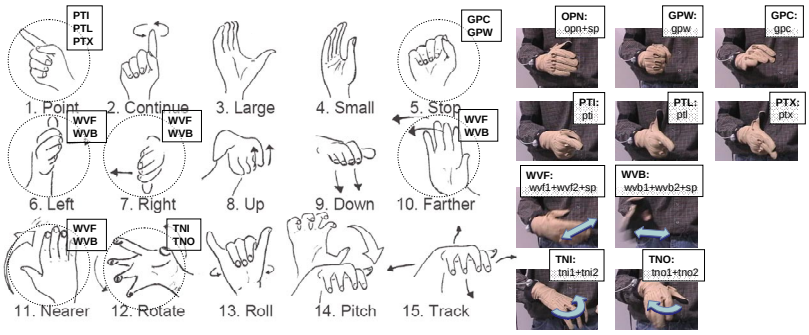

Figure 2. Left (drawings): The list of gesture vocabulary by Quek [6]. Gestures used for our system are circled with the corresponding three-letter gesture symbols (in upper-case). Right (pictures): Gesture symbols recognized by our HTK-based gesture recognition module, along with the gesture phonemes (in lower-case).

The list of hand gestures was selected based on work by Quek [6], who compiled the list of gestures necessary to express spatial information in 3D space (Figure 2). The gesture recognition module is implemented using the Hidden Markov Model Toolkit (HTK) [7] customized to recognize gestures at 60Hz from temporal data streams from two 22-sensor CyberGloves [8], each with a Polhemus 6DOF inductive position sensor. Using HTK, which was primarily developed for speech recognition research, we were able to treat hand gestures as words, and a sequence of hand gestures as a sentence. HTK offers versatile tools and the capability to build HMMs for recognition purposes. Using model adaptation techniques and triphone modeling (e.g.: *gpw-tno1+tno2*) strategy to capture inter-word transitions as well as intra-word transitions provided by the HTK, the gesture recognition module is able to achieve an average accuracy of 92% for previously unseen users.

The spontaneous speech recognition is implemented on a public domain large-volume speech recognition engine. In our implementation, spontaneous speech is translated into words using Microsoft Speech SDK [9], an off-the-shelf speech recognition package. It is responsible for recognition, adaptation, and grammatical parsing of the spoken words. The package's TTS (Text-To-Speech) capability is used for recognition acknowledgments and making user suggestions, as described in Section 5. List of a basic speech vocabulary consists of motion, deictic, name, attribute and programmable command terms [10]. The choice of words is task-dependent and experience-based. Readers interested in vocabulary selection for speech-based robot interactions should refer to [11], which deals with a vocabulary-based human-robot instruction system.

The temporal streams of results from hand gesture recognition and spontaneous speech recognition are combined to generate a semantically correct interpretation to control and program the robot. Results from both recognition processes are streamed into a buffer that gives a one-second window to decrease ambiguity in the speech recognition result (e.g. "this", "that") by grounding proper parameters from the gesture recognition result. Interpretation module uses the semantics database, which is described in [10] to interpret multi-modal recognition results to intended robot task symbols. It is implemented as a lookup table of candidate task symbols and their priorities from input symbols from the multi-modal recognition module.

4 Interactive Robot Control and Sequential Programming

Preemptive execution is crucial in providing the user real-time interaction to control and program a robot on-the-fly. Tasks are prioritized according to a pre-defined rule, and sequential robot actions in the tasks are executed and sometimes overridden based on the arbitration policy. This allows the user to handle situations such as making an emergency stop or avoiding an obstacle during the execution of other tasks. A robot program (task) is stored during interactive robot

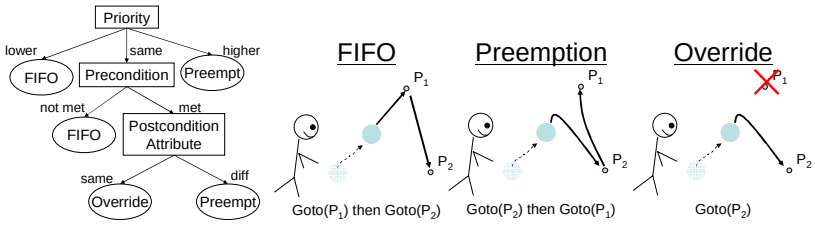

Figure 3. Arbitration Policy Tree and the possible actions for the sequence of instructions Goto(P_1) followed by Goto(P_2). Based on the policy tree, *Override* will be the correct action for the given example.

control. The user sets the module to a learning mode and executes primitives sequentially; the system remembers the sequence as a task.

An illustrative example of task arbitration is given in Figure 3. Imagine that the user first points to a particular position, P_1, and asks the robot to "go there", then points to a different position, P_2, and asks the robot to "go there" while the robot is moving toward P_1. There are three potential outcomes to the above sequential instructions: FIFO, Preemption, and Override. The outcome is decided by the arbitration policy tree described in Figure 3 by using features such as priority, precondition, and post-condition attribute. For the given example with two consecutive "go there" instructions, the outcome is an Override, since they have no preconditions and their priorities (given by the Semantic Database) and post-condition attributes (position) are the same. On the other hand, if the second instruction were "go this way" for a given direction, the outcome would be Preemption, since the post-condition attributes are different (position vs. direction).

To generate a sequential robot program, the user should execute the instructions in the intended order. When an action is overridden during the programming phase, the action up until the point of the override is programmed. The user can edit the program by interrupting its execution and showing an alternative action. In order to compose a non-sequential program which contains loops or if-then statements, the user may edit the program on the iconic programming interface, which is more suitable to display and edit the program flow.

5 Intention Awareness

The system's intention awareness is composed of two capabilities: intention recognition and adaptation. 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 the user may want to execute based on an incomplete sequence of instructions executed by the user. The recognition ability is similar to the auto-completion ability in a text-editing program. It is especially helpful when there is 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, we represent tasks in a probabilistic

framework rather than as a discrete sequence of commands. A Hidden Markov Model (HMM) provides a way to model the task in a probabilistic framework, where both state transitions and observations can be expressed stochastically. Sets of 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.

For each program, a continuous-density HMM representation is generated from a discrete sequence of actions and observations collected during the programming phase. All program HMMs are integrated into a single HMM network, λ_{net}, which is then used to recognize the user's intended program based on the new observation sequence (Figure 4). When a robot is programmed interactively, the system collects an observation sequence $O_n = \{o_{0,n}\, o_{1,n} \ldots\, o_{t,n}\}$ for program action n, where $o_t = \{x_t, y_t, \theta_t\}$ corresponds to the robot position at time t. The sequence O is the collection of observations O_n resulting from program actions *1* to *N*. The robot program is then converted into an HMM by assigning one action per state, and its transition and observation probabilities are calculated from $O_{n=1..N}$. HMMs

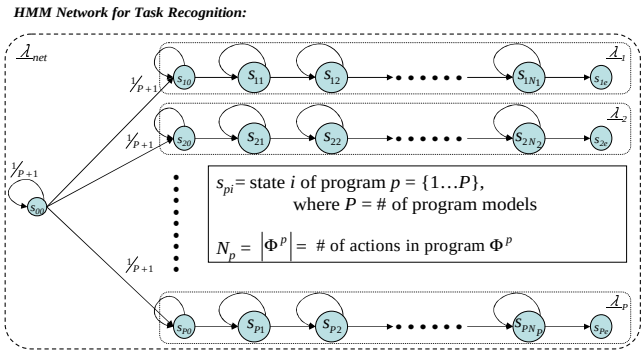

Figure 4. HMM network used for task recognition

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 5. Viterbi Algorithm with Dynamic Garbage Collection

describing different programs are connected in a network forming a single HMM network, λ_{net}

During recognition, the current observation sequence is evaluated and compared to the HMM network. To find the single most likely state out of all states in the shared HMM network for the current observation sequence, we use a modified Viterbi Algorithm described in Figure 5. The Viterbi algorithm, based on the Token Passing paradigm [12], has been modified by adding dynamic garbage collection, that is, recognizing the state s_{00} of the HMM network in which none of the programs is being executed. It is dynamic, in the sense that the state needs no prior training and the evaluation of whether or not the observation sequence is garbage depends entirely on the rest of the HMMs. A model update of each HMM provides online seamless adjustments of the statistics that describe the robot program. The update is performed after execution of the corresponding action associated with the state. The update does not require the entire sequence of previous observations, but rather it is updated using the previous statistics and the most recent observation sequence.

6 Demonstrations

We have conducted two demonstrations to exemplify the interactive programming and plan recognition aspects of the research. The first demonstration is to verify the operation of the 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 having the intention model adapt to the current observations.

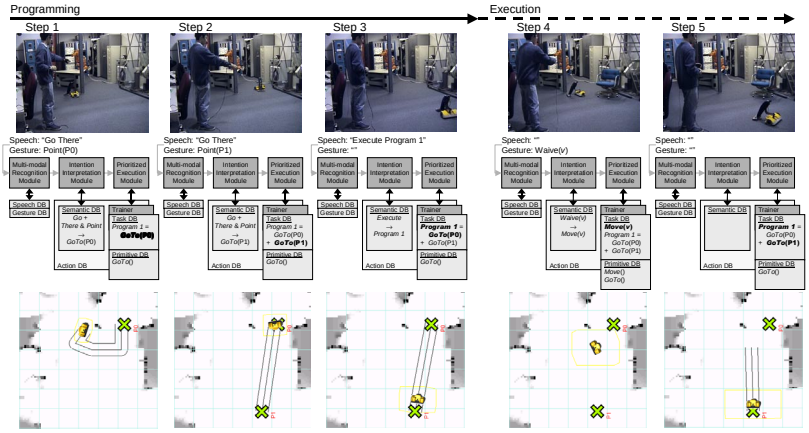

Figure 6: Task level sequential robot programming demonstration

6.1 Task Level Sequential Robot Programming

We have considered an interactive programming scenario, which has a user register numerous via-points to which the robot should navigate using its path planning capability. The robot can accept the user's preemptive speech and hand gesture commands to deal with unforeseen events. Figure 6 illustrates the sequences of the scenario including a sequence of camera snapshots with the corresponding conceptual illustrations of the framework, and the cropped images of the GUI.

In this 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 (step 1). 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. In step 2, the user enters another Goto primitive, but 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 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 "Wave" 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 waving, the robot returns to the execution of "Program One" (step 5).

6.2 Making Suggestions based on the Intention Awareness

This 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(On), AreaCoverage(P_2, P_3), Vacuum(Off), GoHome()}

Φ^2 = {Vacuum(On), 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). Their trajectories and the combined probability distributions of the converted HMMs are shown in the first and second columns in Figure 7. The last three columns in Figure 7 describe the output of the task recognition results from three test observation sequences τ^1, τ^2, and τ^3. Scores in the figure show that for the first test observation sequence, τ^1, the most probable state sequence follows that of the first program, Φ^1. This makes sense, because the trajectory of τ^1 was generated from Φ^1. The second observation sequence, τ^2, was generated by traversing regions covered by both the second and third programs, Φ^2 and Φ^3. The score in the figure also shows that is indeed the case. The third test observation sequence, τ^3, is a random traversal, which is captured by the dynamic garbage collection state, s_{00}, as "non-program".

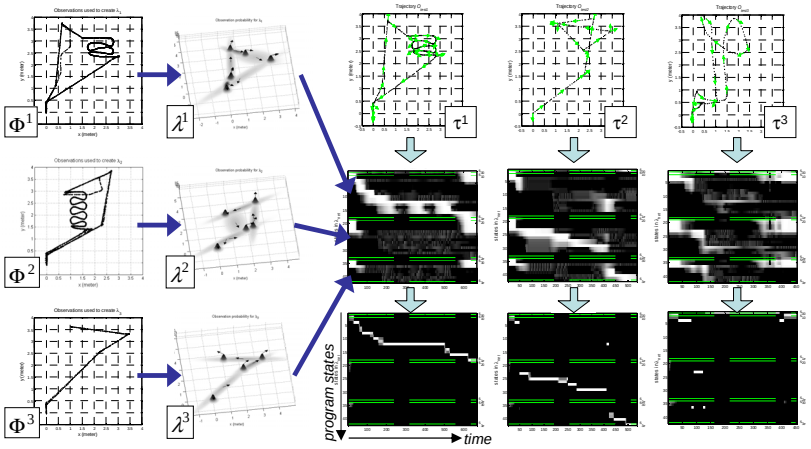

Figure 7. Task recognition results on three test observation sequences τ^1, τ^2, and τ^3. The images describe the state likelihood at any given time.

In each case, suggestions to the user are made through the text-to-speech system and the GUI projected on the wall. When the algorithm determines that there is a most probable state other than the dynamic garbage collection state, the application announces to the user through the text-to-speech system that there is a suggestion to be made. The suggested program is then executed on the simulated robot starting from the instruction associated with the most probable state. Based on the suggestion displayed by the simulated program, the user may or may not choose to execute the suggested program. It is important to note that the user is free to stop or alter the suggested program at any time, so the suggestion does not need to be a perfect match.

7 Conclusion and Future Work

In this paper, we have described an 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 an intuitive interface allows the captured intent to be closer to the user's true intent.

The first demonstration verified interactive multi-modal programming and execution, 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.

To attain 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 define non-sequential flow structures such as conditional branching and looping. Also needed is the ability to learn new primitives from demonstrations. The current implementation assumes that the given primitives cover the entire task space, and it would be convenient to be able to use the current programming paradigm to create new primitives. Discerning user preferences is also an important issue. Currently all programs in the HMM network are equally preferred prior to incoming observations.

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.

References

[1] Musser, G. (2003). "Robots That Suck." Scientific American, 288(2), 84-6.
[2] Gertz, M. W., Stewart, D. B., and Khosla, P. K. (1994). "A human machine interface for distributed virtual laboratories." IEEE Robotics & Automation Magazine, 1(4), 5-13.
[3] Ikeuchi, K., and Suehiro, T. (1994). "Toward an Assembly Plan from Observation, Part I: Task Recognition with Polyhedral Objects." IEEE Transactions Robotics and Automation, 10(3), 368-385.
[4] Fong, T., Conti, F., Grange, S., and Baur, C. (2000). "Novel Interfaces for Remote Driving: Gesture, Haptic and PDA." SPIE Telemanipulator and Telepresence Technologies VII, Boston, MA.
[5] Perzanowski, D., Schultz, A. C., Adams, W., Marsh, E., and Bugajska, M. (2001). "Building a multimodal human-robot interface." IEEE Intelligent Systems, 16(1), 16-21.
[6] Quek, F. (1994). "Toward a Vision-Based Hand Gesture Interface." Virtual Reality System Technology Conference, Singapore, 17-29.
[7] Young, S. J., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. (2000). HTK: Hidden Markov Model Toolkit V3.0, Microsoft Corporation, Redmond, Washington, USA.
[8] CyberGlove Reference Manual (1998). Virtual Technologies Inc., Palo Alto, CA.
[9] Microsoft Speech SDK ver. 5.1. (http://www.microsoft.com/speech/dev/)
[10] Iba, S., Paredis, C. J. J., and Khosla, P. K. (2002). "Interactive Multi-Modal Robot Programming." International Conf. on Robotics and Automations, Washington, D.C., 161-168.
[11] Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., and Klein, A. (2001). "Training personal robots using natural language instruction." IEEE Intelligent Systems, 16(5), 38-45.
[12] Young, S. J., Russell, N. H., and Thornton, J. H. S. (1989). "Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems." Cambridge University Engineering Dept.