In *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 6; Smart Engineering Systems: Neural Networks, Fuzzy Logic and Evolutionary Programming*, ASME Press, 1996

GESTURE-BASED PROGRAMMING, PART 1: A MULTI-AGENT APPROACH

RICHARD M. VOYLES
Robotics Ph.D. Program
Carnegie Mellon University

PRADEEP K. KHOSLA
Electrical and Computer Engineering
Carnegie Mellon University

ABSTRACT:
Gesture-Based Programming is a paradigm for the evolutionary programming of rapidly deployable manipulation systems by human demonstration. The goal is to provide a more natural environment for the user and to generate more complete and successful programs by focusing on task experts rather than programming experts. What makes it unique from other programming by human demonstration approaches are the same things that make it evolutionary: a composable knowledge base of expertise agents and a facility for supervised practice after initial training. Prior knowledge of previously acquired skills (sensorimotor expertise) facilitates the interpretation of "gestures" during training and then provides closed-loop control of execution during run-time. This paper, part one of two, presents a high-level description of the system as well as descriptions of capabilities we've demonstrated on a PUMA robot and Utah/MIT hand. The companion paper provides a detailed account of one method of acquiring, matching, and even transforming sensorimotor expertise.

INTRODUCTION

Programming robotic systems is difficult not only for novices but experts alike. Planning operation sequences and collision-free paths while maintaining realistic cycle times can be hard enough, but add in the complexities of robot/environment contact, uncertainty, and exception handling and the expertise required becomes daunting. As a result, robotic technologies have had minimal impact on most manufacturing domains (in terms of the number or robots versus number of humans employed), particularly as product life cycles and batch sizes decrease. Ironically, these are the conditions under which robots were originally touted as being most effective (Hartley, 1983).

Once implemented, many robotic applications fail to meet expectations or prove insufficiently robust for the desired level of autonomy. A primary reason for this is that programming difficulties maintain a layer of insulation between the system's use and it's development. Because programming requires so much expertise, unskilled users, by definition, cannot be programmers and experienced programmers are too "valuable" to be users. As a result, there is a gap between personnel needs and expertise.

Numerous attempts have been made to ease the discomfort of robot programming with varying degrees of success. Behavior-based and multi-agent systems (Brooks, 1986) seek to modularize software for easier programming by programming experts. Visual programming environments like Chimera/Onika (Stewart et al, 1993)(Gertz, 1994) and commercial packages such as MatrixX/SystemBuild and LabView capitalize on modular, reconfigurable software blocks by iconifying them within "software assembly" environments. These visual environments allow programming at a much higher level, hiding many details of the particular implementation, and lessening the burden of programming expertise.

Despite these advances, a fairly high level of expertise is still required to program and interact with robots. For example, Onika allows novice users to easily build simple pick-and-place robotic applications in a matter of minutes (Gertz, 1994). But, it is imperative that the user understands and consciously considers the importance of via points, collision avoidance, grip selection, and the dynamic effects of transport in order to for the application to be successful. Since these characteristics are inherent in all manipulative tasks and are adeptly mastered by humans, they are usually handled unconsciously. Hence the distinction between task experts and programming experts. Programming experts are trained to bring these subconscious requirements up to the conscious level in order to transcribe them as required by the particular programming environment. Visual programming does little for the bulk of robotic applications that involve substantive contact with the environment, uncertainty, and complex sensing.

A more recent approach to human/robot interaction is the field of *learning by observation* (Kang and Ikeuchi, 1996)(Kuniyoshi et al, 1989). By forcing the robot to observe a human interacting with the world, rather than forcing the human to interact with a *textual representation* of the robot interacting with the world, a more natural, "anthropocentric" programming environment results. But these systems are mostly kinematically-based and operate off-line; if the robot misinterprets the desired trajectory, the whole sequence must be re-taught. (More on learning by observation later.)

It is instructive to look at the inroads robots have made into manufacturing to see how these problems can be mitigated. Two of the most common industrial applications are spray painting and spot welding. Both of these use natural programming methods such as "lead-through teaching" (Todd, 1986) (i.e. human demonstration) and point-teaching, respectively, that are easily mastered by semi-skilled workers. Lead-through teaching, in particular, allows anyone who can perform a "valid" task to program the robot to perform the task by demonstrating it. As such, programming changes can be made reactively or proactively by a task expert, not by calling in a robot programming expert. Tasks such as spray painting involve *skill transfer* from the teacher (i.e., programmer) to the robot. Lead-through teaching provides an intuitive mechanism for a task expert to accomplish this. Unfortunately, it is limited to simple tasks because of its reliance on pure kinematic replay.

GESTURE-BASED PROGRAMMING OVERVIEW

If one views programming as a form of teaching, textual programming is clearly unnatural for humans . Despite a human's ability to master some task -- from high-level planning of sequence and trajectory, to grasp selection, to robust, low-level execution -- there is no guarantee that human will be able to describe the task in syntactically correct prose in a foreign (programming) language. Humans are more effective at

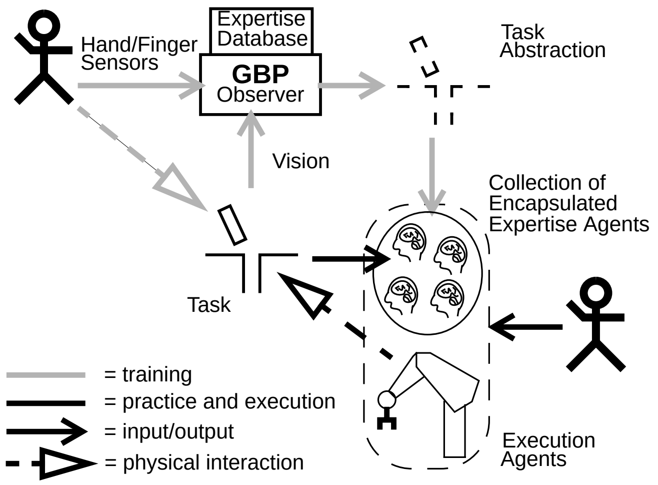

Figure 1: Gesture-Based Programming system overview.

teaching by demonstration and practice and, although there are no guarantees here, either, it has been proven empirically to be the most successful technique between humans (Patrick, 1992). But teaching by demonstration is not guaranteed to work and one of the reasons is it assumes a basic set of *a priori* capabilities. Imagine trying to teach calculus to someone who doesn't know algebra. Gesture-Based Programming (GBP) addresses all of the above issues by providing a more natural, demonstration-based programming paradigm that allows both demonstration and practice phases and relies on an underlying skill base of robust, semi-autonomous primitives from which new tasks can be assembled.

GBP begins by observing a human demonstrate the task to be programmed. (See the stick figure on the left of Figure 1.) Observation of the human's hand and fingertips is achieved through a sensorized glove. The modular glove system senses hand pose, finger joint angles, and fingertip contact conditions. Objects in the environment are sensed with computer vision while a speech recognition system extracts "articulatory gestures." (Gestures will be described later.) Primitive gesture classes are extracted from the raw sensor information and passed on to a gesture interpretation network. These autonomous multi-agent networks extract the demonstrator's intentions with respect to the system's skill base to create an abstraction of the task. In other words, the system is not merely remembering everything the human does, but is trying to understand -- within its scope of expertise -- the subtasks the human is performing ("gesturing"). These primitive capabilities in the skill base take the form of *encapsulated expertise agents* -- autonomous agents that encode sensorimotor dexterity.

The output of the GBP system is the executable program for performing the demonstrated task on the target hardware. This program consists of a network of encapsulated expertise agents of two flavors. The primary agents implement the primitives required to perform the task and come from the pool of primitives represented in the skill base. The secondary set of agents includes many of the same gesture recognition and interpretation agents used during the demonstration. These agents perform on-line observation of the human to allow supervised practicing of the task, if desired. (Stick figure on far right of Figure 1.)

As mentioned above, the human model for teaching by demonstration most often involves a practice phase. The reason for this is that passive observation of a task rarely provides accurate parametrization for the trainee's deduced task "model" (in this case, the model is represented by the collection of primary encapsulated expertise agents) and sometimes the deduced model is wrong (e.g. missing or incorrect encapsulated expertise agents). Incorporating gesture recognition and interpretation agents into the executable provides an intuitive way for the demonstrator, or another user, to fine tune the operation of the program without having to demonstrate the entire task over again. Because all our agents are implemented as autonomous software modules, these observation agents can easily be disabled without recompiling the program to prevent unauthorized modification.

In the real world, it will not be possible to represent most useful tasks with one network of encapsulated expertise agents. Therefore, it is necessary to segment the demonstration into a series of discrete subtasks (e.g. a grasping subtask followed by a manipulation subtask). Each subtask will be embodied by a network as described above. In this case, the executable program will consist of a sequence of networks rather than one network.

What are gestures?

We define gestures as "imprecise events that convey the *intentions* of the gesturer." There are many gestural modalities including *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*, such as "oops!' when an error is made or model-based information such as "tighten the *screw.*" 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.

In effect, gestures form an "alphabet" from which "words" are formed when combined with state information. The state information is also represented by characters in the gestural alphabet. These gestural words are strung together by the gesturer into "sentences" which form the basis for interpretation. The gestural alphabet is like a linguistic alphabet in the sense that characters are parametrized. For instance, in English, the letter "a" has many sounds including the long "a" as in "save," the short "a" as in "sad," and the silent "a" as in "read." This is an implicit parametrization. In contrast, the gestural characters are explicitly parametrized by either a single integer or floating-point argument. An example can be found in (Voyles and Khosla, 1995b).

Gesture recognition is handled by individual gesture recognition agents 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 in Figure 2. It recognizes the instance of a gesture and the type of gesture within the gesture class (mode). The domain specific agent grabs the relevant state information associated with the gesture type for the given domain and appends it to the gestural word.

PROJECT STATUS

Learning by Observation

GBP is a direct extension of the real-time control work by Voyles for the functional demonstration of Kang and Ikeuchi's Learning by Observation system (Kang and

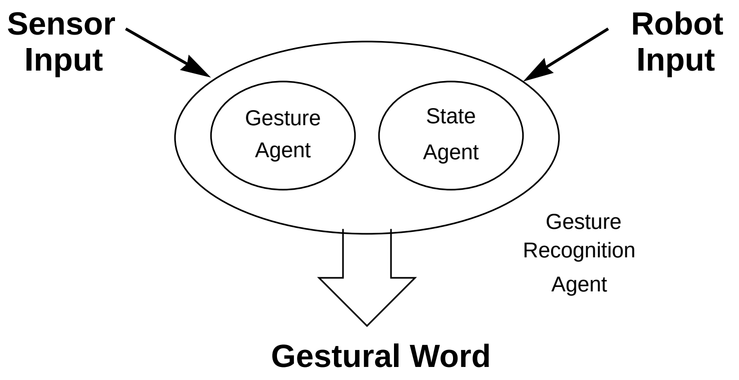

Figure 2: Structure of the gesture recognition agent.

Ikeuchi, 1996). The human is instrumented with a sensorized glove but only kinematic information is reported. As the human demonstrates simple tasks, hand and wrist trajectories are recorded and a multi-baseline stereo machine looks for changes in the pose of detected objects. These data streams are automatically segmented to detect pre-grasp, grasp, and manipulation phases of motion and then mapped from the human hand to the target robot hand. A PUMA and Utah/MIT hand demonstrated lifting, transporting, and re-depositing objects as well as low-tolerance peg-in-hole insertions.

While these demonstrations were impressive when successful, getting them to succeed was difficult. Based solely on kinematic trajectories, duplication by the robot demanded accurate calibration of the workspace since no feedback was provided. The robot was incapable of reacting to even the most insignificant of error conditions. In fact, optimization of the kinematic grasp to fit the surfaces of the manipulated objects prevented the exertion of grasping forces. We had to command trajectories that penetrated the object surfaces in order to hold them stably. These experiences demonstrated the need for a skill-based approach to programming by human demonstration. They also suggested reconfigurable agents for the implementation of those skills.

Gesture Interpretation Networks

An example gesture-based interaction system is shown in Figure 3 and described more fully in (Voyles and Khosla, 1995a). This is not a programming environment but an application scenario utilizing tactile gestures. The robot executes one of four polygonal cyclic trajectories: rectangle, triangle, cross, or pick-and-place. Each trajectory has parameterized height, width, and thickness. The user selects and tunes the trajectories with tactile gestures (nudging the end effector) sensed by a force/torque sensor or trackball. The recognition agents segment gestures from the sensory data streams and pass gestural words on to the interpretation agents. Each interpretation agent contains a fuzzified model of a single hypothesis which gestures either support or refute. The agents individually build a confidence value in their respective hypotheses and vote within agent pools, each pool being winner-take-all. Some of these agents were also applied to a cable harnessing task as described in (Voyles and Khosla, 1995b).

CONCLUSIONS

We have presented our concept for a gesture-based programming system as well as brief portrayals of work we have completed on individual components of that system. Although the complete programming environment is not yet finished, the success of

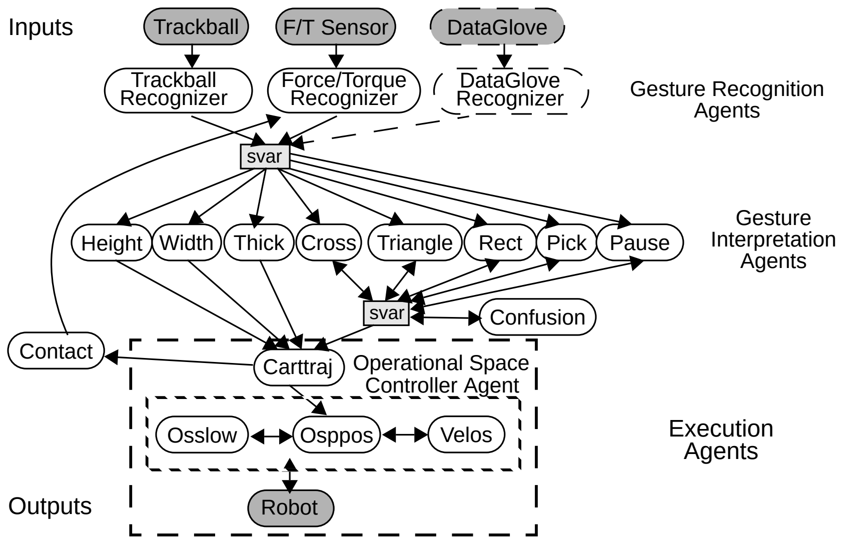

Figure 3: Multi-agent interpretation network for tactile gestures.

our preliminary work leads us to believe it will provide a much more natural and successful programming paradigm for certain classes of contact-based tasks by allowing task experts, rather than programming experts, to specify the operations. Part 2 of this series describes one method of acquiring expertise agents for the skill-base.

ACKNOWLEDGMENTS

Mr. Voyles was supported in part by a National Defense Science and Engineering Graduate Fellowship, a DOE Integrated Manufacturing Predoctoral Fellowship and by Sandia contract AL-3019.

REFERENCES

Brooks, R.A., (1986). “A Robust Layered Control System for a Mobile Robot,” *IEEE Journal of Robotics and Automation*, v.RA-2, n.1, March, pp. 14-23.

Gertz, M.W, (1994). *A Visual Programming Environment for Real-Time Control Systems*, Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University.

Hartley, J., *(1983). Robots at Work*, IFS Publications, Ltd.,Kempston, Bedford, UK, chapters 5, 8.

Kang, S.B., and K. Ikeuchi, (1996). “Toward Automatic Robot Instruction from Perception -- Temporal Segmentation of Tasks from Human Hand Motion,” *IEEE Trans. on Robotics and Auto.*, Mar.

Kuniyoshi, T., M. Inaba, and H. Inoue, (1989). “Teaching by Showing: Generating Robot Programs by Visual Observation of Human Performance,” *Proc. of 20th Intnl. Symp. on Ind. Robots*, pp. 119-126.

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

Stewart, D.B., R.A. Volpe, and P.K. Khosla, (1993). “Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects,” CMU Robotics Institute tech. rpt., CMU-RI-TR-93-11.

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

Voyles, R.M. and P.K. Khosla, (1995a). “Tactile Gestures for Human/Robot Interaction,” *Proc. of IEEE/RSJ Conf. on Intelligent Robots and Systems*, Pittsburgh, PA, v. 3, pp. 7-13.

Voyles, R.M. and P.K. Khosla, (1995b). “Multi-Agent Gesture Interpretation for Robotic Cable Harnessing,” *Proc. of IEEE Conf. on Systems, Man, and Cybernetics*, Vancouver, B.C., pp. 1113-1118.