Task Automation by Interpreting User Intent

Kevin R. Dixon

Submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy at

Carnegie Mellon University
Pittsburgh, PA 15213
24 April, 2001

Contents

List of Figures

List of Tables

1 Introduction and Motivation

Despite the numerous advances in human-computer interaction (HCI), most development systems still require users to convey knowledge to computers and robots through procedural-programming techniques. For the vast majority of the population, this transfer of knowledge is limited by the programming expertise of the user. Indeed, expertise in programming is its own job skill. Most users have neither the experience nor the inclination to program computers and robots to perform their tasks. In industrial settings, many companies do not have the resources to automate production. For example, ABB Flexible Automation estimates that 98% of arc welding is performed by hand due, in part, to the complexity and duration of the programming process. The down-time required to reprogram the facilities may interrupt production and the expense required to obtain programming expertise may be too great. The goal of the proposed research is to create a system that increases knowledge transferred between users and computers or robots. Since the user is not a trained computer programmer, the system should allow a user to "program" automation tasks by demonstration, instead of writing software in a conventional programming language.

The broad paradigm of programming a system to perform a task based on user demonstrations is called *Learning by Observation (*LBO*), Programming by Demonstration, Teaching by Example*, or some permutation thereof. In an LBO system, the inputs are observations obtained from user demonstrations. Since we are interested in using LBO to automate tasks, the output of the system is some type of generative model, *e.g.* , a controller, production program, etc. In this work, a task is defined to be a sequence of states that accomplish some overall goal. The states sufficient to complete the task are called subgoals. Because LBO is a general approach to a wide variety of tasks, addressing its major issues will find use in many domains (Pomerleau, 1994; Cypher, 1993; Ikeuchi et al., 1993; Kuniyoshi et al., 1994; Yang et al., 1994; Schaal, 1997; Bentivegna & Atkeson, 1999; Matarić, 1999). Each time an LBO system observes the user or performs the task, the environment may be different. Even though minor perturbations in the environment can derail controllers and production programs, most previous LBO systems assume that the environment is static. An LBO system that can operate in a dynamic environment will facilitate its mainstream acceptance.

As with all sensor-based systems, LBO must contend with uncertainty in raw sensor readings and in determining the configuration of the environment (Thrun, 2001). In addition to uncertainty in perception, an LBO must system must also deal with uncertainty in observing user demonstrations. If the user demonstrates a task several times, the inherent imprecision and nonrepeatability of human actions during demonstration mean that there will be uncertainty in determining if an action was necessary to complete the task or an unintended byproduct due to a specific setup of the task. Since tasks are sequences of goal-directed states, the *intent* of the user should be consistent throughout all demonstrations, despite the noise and imprecision of her actions. The LBO system should attempt to capture this intent to determine the subgoals and, consequently, the overall goal, of the task. For many domains, it may not be acceptable for the LBO system merely to repeat the actions of a user verbatim. In many instances, humans perform tasks differently than a computer system would. Also, if the LBO system does not generalize and consider the intent of the user then it may be impossible to accommodate perturbations in the environment. The main issues to address in this proposed research are designing an LBO system that addresses uncertainty in sensing and user demonstrations, identifying the intent of the user, generating a robust automation program insensitive to changes in the environment, and incorporating multiple demonstrations to improve system performance.

2 Related Work

Virtually all task-based programs used in practice today replay a series of open-loop commands. In the user-interface domain, these programs are called “macros”. To create a macro, a user specifies a sequence of commands which can be recalled later. Generally, macros are created to simplify repetitive tasks with no variation between them (Cypher, 1993). In the industrial setting, automation programs are usually created by moving the robot (or other system) through a sequence of desired locations, called “waypoints”. Sometimes sensors are used to trigger the beginning and end of the program, but most move through the waypoints in an open-loop fashion. Integrating sensors into the locus of task-based programs (called sensor-based control or visual servoing (Morrow & Khosla, 1995)) tends to slow their execution.

Learning by Observation has been the subject of much study for motor-skill acquisition tasks. In (Pomerleau, 1994), researchers constructed a system that allowed users to generate an essentially limitless supply of training examples steering a car. From these examples, the computer system learned to imitate the car-steering skill of the user, using a camera as input. In this work, the skill was a memoryless, nonlinear control law. In (Kositsky, 1998), researchers performed psychophysical tests to determine patterns in the behavior of users performing sample manipulation tasks. These tests looked for clusters, or *subgoals*, in the state-space trajectory of the plant (a simulated two degree-of-freedom manipulator). A stochastic search algorithm is then used to move through the subgoals, using an approximated forward model of the manipulator. However, these state-space subgoals have no grounding in the physical configuration of the environment and any alteration in the work space would require complete retraining. Furthermore, the method of state-space clustering and the stochastic forward-model search does not scale well with the state-space dimensionality.

Learning by Observation has also been used to decompose a task into a temporal sequence of primitives. In (Yang et al., 1994), researchers used Hidden Markov Model (HMM) training techniques to learn telerobotic manipulation skills. These skills were open loop, in that there was no force or visual feedback to the user demonstrating the task. In this work, a skill was a finite-state automaton (FSA), with each node executing a memoryless, nonlinear control law and transitions determined by HMM training on user examples. In (Friedrich et al., 1996) a task, demonstrated by a user, is decomposed into a sequence of predefined primitives, called *Basic Operations*, using Time-Delay Neural Networks (TDNNs). The decomposition is then stored in a symbolic fashion. A search, using the pre- and post-condition STRIPS model (Fikes & Nilsson, 1971), is then executed to determine a sequence of primitives that describes the sequence observed from the user. The segmentation derived from the TDNNs serves as an initial bias on the search and the user is allowed to make modifications to the automatically generated program. This type of symbolic system does not consider the uncertainty inherent in observations and adding a new primitive to the system requires training a new TDNN, as well as a creating a new STRIPS pre- and post-condition model of the primitive. Furthermore, all programs (whether automatic or user modified) run without any sensor

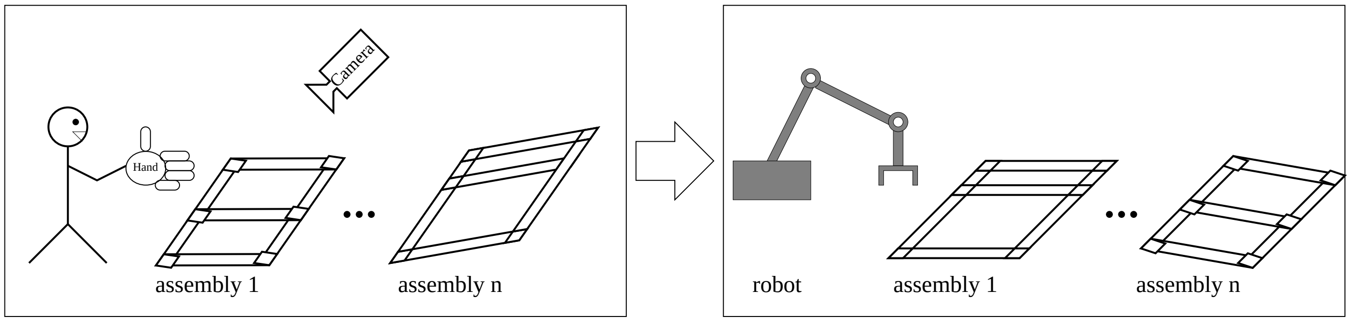


Figure 1: Task setup for a typical LBO problem.

input and are unsuitable for most realistic operating conditions. In (Matarić, 1999), researchers used neuroscience and psychophysical models to select a "basis set" of primitives. Observations from users are segmented to determine the primitives to execute in a temporally sequenced or additive fashion. Dynamic Time Warping was used in (Kang & Ikeuchi, 1995) to decompose observations into symbolic primitives for assembly tasks. In (Bentivegna & Atkeson, 1999), researchers used application-specific *ad hoc* methods to create symbolic descriptions of human actions in simulation.

Another use of LBO is using observations to prime approximate, iterative optimal-control methods. In (Schaal, 1997), a demonstration of a nonlinear control problem (the inverted pendulum) is used to bias the initial estimate of a Linear Quadratic Regulator (LQR) using Q-Learning. This method arrived at a satisfactory solution better than an order of magnitude faster than standard Q-Learning (Watkins & Dayan, 1992). However, it is unclear that the approach will scale with the dimensionality of the task.

3 System Design

A typical problem to be solved by an LBO system is given by Figure 1. A user demonstrates a task while the LBO system observes. The user can repeat the task as many times as she wishes, and the environment may change between demonstrations. Once the task demonstrations are finished, the LBO system designs a program to perform the task based on user observations. This program may encounter situations that it did not observe previously and must be able to adapt to modified environment configurations.

The conceptual flow diagram of the proposed LBO system is given in Figure 2. First, the configuration of the environment is determined (locations of parts in the work space). Next, the LBO system observes the user performing the task and these observations are fed into another subsystem which extracts important aspects, or subgoals, from the observations. The subgoals are then associated with objects in the environment so that the subgoals will be updated as the corresponding objects are altered. At this point, the user can demonstrate the task again, or allow the LBO system to design a production program to perform the task. In designing the program, the LBO system must first consider observations from all user demonstrations. The LBO system maps all the demonstrations to a common environment configuration so that it can determine the structure of the task based on information gained from all demonstrations. The LBO system then performs the task and incorporates any

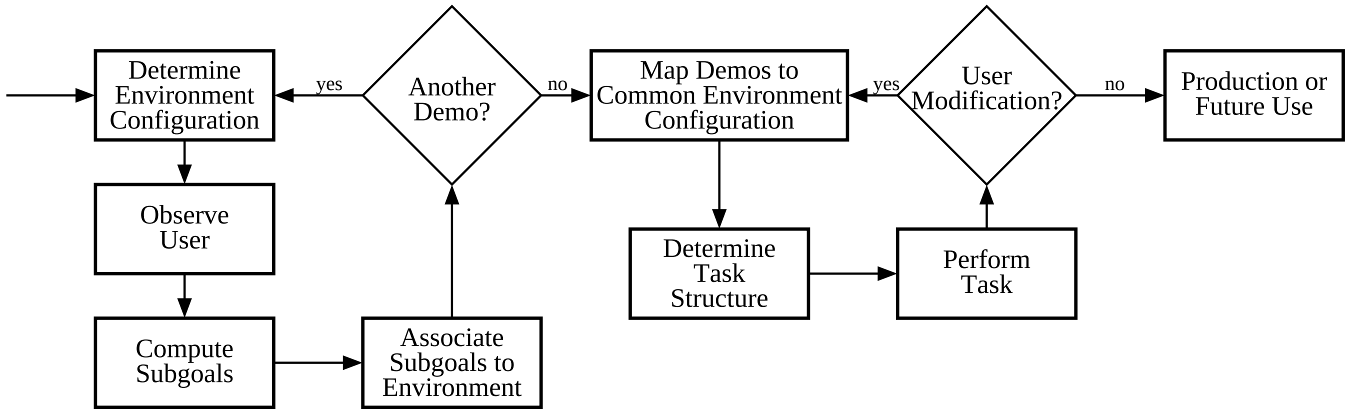


Figure 2: Flow diagram of the overall system.

modifications that the user may have. A more detailed description of the subsystems in the flow diagram will be given in the following subsections. A data-flow chart of the overall system is given in Section 3.9.

3.1 Observing the User

From the standpoint of this research, there are two broad types of observations: direct and indirect. Direct observation requires that users perform the task with the computer or robot. Thus, if a joystick is used to position a mobile robot or a teach pendant is used to guide the end effector of a manipulator through the task, then the LBO system can observe the state of the robot directly through GPS, shaft encoders, etc. Direct demonstration makes less work for the LBO system, but the user must transfer domain expertise through factitious devices like joysticks and teach pendants. If the user demonstrates the task without using the computer or robot, the LBO system must determine the state indirectly. For example, if the user demonstrates the task "by hand", then the LBO system must map sensor readings to the state of the computer or robot, from a camera, instrumented glove, or other device. Indirect demonstration allows the user to transfer knowledge to computers or robots in the most natural, concise manner possible. However, indirect demonstration raises the issue of mapping sensor capabilities and observability, though this issue is present in direct demonstration to a lesser extent. When humans perform tasks, their actions are based on feedback from their senses and even the most sophisticated computer sensor suites are do not offer the wide range of sensory feedback as that of a human. Due to these sensing differences, clearly there will be some tasks outside the scope of an LBO system. An assumption in this work is the state of the user is observable using the available sensors.

This research will focus on cameras as the primary source of sensory input for indirect observation in reality-based tasks since we feel that cameras are the most general and least intrusive sensor available. Reality-based tasks are those where some portion of the task occurs in the real world. Non-reality-based tasks occur entirely within a computer environment (*e.g.* , simulators and Web surfing). While some devices, such as tactile gloves, can increase the sensing capabilities of the LBO system, they tend to require that the user modify the environment or her behavior. Since one of the goals of this research is to maximize knowledge transfer between users and computer systems, we would like to avoid instrumenting the user in any obtrusive fashion. Additional modal

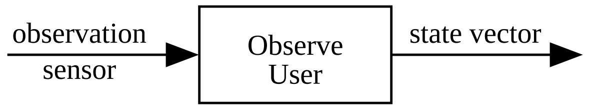


Figure 3: Input/output model of the "Observe User" subsystem.

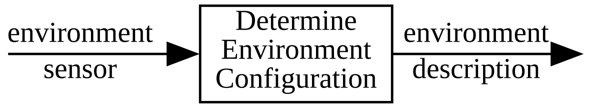


Figure 4: Input/output model of the "Environment Description" subsystem.

inputs, such as speech, may be considered as observations since this may help users convey domain knowledge and may be a natural process for many tasks.

The input to this subsystem is some type of observation sensor (*e.g.* , a camera, speech recognition, laser range finder, etc.) and the output is a discrete-time state vector, $\boldsymbol{x}_n$, describing the (possibly stochastic) configuration of the user demonstration (*cf.* Figure 3).

3.2 Determining the Environment Configuration

To create a system resilient to perturbations, the LBO system must be able to describe the configuration of its environment. This description should be sufficient such that any changes affecting the task are captured. For example, if the task were painting different blocks, a description of the environment might include the position, orientation, and size of the blocks. Thus, changes in the environment, by moving blocks, would be conveyed through this description. How this information can be used to handle perturbations in the environment will be discussed in later subsections.

The description of the environment comes from some type of sensor capable of conveying the necessary information, potentially the same sensor that observes the user performing the task. In determining the configuration of the environment, object occlusion will be a problem that must be addressed. If certain objects in the environment are not visible, then the description of the environment configuration is incomplete.

The input to this subsystem is some type of environment-configuration sensor and the output is a description of the environment, $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_M\}$ (*cf.* Figure 4), where $\boldsymbol{\lambda}_i$ describes the ith object in the environment.

3.3 Computing Subgoals

The states sufficient to complete the task are the subgoals. From this perspective, subgoals mark important locations during demonstrations of the task and provide a layer of abstraction so that the LBO system can better interpret the intent of the user. Using the raw state vectors, the LBO system could attempt to repeat the actions of the user verbatim, but this would lead to several problems. First, how a computer system or robot performs a task may not be the same as a human. So, attempting to repeat identically the actions of a human with a system that has fewer degrees of freedom, different controllers, and different *a priori* knowledge may be an ill-posed problem. Also, humans tend to be imprecise and are generally unable to repeat tasks exactly; any system that replays the

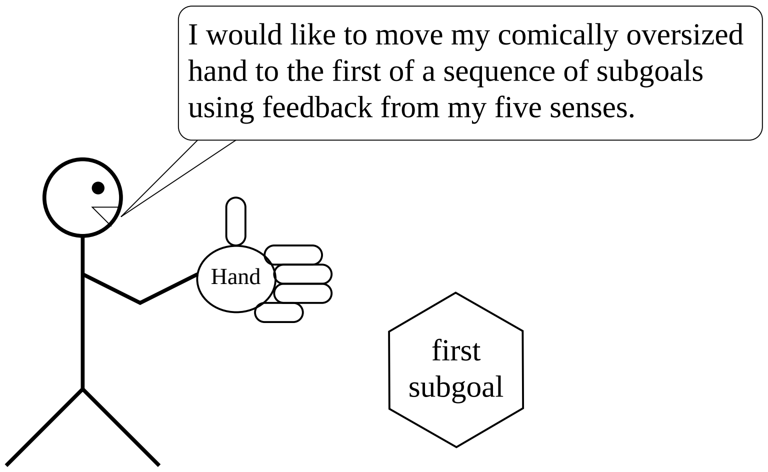

I would like to move my comically oversized hand to the first of a sequence of subgoals using feedback from my five senses.
Hand
first subgoal

Figure 5: High-level view of how humans perform tasks.

sequence of human actions is subjecting itself to the same imprecision. Furthermore, it is quite difficult to determine how *every* action of the user should be modified according to changes in the environment. Estimating fewer parameters generally results in better performance, if the structure of the model is sufficiently complex. Mapping a few subgoals to a new environment should be easier than mapping every action taken, especially since subgoals represent important locations and should be easier to identify in modified environment configurations.

Various researchers have incorporated some notion of subgoals into their LBO systems. Many attempt to segment a demonstration according to a discrete set of predefined primitives (Ikeuchi et al., 1993; Yang et al., 1994; Kang & Ikeuchi, 1995; Friedrich et al., 1996; Matarić, 1999; Bentivegna & Atkeson, 1999). These methods use tools common in the speech-recognition domain (hidden Markov models, dynamic time warping, time-delay neural networks, and *ad hoc* methods), while (Kositsky, 1998) explored numeric clustering techniques. It seems that the purpose of the segmenting and clustering is to abstract away from raw observations and allow the LBO system to analyze the intent of the user more easily. However, attempting to fit observations to a discrete set of symbols ignores the sources of uncertainty outlined in Section 1. Furthermore, it assumes that users will not deviate from the predefined primitives, though allowing additive combinations and parameterizing the primitives does address some of these issues (Matarić, 1999).

Of course, there is no closed-form, optimal solution to determine the *intent* of the user and assumptions must be made to make the problem tractable. We assume that the user follows smooth trajectories between important locations. For example, the user moves her hand to an important location, then toward the next. A high-level example of how a human performs a task is given in Figure 5. There will most likely be some type of discontinuity at the location of the first important location, another at the second, and so forth. The state of the user when the discontinuities occur are the subgoals.

We propose determining subgoals through probabilistic, nonsymbolic methods. Based on the available observations $\boldsymbol{X}_n = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, a model of the actions of the user can be constructed, $\widehat{\boldsymbol{x}}_{n+1} = \boldsymbol{f}(\boldsymbol{X}_n)$, where $\boldsymbol{x}_{n+1}$ is the state of the user at time $n+1$ (the output of the observe user subsystem, Section 3.1), $\widehat{\boldsymbol{x}}_{n+1}$ is an estimate of $\boldsymbol{x}_{n+1}$, and $\boldsymbol{f}(\cdot)$ is the predictive model. Since observations are noise corrupted, any estimate based on

state vector
Compute Subgoals
subgoals

Figure 6: Input/output model of the "Compute Subgoal" subsystem.

these observations will be a random variable, and we can compute the likelihood of $\widehat{\boldsymbol{x}}_{n+1}$ when $\boldsymbol{x}_{n+1}$ becomes available. In general, this likelihood can be expressed as

$$p(\widehat{\boldsymbol{x}}_{n+1}|\boldsymbol{x}_{n+1}, \boldsymbol{X}_n, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}$ represents the parameters of the model describing the prediction error (*e.g.* , a mixture of Gaussians, artificial neural network, etc.). Since it is assumed that the user follows smooth trajectories except at subgoals, drops in the prediction likelihood may be due to the user reaching one subgoal and beginning a trajectory for the next. States where declines in the likelihood occur are the subgoals. This formulation requires only a choice of a prediction error model, $\boldsymbol{\theta}$, but does not require that the observations be segmented according to preconceived "primitives". Also, note that Equation 1 is causal and can be computed online, while the user demonstrates the task. A more-involved example formulation of this problem is given in Section 4.3.

The input to this subsystem is the stream of observations in the form of state vectors. The output is a stream of subgoals and (possibly) the likelihood that the state is, in fact, a subgoal (*cf.* Figure 6).

3.4 Associating Subgoals in the Environment

Most tasks are defined with respect to objects in the environment. For example, arc-welding tasks are defined with respect to the locations of parts to be welded as well as locations to avoid obstacles. In order to achieve the generality required to deal with changes in the environment, subgoals are associated with objects in the environment. As objects are moved, the subgoal locations are changed according to the association relationships. One simple type of association is setting the location of the subgoal to co-occur with the corner of a block. As the block is translated or rotated, the subgoal would be moved accordingly. This subsystem identifies subgoal-environment associations for a single demonstration; refining subgoals from multiple demonstrations is described in Section 3.6.

Associating subgoals in the physical environment was addressed in (Morrow & Khosla, 1995). This association was achieved by determining the corners of objects in the work space from camera images. The user was required to supply the subgoals to the system and these subgoals were manually associated with the corners by the user. As the objects in the work space were altered, a corner-tracking algorithm updated the subgoals for the system accordingly.

We propose to create a system to determine the subgoal-environment associations automatically. Subgoal-environment associations offer substantial benefits to an LBO system. By correctly determining the relationship of the subgoals to objects in the environment, the accuracy of the LBO system is no longer limited by the accuracy of the user, only by its accuracy to determine the configuration of the environment. Also, these associations are one of the primary reasons that the LBO system can handle perturbations in the work space. However, there are

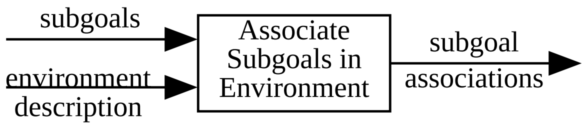


Figure 7: Input/output model of the “Associate Subgoal” subsystem.

several issues to address in making the subgoal-environment associations. Determining the type of relationship assumed to exist between a subgoal and the objects in the environment is paramount to the performance of the LBO system. If this relationship is incorrect, then the benefits mentioned above could harm the performance of the LBO system. For example, if the LBO system assumes that a subgoal is associated with one corner of a block when the relationship should be the average of the nearest three corners, then the subgoal-environment association may cause the LBO system to perform worse. Object occlusion will also be an issue. If an occluded object is needed for a subgoal-environment association, the LBO system must reformulate the association based on available information.

The inputs to this subsystem are subgoals and an environment description. The output is a set of subgoal associations that describe subgoal-environment relationships (*cf.* Figure 7).

3.5 Mapping Demonstrations to the Same Frame

In order to use multiple demonstrations to improve performance, the LBO system must first account for differences caused by the corresponding environment configuration for each demonstration. While there are LBO systems that allow users to demonstrate tasks multiple times, these systems require that the configuration of the environment be identical each iteration (Yang et al., 1994; Kositsky, 1998). Since the user will perform the task differently according to configuration-specific artifacts, the LBO system should map all demonstrations into a canonical, or common, environment configuration, $\mathbf{\Lambda}_c$. After determining the mapping, the LBO system uses the subgoal-environment associations to determine the locations of the subgoals for each demonstration in the canonical environment. At this point, the LBO system essentially is “pretending” that the user demonstrated all tasks in the same environment configuration.

This mapping can be approached in several ways and, in general, these methods optimize an objective function. The type of objective function, linear or nonlinear, determines the algorithms appropriate to use. With a linear objective function, algorithms exist to optimize the function optimally in a reasonable amount of time (Papadimitriou & Steiglitz, 1998). However, a linear objective function may not describe the original intent of the optimization well, so nonlinear objective functions may have to be incorporated. Most nonlinear optimization algorithms are heuristic in nature and tend to be satisfied with local maxima (Duda et al., 2001).

In the general problem setup, shown in Figure 1, an assembly enters environment of the robot and production begins. When completed, the first assembly leaves and the second enters, and so forth. Given these conditions, the mapping algorithm cannot use object-tracking methods, such as (Shi & Tomasi, 1994), because the LBO system is not aware of how the objects were placed on the assembly, since this occurs outside the work space of the robot. This may seem like an artificial restriction, but it is a reasonable assumption in practice. In industrial settings,

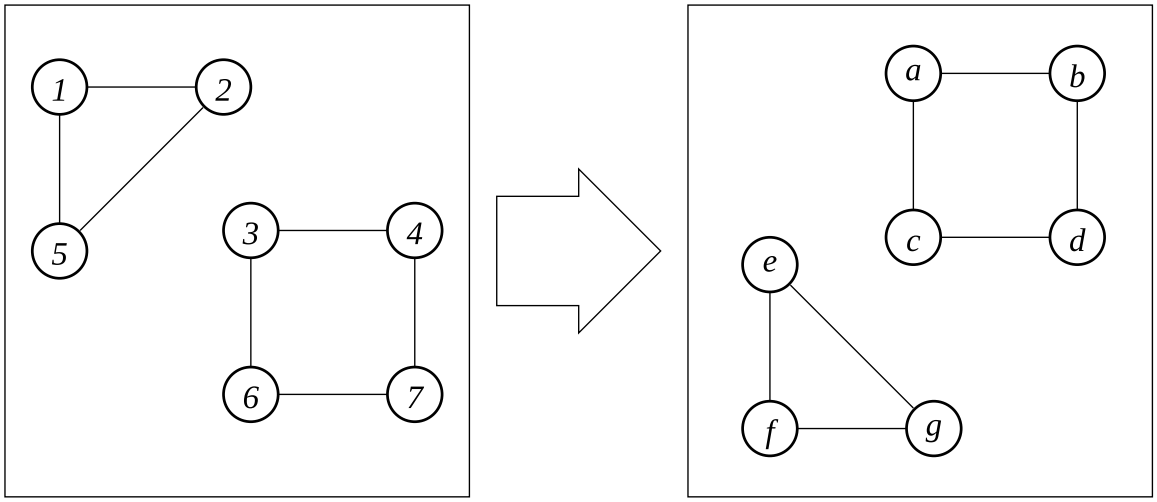


Figure 8: The objects on the left must be mapped to the "canonical configuration" on the right.

the assemblies may be constructed over many hours by different people in different locations. Thus, the only information that should be used to determine the mapping from each demonstration to the canonical configuration is the environment description of the ith demonstration, $\mathbf{\Lambda}_i$, and the canonical configuration, $\mathbf{\Lambda}_c$.

The issues to address in computing the mapping are related to creating an appropriate objective function. The objective function must be able to cope with object occlusion and extraneous objects; it is not reasonable to assume that the mapping will be bijective in all cases. Since we cannot rely on bijective mappings from environment configurations to all demonstrations, then it is not reasonable to assume that all demonstrations can be mapped to an arbitrary canonical configuration (since functions are invertible if and only if the function is bijective (Gilbert & Vanstone, 1993)). So, care must be taken to select a canonical environment description such that all environment configurations can map to it (such that there exists an injective function mapping each environment configuration to the canonical environment configuration).

Also, built into the objective function are implicit assumptions regarding likely perturbations to the environment, such as translations being more likely than rotations. For example, in Figure 8 the objects on the left, $\{1, 2, 3, 4, 5, 6, 7\}$, must be mapped to the objects on the right, $\{a, b, c, d, e, f, g\}$, and there are many plausible solutions. Any objective function that gives preference to one possible mapping over another is implicitly assuming that some mappings are more likely to have occurred than others.

The inputs to this subsystem are the canonical environment configuration and the subgoals, subgoal-environment associations, and environment configuration from one or more previous task demonstrations. The output is the location of the subgoals in the canonical environment configuration for each of the task demonstrations (*cf.* Figure 9).

3.6 Determining Task Structure

Once all the demonstrations have been mapped into a canonical environment configuration, the LBO system can analyze all demonstrations to determine the intent of the user. As stated earlier, tasks are defined to be goal-directed

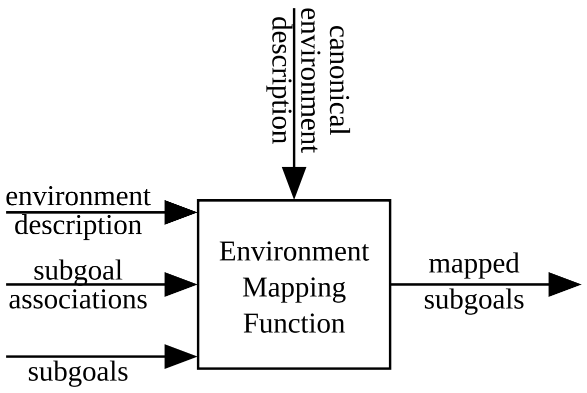


Figure 9: Input/output model of the “Environment Mapping Function” subsystem.

sequences of states and the set of states sufficient to achieve the overall goal are the subgoals. The objective of determining the model structure is to create a finite-state automaton (FSA) that captures the intent of the user. The states of the FSA should be the subgoals of the task and the FSA should terminate only when the overall goal of the task has been attained. Also, the FSA should include branching (*if-then-else*-type) statements to allow the LBO system to recover from errors or unexpected post-conditions. Researchers have also proposed using concurrent execution of states (Matarić, 1999) (parallel automata such as Petri Nets (Murata, 1989) are a subject of much study). There has been much work on finding optimal FSA topographies from example inputs given different assumptions (acyclic, serial, nondeterministic, probabilistic, etc.), such as (Gold, 1967; Rudich, 1985; Rabiner & Juang, 1993; Ron et al., 1998).

Despite the wide range of algorithms for incorporating multiple training examples, there are very few LBO systems that attempt to exploit this information. The precision required by many tasks may be greater than sensors can provide. For high-precision tasks (*e.g.* , vitreoretinal microsurgery (Riviere & Jensen, 2000)) there have been efforts to increase the accuracy of sensor readings, but these methods are prohibitively expensive and impractical for mainstream applications. An LBO system could incorporate multiple demonstrations of the task to achieve greater accuracy than that afforded through a single demonstration. However, incorporating disparate examples from an imprecise and nonrepeatable source like a human is a challenging problem. In (Yang et al., 1994), a Markov chain structure was decided *a priori* and the system was trained using the forward-backward (Baum-Welch) algorithm (Rabiner, 1989). However, (Yang et al., 1994) writes that one of the primary issues yet to be addressed is “selecting a reasonable structure and number of states”. Also, since (Kositsky, 1998) used clustering as the means of determining subgoals, multiple demonstrations were required. Researchers that used predicate-calculus statements in (Ikeuchi et al., 1993; Kuniyoshi et al., 1994; Kang & Ikeuchi, 1995; Friedrich et al., 1996; Bentivegna & Atkeson, 1999) to represent user actions and performed theorem-proving searches to determine the sequence of commands that describes the task.

Once the necessary set of subgoals has been computed through FSA-training methods based on the subgoals from Section 3.3, they are associated in the canonical environment so that they can be used in future environment configurations. The input to this subsystem is the location of the mapped subgoals in a common frame from one or more demonstrations and the canonical environment description. The outputs are a list of subgoals necessary to

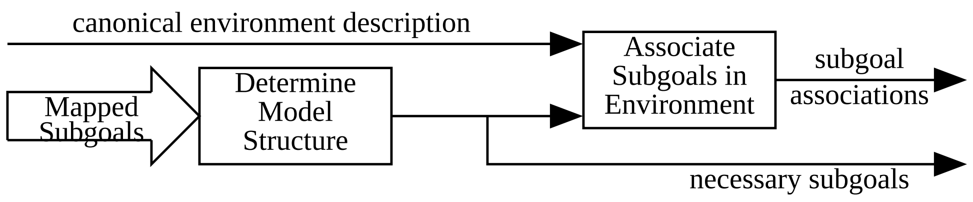

canonical environment description
Mapped Subgoals
Determine Model Structure
Associate Subgoals in Environment
subgoal associations
necessary subgoals

Figure 10: Input/output model of the “Determine Model Structure” subsystem.

complete the task and subgoal associations in the canonical environment (*cf.* Figure 10).

3.7 Performing Task

At this point, the LBO system has determined the set of necessary subgoals, the corresponding task structure modelled as an FSA, and the respective associations of the subgoals in the canonical environment. When the system reproduces the task, it must first map the canonical environment configuration to the current configuration of the environment. This is the reason that environment mapping functions, mentioned in Section 3.5, should be bijective. This allows the LBO system to determine the locations of subgoals in the current environment. When the LBO system begins performing the task, the locations of the subgoals are assumed to be static so that constant sensor feedback from the environment is not necessary. Now, the system must cause the plant to move through the sequence of states activated from the FSA. Because the automaton may include branching statements, information regarding the results of actions must be fed back into the FSA to ensure that the correct branch is taken. If the automaton has concurrent active states then the controller must be able to execute multiple commands or combine commands in a sensible fashion. Depending on the plant, concurrent execution may not be feasible, combining commands may be implausible, and physical limitations may render a subgoal unreachable. All these issues are addressed in the control-system design. Furthermore, it is unlikely that a single controller will achieve all subgoals (consider a system performing linear interpolation when an arc is appropriate). Therefore, the LBO system may need to incorporate controller scheduling (Slotine & Li, 1991), a scheme that allows different controllers to become active when conditions dictate.

The inputs to this system are the set of necessary subgoals, the subgoal associations, the canonical environment description, and the current environment description. The output is a list of locations of the subgoals and the performance of the LBO system (*cf.* Figure 11).

3.8 User Modification

After watching the LBO system perform the task, the user may want to make modifications to the program. The user should be integrated “into the loop” to achieve the required accuracy by fine-tuning subgoals, removing superfluous subgoals, and adding necessary subgoals. This online modification could include classical methods of fine tuning, like a teach pendant or a joystick, or more sophisticated techniques like tactile gestures (Voyles & Khosla, 1995) or gesture-based programming (Iba et al., 1999). The online modifications should serve to improve the way by which the set of necessary subgoals is determined. The user should also be able to demonstrate the

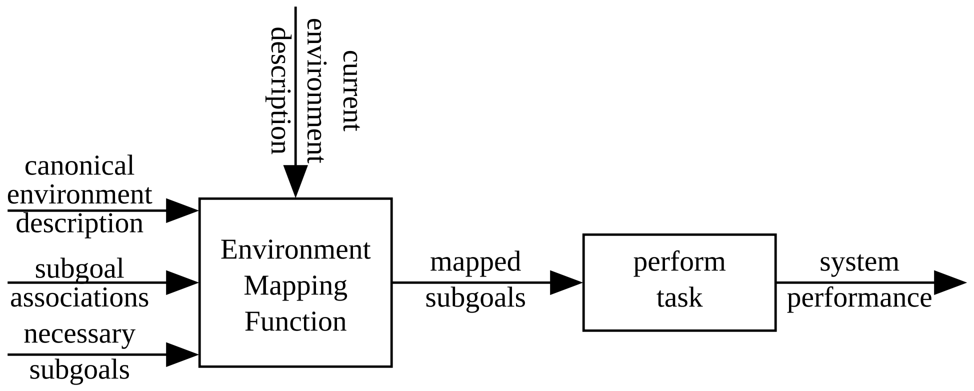

current environment description
canonical environment description
subgoal associations
necessary subgoals
Environment Mapping Function
mapped subgoals
perform task
system performance

Figure 11: Input/output model of the “Perform Task” subsystem.

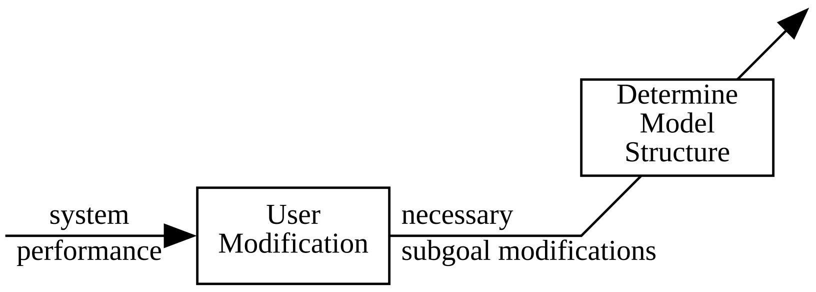


Figure 12: Input/output model of the “User Modification” subsystem.

task again if desired.

Despite the shortcomings of LBO to achieve high precision and exact interpretations of user demonstrations, very few researchers allow a user to modify the output of the system. Researchers in (Friedrich et al., 1996) allowed modification of the parameters for the “filtering” process by which their LBO system determines the structure of the output program. Also, the user could edit predicate-calculus-like statements to remove *superfluous* actions, called “Object Groups”, or add *intended* and *not intended* actions if any were omitted.

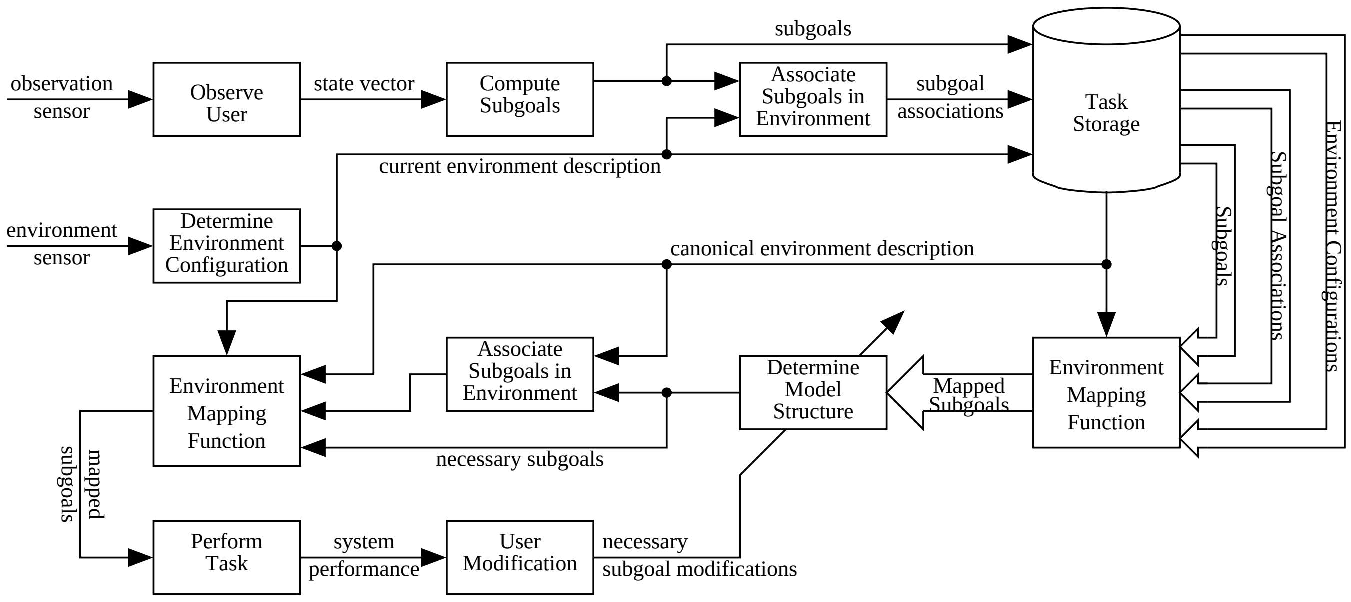
The input to this subsystem is the performance of the LBO system and the output is some set of modifications to the necessary subgoals, used to modify the way by which the model structure is determined (*cf.* Figure 12).

3.9 Data Flow

Figure 13 gives an overview of the data flow in the abstract LBO system using the subsystems described in the preceding subsections.

4 Sample Implementation

We have developed a simulator to demonstrate the viability of some of the ideas in this proposal. The simulator allows users to demonstrate tasks using a mouse in an environment of planar geometric shapes. The tasks that



observation sensor
Observe User
state vector
Compute Subgoals
subgoals
Associate Subgoals in Environment
subgoal associations
Task Storage
current environment description
environment sensor
Determine Environment Configuration
canonical environment description
Subgoals
Subgoal Associations
Environment Configurations
Environment Mapping Function
Associate Subgoals in Environment
Determine Model Structure
Mapped Subgoals
Environment Mapping Function
necessary subgoals
mapped subgoals
Perform Task
system performance
User Modification
necessary subgoal modifications

Figure 13: Data flow in the abstract LBO system.

can be demonstrated consist of sequences of clicking and dragging operations. Once the user has completed the demonstrations, the simulator reproduces the task using a three-degree-of-freedom plant. A screen shot of the simulator is shown in Figure 14. While the simulator is clearly a simplified system, we feel that many of the subsystem implementations are relevant to a complete LBO system.

4.1 Observing the User

The position of the cursor and the state of the mouse button comprise the user observations, which are given directly to the LBO system by the simulator with the state-variable velocities estimated for control purposes. For each time step, the observations are $\boldsymbol{x}_n = [x_{pos}\ \dot{x}_{pos}\ y_{pos}\ \dot{y}_{pos}\ b]^{\mathsf{T}}$.

4.2 Determining the Environment Configuration

The state of the environment is determined from a screen capture and the resulting image is processed using an algorithm to detect corners of shapes in the environment (Deriche & Giraudon, 1991). The environment configuration description is comprised of two-dimensional corner locations.

4.3 Computing Subgoals

A system-level model of the subgoal-estimation method that follows is given in Figure 15. Essentially, we would like to estimate the parameters of a time-varying linear system based on user observations. When the likelihood of predicting the next observation drops dramatically, the current observation is marked as a subgoal.

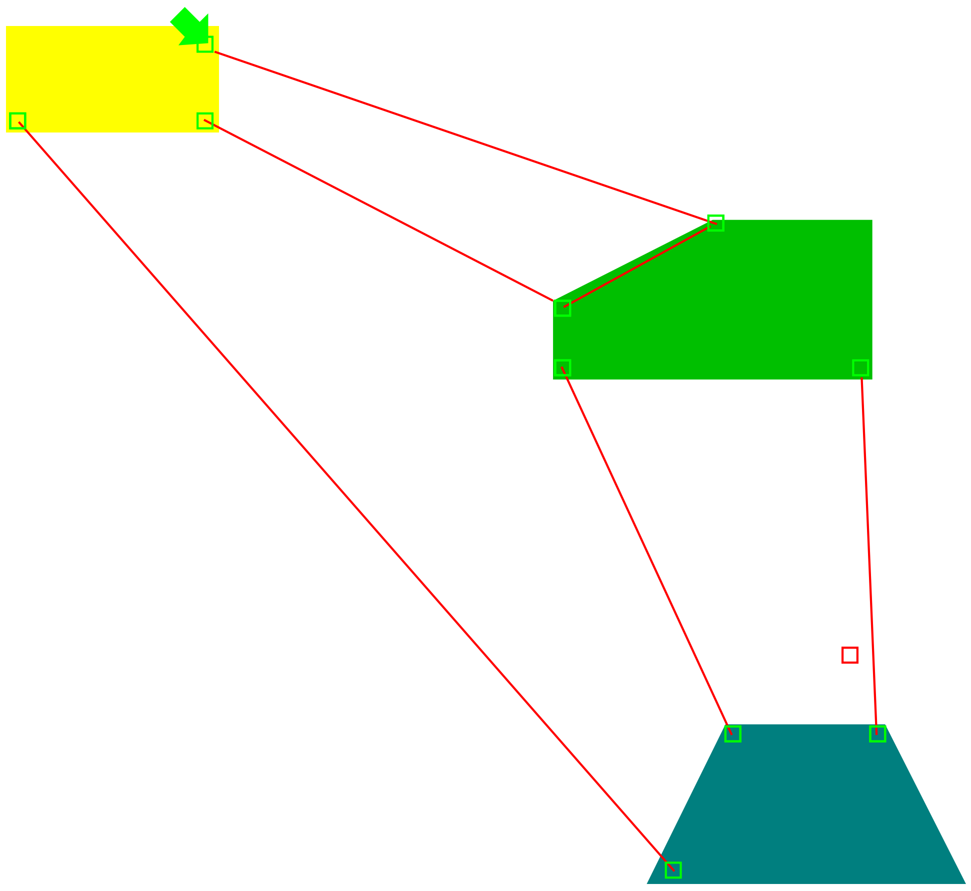


Figure 14: Screen shot of the simulator.

The observations from Section 4.1 are assumed to be generated by the nonlinear difference equation

$$\boldsymbol{x}_n = \boldsymbol{f}(\boldsymbol{x}_{n-1}, \boldsymbol{u}_n) + \boldsymbol{\nu}_n, \tag{2}$$

where $\boldsymbol{x}_n$ is an $(r \times 1)$ state vector, $\boldsymbol{u}_n$ is the $(c \times 1)$ control vector, and the additive Gaussian noise $\boldsymbol{\nu}_n \sim N(\mathbf{0}, \boldsymbol{C})$. The control vector, $\boldsymbol{u}_n$, represents external forces on the hand of the user controlling the mouse, as well as "clicking" the mouse button. The noise term, $\boldsymbol{\nu}_n$, encapsulates the sources of uncertainty mentioned in Section 1. Linearizing Equation 2 at each time step gives

$$\boldsymbol{x}_n = \boldsymbol{A}_n \boldsymbol{x}_{n-1} + \boldsymbol{B}_n \boldsymbol{u}_n + \boldsymbol{\nu}_n, \tag{3}$$

where $\boldsymbol{A}_n$ is the discrete-time Jacobian matrix and $\boldsymbol{B}_n$ is the control-gain matrix. Since subgoals are destination states of the system, we assume that

$$\boldsymbol{u}_n = \boldsymbol{G}_n(\boldsymbol{\gamma}_i - \boldsymbol{x}_{n-1}), \tag{4}$$

where $\boldsymbol{G}_n$ is a closed-loop stable feedback control law matrix, and $\boldsymbol{\gamma}_i$ is the ith subgoal. Essentially, Equation 4

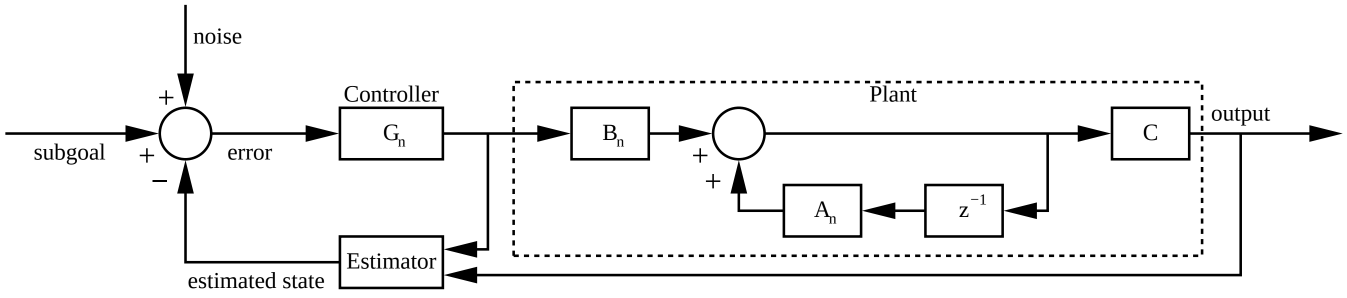


Figure 15: System-level model of human task demonstration.

means that the user exerts forces on her hand to ensure that it moves toward the current subgoal $\boldsymbol{\gamma}_i$. However, once the user reaches the current subgoal, denoted by $\boldsymbol{\gamma}_i = \boldsymbol{x}_n$, she will select another subgoal, $\boldsymbol{\gamma}_{i+1}$, and move toward it. We would like to know when the user makes this switch. Substituting Equation 4 into Equation 3 and verifying that $\boldsymbol{\gamma}_i = \boldsymbol{x}_n$ gives

$$\begin{aligned} \boldsymbol{x}_n &= (\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{G}_n)\boldsymbol{x}_{n-1} + \boldsymbol{B}_n\boldsymbol{G}_n\boldsymbol{\gamma}_i + \boldsymbol{\nu}_n \qquad (5)\\ &= (\boldsymbol{I} - \boldsymbol{B}_n\boldsymbol{G}_n)^{-1}(\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{G}_n)\boldsymbol{x}_{n-1} + (\boldsymbol{I} - \boldsymbol{B}_n\boldsymbol{G}_n)^{-1}\boldsymbol{\nu}_n \\ &= \boldsymbol{R}_n\boldsymbol{x}_{n-1} + \widetilde{\boldsymbol{\nu}_n}. \end{aligned}$$

If there are $N > r$ samples of Equation 3 then we can form the matrix equation

$$\begin{aligned} [\boldsymbol{x}_n \cdots \boldsymbol{x}_{n-N+1}] &= (\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{G}_n)\,[\boldsymbol{x}_{n-1} \cdots \boldsymbol{x}_{n-N}] + \boldsymbol{B}_n\boldsymbol{G}_n\,[\boldsymbol{\gamma}_i \cdots \boldsymbol{\gamma}_i] + [\boldsymbol{\nu}_n \cdots \boldsymbol{\nu}_{n-N+1}] \\ \equiv \boldsymbol{X}_n &= (\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{G}_n)\boldsymbol{X}_{n-1} + \boldsymbol{B}_n\boldsymbol{G}_n\boldsymbol{\Gamma}_i + \boldsymbol{\mathcal{N}}_n. \qquad (6) \end{aligned}$$

Solving for the minimum mean-squared error (MMSE) for the unknowns gives the moving-average estimates

$$\begin{aligned} \widehat{\boldsymbol{R}} &= \boldsymbol{X}_n\boldsymbol{X}_{n-1}^R, \\ \widehat{\boldsymbol{BG}} &= (\boldsymbol{X}_n - \widehat{\boldsymbol{R}}\boldsymbol{X}_{n-1})(\boldsymbol{\Gamma}_i - \widehat{\boldsymbol{R}}\boldsymbol{X}_{n-1})^R, \\ \widehat{\boldsymbol{A}} &= \widehat{\boldsymbol{R}} + \widehat{\boldsymbol{BG}}(\boldsymbol{I} - \widehat{\boldsymbol{R}}), \qquad (7) \end{aligned}$$

where $\boldsymbol{X}^R$ denotes the right pseudo-inverse of $\boldsymbol{X}$. These MMSE equations assume that the current state is the subgoal. If the user has selected another subgoal then the older samples of the moving-average Equation 6 assume that the reference is $\boldsymbol{\gamma}_i$ while the newer samples assume the reference is $\boldsymbol{\gamma}_{i+1}$, which will result in a large noise term, $\widetilde{\boldsymbol{\nu}_n}$. The likelihood of the noise corrupting the model is found from Equation 5 using the MMSE estimates,

$$\begin{aligned} p(\widetilde{\boldsymbol{\nu}_n}) &= \mathcal{N}(\widetilde{\boldsymbol{\nu}_n}, \widetilde{\boldsymbol{C}})r \\ &= \mathcal{N}(\Big((\boldsymbol{I} - \widehat{\boldsymbol{BG}})\boldsymbol{x}_n - (\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{BG}})\boldsymbol{x}_{n-1}\Big), \widetilde{\boldsymbol{C}})r. \end{aligned}$$

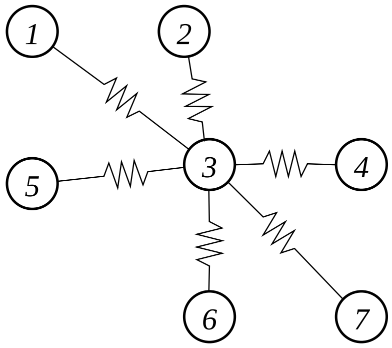


Figure 16: Intuitive representation of the "spring" optimization.

When the user changes subgoals, the likelihood $p(\widetilde{\boldsymbol{\nu}_n})$ falls dramatically such that

$$\frac{p(\widetilde{\boldsymbol{\nu}_n})}{p(\widetilde{\boldsymbol{\nu}_{n-1}})} < \tau,$$

where $\tau \in (0, 1)$ is a pre-determined threshold. Since the user changed subgoals $\boldsymbol{\gamma}_i \neq \boldsymbol{x}_n$, therefore $\boldsymbol{\gamma}_i \equiv \boldsymbol{x}_{n-1}$, and this vector is saved for further processing. We have found empirically that a threshold of $\tau = 0.1$ works well when $N = 10$ with the state vector described in Section 4.1.

4.4 Associating Subgoals in the Environment

We assume that the user makes errors which follow a Gaussian distribution while demonstrating tasks. The probability that a subgoal occurred at each corner is computed for all subgoals. If the most-likely probability exceeds a threshold then we assume that the user meant to place the subgoal on the corner and the subgoal is associated with the corner outright. Otherwise, the subgoal is probably intended for obstacle avoidance and depends on the locations of several nearest corners. We compute an association such that the position of the subgoal remains unchanged but is linearly related to the locations of the four most-likely corners. If any of those corners move, it would affect the position of the subgoal. In Figure 14, small green squares represent subgoals associated directly with corners, while small red squares represent subgoals associated with the four nearest corners (hopefully, this copy of the proposal is in color).

4.5 Mapping Demonstrations to the Same Frame

Between iterations of demonstrating the task or allowing the LBO system to perform the task, the user may move objects around in the simulator. In order to complete the task successfully, the LBO system must determine which corners in an earlier environment description map to those in a later one. Suppose there were two configurations of objects, such as in Figure 8. The mapping function must map each of $\{1, 2, 3, 4, 5, 6, 7\}$ to one of $\{a, b, c, d, e, f, g\}$. One possible formulation is to consider each corner connected to all others by "springs", for example corner 3 is connected to all other corners in Figure 16.

The spring constant is given by

$$k_{i,j} = \|\boldsymbol{\lambda}_j - \boldsymbol{\lambda}_i\|_{\boldsymbol{Q}}^{-2}, \tag{8}$$

where $\boldsymbol{\lambda}_i$ is the location of the ith corner and $\boldsymbol{Q}$ is some symmetric weighting matrix. This formulation puts stiff springs on close corners and weak springs on distant ones. The resulting force on the ith corner in the nth demonstration is

$$\boldsymbol{f}_{i,n} = \sum_{\substack{\boldsymbol{\lambda}_j \in \boldsymbol{\Lambda}_n \\ \boldsymbol{\lambda}_j \neq \boldsymbol{\lambda}_i}} \frac{k_{i,j}\left(\boldsymbol{\lambda}_j - \boldsymbol{\lambda}_i\right)}{\|\boldsymbol{\lambda}_j - \boldsymbol{\lambda}_i\|_{\boldsymbol{Q}}}.$$

If the ith corner were mapped to the jth corner in a subsequent demonstration, then the magnitude change in force caused by this mapping is

$$c_{i,j} = \left\lceil \|\boldsymbol{f}_{i,n} - \boldsymbol{f}_{j,n+1}\| \right\rceil, \tag{9}$$

where $\lceil a \rceil$ is the ceiling operation on a, which rounds $a \in \mathbb{R}$ up to next greatest integer. We would like to find a mapping such that the *total* change in force is minimized. Thus, Equation 9 is the objective function in this optimization problem. From here, we can formulate the problem into a weighted bipartite-graph matching problem, which can be solved by Linear Programming (optimality for this formulation shown by (Heller & Tompkins, 1956)) and we currently use the two-phase Simplex algorithm.

Although the Simplex is, worst case, an exponential algorithm, we have found empirically that this problem formulation does not lead to a computational explosion. We have computed bijective mappings for 50 corners in a reasonable amount of time. Problems of smaller and possibly more realistic size, about 15 corners, can be computed in near real time. However, this formulation does not always return an acceptable mapping, particularly when the environment changes drastically between iterations. Also, the Linear Programming formulation can only compute mappings when objects are not occluded. To address these two issues, more sophisticated nonlinear optimization formulations may have to be developed. More information may also be used to determine the mapping correctly. This description could include information like edges, color, or texture.

4.6 Determining Task Structure

Several methods have been investigated to determine an appropriate algorithm to construct an FSA that captures the intent of the user given subgoals from multiple demonstrations. One such approach was similar to recent research in automatic phoneme generation in speech recognition (Singh et al., 2001). This approach trains an HMM based on transcripts of speech utterances to determine the atomic units of the spoken language. However, these algorithms require transcripts of the speech, as well as a large cross-validation set to gauge when the recognizer has determined an appropriate HMM. If an incorrect structure for the HMM is constructed, then the forward-backward HMM training method places the target subgoals at meaningless averages in the state space of the subgoals. Since LBO systems lack a large transcribed corpus, this method never performed adequately and took several hours for

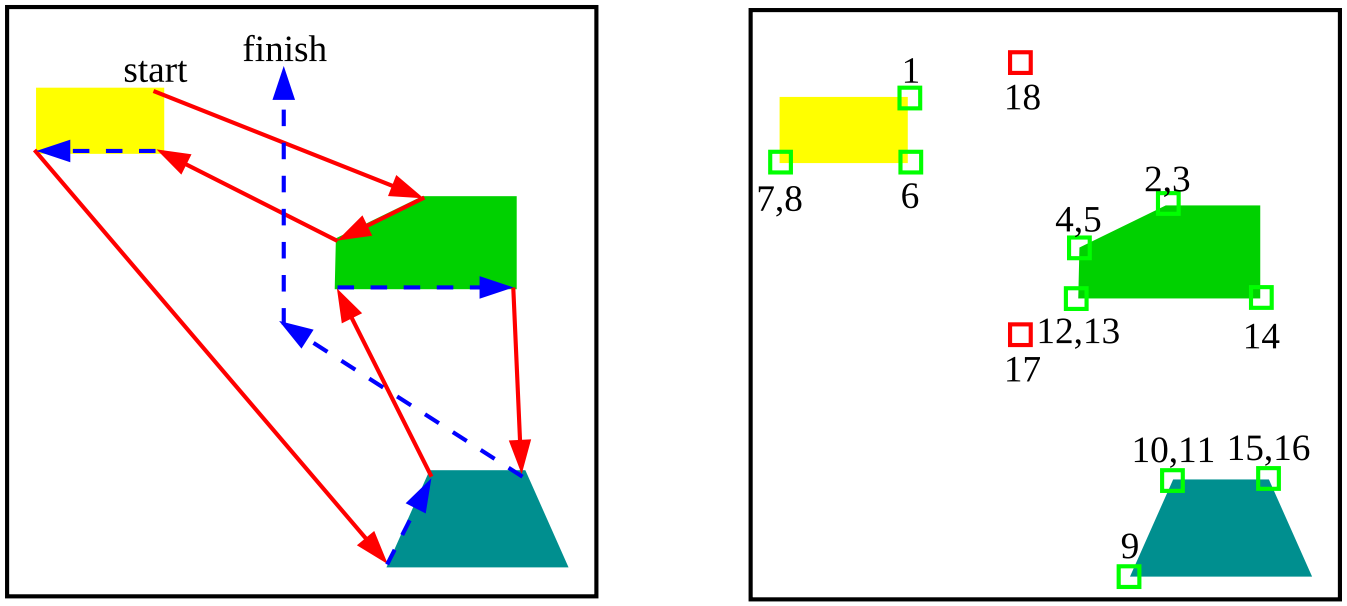


Figure 17: Target path for users and the corresponding location of subgoals, determined by hand.

moderately sized tasks. Another, more promising, method involves determining the structure of a special class of FSAs called Acyclic Probabilistic Finite Automata (APFAs). An algorithm, presented in (Ron et al., 1998), gives a method by which systems can compute the structure of an acyclic FSA based on stochastic inputs. With some modification, the algorithm has been applied to the LBO problem of determining the structure of an APFA based on the subgoals from multiple demonstrations. Essentially, the modified algorithm looks for clusters in the state space of the subgoals with consideration given to their temporal ordering in the demonstrations. States in the APFA can be merged based on similarity statistics of the clusters. The resulting APFA is used to control the simulated plant to perform the task. Based on tasks performed in the simulator, this training algorithm performs well in the sense that the states of the APFA (the set of necessary subgoals mentioned in Section 3.6) are the intended destination states for the system and training takes less than a second for moderately sized tasks.

An example task is shown in Figure 17. The path is shown on the left, where solid red arrows mean click and drag the mouse while dashed blue lines mean move without clicking. The necessary set of subgoals, determined by hand, is shown on the right of Figure 17. The locations of all subgoals collected from various users in twenty demonstrations are shown on the left of Figure 18. The user demonstrations were given to the APFA training algorithm described earlier. The simulator is using the APFA to reproduce the task on the right of Figure 18, where the small squares in the figure represent the subgoals that the APFA training algorithm determined were necessary to complete the task. The subgoals from the APFA training algorithm appear quite similar to the hand-crafted subgoals in Figure 17. The only discernable differences are that the APFA training algorithm placed the obstacle-avoidance subgoals at incorrect locations, though all obstacles were avoided and the overall goal of the task was still accomplished.

Quantifying the amount of similarity between subgoal sets is a difficult procedure. One possible method is to consider the hand-crafted set of subgoals to be a “target string” and the output of the APFA algorithm a “test

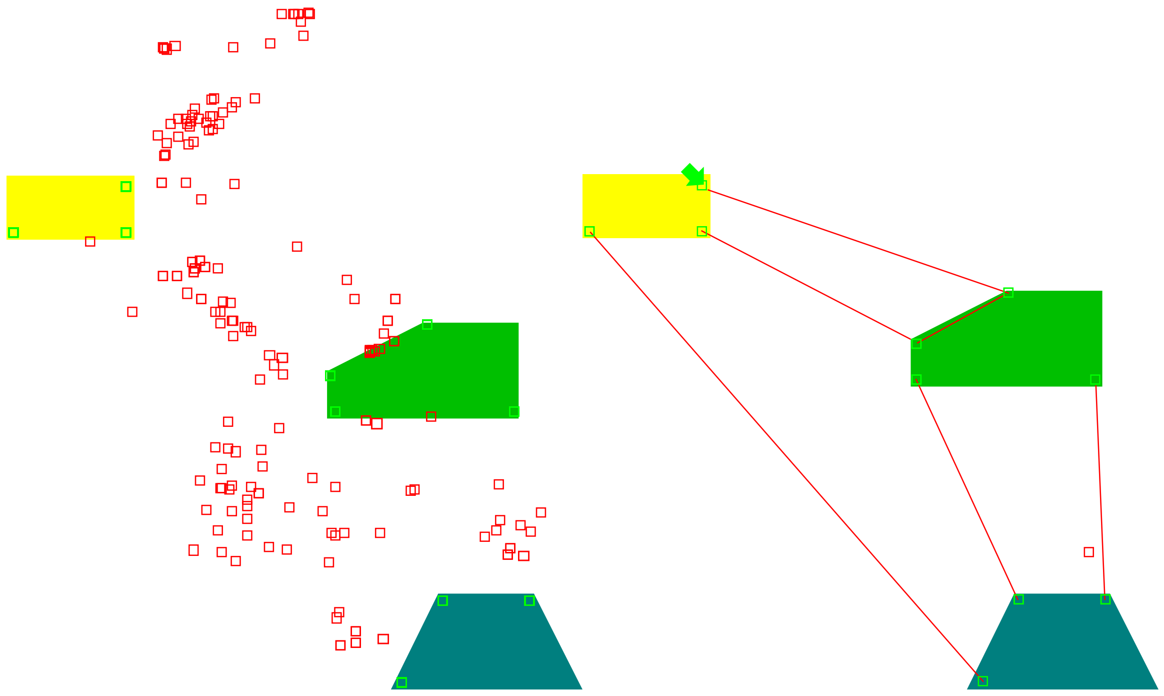


Figure 18: Locations of subgoals from 20 user demonstrations and APFA training algorithm output.

string", with each state or subgoal being a letter in the string. The normalized string edit distance can provide a disparity measure between these two strings (Marzal & Vidal, 1993). Intuitively, this distance is the number of subgoals that must be added, modified, or removed to get the output of APFA algorithm to be equal to the hand-crafted set, normalized by the length of the string. For the APFA algorithm, the normalized string edit distance is 0.11, roughly meaning that 11% of the subgoals must be modified to get an exact match. Using the HMM-based method mentioned previously, the normalized distance is 0.95, which is a miserable result meaning that 95% of its subgoals would have to be modified to get an exact match. While the normalized string edit distance gives some measure of performance, there are some obvious drawbacks. It cannot quantify the *amount* of change that each subgoal must be modified to get a match, only whether modification is needed. Also, it does not consider how well the system interpolates between subgoals, or even if the overall goal of the task was accomplished.

4.7 Performing Task

To perform the task, the simulator uses a dynamical system with the ability to move and click and drag. The plant is the small, green arrow in the top-left of Figure 14. The plant is governed by the differential equations

$$\begin{aligned} \ddot{x}_{pos} &= -b_x m(\dot{x}_{pos} + \dot{y}_{pos}) + u_x/m, \\ \ddot{y}_{pos} &= -b_y m(\dot{x}_{pos} + \dot{y}_{pos}) + u_y/m, \\ \dot{b} &= -b + u_b, \end{aligned}$$

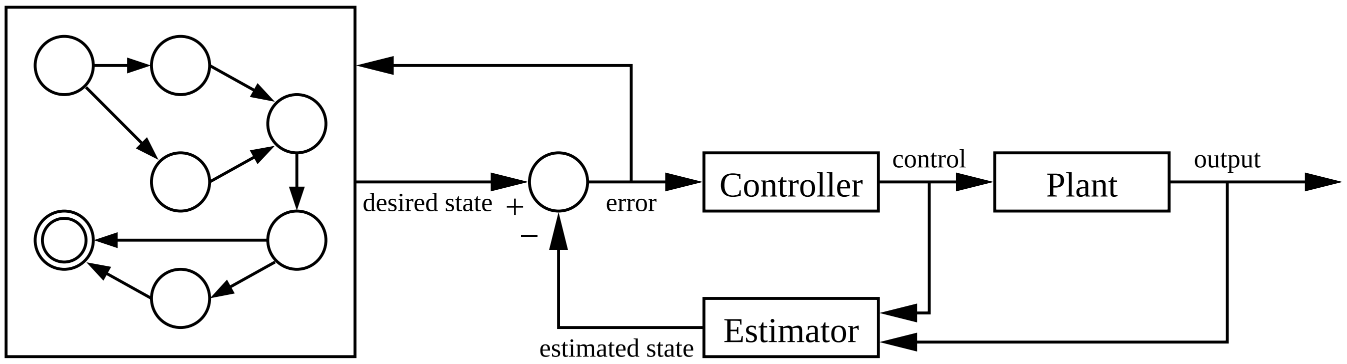


Figure 19: General view of the production model for the simulator.

where u_x, u_y, and u_b represent the controls for the x, y, and mouse-button coordinates respectively, b_x and b_y are directional damping-type coefficients, and m is the mass of the plant. Only the position of the coordinates of the plant are measurable, so a full state estimator was created (Stengel, 1986) because the closed-loop stable PD controller designed for this plant requires full-state feedback. An FSA representing the necessary set of subgoals serves as a reference to the PD controller. When the state error reaches an acceptable level, the FSA transitions to the next state until the task is completed. The production model for the simulator is shown in Figure 19.

5 Proposed Work

Using the experience gained from working with the simulator, we would like to build on our existing algorithms and transition to real-world applications. While we feel that some of the algorithms used in the simulator will scale to the final system, others will have to be reformulated. Future efforts in this proposed research will be split amongst theoretical and applied work.

A large part of the theoretical aspect will involve determining the underlying FSA structure that describes observations from multiple demonstrations. The performance of the overall LBO system is tightly coupled with this subsystem and previous work in this area does not address many of the most important issues. The APFA training algorithm, outlined in Section 4.6, performs well but development of the theory behind the algorithm will create a better understanding of its uses and limitations.

Another area of work is to create a robust algorithm to map demonstrations to a common environment configuration. The current formulation makes the unrealistic assumption that object occlusion cannot occur. We propose to develop an algorithm that addresses object occlusion. This algorithm will be a nonlinear-optimization problem and creating the objective function will require some time. Current avenues being explored are based on topography matching subject to affine transformations, graphically similar to a Kohonen Network.

We also proposed to develop a robust metric that describes the similarity between two demonstrations of a task. This metric must consider if the overall goal of a task was achieved, the relative weight for interpolating between subgoals, and a meaningful quantity describing how similar disparate subgoals are.

Incorporating users into the design loop on the LBO system somewhat blurs the line between theory and applied research, but is important to the success of the system. Future work in this area will involve determining practical ways that the user can aid the LBO system to improve performance. Specifically, gesture-based programming seems to be the most-appealing modal input for users. However, an area of active research is multi-modal interfaces, such as combining speech and gesture recognition. These interfaces could be incorporated as a way of giving users more input to an LBO system.

On the applied side of the proposed work, the target task is arc welding. While manual arc welders have extensive domain expertise, most have never used a computer and virtually none have computer-programming experience. The process that human arc welders perform to accomplish a task is much different than the way that a robot arc welder would behave performing the same task. Improving the programming process for arc welding will appeal to an industry that, so far, has resisted automation. As stated earlier, 98% of arc welding is still performed by hand. Because of these reasons, we feel that arc welding encompasses many of the most interesting issues and the results will be widely applicable. Machine-vision algorithms are central to the LBO system and much development is needed. We plan on using off-the-shelf equipment as much as possible, but there will inevitably be time-consuming integration issues.

6 Evaluation of Research

We expect to evaluate the theoretical aspects of the proposed research using limited cross-validation sets. While requiring an assembly to be arc welded several hundred times defeats our motivation, the CV sets must only be used for the purpose of validating the subsystems of this research. However, algorithms developed for the LBO system will not rely on large CV sets to perform adequately. Feedback from domain experts using the system to automate arc-welding tasks will serve to evaluate the LBO system from the practical side. Their comments will be used to refine the algorithms of the LBO system address application-specific concerns and to determine the viability for practical uses of the system.

6.1 Expected Contributions

We expect to demonstrate that an LBO system is a viable automation tool and will allow users to “program” automation tasks without relying on a trained computer programmer to transfer their domain knowledge. Part of being a viable automation tool requires resilience to perturbations in the environment. Most previous LBO systems do not address this issue, and we feel that preliminary results from the simulator are encouraging. Another key aspect to move LBO into mainstream use is dealing with uncertainty. Our system is designed specifically to address the uncertainty inherent in LBO.

We expect that the method for determining subgoals in a non-symbolic fashion could find its way into different domains. Clearly this algorithm is useful in automation tasks, but there are potential applications to segmenting speech for better phoneme alignment in the speech-recognition domain. We also expect that incorporating multiple demonstrations to improve system performance will prove useful in many areas, such as Intelligent Assistant Agents (Lashkari et al., 1994). These agents assimilate multiple examples of user behavior and attempt to help the user automate daily computer-based tasks. Devising an algorithm that determines the underlying structure of the task in a reasonable amount of time with a small set of examples would be extremely useful, especially if the theoretical aspects of the algorithm are well understood, and we fell that the APFA training algorithm is a good first step.

6.2 Preliminary Timetable

Task	**Start date**	**End date**	**Duration**
Verification of algorithms and concepts on toy applications and simulators	May 01	August 01[1]	4 months
Development and integration of stereo vision and object tracking algorithms	September 01	November 01[1]	3 months
Controlled experiments and verification on robot manipulators	December 01	March 02	4 months
Experiments with domain experts and incorporation of their feedback	April 02	July 02	4 months
Write report	July 02	September 02	3 months
Total	May 01	September 02	18 months

Table 1: Preliminary timetable of research milestones.

The first step in this research, verification of algorithms, is to ensure that the concepts outlined in this proposal are viable. The allotted time is to devise follow-on algorithms, as well as finishing the undeveloped sub-systems.

The second step, machine-vision algorithm development, is the beginning of the transition to real robots. These algorithms are critical to the success of the LBO system, since our system depends on visual input for observations. These algorithms must be able to determine the configuration of three-dimensional work spaces with a high degree of accuracy track the user in real time. Off-the-shelf vision systems will be considered, as will collaboration with other researchers.

For the next step, controlled experiments on robots, the entire system will be functioning on manipulators. The experiments will focus on isolating and evaluating the performance of specific components of the LBO system and incorporating improvements. In last stage of development, experiments with domain experts, the performance of the entire system will be compared to existing methods. Furthermore, feedback from experts will be refine the LBO system.

[1]The author will be on leave from Carnegie Mellon University at ABB Corporate Research in Västerås, Sweden until the end of 2001.

References

Bentivegna, D. C., & Atkeson, C. G. (1999). Using primitives in learning from observation: A preliminary report. *Eighth AAAI Mobile Robot Competition Workshop* (pp. 40–46).

Cypher, A. (Ed.). (1993). *Watch what i do: Programming by demonstration*. Cambridge, MA: MIT Press.

Deriche, R., & Giraudon, G. (1991). *Accurate corner detection: An analytical study* (Technical Report 1420). Robotics, Image, and Vision Program, INRIA.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. Wiley-Interscience. Second edition.

Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Friedrich, H., Münch, S., Dillmann, R., Bocionek, S., & Sassin, M. (1996). Robot programming by demonstration (RPD): Supporting the induction by human interaction. *Machine Learning*, *23*, 163–189.

Gilbert, W. J., & Vanstone, S. A. (1993). *Classical algebra*. Waterloo, Ontario, Canada: Waterloo Mathematics Foundation. Third edition.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, *10*, 447–474.

Heller, I., & Tompkins, C. (1956). An extension of a theorem of Dantzig's. *Linear Inequalities and Related Systems* (pp. 247–252). Princeton, NJ: Princeton University Press.

Iba, S., Vande Weghe, J. M., Paredis, C., & Khosla, P. (1999). An architecture for gesture based control of mobile robots. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 851–857).

Ikeuchi, K., Kawade, M., & Suehiro, T. (1993). Towards assembly plan from observation: Task recognition with planar, curved and mechanical contacts. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2294–2301).

Kang, S. B., & Ikeuchi, K. (1995). A robot system that observes and replicates grasping tasks. *IEEE Fifth International Conference on Computer Vision* (pp. 1093–1099).

Kositsky, M. (1998). *Motor learning and skill acquisition by sequences of elementary actions*. Doctoral dissertation, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science.

Kuniyoshi, Y., Masayuki, I., & Inoue, H. (1994). Learning by watching: Reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, *10*, 799–822.

Lashkari, Y., Metral, M., & Maes, P. (1994). Collaborative interface agents. *Proceedings of AAAI Conference*.

Marzal, A., & Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*, 926–932.

Matarić, M. J. (1999). Visuo-motor primitives as a basis for learning by imitation. *Imitation in Animals and Artifacts*. Cambridge, MA: MIT Press.

Morrow, J. D., & Khosla, P. K. (1995). Sensorimotor primitives for robotic assembly skills. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1894–1899).

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*, 541–580.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola, New York: Dover Publications. Second edition.

Pomerleau, D. (1994). *Neural network perception for mobile robot guidance*. Doctoral dissertation, Carnegie Mellon University.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Rabiner, L. R., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall.

Riviere, C. N., & Jensen, P. S. (2000). A study of instrument motion in retinal microsurgery. *Proceedings of the 22nd International Conference of the IEEE Engineering in Medicine and Biology Society*.

Ron, D., Singer, Y., & Tishby, N. (1998). On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, *56*.

Rudich, S. (1985). Inferring the structure of a Markov chain from its output. *IEEE Symposium on Foundations of Computer Science* (pp. 321–326).

Schaal, S. (1997). Learning from demonstration. *Advances in Neural Information Processing Systems* (pp. 1040–1046).

Shi, J., & Tomasi, C. (1994). Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*.

Singh, R., Raj, B., & Stern, R. M. (2001). Automatic generation of sub-word units for speech recognition systems. Submitted to *IEEE Transactions on Speech and Audio Processing*.

Slotine, J.-J. E., & Li, W. (1991). *Applied nonlinear control*. Englewood Cliffs, New Jersey: Prentice Hall.

Stengel, R. F. (1986). *Stochastic optimal control: Theory and applications*. New York: Wiley-Interscience.

Thrun, S. B. (2001). Probabilistic algorithms in robotics. To appear in *AI Magazine*. Also appeared as Carnegie Mellon University School of Computer Science Technical Report CMU-CS-00-126.

Voyles, R. M., & Khosla, P. K. (1995). Tactile gestures for human/robot interaction. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 7–13).

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.

Yang, J., Xu, Y., & Chen, C. S. (1994). Hidden Markov model approach to skill learning and its application to telerobotics. *IEEE Transactions on Robotics and Automation*, *10*, 621–631.