A MULTI-AGENT SYSTEM FOR PROGRAMMING ROBOTS 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

Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
pkk@ece.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 (called "sensorimotor primitives") which are intended to be the robotic equivalent of that which humans acquire through everyday physical interactions. 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.

1. 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 itself and of the development of the application.

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. This is not true for skills in general. 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.

2. 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 et al, 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.

Kaiser and Dillman (1996) looked specifically at extracting contact-based skills from human demonstration, paying particular attention to the inconsistency of human performance. This demands the ability to incrementally re-train and to prune ineffective actions from the training set. They used radial basis function networks to represent their skills, which do not offer a unified approach for acquisition and identification.

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".

3. 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 these 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 and relate them to the Wooldridge and Jennings definition.

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 variable during run-time, while resource ports are considered variable only 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 the combination of reactivity and social ability as defined by Wooldridge and Jennings.

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. Based on the above, we can not rule out a subroutine as an agent 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 Wooldridge and Jennings' definition of autonomy.

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 exhibit some non-trivial behavior, but, we acknowledge we are unable to quantify or define "non-trivial" at this time.

4. AGENTS FOR GESTURE INTERPRETATION

Why use an agent-based architecture for GBP? The answer lies in Wooldridge and Jennings' argument that it is reasonable to ascribe mentalistic notions when it enhances understanding. Programming languages are deterministic and can be interpreted algorithmically, so it does not enhance our understanding to ascribe mentalistic notions to their compilation. Furthermore, it is important to transcribe the input from the programmer with perfect accuracy. On the other hand, a human's actions are unrepeatable, error-prone, and unique to the individual. In programming contact tasks by human demonstration, we generally do *not* want to repeat the programmer's input with perfect accuracy. Instead, we want to interpret the human's *intention* from noisy observations of noisy actions. This is inherently a qualitative task requiring qualitative or heuristic-based solutions. Our approach is to embody a variety of different qualitative and heuristic solutions (or parts of solutions) within agents and let them decide if they are relevant or not.

This rationale has focused exclusively on the interpretation side of the problem. Another agent-based aspect of our approach is on the execution side. Because GBP assumes a decomposed set of sensorimotor primitives -- which map sensing to action -- it is natural to instantiate this executable expertise as a set of agents. They are autonomous; they have input/output relationships; and they're socially reactive. Plus, they can be recomposed easily and in various configurations.

But the focus of this section is the agent architecture for understanding the demonstration of the task, which we call gesture interpretation, not on the agent architecture for execution (which is similar in many ways). Gestures, within the scope of this work, are defined as "imprecise actions or 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 hand before a grasp, *motion gestures*, such as the path taken in transporting 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*." (This is an application of word spotting, which we won't discuss in this paper.) 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 (Voyles et al, 1997a) for more information and examples.

The architecture we use is a variant of the Tropism System Cognitive Architecture (TSCA) (Agah and Bekey, 1996) called the Fine-Grained TSCA that is described in (Voyles et al, 1997a). In the TSCA the world is sensed by an agent (e.g. a gatherer or a predator) and the sense information is matched to the agent's likes and dislikes (see Figure 1). The set of all matches suggests a set of corresponding possible actions with a probability for taking each action. For example, sensing the presence of a predator may suggest the actions "flee" and "freeze" with equal probability. Sensing a predator approaching may suggest "flee" with much greater probability than "freeze." From the suggested set of actions, the agent then probabilistically selects a particular action to take

The Fine-Grained TSCA works on the same principle but rather than assuming each agent is a rather complex, multi-modal entity with multiple goals (such as a gatherer that can search for food and avoid predators), it assumes each agent is a simple software entity with one very narrow goal: to prove or disprove a hypothesis. From an external perspective, each agent can only prove or disprove a hypothesis regarding the interpretation of a gesture. From the agent's perspective, however, it has a number of actions to take that either increase or decrease its confidence in its sole hypothesis. Complexity results from combinations of many of these simple agents.

In the nomenclature of TSCA, there are entities, ε, which are instances of gestures and their associated state, σ, is the context in which they occurred. The tuple (ε, σ) becomes a "gestural word" in our parlance, but because σ consists of continuous variables (state variables of the environment), it must be classified into fuzzy sets to create discrete tropism elements that can be matched. Because the fine-grained TSCA limits each agent to the single goal of confirming or refuting its hypothesis over a series of inputs (a "gestural sentence," in our parlance), the actions, α, involve only increases or decreases in the agent's confidence in its hypothesis. Hence, actions become scalar functions that are integrated over time. Agah and Bekey (1996) represent a tropism element as:

$$(\varepsilon, \sigma_F, \alpha(\sigma), \tau)$$

where σ_F is the fuzzy set to which σ belongs, $\alpha(\sigma)$ is a scalar function proportional to membership value, and τ is the preference

the agent has to take this action. In operation, there is a high degree of variability in the gestural words generated by the operator so the interpretation of a gesture is a random variable sampled many times. As a result, $\alpha(\sigma)$ represents a probabilistic distribution when considered over several samples (a gestural sentence) so we set τ to 1 without loss of consistency to the TSCA formulation of Agah and Bekey:

$$(\varepsilon, \sigma_F, \alpha(\sigma), 1)$$

Each agent has a set of these tropism elements which associate gestures and their context with greater proof or disproof of their respective hypothesis. This is illustrated diagrammatically in Figure 2. In the illustration "GRA" stands for Gesture Recognition Agent and "GIA" stands for Gesture Interpretation Agent. The "world" of traditional TSCA consists of a demonstrator and the objects with which she interacts during a task demonstration. The GRA's perform the sensing function of the standard TSCA and may read real sensors directly or abstract sensor information to create virtual sensors. In either case, the GIA's consume the gestural words spotted by the GRA's and build an interpretation by matching them to their own tropism set. The actions of a given entity are the increasing or decreasing of its internal confidence. But a "meta-action" is taken based on a community of agents through voting. In a nutshell, cognition is the result of matching observations of the world with *tropisms* -- the likes and dislikes of the agents involved.

4.1Gesture 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 3. 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 force sensor (Voyles et al, 1996) data into tactile gestures. As an example, consider the recognition of tactile gestures. Recognition of an instance of a gesture with the intrinsic (net force) fingertip sensor module consists simply of comparing the force magnitude to a threshold. If the magnitude exceeds the threshold, the gesture agent recognizes it as an instance of a tactile gesture and uses the direction of the force relative to the sensor to determine its type (pinch, anti-pinch, or nudge).

The state agent appends application-specific information which, for the types of manipulation tasks we attempted, included time histories of the force components and motion of the wrist for nudges and instantaneous values of the same for pinches. How this information is used is described in the following section.

4.2Gesture Interpretation Agents

The Gesture Interpretation Agents are task specific and must be described within the context of a particular application. The authors are interested in GBP for simple manipulation tasks (with the hope of extending to more complex assembly tasks in the future) and in the previous section examples of GRA's for pinch gestures were described. That example is continued here by providing an illustration of tropism elements for a pinch-gesture interpreter.

An agent was constructed for the purpose of determining if the demonstrator is in the process of grasping an object. Its tropism set includes a tropism element that associates a large increase in confidence of a grasp with a pinch gesture that occurred while the wrist was almost stationary; in other words, it likes that type of observation. It also included an element that dislikes any anti-pinch gesture (force direction opposite to that of pinching). Another element slightly dislikes a nudge (force impulse) because it assumes the human is placing an object and, therefore, is about to ungrasp. These are essentially the fuzzy rules that drive the behavior of the pinch GIA.

The recognition and interpretation of pinch gestures are rather trivial, but serve to illustrate the concept. More complex interpretations of other gestures are determined using the same methods.

5. 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 4, drawn in a manner consistent with Figure 2.

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 grasping an object. This gesture is parametrized based on the integrated force of the grasp. Like-

wise, “force” provides grasp gestures as described in Section 4.1. More importantly, the “force” recognizer also passes on interesting signatures from the fingertip intrinsic force sensor for interpretation of manipulation primitives. The “hand motion” recognizers look for approximately straight-line motions and also gross sweeping motions. 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. Some of the GIAs contain eigenspace encodings of individual primitives that have been previously taught to the robot using *shape from motion primordial learning*. (The learning process is described in (Voyles et al, 1997b).) These interpretation agents are more algorithmic than others, but still fit the fine-grained TSCA. 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. The tropisms for these agents involve the degree of parallelism that is likable. “Guard Move” represents the guarded move primitive while “Rotate X” represents a similar primitive that accommodates torques around the x-axis.

Finally, the “traj” agent extracts trajectory segments from the hand motion gestures to fill in via points naturally selected by the human’s motion. All agents make use of the volume sweep rate gestures as a cue for the particular phase of action: pre-grasp, grasp, or manipulation (Kang and Ikeuchi, 1996).

The GBP system is shown during the demonstration phase of a low-tolerance peg-in-hole insertion in Figure 5. 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 the 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 as in previous systems (Kang, 1994). This more closely matches what humans actually do and results in greater robustness to uncertainties in the environment.

To abstract the task, a multi-agent network, based on the fine-grained tropism system cognitive architecture represented in Figure 4, 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 6 as a finite state machine (FSM) using Morrow's Skill Programming Interface (SPI) (Morrow, 1997). This is an important step in the development cycle because it provides a visual check of the system's interpretation of the demonstration. The output of the multi-agent network is automatically post-processed and formatted for display by the SPI. The user can quickly determine if the program "looks correct" before actually executing it on the robot. Each bubble in the FSM represents a node that consists of several agents executing in parallel to achieve the desired action. The SPI provides an interface to inspect the internals of each bubble if the user suspects a problem with the program or desires more information.

The autonomous execution of the above program on a PUMA robot with a Utah/MIT hand is shown in Figure 7.

6. DISCUSSION

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

The programming of a new task is done by demonstration of that task in real time. No special actions or additional time is required of the demonstrator other than that incurred as the result of wearing an unfamiliar glove and fingertip coverings. (The current generation of tactile sensors does significantly impact manual dexterity, as expected.) Although we have performed no trials with naive users, the several trials we have performed have been completely successful about 70% of the time. (Detailed statistical trials have not been investigated.) Errors are generally caused by unstable grasps of the peg by the robot which introduces error into the initial positioning of the peg for the insertion. A separate project is investigating the inclusion of visual alignment of the peg and hole into the GBP paradigm.

One would like the domain of understanding to be equivalent to the domain of application, but we have not yet begun to investigate the implications of this. It remains very much an open research 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 given 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 elsewhere described learning approaches to extract the primitives automatically (Voyles et al, 1997b) and successfully employed them in this work, but considerable foresight is still required on the part of the demonstrator.

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

7. 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.

8. REFERENCES

Agah, A, and G.A. Bekey (1997) "Phylogenetic and Ontogenetic Learning in a Colony of Interacting Robots," *Autonomous Robots* 4(1):85-100.

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

Kaiser, M. and Dillman, R. (1996) "Building Elementary Robot Skills from Human Demonstration," *Proceedings of the IEEE International Conference on Robotics and Automation* 3:2700-2705.

Kang, S.B. (1994) *Robot Instruction by Human Demonstration*, Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, CMU-RI-TR-94-33, November.

Kuniyoshi, Y., M. Inaba, and H. Inoue (1994) "Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance," *IEEE Transactions on Robotics and Automation,* 10(6):799-822.

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

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,* 1:866-877.

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*, 6(2):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,* 1:13-17.

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

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 (1997b) “Towards Gesture-Based Programming: Shape from Motion Primordial Learning of Sensorimotor Primitives,” *Journal of Robotics and Autonomous Systems*, 22(3-4):361-375.

Wooldridge, M. and N.R. Jennings (1995) “Intelligent Agents: Theory and Practice,” *Knowledge Engineering Review*, 10(2):115-152.

.

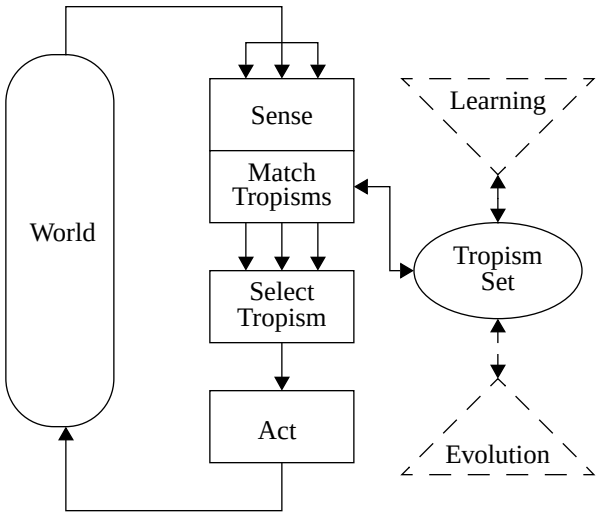


FIGURE 1: STANDARD TROPISM SYSTEM COGNITIVE ARCHITECTURE

,

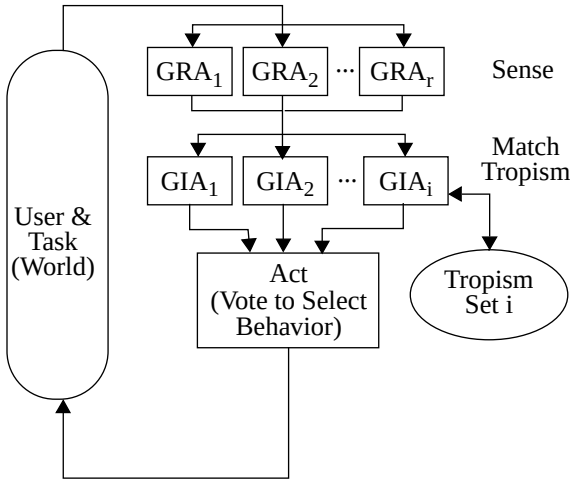


FIGURE 2: STATIC FINE-GRAINED TROPISM SYSTEM COGNITIVE ARCHITECTURE AS IMPLEMENTED FOR GESTURE INTERPRETATION.

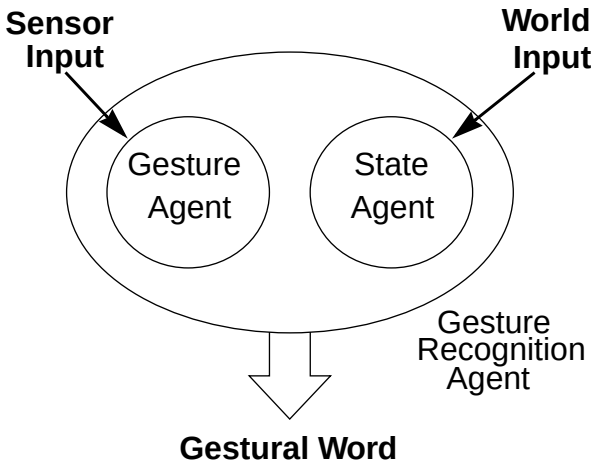


FIGURE 3: BASIC STRUCTURE OF THE GESTURE RECOGNITION AGENT

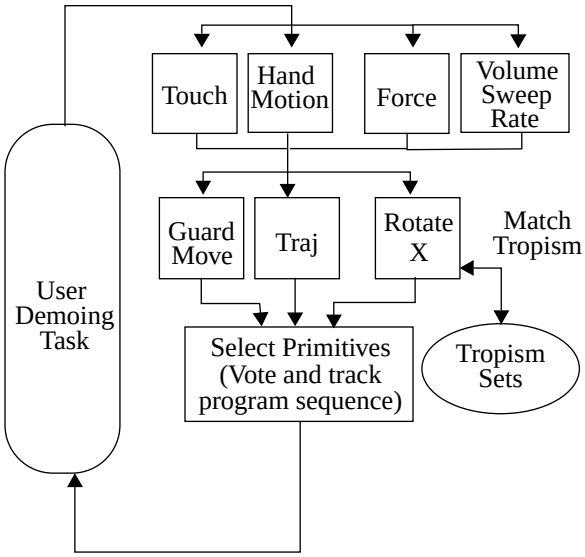
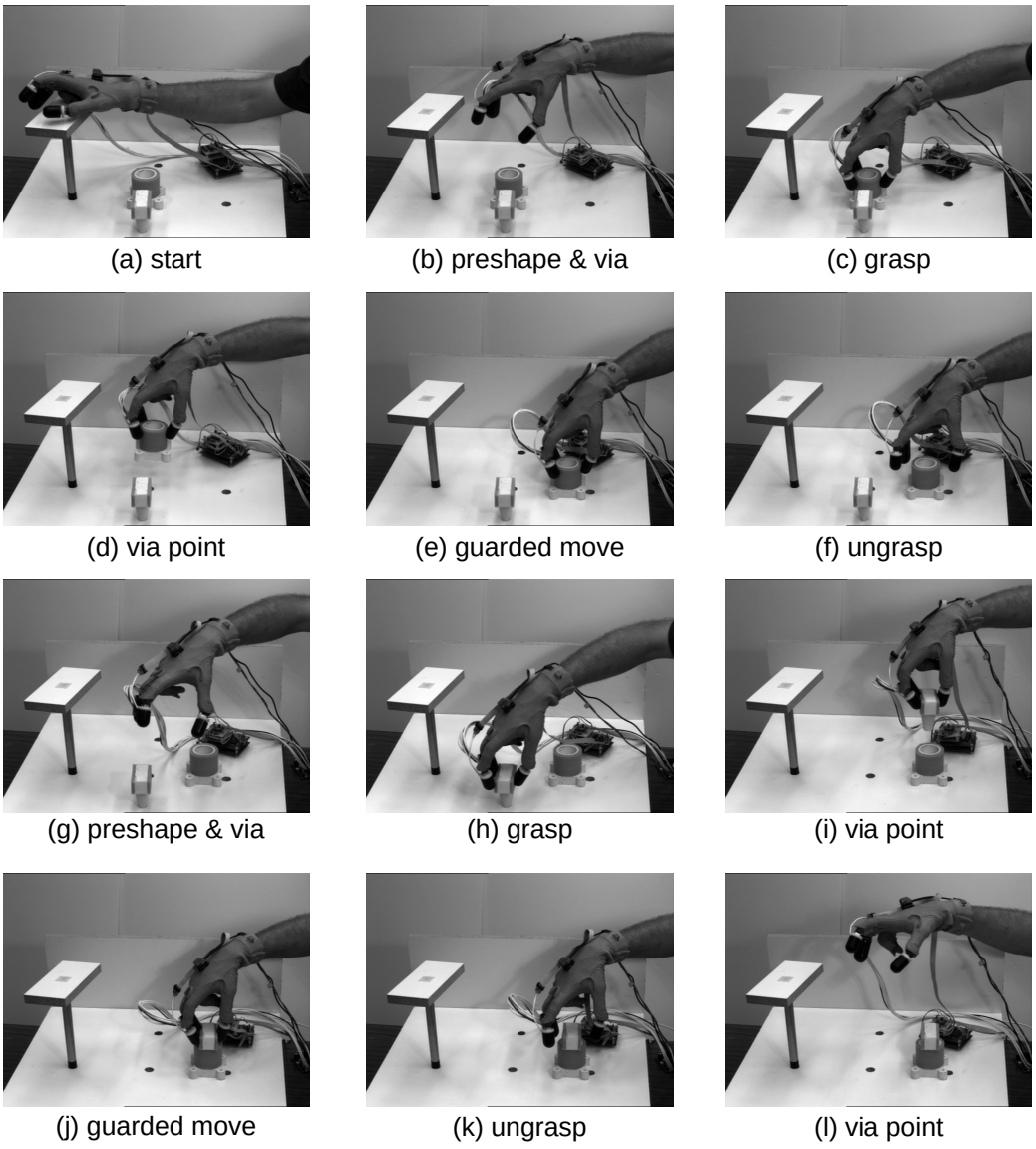


FIGURE 4: GBP NETWORK FOR SIMPLE MANIPULATION TASKS.



(a) start (b) preshape & via (c) grasp

(d) via point (e) guarded move (f) ungrasp

(g) preshape & via (h) grasp (i) via point

(j) guarded move (k) ungrasp (l) via point

FIGURE 5: DEMONSTRATION OF THE LOW-TOLERANCE PEG-IN-HOLE TASK.

.

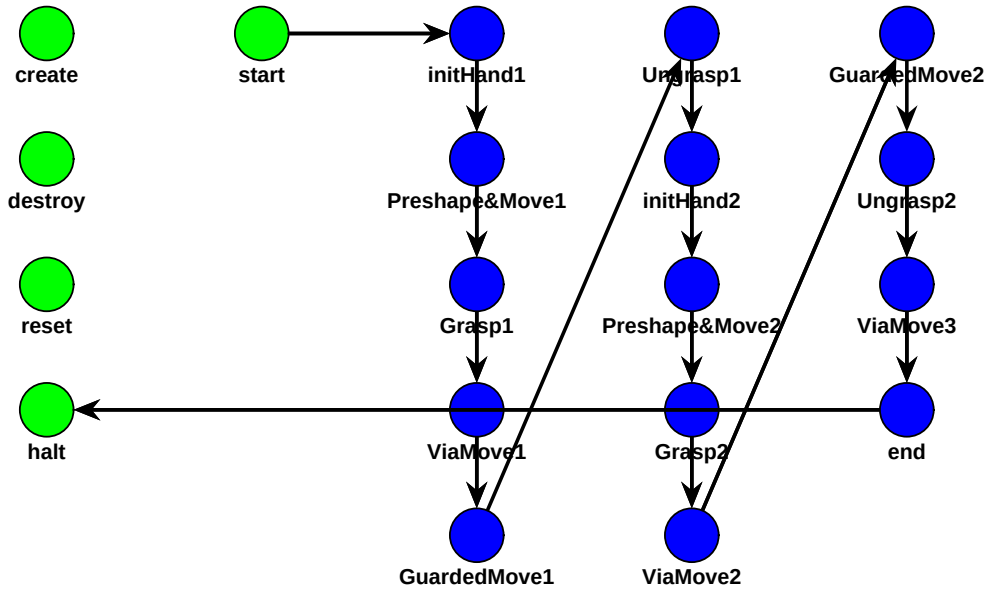
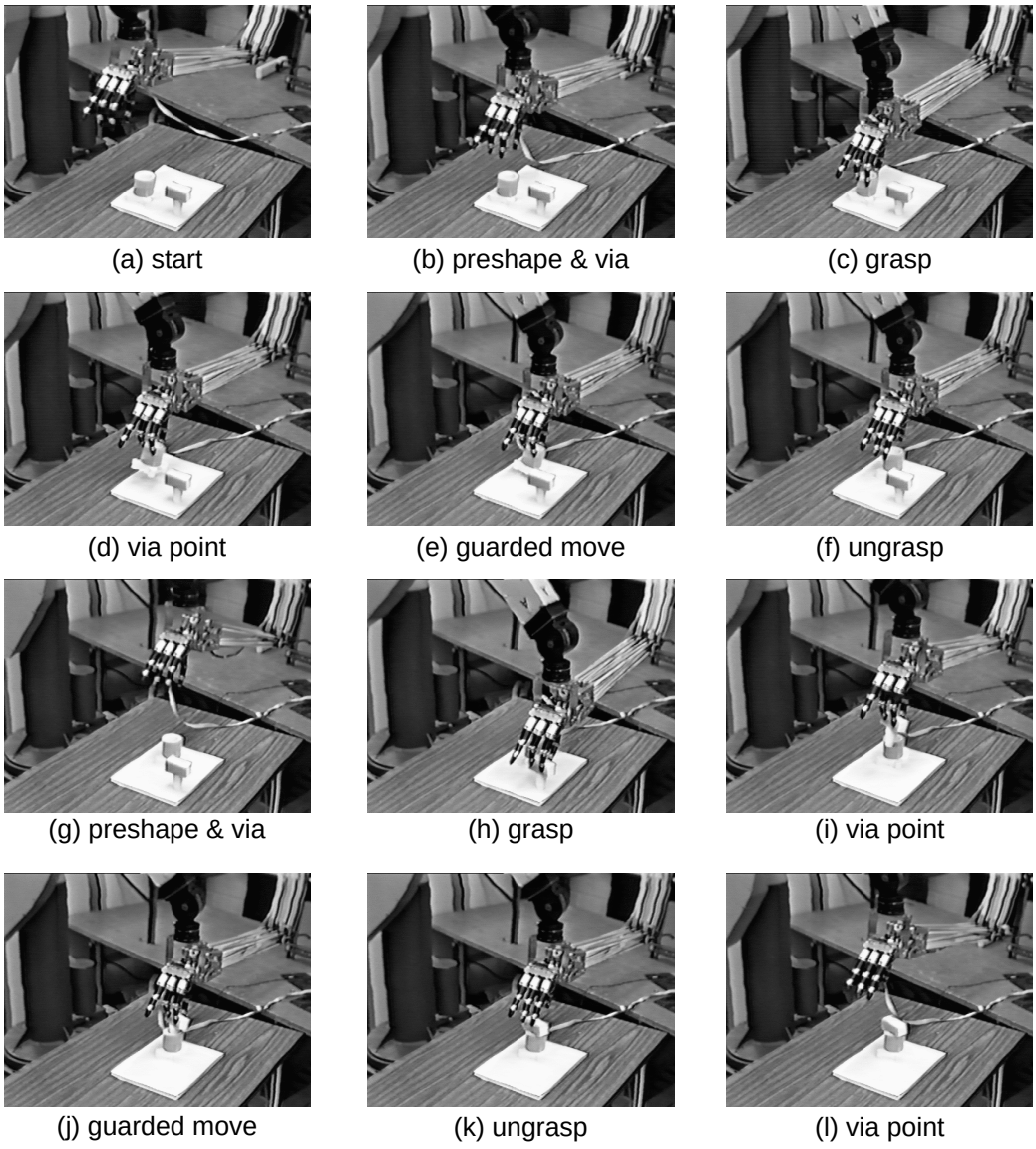


FIGURE 6: SPI GRAPHICAL DISPLAY OF PROGRAM RESULTING FROM PEG-IN-HOLE DEMONSTRATION.



(a) start (b) preshape & via (c) grasp

(d) via point (e) guarded move (f) ungrasp

(g) preshape & via (h) grasp (i) via point

(j) guarded move (k) ungrasp (l) via point

FIGURE 7: ROBOTIC EXECUTION OF THE LOW-TOLERANCE PEG-IN-HOLE TASK.