6075

AIAA-94-1232-CP

An Intelligent Robotic Aid System for Human Services

N94-30580

K. Kawamura, S. Bagchi, M. Iskarous, R. T. Pack, and A. Saad
Center for Intelligent Systems
Box 1804, Station B Vanderbilt University
Nashville, TN 37235

207677
P. 8

Abstract

The long term goal of our research at the Intelligent Robotic Laboratory at Vanderbilt University is to develop advanced intelligent robotic aid systems for human services. As a first step toward our goal, the current thrusts of our R&D are centered on the development of an intelligent robotic aid called the ISAC (Intelligent Soft Arm Control). In this paper, we describe the overall system architecture and current activities in intelligent control, adaptive/interactive control and task learning.

I Introduction

The goal of our current research is to develop an intelligent robotic aid system for the service sector such as hospitals and home. The main benefit of such a system is to provide the sick and physically challenged person with means to function more independently at home or work place. As a first step toward our goal, we have developed a prototype robotic aid system called the ISAC (Intelligent Soft Arm Control).[1] To insure ease of use, safety, and flexibility of the system, we have integrated several sensors such as vision, voice, touch and ultrasonic ranging. The user interacts with the system in natural language like commands such as *'feed me soup.'* Other related R&D activities being conducted include the development of an ISAC/HERO cooperative aid system with a HERO 2000 mobile robot to extend the system capabilities and work on a flexible microactuator robotic hand. In this paper, the overall system architecture is first presented. Next, the construction and performance of a parallel controller is described, followed by a discussion on various command interpreters and reflex control. Very preliminary results from recently constructed macro action builder and task learning module follow to illustrate the ease of use. We conclude with a discussion of the remaining technical issues needed to be addressed.

II System Architecture

ISAC is a robotic aid system for feeding the physically handicapped. It uses a unique manipulator called Soft Arm. The Soft Arm is a pneumatically-actuated manipulator. It is lightweight and suitable for operation in close proximity to humans. The actuators are fiber-reinforced rubber tubes called *rubbertuators*, whose length depends on the pressure of the air inside the tube. Two rubbertuators control a joint in much the same way as human muscles.

The feeding task requires the recognition and the location of objects such as spoon, fork, and bowl on the table so that the arm can manipulate them. These objects are recognized from an image taken by an overhead camera. The recognition is independent of the size and orientation of the objects, a requirement characteristic of a normal feeding environment where utensils of different sizes are present at various orientations. ISAC also uses stereo cameras to track the face of the user in 3-D. This allows the arm to reach the mouth of the user even when he moves his head. Real-time face tracking also allows the detection of a sudden motion of the user. This could be caused by a sneeze or a muscle spasm. In such a case, the arm uses reflex action to move away from the path of the user.

Figure 1 illustrates the integrated hardware/software configuration of the ISAC system.[2] As shown in the figure, ISAC has a distributed architecture. The distributed architecture will allow us to easily add new modules and, therefore, new functionality.[3] The breakdown of one module will not halt the system, but rather, it will only result in a degradation of the activities that the system as a whole could previously perform.

The nature of communication between the modules must be such that each can exist without assuming the existence of other modules. This is achieved with a "blackboard," which is a means of indirect communications between modules.[4] Whenever a module requires a service to be performed by some other module, it posts the request to the blackboard. The requests in the blackboard are monitored by the modules, which perform the ones they are capable of. A brief description of key modules in ISAC are given below.

Object Recognition The object recognition module captures an image of the environment and identifies the location of all recognizable objects. The recognition algorithms used in this module is described in Bishay *et al.*[5] Recognition is model based, using a normalized distance histogram to find the best

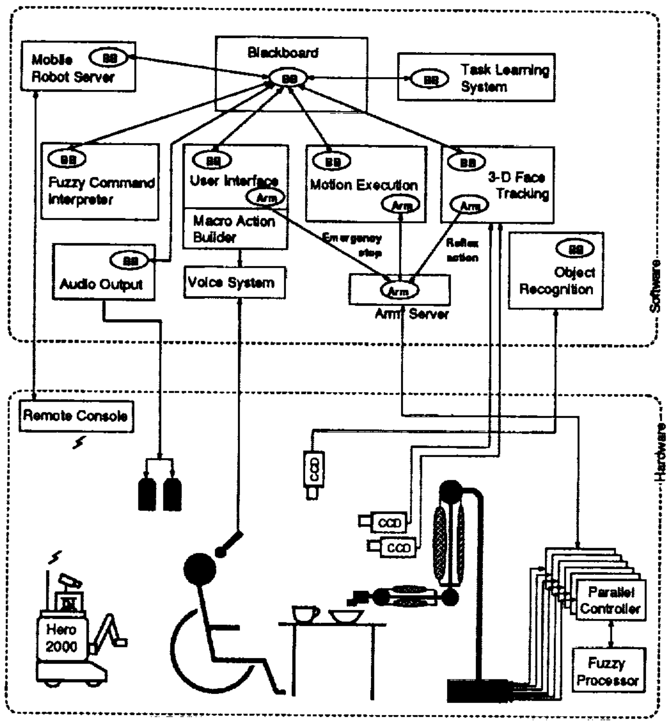

Fig. 1. Integrated ISAC architecture.

match with histograms in a database of possible target objects. An orientation histogram is used to determine the orientation of the object. Example objects in the environment are spoon, fork, knife, cup, and bowl. The recognition algorithm is robust enough to recognize various types and sizes of generic objects such as a spoon.

Face Tracking The face tracking module uses a pair of cameras to determine the position of the face in 3-D. The forehead of the user is tracked by both cameras. The disparity between the 2-D position obtained from each camera provides the third dimension: the distance of the user from the camera. Details of the tracking algorithm and camera calibration are described in Ernst *et al.*[6] In addition to specifying the position of the user's face for accurate feeding, the face tracking module can also detect any sudden motion made by the user. If the direction of the motion is towards the arm, such that a collision with the arm is possible, the arm is moved away from the user in a reflex action.

Voice Recognition A voice recognition system replaces the keyboard as the main user interface. Currently we are using the IN^3 commercial voice recognition system. It is running in parallel with the planning process, allowing the user to intervene the task execution if necessary.

Parallel Control The Soft Arm is controlled by a transputer-based parallel controller. It uses a network of transputers that can be reconfigured in case of a fault in the controller. Details of this module is given in Section III. We are developing a control system which can learn the best control strategy using a neural network and fuzzy logic. The neural network will be used to generate the knowledge base which will be used by the fuzzy controller.

Macro Action Builder This module acts as a voice-based "teach pendant" for the system designer or user. It provides the user with the ability to teach ISAC new actions and later retrieve them. It also allows the user to use fuzzy and context dependent commands such as *move closer*. This module enhances the extensibility of the ISAC's tasks as described in Section V.

Task Learning This module, currently under development, adds the capability of learning from obeying user commands and observing their effects. While the action building facility allows the system to learn *how* to perform an action, this module learns *when* and *why* to perform it, allowing the system to learn how to plan. The learning mechanism is described in Section V.

The ISAC system identifies some key requirements of service robot systems. These form the objectives of many research areas such as control, user interface, planning, and learning.[7,8] The following sections highlight these important research issues and describe the work being performed in greater detail.

III Intelligent Control

Parallel Controller

One of the key issues in the ISAC system is intelligent control. Since the Soft Arm exhibits a highly nonlinear joint dynamics due to rubbertuators[9], a special controller is needed to allow different control techniques to be used. Currently a transputer-based parallel controller is designed and implemented to control the Soft Arm as shown in Figure 2. It consists of a network of eight transputers. Each joint of the Soft Arm is connected to one transputer as its joint controller. A master transputer is then used to communicate with the host computer and supervise the joint controllers. The master node contains the robot command interpreter and the kinematic model of the Soft Arm.[10] It is also connected to the fuzzy processor FP-3000 which acts as a fuzzy coprocessor for control and path planning.

Currently, a PID controller is implemented in each joint node. A cubic spline is generated as the trajectory for the joint motion to follow. This reduces the amount of energy stored while moving the joint, thus allowing the speed to be increased without considerable jerky motion due to the nonlinearities of the rubbertuator joint.[11] The controller can access all the motion data and issue a new command at any time even when

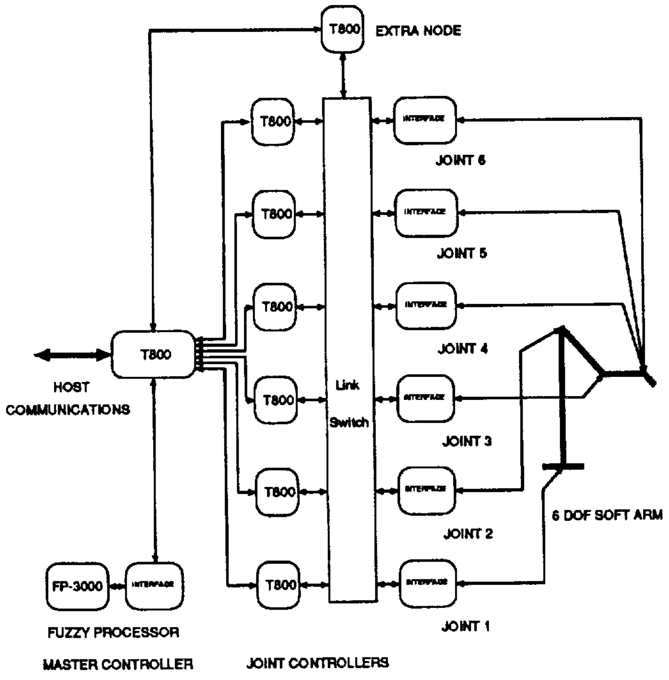

Fig. 2. The Transputer-based Parallel Controller.

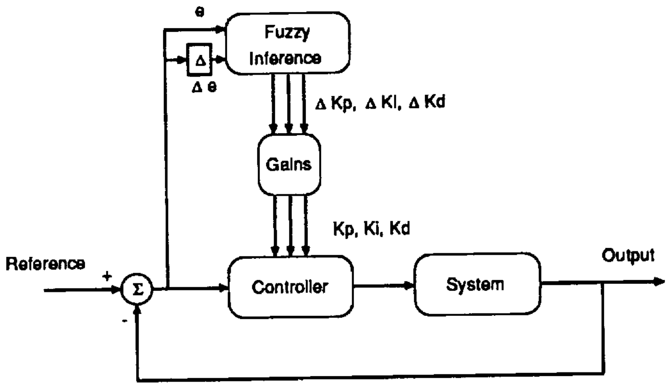

Fig. 3. Fuzzy Tuning for the Joint PID Controller.

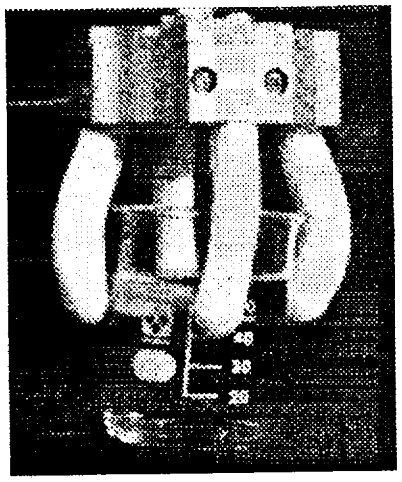

Fig. 4. The Flexible Microactuator.[14]

the robot is still moving. This is very important to implement the reflex action.

The nodes of the parallel controller are connected together via a programmable link switch. This feature allows the controller to be reconfigured in case of a fault detected in one of its nodes. A spare transputer can replace the faulty one by reconfiguring the connections of the network.

Fuzzy Control

Currently, a fuzzy tuning mechanism is used to tune the parameters of each PID controller as shown in Figure 3. This is useful with rubbertuators because of their nonlinear behavior. This mechanism uses three fuzzy matrices to tune the proportional, integral, and differential gains of the PID controller. Each matrix is similar to the Macvicar-Whelan matrix described in Tzafestas*et al.*[12] The fuzzy supervisor continually updates the controller parameters based on heuristic rules. The output of the fuzzy supervisor is the amount of change in each parameter of the PID controller. This allows the designer to specify different conflicting performance indices such as the trajectory following and disturbance rejection which leads to improved performance of the transient and steady state behavior of the closed loop system. In this case, the fuzzy system handles joint couplings as disturbances.

Combined with fuzzy logic, neural networks can also be used to learn the human-like trajectory to be followed by the robot. This technique uses neural networks to generate the knowledge base used by the fuzzy system.[13]

Flexible Gripper

Recently, research on the use of a flexible gripper has started. The gripper is composed of four flexible microactuators which act as fingers (shown in Figure 4). Each finger has three degrees of freedom — pitch, yaw and stretch.[14]

The flexible microactuator is made of fiber-enforced rubber. Internally, it is divided into three chambers whose individual pressures are controlled independently. The tip of each microactuator can be positioned depending on pressure differences inside its chambers.

These microactuators are very useful in applications that require flexible grippers. The flexible microactuator can also be very useful in zero gravity applications since it will not be loaded with object weight. The use of the flexible gripper will also increase the variety of tasks ISAC can perform such as handling fragile objects.

Issues on position sensing and closed loop control of the microactuator need to be investigated. Currently, open loop control is used to drive the flexible microactuator. To use closed loop control for the microactuator, force sensitive resistors will be used for position and force sensing.

C-6.

IV Interactive Control

The ISAC system's chief purpose is to interact with its user in a friendly and beneficial fashion. ISAC must provide a simple and consistent user interface with plenty of feedback so that the user is not intimidated or misunderstood. The system should be capable of handling the terms people use in normal language and it should understand when context applies to a command. At the same time, the system must keep a vigil for potentially dangerous situations and react to these without needing user interaction.

Command Interface

If ISAC is to be useful as a tool for a physically challenged person, the system must have a simple and flexible user interface. We elected to use voice commands to drive the system. The user's voice is captured by a microphone and is processed by a voice recognition system. Only a short training session is required to handle any speaker.

After the words are recognized, the planning module takes over and breaks down the user commands into actions for the system. Thus, the simple command *'feed me soup'* is broken down into picking up the spoon, going to the bowl, dipping into the bowl, tracking the user's face, and positioning the spoonful of soup at the user's mouth. These commands are also broken down into direct actions and coordinates for the Soft Arm. The current system handles natural-language-like commands by repeatedly breaking down the commands into subcommands until primitive actions are reached.

Another important aspect of the ISAC user interface is feedback. The ISAC system provides voice feedback by means of digitized messages that are replayed under certain conditions. These messages acknowledge user commands and assure the user that the system is doing what is expected, before things have gone too far. The messages also transmit error conditions to the user. At the moment ISAC detects a situation when it cannot pick up a spoon, even though the user requested soup, it will report an error to the user. Thus voice feedback is used to make ISAC more natural to use and to report error conditions in a straightforward manner.

Fuzzy Command Interpreter

A recent addition to the ISAC system is a fuzzy command interpreter. This module looks at user commands that contain fuzzy linguistic terms and translates them into crisp outputs for the rest of the ISAC system given the current context as shown in Figure 5. This context-based translation is a very powerful mechanism and allows a series of commands to be replaced by one fuzzy command (for example, *'move a lot closer'* would position the robot close to the user, and the subsequent command *'move closer'* would only move the robot a

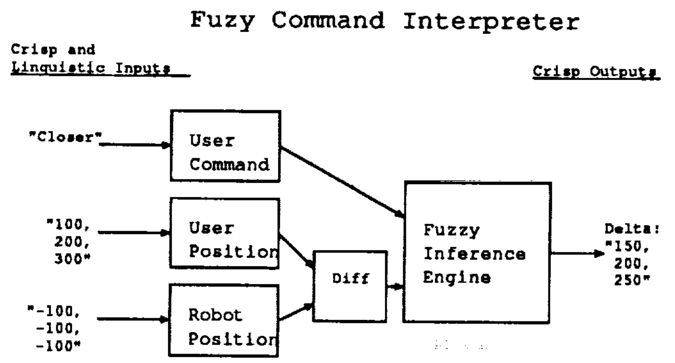

Fig. 5. Fuzzy Command Interpreter.

tiny bit because the arm is already "close" to the user). Fuzzy inference is used to take these linguistic terms and generate the crisp outputs which ISAC can use. This mechanism adapts user commands based on current system context information.

By having the ability to understand fuzzy linguistic terms in commands, ISAC's user interface is much friendlier to potential users and is also more powerful due to the fact that these commands include context as well as "fuzziness."

Reflex Action

One important characteristic of an intelligent robotic system is the ability to detect a potentially dangerous condition and react to this condition without user intervention. In particular, when a robot is in close proximity to people it is very important that the robot should not injure the person even in "emergency" situations. To this end ISAC is equipped with a reflex system like the one described by Kara *et al.*[1] The key system components related to reflex action are the stereo real-time face tracking system, the sonar sensor, and the parallel controller.

The reflex system monitors the user's position relative to the arm position and when the user makes a sudden motion toward the arm (as in a sneeze or convulsion), a high speed signal is sent to the arm controller to immediately move the robot out of the user's way. This type of intelligence is crucial in insuring that ISAC performs well under user command as well as situations that the user did not expect.

Stereo Face Tracking The ISAC system relies on stereo face tracking[6] to get the user's position. This tracking system locks on to the user and can track the user's position at 10-12 frames/sec. By using stereo cameras the face tracking system can track objects in 3 dimensional space and provides z (depth) values as well as x,y position. This depth value is the one that is crucial to the reflex action. The 3D face tracking system is shown in Figure 6.

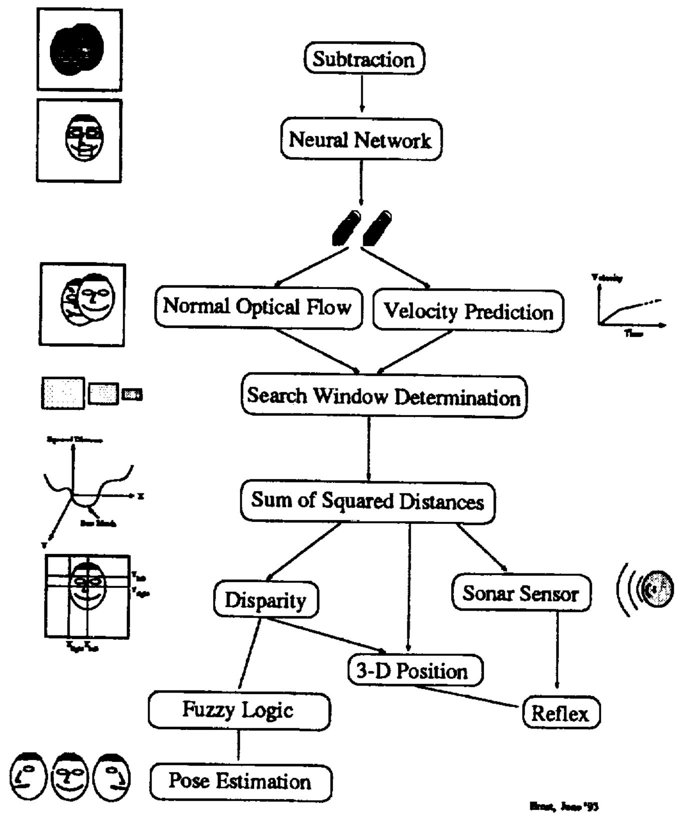

Fig. 6. 3D Face Tracking System.

Gripper Mounted Sonar Sensor In addition to the stereo face tracking, the ISAC system uses a sonar sensor mounted on the gripper to measure the relative distance from the robot to the user. This sensor provides additional information about the relative position and velocity of the user with respect to the arm. The sampling rate of the sonar sensor is slightly faster than the cameras in the face tracking system and thus provides better tracking information to control the reflex action.

Parallel Controller and Fuzzy Supervisor The reflex action depends on the ability of the arm to move quickly out of the way. Ideally the servo system for the arm could complete any motion that we desire, but this is not the case. One type of control is useful for making smooth steady motions during normal system operation, but the control system response should be entirely different during the reflex action, where response speed is critical. Due to this need, our parallel controller[10] uses a fuzzy supervisor.[12] to tune the control loops as described in Section III. During normal operation, the fuzzy supervisor sets the controller gains for steady smooth motions, with low overshoot and high damping. When a motion command meets the requirements of reflex, the gains are set to minimize the rise time only. Thus, the damping ratio and other indices are ignored during the reflex motion. This results in quick, but jerky motions.

Reflex The two sensor systems allow the reflex system to keep constant watch over the possibility of user injury and the fuzzy supervisor in the controller tunes the arm for the best response in the emergency situation. The combination of these systems leads to an effective reflex action to protect the user from injury.

V Task Learning

An aid system that comes preprogrammed with a fixed repertoire of tasks will not be of help to users with unanticipated needs. The advantage of using a general purpose robot manipulator over the specific-purpose aid devices cannot be fully realized unless the user can create new tasks for the robot. Teleoperation has traditionally been the way by which the user can move the arm to perform the action desired by the user. However, users find teleoperation very tiring and prefer to substitute them with high level commands.[15] To achieve this, the system must be able to create high level actions out of teleoperated commands and then use a sequence of these actions to carry out tasks.

The knowledge to be learned can be divided into three types: how to perform an action, when to perform it, and what are its effects. To address the first type, we have designed an action builder which allows the user to create macro actions from primitive motion commands and existing macro actions. This process is described in the next subsection. For the system to plan it must learn the two remaining knowledge types which represent the preconditions and effects of an action. These can be learned when the user prompts the robot to perform an action. The conditions existing in the environment just before the action was performed and the conditions changed as a result of the action are used to induce the preconditions and effects. The learning algorithm is described later in this section.

Figure 7 shows the knowledge representation used for planning. Actions and conditions are represented as nodes and the relation between them as links. Learning the relations between actions and conditions involves formation of these links. Planning occurs through a spreading activation process: "Potential" from the goal conditions are spread backwards to the actions that can achieve them. Similarly, potential from the current state of the conditions is spread forward to actions. An action is performed when its potential rises above a predefined threshold value. Details of the task planning and learning mechanisms are described in Bagchi *et al.*[16]

Action Builder

Traditionally, the user of ISAC was supposed to rely on the knowledge precompiled by the system developer for

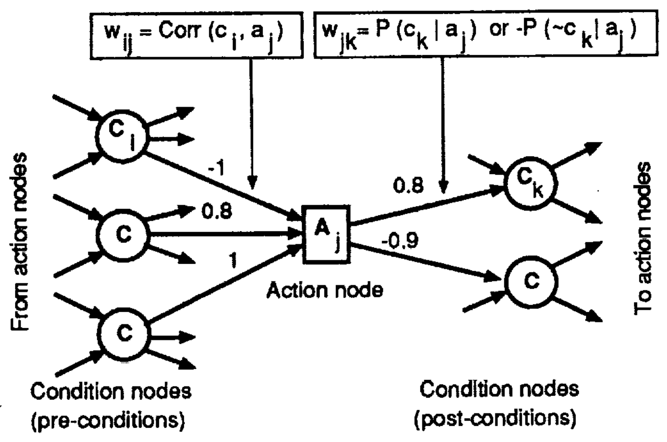
w_{ij} = Corr (c_i, a_j)
w_{jk} = P (c_k | a_j) or -P (~c_k | a_j)
From action nodes
C_i
C
C
-1
0.8
1
A_j
Action node
0.8
-0.9
C_k
C
To action nodes
Condition nodes (pre-conditions)
Condition nodes (post-conditions)

Fig. 7. Representation of an action, its preconditions, and effects.

the execution of a high level command such as *'feed me soup'*.[2] The objective of the Action Builder module is to enable the user to increase the system's repertoire of actions, by teaching new actions and integrating them with those already provided to the user.

Any complex robotic action is made of a sequence of several primitive actions. Primitive actions are those that can be directly executed by the robot. In our case, they fall under two categories:

- *unconditional* motions which instruct the Soft Arm to get the tip of the gripper to a certain location within its workspace, such as 'move to location xyz,' and
- *conditional* motions which are tied to the input from any of the sensors mounted on the Soft Arm, such as 'move down until an input from the photocell sensor is detected.'

The role of the system developer is then to provide the user with an initial set of complex actions, and with a library of primitive actions. The primitive actions incorporated into such a library enable the user to exploit the Soft Arm, as well as any of the other modules that ISAC comprises. The user then can tailor a complex action by combining any number of primitive and/or complex actions. A user-defined complex action could, in turn, be at the basis for creating actions of higher complexity. Hence, this module ensures the extensibility of ISAC's repertoire of actions in order to accommodate the specific needs of its user.

User Interface The user interface provided by the the Action Builder module is highly user-friendly. From the user's perspective, the Action Builder module acts as a voice-activated "teach pendant." It accepts the user-defined complex actions, stores them, and retrieves them whenever the user deems it necessary. Once retrieved, the user can modify or delete any of the previously stored complex actions or alternatively use them, in conjunction with the primitive and complex actions provided by the system developer, in order to build a more complex action.

Learning from Observation

The learning task can be divided into two related parts: learning the effects of an action and learning its preconditions. For both, learning is supervised. The user asks the robot to perform a set of actions. As the robot performs them, it observes the change of conditions in the environment and induces relations between conditions and actions. The system can plan for a task once the correct relations have been learned. It should be noted here that the system does not learn the sequence of actions that can achieve the goal. Instead, it learns how to make the procedural actions "transparent," by associating preconditions and effects with each of them.

Learning the Effects of an Action The effects of an action can be easily identified if they can be detected as soon as the action is complete. For robotic tasks where objects have to be manipulated, this requirement is true. The task of the learning system is not only to detect the conditions that change after an action is performed, but also to maintain a probability for this change. The strength of the connection between an action a_j and a condition c_k (see Figure 7) is given by

$$w_{jk} = \begin{cases} P(c_k \mid a_j) = \mathrm{num}(c_k, a_j)/\mathrm{num}(a_j) \\ \qquad\qquad \text{if } c_k \text{ changes to true} \\ -P(\neg c_k \mid a_j) = -\mathrm{num}(\neg c_k, a_j)/\mathrm{num}(a_j) \\ \qquad\qquad \text{if } c_k \text{ changes to false} \end{cases} \tag{1}$$

where $\mathrm{num}(c_k, a_j)$ is the number of times c_k is true after action a_j was performed and $\mathrm{num}(\neg c_k, a_j)$ is the number of times it is false. $\mathrm{num}(a_j)$ is the number of times the action was performed.

It is possible for this approach to identify spurious conditions as effects. Conditions can change at random or as a result of other agents in the environment. However, as the action is performed a number of times, the strength of the link to a uncorrelated effect will decrease.

Learning the Preconditions of an Action
Preconditions define the situations under which an action will be successful in enabling all its effects. It is difficult to discover the preconditions because one cannot be sure that an action failed because of incorrect preconditions or because of the unreliability of the environment. When the robot is asked by the user to perform an action, the entire state of the environment may be assumed to be the precondition. This, however, is too specialized: the learning mechanism must generalize the precondition set over multiple instances of successful operation of the action.

This generalization is performed by maintaining correlation statistics between the state (true/false) of the conditions and the successful execution of an action. The correlation measure used is given by

$$w_{ij} = \text{Corr}\ (c_i, a_j) = P(a_j \mid c_i) - P(a_j \mid \neg c_i), \quad (2)$$

where $P(a_j \mid c_i)$ is the probability of action a_j succeeding given c_i is true, and $P(a_j \mid \neg c_i)$ is the probability of action a_j succeeding given c_i is false. These probabilities are approximated from the statistics kept from multiple executions of the action. For "hard" preconditions, those that *must* be in the desired state for the action to succeed, the correlation will be 1 (if the condition must be true) or -1 (if the condition must be false). When the value is between these extremes, the precondition is termed "soft," denoting desirability but not necessity. Finally, a value of 0 (or close to it) denotes no correlation between the action and the condition.

Examples

Equipped with an initial set of primitive actions, the user can create a complex action to pickup a fork by using the following steps:

1. *'locate objects'* to locate the objects on the table, using the object recognition module.
2. *'goto fork'* instructs the Soft Arm to move the tip of the gripper on top of the fork's location.
3. *'move down'* until an input from the photocell sensor is detected.
4. *'close gripper'* to grasp the fork (now positioned between the gripper's fingers).

These steps are then stored as the *'pickup fork'* complex action.

Once, the system has been taught how to perform the action *'pickup fork'* it has to learn its preconditions and effects. Consider the following state of the conditions when the user asked the action to be executed for the first time (the ˜ symbol denotes false and its absence denotes true):

```
located (spoon),
located (fork),
holding (nothing),
~in (soup, bowl),
in (plate, fries),
...
```

These states form the initial preconditions for the action. After *'pickup fork'* is executed, the conditions that changed are observed. For this example, the state of the changed conditions are:

```
holding (fork),
~holding (nothing).
```

These form the initial effects.

At this stage, the preconditions are too specialized. For example, `located (spoon)` should not affect the outcome of the action in any way. The preconditions are generalized from repeated observations. For example, if `located (spoon)` is false when *'pickup fork'* is performed for the second time, the correlation is changed to zero. After a series of such observations, it is expected that all uncorrelated conditions will have low link strengths and the preconditions will converge to:

```
located (fork),
holding(nothing).
```

The effects are also updated every time the action is performed. This allows the maintenance of statistics that allow the determination of the probability of an effect occurring when the action is performed. This information is used by the planner when it has to choose between multiple action sequences in order to achieve its goals with the highest reliability.

VI Conclusions and Future Directions

We have presented the design and implementation of an intelligent robotic aid system for human services. The prototype system, termed the ISAC (Intelligent Soft Arm Control) has been shown to be an excellent testbed for such a system which may be used in the service sector in the future. Figure 8 shows ISAC in its current working environment. Remaining technical issues to be addressed include the development of intelligent control mechanisms for the flexible microactuator, integration of the learning algorithm with the ISAC system and the development of a robust real-time sensor fusion algorithm for the ISAC/HERO system.

VII Acknowledgments

The authors would like to thank the members of the Intelligent Robotics Laboratoty for their contributions to the ISAC project. This research is partially supported by Vanderbilt University, Bridgestone Corporation and Toshiba Corporation.

References

[1] A. Kara, K. Kawamura, S. Bagchi, and M. El-Gamal, "Reflex control of a robotic aid system for the physically disabled," *IEEE Control Systems Magazine*, vol. 12, pp. 71–77, June 1992.

[2] S. Bagchi and K. Kawamura, "An architecture of a distributed object-oriented robotic system," in *IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS '92)*, (Raleigh, NC), pp. 711-716, July 1992.

[3] H. Asama, M. K. Habib, I. Endo, K. Ozaki, A. Matsumoto, and Y. Ishida, "Functional distribution among multiple mobile robots in an autonomous and decentralized robot system," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, vol. 3, (Sacra-

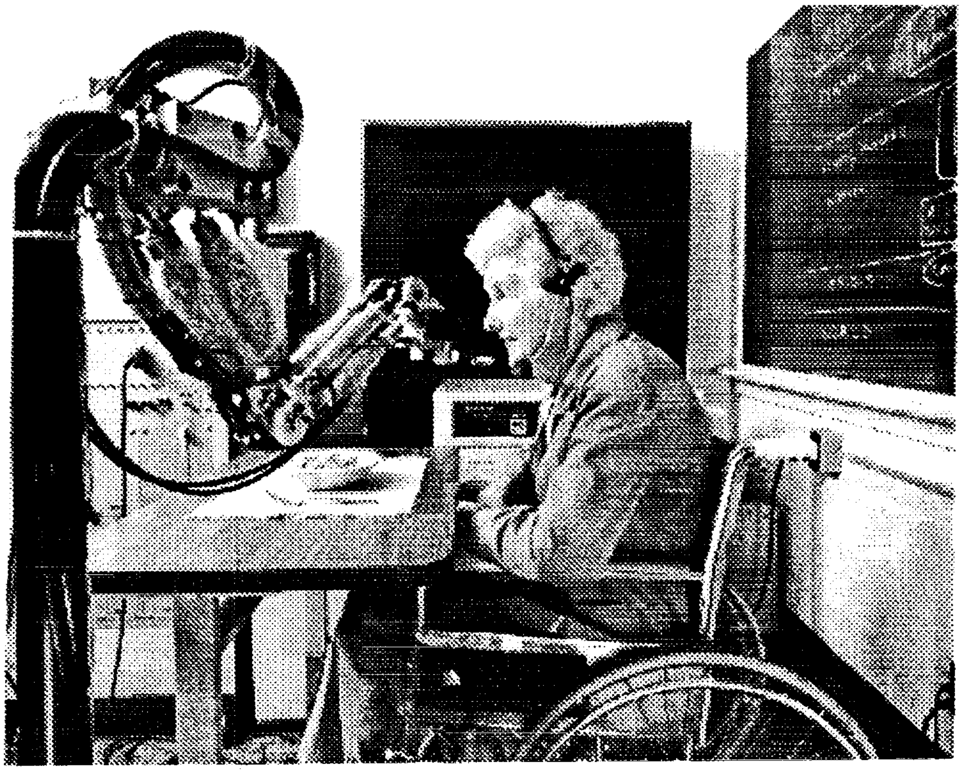

Fig. 8. ISAC at work.

mento, CA), pp. 1921–1926, IEEE Computer Society Press, 1991.

[4] R. Engelmore and T. Morgan, eds., *Blackboard systems*. Reading, MA: Addison–Wesley, 1988.

[5] M. Bishay, K. Homma, C. Swain, K. Fujiwara, and K. Kawamura, "Two-dimensional object recognition," Tech. Rep. CIS-93-01, Center for Intelligent Systems, Vanderbilt University, Nashville, TN, 1993.

[6] R. Ernst, M. El-Gamal, R. A. Peters II, and K. Kawamura, "3-D face tracking in ISAC," Tech. Rep. CIS-93-06, Center for Intelligent Systems, Vanderbilt University, Nashville, TN, 1993.

[7] J. F. Engelberger, *Robotics in Service*. Cambridge, MA: The MIT Press, 1989.

[8] K. Kawamura and M. Fashoro, "Biorobotics," Tech. Rep. CIS-92-05, Center for Intelligent Systems, Vanderbilt University, Aug. 1992.

[9] Bridgestone Corporation, *Rubbertuators and Applications for Robotics, Technical Guide No. 1*, 1986.

[10] M. Iskarous and K. Kawamura, "A fault-tolerant transputer-based parallel controller for the soft arm robot," *Proc. of the 6th Conference of the North American Transputer User Group (NATUG6)*, pp. 234 - 246, May 1993.

[11] M. Iskarous, R. T. Pack, and K. Kawamura, "A reconfigurable transputer-based parallel controller," *Submitted to the 1994 American Control Conference*, 1994.

[12] S. Tzafestas and N. P. Papanikolopoulos, "Incremental fuzzy expert pid control," *IEEE Transactions on Industrial Electronics*, vol. 37, pp. 365–371, Oct. 1990.

[13] B. Kosko, *Neural Networks and Fuzzy Systems*. Prentice Hall, 1992.

[14] K. Suzumori, S. Iikura, and H. Tanaka, "Applying a flexible microactuator to robotic mechanisms," *IEEE Control Systems*, pp. 21–27, Feb. 1992.

[15] M. A. Regalbuto, T. A. Krouskop, and J. B. Cheatham, "Toward a practical mobile robotic aid system for people with severe physical disabilities," *Journal of Rehabilitation Research and Development*, vol. 29, no. 1, pp. 19–26, 1992.

[16] S. Bagchi, G. Biswas, and K. Kawamura, "A spreading activation mechanism for decision-theoretic planning," in *AAAI Spring Symposium on Decision-Theoretic Planning*, Mar. 1994. In press.