A Multi-Agent System for Programming Robotic Agents by Human Demonstration

Richard M. Voyles

Dept. of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
voyles@cs.umn.edu

Pradeep K. Khosla

Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213
pkk@ices.cmu.edu

Abstract

This paper explores Gesture-Based Programming as a paradigm for programming robotic agents. Gesture-Based Programming is a form of programming by human demonstration that focuses the development of robotic systems on task experts rather than programming experts. The technique relies on the existence of previously acquired robotic skills (which are called "sensorimotor primitives") which are approximately the robotic equivalent of everyday human experiences. The interpretation of the human's demonstration and subsequent matching to robotic primitives is a qualitative problem that we approach with a community of skilled agents. A simple manipulative task is programmed to demonstrate the system.

Introduction

Historically, machine loading, welding and spray painting have claimed the largest installed bases of robots in the manufacturing sector. While applications such as assembly and deburring have been gaining favor, they still don't come close to the proportionate use of the former application areas, even within narrow specializations. In these more complex domains, there are still important issues of suitability; both of the robot and of its development.

Industrial robots are inherently position-controlled devices and are well-suited to position-based applications such as machine loading, welding and spray painting. Deburring and assembly, on the other hand, are contact intensive and not well-suited to position-based control schemes in which small errors in position cause large forces to be exerted. But there is another, more subtle suitability issue: skill transfer. Deburring and assembly require some non-trivial level of skill. While the skill level required can be minimized with clever fixtures and tools, *some* level of skilled execution is required. Because "skill" is difficult to quantify, it is also difficult to transfer and, hence, taxes the development of the application. In general, the more skill transfer involved, the less suited the application is to the traditional text-based development paradigm.

So, why have machine loading, welding and spray painting become popular robotic applications? One reason, we claim, is the relative ease with which the applications are developed. Spot welding and machine loading, in particular, require no real skill transfer and are generally "programmed" by teaching a series of points. These points can be taught and re-taught by semi-skilled workers on the factory floor (*task experts*) in response to changing conditions, rather than by development engineers (*programming experts*). Although much text-based programming may be required in installing and setting up the application, the fine-tuning and re-programming uses a completely different and more natural paradigm, which strongly contributes to the popularity of these applications.

Spray painting is different because a certain degree of skill is required to do a good job. The amount of overspray at the ends of a pass, the degree of overlap between passes, and proper attention to contours all impact finish quality. To address this, spray painting applications employ a different programming paradigm called *lead-through teaching* (Todd 1986). Even more natural for humans than point teaching, lead-through teaching allows a human task expert, with no traditional programming expertise, to program the trajectories by actually painting a sample panel.

Lead-through teaching is a primitive form of *programming by human demonstration* that includes the requisite skill transfer for appropriately simple tasks. In the case of spray painting, the skill involved is encoded entirely by the kinematics of motion and, hence, is transferred via lead-through teaching. In general, this is not the case. The types of skills that humans acquire in their daily interaction with the world are mainly contact-based: mating surfaces, turning a hinge, or inserting a screw, for example. Still, we argue that programming by demonstration is the most natural paradigm for human programmers because training by demonstration and practice is the most often used method between humans (Patrick 1992). If we assume the robot already possesses the underlying skills required -- as we assume when training another human -- programming by demonstration becomes a task of identifying the component skills in a task and appropriately parametrizing those identified skills.

We call our approach to programming by human demonstration "Gesture-Based Programming" (GBP) (Voyles 1997). We assume a set of robotic skills (which we call *sensorimotor primitives* (Morrow and Khosla 1995)) that approximate the set of primitives that a human uses to perform a task from the particular application domain. The demonstration consists of a series of "gestures" that only qualitatively hint at the underlying primitives. The job of the GBP system is to interpret these qualitative gestures and map them onto the executable primitives the robot already "knows." Because this cannot be modeled analytically, we employ a community of intelligent agents to interpret the demonstration.

Prior Work

This approach leverages the work of Kang and Ikeuchi on robot instruction by human demonstration (Kang and Ikeuchi 1996) but is distinct from it in several ways. Their work was not skill-based and the output of their system was not an executable program. Instead, they generated kinematic trajectories that the robot followed open-loop. Also, they employed a monolithic rather than a multi-agent approach to interpret the demonstration.

There are more similarities with "Learning by Watching" (Kuniyoshi, Inaba, Inoue 1994). They attempted greater abstraction of the task being learned, but still focused on kinematic relationships of the objects in the scene (including the demonstrator's hand). Much of their work was on real-time visual detection and tracking of relevant blobs in image sequences.

The idea of primitive-based control applied to the Utah/MIT hand was pioneered by Speeter (Speeter 1991). He developed a more diverse set of primitives than we present here that included a utility to combine and modify primitives to create task-specific actions. However, his primitives were purely kinematically-based and employed no sensor feedback so they cannot be considered "sensorimotor primitives".

Intelligent Agents

Wooldridge and Jennings (1995) define "agents" in two ways. Their first notion of agency is a "weak" definition with which few disagree. It specifies that an agent must possess the properties of *autonomy*, *social ability*, *reactivity*, and *proactiveness* as necessary attributes. Unfortunately, this weak notion barely differentiates an agent from a subroutine. Their "strong" definition adds to the above the *mentalistic* notions of *knowledge*, *belief*, *intention*, etc. Their argument is that it is appropriate to ascribe mentalistic notions to an artificial agent when these notions help us understand its behavior.

We subscribe to this stronger notion of agents, and add that "non-trivial" behavior and the ability to influence agents other than its own parent are necessary attributes, as well. Furthermore, because we feel the term "agent" becomes a meaningless wild card in the absence of a definition, we will present the attributes we feel are important.

Input/output relationship. Agents must do something within their world, whether that world is real or simulated. They must also be responsive to that world or, at least parametrizable. Our model of port-based objects (Stewart and Khosla, 1996) allows agents to possess both "input/output ports" and "resource ports." Input/output ports are considered variables during run-time, while resource ports are considered variables during initialization. In either case the ports are inputs and outputs to which the agent responds or effects its world. The key difference is the dynamism during run-time. This is similar to reactivity and social ability.

Autonomy. Agents must be able to function on their own. It is not necessary that they be able to fully achieve their goals on their own nor must they be able to survive on their own. Instead, teams of agents may be needed for certain tasks, but each agent must have some level of autonomy in processing its inputs and outputs. This is slightly different than that of Wooldridge and Jennings, see below.

Persistence. There is a need to distinguish a subroutine from a child process. To this point, we can not rule out a subroutine because it can have resource ports (arguments and return value) and input/output ports and is autonomous for the duration of the call. Of course, a subroutine is wholly dependent on the main execution thread so, at best, it can be considered a *piecewise autonomous agent*. Yet, persistence is the key idea that differentiates a child process from a subroutine. Our combination of autonomy and persistence is similar to autonomy of Wooldridge and Jennings.

Non-parental influence. To be truly independent, an agent must be able to influence agents other than its parent. This helps distinguish levels of decomposition but does not exclude hierarchies in which a collection of agents can be considered an agent in its own right. This property also requires that the environment be considered an agent.

Cognitive behavior. Cognitive behavior, or, perhaps more appropriately, *nonlinear behavior*, is the most controversial property because it seems to be the most arbitrary. Nonetheless, we feel a need to exclude such trivial things as parametrizable constants. In essence, agents should possess some non-trivial behavior, but it is difficult to quantify or define "non-trivial."

Agents for Gesture Interpretation

This definition of agents is motivated by our need to solve a complex qualitative problem, namely interpreting the demonstration of a task. Because of its qualitative nature, we found it helpful to describe the problem and its approach to solution in mentalistic terms. But before we describe the multi-agent architecture for solving the problem, we'll briefly define gestures.

Gestures, within the scope of this work, are defined as "imprecise events that convey the intentions of the gesturer." While this is a broad definition that encompasses many gestural modalities, we are interested in a small subset of four manual and articulatory gestures for GBP. These include *symbolic gestures*, such as the pre-shape of a dextrous grasp, *motion gestures*, such as transporting or manipulating an object, *tactile gestures*, such as fingertip contact force, and *articulatory gestures*, which are utterances such as "oops!' when an error is made or geometric model-based information such as "tighten the *screw*." (The whole phrase is irrelevant; only the keyword "screw" is important.) Other gestural modes exist but this is the subset we consider for our GBP system. Because gestures are imprecise events, they are context dependent; in isolation they convey little information. Yet, when combined with the state of the system and the environment at the time the gesture occurred, perhaps augmented with some past history of gestures, the intention becomes interpretable. The key point is the interpretation of non-symbolic gestures develops over a series of gestures rather than instantaneously. See for more information and examples.

The architecture we use is a variant of the Tropism System Cognitive Architecture (Agah and Bekey 1996) that is described in (Voyles et al 1997) and illustrated in Figure 1. In the illustration, "GRA" stands for Gesture Recognition Agent and "GIA" stands for Gesture Interpretation Agent. In a nutshell, cognition is the result of matching *tropisms* -- the likes and dislikes of the agents involved.

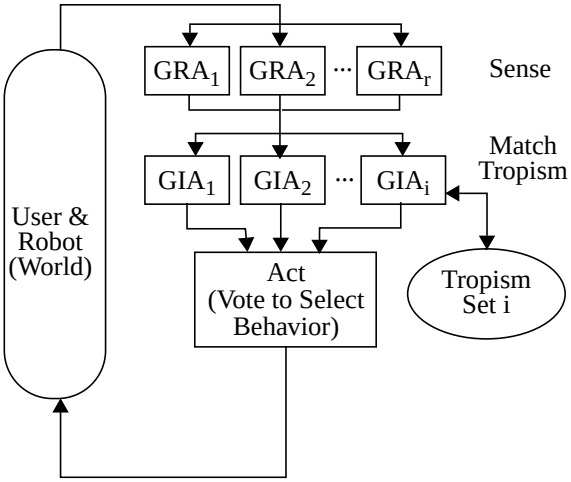

Figure 1: Static Fine-Grained Tropism System Cognitive Architecture as implemented for gesture recognition.

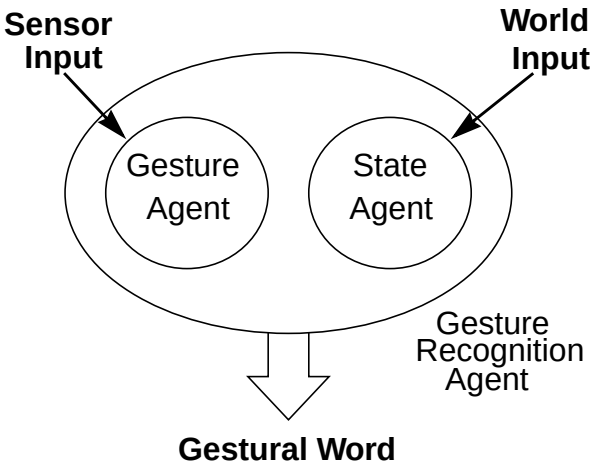

Figure 2: Structure of the Gesture Recognition Agent

Gesture Recognition Agents

Because gestures are context dependent, state information must be associated with each gesture. This state information is generally application specific. We modularize the gesture recognition agents by separating the application-specific state specification agent from the raw gesture agent.

Gesture recognition is handled by individual GRAs for each gestural mode. Each recognition agent is made up of two subordinate agents, one general, one domain-specific. The general agent is the "gesture agent" of Figure 2. It recognizes the instance of a gesture and the type of gesture within the gesture class (mode). The domain specific agent (the "state agent") grabs the relevant state information associated with the gesture type for the given domain and appends it to the type specifier.

We have implemented GRAs for segmenting CyberGlove data into symbolic gestures, Polhemus data into hand motion gestures, and fingertip sensor (Voyles, Fedder, Khosla 1996) data into manipulation primitives.

Gesture Interpretation Agents

The Gesture Interpretation Agents are task specific and must be described within the context of a particular application. We are interested in GBP for simple manipulation tasks (with the hope of extending to more complex assembly tasks in the future).

For training demonstrations, trajectory extractors are re-

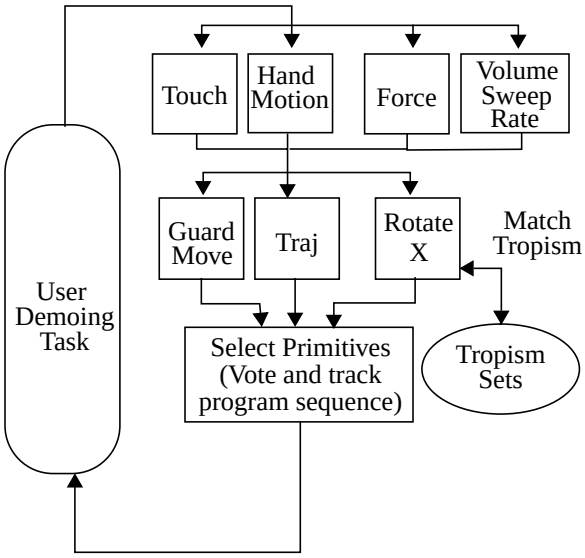

Figure 3: GBP Network for Simple Manipulation Tasks.

quired to infer straight-line motions and select via points based on hand motion gestures. Also, each pre-taught primitive requires a corresponding interpretation agent to identify it during the task

Demonstrations of a Peg-in-Hole Task

To illustrate the intended use of the system, we programmed a low-tolerance peg-in-hole task by demonstration. In fact, the peg is in the shape of a truncated cone, providing a long and forgiving chamfer with large initial clearance. As a result, the task reduces to a combination of kinematic motions and force accommodation. The set of primitives includes straight-line motion (robot only), guarded moves in several axes, joint-interpolated motion (robot and hand) and force-controlled gripping. For adequate sensing during the demonstration, the demonstrator must wear a CyberGlove and Polhemus with special force-sensing fingertips to observe the tactile, symbolic, and motion gestures. A simplified diagram of the agent network is shown in Figure 3, drawn in a manner consistent with Figure 1.

The "world" consists of the demonstrator and the task. The "touch" gesture recognizer looks for signals from the array tactile sensor that are consistent with gripping an object. This gesture is parametrized based on the integrated force of the grasp. The "hand motion" recognizers look for approximately straight-line motions and also gross sweeping motions. The "force" recognizer passes on interesting signatures from the fingertip intrinsic force sensor translated into a pre-determined frame. Finally, the "volume sweep rate" (Kang and Ikeuchi 1996) is a measure of the volume the fingertips are tracing out in space and is useful in finding breakpoints in natural motions of a human's hand.

The gesture recognition agents are effectively preprocessors of the sensor data. They pass their information on to the gesture interpretation agents that understand more about the robotic system and potential tasks. Specifically, they individually possess encodings of each of the primitives that have been previously taught to the robot. (The learning process is described in (Voyles, Morrow, Khosla 1997).)

The primitive-based interpretation agents are more algorithmic than other types of tropism-based agents. Rather than consisting of heuristic rules that model the agent's behavior, primitive-based tropism sets consist of a group of eigenvectors that represent the acquired primitive. These are matched using the dot product as a measure of parallelism. "Guard Move" represents the guarded move primitive while "Rotate X" represents a similar primitive that accommodates torques around the x-axis.

The "traj" agent extracts trajectory segments from the hand motion gestures. All agents make use of the volume sweep rate gestures as a cue for the particular phase of action: pre-grasp, grasp, or manipulation.

The GBP system is shown during the demonstration phase of a low-tolerance peg-in-hole insertion in Figure 4. First, the hole is grasped, transported, and placed on the table. Then the peg is grasped, transported, and inserted into the hole. In both cases of placing t he hole and the peg, a guarded move is used. In effect, the hole is pressed onto the table (as opposed to being dropped) and the peg is pressed into the hole. Contact force is used as the terminating condition rather than reaching a location in space.

To abstract the task a multi-agent network, based on the fine-grained tropism system cognitive architecture interprets the demonstrator's "intention" during the various task phases. (Although the agents are designed to execute in real time, in fact, some of the task abstraction processing was performed off-line for this implementation.)

The program that results is displayed in Figure 5 as a finite state machine (FSM) using Morrow's Skill Programming Interface (SPI) (Morrow 1997) (The output of the multi-agent network is automatically post-processed and formatted for display by the SPI as a visual check of program accuracy.) Each bubble in the FSM represents a node that consists of several agents executing in parallel to achieve the desired action.

The autonomous execution of the above program on a PUMA robot with a Utah/MIT hand is shown in Figure 6.

Discussion

Demonstration is a natural and intuitive method for human-to-human training of manipulative tasks and has niches of success in human-to-robot training. We feel it is appropriate

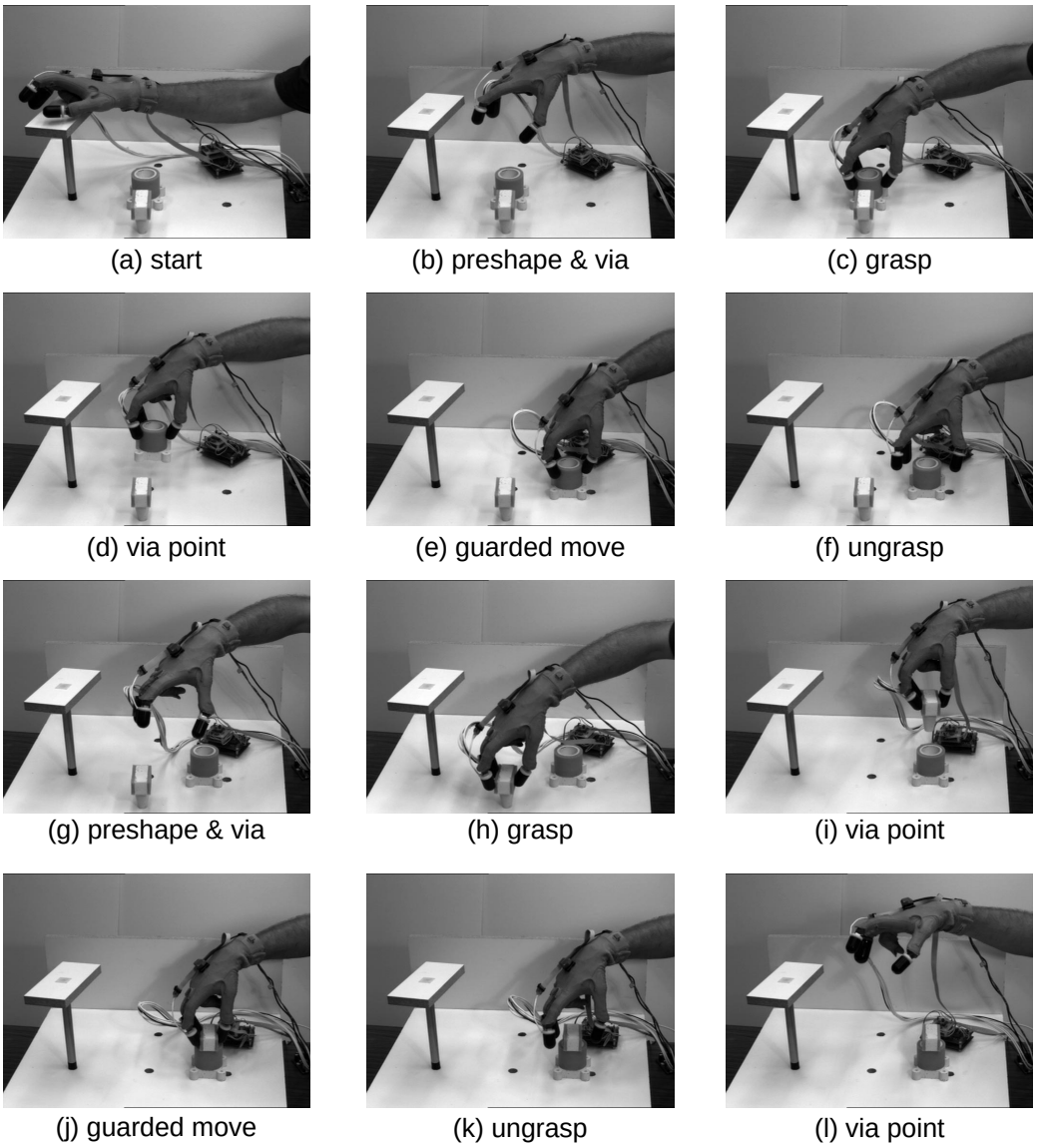

Figure 4: Demonstration of the Low-Tolerance Peg-in-Hole Task.

to extend this training paradigm to programming robots for contact-intensive tasks. We have argued for and briefly outlined a skill-based approach that assumes an existing set of "sensorimotor primitives" that defines both a domain of application and a domain of understanding. The domain of application is the range of tasks that is achievable given the primitives while the domain of understanding is the range of tasks that is readily interpretable given the primitives.

One would like the domain of understanding to be equivalent to the domain of application, but we have not yet begun to investigate this. It remains very much an open problem to define what the domains of application and understanding are given a set of primitives. It is even harder to define a set of primitives sufficient for a specific domain.

The domain of application of the set of primitives we have implemented for the peg-in-hole demonstration is quite small and does not approach that required even for a few simple tasks in a real manufacturing environment. The set was also selected mostly ad hoc. We have explored learning approaches to extract the primitives automatically (Voyles, Morrow, Khosla 1997) but considerable foresight is still required on the part of the demonstrator.

Despite these obstacles, we remain convinced of the usefulness of the paradigm. We successfully applied GBP to a

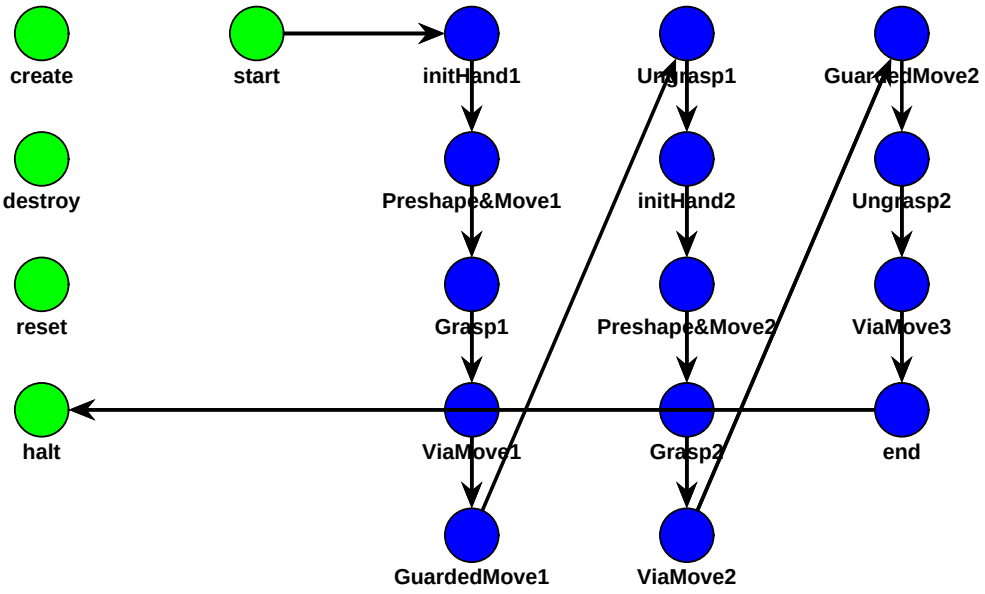

Figure 5: SPI Graphical display of program resulting from Peg-in-Hole demonstration.

contact-based task. (Though, perhaps, not contact *intensive*.) It is certainly more intuitive for the programmer and the success rate for the interpretation was acceptably high for a prototype system.

Acknowledgments

Dr. Voyles was supported during the bulk of this work in part by a National Defense Science and Engineering Graduate Fellowship and a DOE Integrated Manufacturing Predoctoral Fellowship.

References

Agah, A, and G.A. Bekey, 1996, "Phylogenetic and Ontogenetic Learning in a Colony of Interacting Robots," *Autonomous Robots*.

Kang, S.B., and K. Ikeuchi, 1996, "Toward Automatic Robot Instruction from Perception -- Temporal Segmentation of Tasks from Human Hand Motion," in *IEEE Transactions on Robotics and Automation*, v. 11, n. 5, October, pp. 670-681.

Morrow, J.D. and P.K. Khosla, 1995, "Sensorimotor Primitives for Robotic Assembly Skills," in *Proceedings of the IEEE International Conference on Robotics and Automation*, v. 2, May, pp.1894-1899.

Morrow, J.D., 1997, *Sensorimotor Primitives for Programming Robotic Assembly Skills*, Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, May.

Patrick, J., 1992, *Training: Research and Practice*, Academic Press, San Diego, CA

Speeter, T.H., 1991, "Primitive Based Control of the Utah/MIT Dextrous Hand," *Proceedings of the IEEE International Conference on Robotics and Automation*, v1, pp. 866-877.

Stewart, D.B. and P.K. Khosla, 1996, "The Chimera Methodology: Designing Dynamically Reconfigurable and Reusable Real-Time Software Using Port-Based Objects," *International Journal of Software Engineering and Knowledge Engineering*, v. 6, n. 2, pp. 249-277.

Todd, D.J., 1986, *Fundamentals of Robot Technology*, John Wiley and Sons, chapters 3, 7.

Voyles, R.M., G. Fedder and P.K. Khosla, 1996, "A Modular Tactile Sensor and Actuator Based on an Electrorheological Gel," *Proceedings of the IEEE International Conference on Robotics and Automation*, v1, pp. 13-17.

Voyles, R.M., A. Agah, P.K. Khosla, and G.A. Bekey, 1997, "Tropism-Based Cognition for the Interpretation of Context-Dependent Gestures," *Proceedings of the IEEE International Conference on Robotics and Automation*, v.4, pp.3481-3486.

Voyles, R.M., 1997, *Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation*, Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, August.

R.M. Voyles, J.D. Morrow, and P.K. Khosla, 1997, "Towards Gesture-Based Programming: Shape from Motion Primordial Learning of Sensorimotor Primitives," *Journal of Robotics and Autonomous Systems*, v. 22, pp. 361-375.

Wooldridge, M. and N.R. Jennings, 1995, "Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*, v. 10, n. 2, pp. 115-152, June.

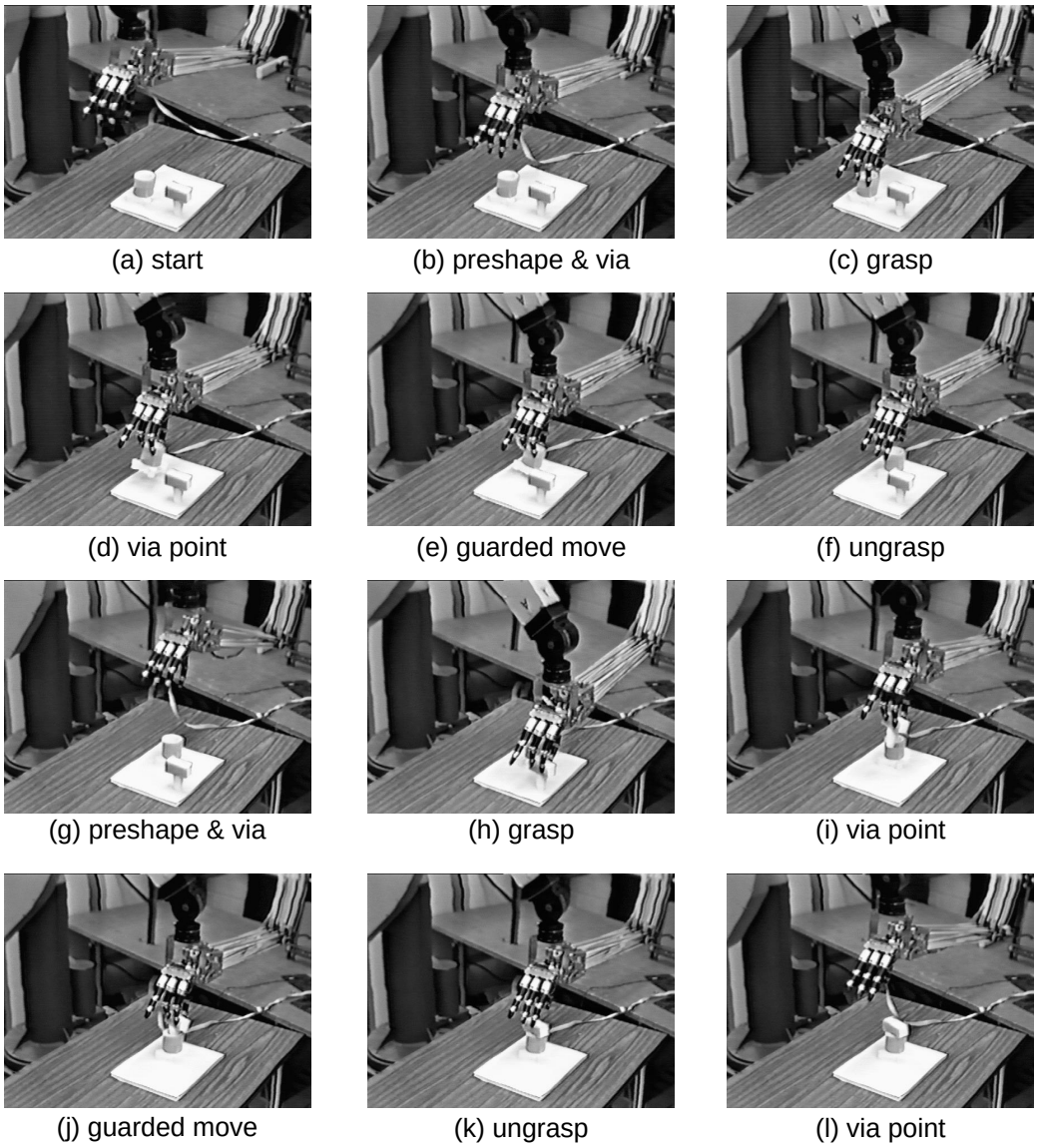

Figure 6: Robotic Execution of the Low-Tolerance Peg-in-Hole Task.