Shape Description and Grasping for Robot Hand-Eye Coordination

Kashipati Rao, Gerard Medioni, Huan Liu, and George A. Bekey

ABSTRACT: The successful execution of grasping by a robot hand requires translation of visual information into control signals to the hand, which produce the desired spatial orientation and preshape for grasping an arbitrary object. This paper presents an approach to this problem based on separation of the task into two modules. A vision module is used to transform an image into a volumetric shape description using generalized cones. The data structure containing this geometric information becomes an input to the grasping module, which obtains a list of feasible grasping modes and a set of control signals for the robot hand. Features of both modules are discussed.

Introduction

The control of robot hands while reaching and grasping an object is a complex task, involving the interaction of sensory inputs and motion commands [1]-[3]. We assume that the process can be divided into two phases, a target approach or preparation phase and a grasp execution phase [4]. The target approach phase is based entirely on interpretation of visual input, whereas the execution phase relies heavily on tactile sensing.

This paper concerns itself with the target approach phase, i.e., the sequence of actions that lead the hand from an arbitrary position and orientation to the vicinity of the object to be grasped with the appropriate pose. It consists of the following three subtasks:

1. Reasoning from visual inputs to provide the geometric properties, spatial location, and orientation of the target object; and decomposition of an object and simplification as necessary to yield a connected set of primitive geometric shapes;
2. Selection of a feasible and appropriate grasp mode and hand shape on the basis of task and geometry for each object;
3. Commanding the hand to reorient, preshape, and move toward the target.

The following sections describe our work in the design and implementation of these aspects of the target approach process, with an emphasis on the generation of volumetric descriptions suitable for the required geometric reasoning, and the transformation of the geometric information into grasp modes and motion commands. Because grasping with a dextrous hand is most useful in an unconstrained environment, we impose the requirement that the system should be very general and handle a large variety of shapes. In addition, it should be completely data-driven, i.e., have no expectation about the objects in its field of view, and be capable of working with scenes with multiple objects. The output of the vision module is a set of generalized cones [5] as described later. These generalized cone descriptions are computed from three-dimensional data, as obtained from a pair of stereo images (in which case the data are sparse) or from an active range finder. In the latter case, the volume inference task is somewhat simpler and also more robust.

Basic issues in the choice of a shape description mechanism are discussed in the next section, followed by an overview of the vision module. More detailed explanations have been published elsewhere [6], [7]. The subsequent section deals with reasoning using generalized cone descriptions to generate a grasp mode, and the last section describes the grasping module and presents results. This work is still in progress, and plans for future research are outlined in the conclusion.

Shape Representation

Shape description is a problem of fundamental importance in human vision and in machine vision [8], [9]. Humans are capable of recognizing an object from the shape description. Shape is intrinsic to an object and is independent of lighting intensity or color. Shape description is also an important component of machine vision, especially indoor robotics vision.

The requirement that the system be data-driven means that it should be able to grasp objects without having a priori models and it need not recognize objects in order to grasp them. Also, since the task is grasping, there should be volumetric shape descriptions of the objects in the scenes. Therefore, model-based recognition systems, such as those of Faugeras and Hebert [10], Oshima and Shirai [11], and Bolles and Horaud [12], are not applicable to the task. These systems have a database of models (usually surfaces of some objects) and achieve recognition by matching a description of the scene to the models.

Criteria for Shape Representation

What are the criteria for good shape representation? The shape representation scheme should be robust to the usual changes in viewing conditions, such as rotation, perspective, scale, illumination, etc. It should be stable, i.e., small changes in shape should not cause radical changes in the description. The representation should be rich, meaning that it should be information-preserving, so that partially visible objects can still be identified. The representation should have local support, meaning that it can be computed locally, in order to deal with occlusion and to perform detailed inspection.

Although a number of shape representation schemes are possible, the preceding criteria suggest a segmented, hierarchical representation. (That is, the description should be given in parts and should be available at multiple levels.) Also, the representation scheme should be volumetric and object-centered because of the grasping application. Even though there exist many usable volumetric description mechanisms in fields such as graphics and computer-aided design, the only demonstrated approaches in vision have used superquadrics [13] and generalized cones [5], [8]. Superquadrics suffer from the weakness that extracting them from an image requires a priori segmentation of the scene. The superquadric surfaces are very sensitive to some of the surface parameters. Also, superquadrics are a subset of generalized cones. Therefore, superquadrics are not used for our
shape description, and generalized cones are used instead.

Generalized Cones

Generalized cones were introduced by Binford as useful volume descriptions for three-dimensional objects [5]. A tutorial description may be found in [8]. A generalized cone or a generalized cylinder (GC) may be defined as consisting of an arbitrary planar shape called a cross section, swept along an arbitrary three-dimensional curve called an axis. In general, the size and even the shape of the cross section may change along the axis; the rule describing the change is called the sweep rule or the cross-sectional function. Although the axis, cross section, and the sweeping rule could be arbitrary analytic functions, in practice, only simple functions are used. Typically, the axis is straight or circular, the sweep rule is constant or linear, and the cross section is rectangular or circular. Thus, a normal cylinder consists of a circular cross section, swept along a straight axis with no change in shape or size of the cross section. For a normal cone, the size changes linearly along the axis.

The precise restrictions for a generalized cone have been different for various researchers; e.g., some do not allow the cross-sectional shape to change. Shafer and Kandade have developed a terminology for describing the variants of a generalized cone [14]; this terminology shall be followed where appropriate. In this terminology, a linear cone has a linear cross-sectional function, a straight cone has a straight axis, and a homogeneous cone has an invariant cross-sectional shape.

The generalized cone representation is rich and has local support. Using the functions discussed earlier, all points on the generalized cone can be described (except at sharp bends, see [15]). Thus, the representation is information-preserving, and is therefore rich. At each point on the axis, a local coordinate system can be defined using the tangent, normal, and binormal of differential geometry [16]. The generalized cone description is invariant to rotation, scale changes, perspective, or change in illumination.

Because of the above-mentioned properties, the generalized cone representation deservedly has received considerable attention in the vision and robotics community. However, there are some difficulties with the generalized cone representation; they are difficult to compute. The generalized cone description is not unique for an object, and ad hoc rules may have to be used to choose a description. The generalized cone representation may not be the best if there is interest in very fine details in the object.

Vision Module

The functions of the vision module are to acquire three-dimensional data, compute symbolic surface descriptors, infer the volumes of objects, and present the final output. Figure 1 summarizes the steps in the vision module.

Obtaining Three-Dimensional Data

One method of obtaining three-dimensional data is to take a stereo pair of the scene and match the two images to get three-dimensional data. We have developed some successful algorithms to perform this task [17], but for reasons of robustness and processing time, the vision module uses range images.

Several range acquisition systems have approximately the same performance, and the final choice has not been made. One candidate is a commercial system purchased from the Technical Arts Corporation, Seattle, Washington. It consists of a plane of coherent light projected onto the scene and recorded by a charge-coupled-device camera. Each position of the laser gives a set of three-dimensional points. Another candidate system is an active stereo bench [18], which consists of two cameras recording the intersection of a plane of coherent light with the scene.

Symbolic Surface Descriptions

The goal of this step is to extract important curves, corresponding to physical boundaries of objects, and to segment the scene into surface patches. It is desired to decompose a complex surface into simple surface patches. The approach is to determine the boundaries of the surfaces by computing local properties and then inferring the patches from them. A complete description of the process can be found in [19]. In particular, the types of boundaries are (1) jump boundaries, where the surface undergoes a discontinuity, and (2) creases, which correspond to surface orientation discontinuities.
These boundaries are inferred from curvature properties of the surface, such as curvature zero crossings and extrema [19]. A jump boundary creates a zero crossing of the curvature in a direction normal to that of the boundary; a crease causes a local extremum of the curvature at that point. Crease boundaries may also create zero crossings away from the location of the boundary itself.

These features, when connected using contiguity, give us partial boundaries for patches in which the surface should be segmented, but do not necessarily give a complete segmentation. These partial boundaries are then completed by simple extension [19]; this process worked satisfactorily for a large number of examples tested. The resulting regions are assumed to correspond to elementary surface patches. These regions could be segmented further, based on either the region shape or the results of surface fitting; however, it was not found necessary for the examples tried.

Figure 2 illustrates this early processing on a synthetic image for reasons of clarity. Figure 2(a) shows the shaded image representing three objects partially occluding each other, Fig. 2(b) shows the curves obtained by our feature detection process, and Fig. 2(c) shows the regions bounded by the previous curves. A shaded scene is reconstructed after each patch has been approximated by a quadratic polynomial.

Inferring Volume Descriptions

The submodule for inferring volume works with edge data as input, where the edges may be obtained from range data. The submodule finds symmetries in a particular direction by making lines perpendicular to that direction. The points of intersection between the lines and the edges are found, and symmetry points are located as midpoints of these points of intersection. The symmetry points are first linked by a near-neighbor algorithm. The linked symmetries are called “threads,” and the areas they describe are called “ribs” or two-dimensional generalized cones. Local tangents are computed at each point in each thread and those points are filtered out, where the tangent deviates a large amount from the current direction. This process of thread finding and filtering is repeated in some chosen directions (usually equiangular directions from 0 to 180 deg, say every 45 deg). Then the threads from all directions are considered, and “superthreads” are formed by linking threads based on nearness, continuity in tangents, nearness of the directions they came from, and disjointedness of the areas covered. The areas covered by these superthreads are called “superribbons” and are the two-dimensional generalized cones in the scene.

Several generalized cone descriptions thus may be found for one object. The generalized cones found are then verified—that is, they should be terminated by two ends or terminators. The verification may be based on edge data, but if unsuccessful, linear segments of the edges may be used to find terminators (using the algorithms explained in [6]). The generalized cones are then chosen based on verification, larger length-to-width ratio, and larger area covered. The “best” generalized cone description (according to these criteria) is the one retained.

To generate a three-dimensional generalized cone description from the two-dimensional description, it is currently assumed that the cross section is circular. The radius of the cross section is obtained from half the width of the two-dimensional generalized cone at each point on the axis to give the cross-sectional function. The axis is obtained from the two-dimensional axis. If dense three-dimensional data are available, the depth of the axis is obtained from the (reliable) depth of the object surface at that point minus the radius of the cross section at that point. (This method is used as the depth at the boundaries of the object changes very fast and is highly unreliable.) If three-dimensional data are not available, it is assumed that the axis is flat. The cross sections are fit so that they are perpendicular to the local tangent of the axis.

The output of this submodule is a data structure, whose fields are (1) the x, y, and z coordinates of the axis, (2) the cross-sectional function or sweep rule at each point, and (3) the cross section of the generalized cone in polar coordinates—angle θ and distance ρ. (There is only one cross-section output since we assume that the generalized cone is homogeneous.) Some additional attributes of the generalized cone added for convenience are (4) the axis type (straight or non-straight), (5) the cross-sectional function type (constant, linear, or nonlinear), (6) the cross-sectional type (currently circular), (7) whether the generalized cone is homogeneous or nonhomogeneous (currently always homogeneous), and (8) the center of gravity of the generalized cone.

Using these additional attributes, an abstract description of the generalized cone is given in terms of the terminology discussed earlier. For example, the snakelike object in Fig. 3 (whose generalized cone data structure is given in Fig. 7) is a Nonlinear Nonstraight Homogeneous Generalized Cone with circular cross section. More details of the preceding algorithm may be found in [7].

The generalized cone-finding algorithm is general and is capable of describing scenes with imperfect data (such as that even from an intensity image)—with breaks in the object boundaries and surface markings on the object and the background. Thus, the algorithm handles the segmentation problem and does not require a priori knowledge of the figure and ground (closed boundaries). This is the major contribution of the system. The novel aspects of the algorithm are the verification phase for the choice of generalized cone descriptions and the utilization of the cover idea for obtaining superribbons.
Because the Model 1 Belgrade-USC hand has four three-jointed fingers capable of parallel motion, plus a nonarticulated thumb, only the four grasp modes shown in Fig. 4 can be implemented currently: (1) power grasp, (2) hook grip, (3) pulp pinch, and (4) lateral pinch. The Model 2 version of this hand will make more grasp modes available by including thumb articulation and fingers capable of spreading.

The hand is activated by three motors, which control the thumb and four fingers. The control variables are defined as angle $\alpha_1$ for the thumb, $\alpha_2$ for one pair of fingers, and $\alpha_3$ for the other pair of fingers. The four grasp modes are implemented using the three control variables.

Recognizing and Isolating Primitives

The knowledge system infers suitable grasp modes from object information. The descriptions furnished by the vision module are very general in order to be used by other modules besides grasping (for instance, recognition or classification). In order to compute a grasp mode, the generalized cone description is transformed into a format more suitable for grasping. Currently, the generalized cone description is decomposed into a set of six primitives.

Transforming the generalized cone description into a format suitable for reasoning about grasping is a two-step procedure. The first step is to recognize one of the six pre-defined primitives [23], [24]: box, ball, cylinder, cone, torus, and pyramid. If an object is a primitive, the recognition is completed. If not, it is necessary to divide the object into primitives by segmentation. This segmentation is partially achieved by the vision module itself, in the case where the object must be represented by multiple generalized cones; e.g., a cup and handle. Some objects can be represented in this way, such as a snakelike object. In the current implementation, segmentation is done by hand. Au-
Automatic segmentation is a difficult research problem in itself. If the object or its segments are not primitives, it is necessary to perform smoothing operations in order to obtain approximate primitives. For example, a cucumber can be approximated by a circular cylinder for the purpose of selecting a grasp.

The second step is to compute features related to grasping, such as volume, measurements, and orientation. The extracted features are stored in schemas [2], [25]. Schemas are concepts for a style of cooperative computation that involve concurrent activities in a dynamic network. Schemas combine attributes of both control elements and instantiations. The distinction is made between a prototype, which describes each primitive shape, measurements, orientation, etc., and an instance of a prototype, which describes the unique aspects of individual instantiations of primitives. With demons and relations to other instantiations, schemas describe object information symbolically and explicitly. A primitive schema prototype contains the primitive name, which describes object shape, measurements, orientation, center of gravity, etc. The primitive cylinder gives the object shape, as well as two measurements (radius and height), orientation, and the center of gravity. The center of gravity gives spatial information as well as suggesting a grasping position.

Generally, geometric and spatial information about an object are needed in the knowledge system. Object shape (cylinder) measurements (r, h) plus orientation give object geometric data. A circular cylinder is detected by noting that the cross-sectional function is constant and the cross-sectional type is circular. Its measurements (r and h) are easily computed from its x-, y-, and z-axis lists when the orientation is known. The orientation can be obtained from the generalized cone coordinates by translation and rotation. In this implementation, the location of the center of gravity and the shape are used to give spatial information.

Different shapes may lend themselves to different geometric measurements. For example, although both a box and a cylinder are three-dimensional objects, a box needs three measurements (length, width, and height) but a cylinder needs only two (radius and height). In other words, for different shapes of objects, it is necessary to extract different measurements from the generalized cone data structures. After obtaining the object shape, it is relatively easy to obtain measurements. The fundamental method to get object shape is to decompose an object into primitives, as indicated earlier. Decomposition is accomplished by detecting sharp changes in either cross section or axis direction.

Grasp Mode Formation

Our hypothesis is that there exist certain grasp modes that are preferred for primitive objects. Grasp modes for a complex object can be formed by combining the grasp modes for primitives. It is important to note that these grasp modes are not necessarily unique, since more than one form of grasping may be feasible for a given geometric shape. The question of selection among feasible grasps is considered later. Therefore, there are two basic steps to obtain general grasp modes. The first is to find grasp modes for primitives by table lookup. A table is built for six primitives with eight grasp modes [23]. The table entries are geometric primitive, primitive parameters, grasp modes, etc. Then, given a primitive and its parameters, a set of feasible grasp modes can be selected. A portion of the table for the Model 1 Beldrage-USC hand is shown in Fig. 5.

The second step for grasp mode selection is to combine all grasp modes for the primitives and find a sorted grasp mode list in which the preferred one is first. Each primitive has center of gravity (COG) and constraint slots (defined later), which contain its spatial position in the working environment and relative position to its neighboring primitives. The generation of a sorted grasp mode list is based on heuristics derived from observations of human behavior in grasping tasks with primitive objects of various dimensions.

For convenience, the heuristics are grouped into categories:

- Heuristics about center of gravity: Grasping position is chosen as close to the object center of gravity as possible to ensure stability with minimum applied force. The distance between the primitive COG and the object COG is calculated for this purpose.

- Heuristics about measurements: Grasp the largest feasible segment. The volume of each segment is compared and its constraints are checked in order to find the largest feasible segment.

- Heuristics for approaching orientation selection: The preferred orientation is the one requiring least hand movement.

- Heuristics for grasp mode selection: The more flexible grasp mode has higher priority to execute. The four grasp modes available with our hand have different flexibilities when they are used to grasp an object. For manipulation purposes—stability can be ensured—the more flexibility, the better. For the Beldrage-USC hand, a power grasp has the least flexibility and a pulp pinch has the most flexibility.

- Heuristics for action selection: Ensure at least a minimum contact area with the grasp surfaces. The amount of overlap should depend on object properties such as weight and surface smoothness.

It is evident that the selection of specific heuristics is related to the task to be performed, which implies the selection criteria. Such criteria include stability, flexibility in manipulation, and the ability to apply large torques. Since stability is the major concern in this paper, the heuristics are applied in the order of grasping position (center of gravity and measurement heuristics) first, approach-
Current Status and Results

The vision, transformation, and knowledge systems have been implemented as a set of independent modules. The vision module is used to obtain generalized cone representations from the working environment. There are two submodules in the grasp mode selection module: the feature-computing module is used to compute and fill features into schemata from generalized cones, whereas the reasoning module is used to obtain grasp modes for the target object. Generalized cone data structures in LISP are read by the knowledge system, which also gets task information from the user. Both task and object data are stored in schemata. The knowledge system is implemented using Knowledge Craft, an expert system development environment on which LISP, CRL-OPS, and PROLOG can be run. The schema representation of a primitive is obtained from generalized cone data structures by LISP functions. Then the knowledge system works on schemata and gives an output list to the hand control module. There are two steps in this process. First, a rule-based inference mechanism is employed to infer grasp modes for each primitive. The set of sorted grasp modes is generated by comparing and combining according to the criteria and the heuristics. For example, stability is a criterion for grasp selection. Second, the grasp modes are transformed into control variables with measurements, position, and orientation. Table lookup is used to obtain the three control variables. Position and orientation are passed to the hand control module.

The current results are obtained from objects with a circular cross section, such as a cylinder, torus, snake, etc. In the present implementation, grasp mode formation is based on object information, and the PUMA 560 is used to move the hand toward the object. Therefore, only the object shape, measurements, orientation, constraint, and center of gravity are considered. For simplicity, our measurements take only three symbolic values: large, medium, and small. They are defined as follows, where Lh is the hand width.

Large: greater than $2Lh$
Medium: less than or equal to $2Lh$ and greater than one-third $Lh$
Small: less than or equal to one-third $Lh$

The orientation of approach takes three values in a vertical plane in which lies the hand and the object to be grasped: from top, horizontal, and slant. The definition refers to angular regions in the plane (see Fig. 6).

Presently, each primitive is represented by a generalized cone data structure. For example, any object is eventually decomposed into primitives. Therefore, the number of generalized cone data structures defines the number of primitives. The connection of the primitives is given by the slot organization of the schema object, which contains overall information about an object. The content of constraint for each primitive is inferred from the content of the organization slot, which has predicates such as right-connect, left-connect, etc. The constraint is described by the relationships of how primitives are connected, such as right, left, up, down.

A complete grasp is a four-tuple, containing the mode, hand-opening size of the mode, orientation of approach, and position of approach. The four-tuples are ordered as (mode, size, position, orient). When this information is passed to the hand, the four-tuple is translated into a three-tuple, which contains (list of variables, position, orient). A snakelike object does not fall into any of the classes, but it is approximated by three cylinders—two horizontal and one tilted. The approximate data are the input to the grasping control system.

In the following paragraphs, the successive stages in the process of obtaining the preferred grasp mode are shown. At each stage, the computer code (or pseudocode) is shown. It should be noted that the output of our grasp mode inference system is computer code, so that the listings below represent system output data, which eventually are fed to the robot hand controller.

A snakelike object is treated as three cylinders, two small horizontal cylinders and one large slanted cylinder. For the grasp mode selection module, the input is the generalized cone data structure, which is output of the vision system. The generalized cone data structure for a snakelike object is shown in Fig. 7.

The second step is to approximate the object by three cylinder primitives; two are horizontal, and one is slanted. Each primitive instantiation has its own name, such as cld1, cld2. At this step, the slot for grasp modes is empty. Each primitive has an associated name, which is described in the slot primitive-of. The three primitives compose the object snake, which is given in an object schema with the name snake. In the object schema, the relationship among the primitives is described. The relationship, in turn, gives the geometric constraints for each primitive.

The inference of grasp modes is based on the information included in the primitive schemata. Grasp modes with different approach orientations and hand openings are generated for different primitives. The results for the snakelike object are shown in Fig. 8.

To control a robot hand, digital signals are needed. Therefore, the grasp modes given are transformed into a control list for the hand control in a digital format. The effect of the preceding process can be seen in Fig. 9, which shows the snake-like object before and after a successful grasp.

Conclusion

Robot hand-eye coordination has been demonstrated by means of two modules. First, a vision module is used to transform an image into an object description. Second, a grasp mode selection module is used to transform the object description into a set of control signals, which yields the desired hand orientation and prehension. Object representation is by means of generalized cones. Complex objects are decomposed into geometric primitives and approximated by primitives. The output of the vision module is a data structure. Operations on this data structure ultimately yield an ordered list of possible grasp configurations. For example,
Fig. 7. Data structure for snakelike object.

```
((power-grasp medium (59 71 152) slant)
 (pulp-pinch medium (59 71 152) slant)
 (pulp-pinch medium (21 64 103) from-top)
 (pulp-pinch medium (99 65 220) from-top)
 (pulp-pinch medium (99 65 220) horizontal))
```

Fig. 8. Results for snakelike object.

using images of a snakelike object, generalized cone data structures, schemas, grasp modes, and hand/finger control signals are obtained.

We have a videotape that illustrates the entire process from image acquisition to hand preshaping, where a PUMA 560 robot is used to lead the hand toward the object and orient it as required for grasping.

In the future, we plan to extend the vision module so that it can work with compound objects (e.g., a human or animal shape). In the grasping module, our major goal is to automate the process of object decomposition and primitive approximation. Task requirements will be considered as a major constraint to select a task-oriented grasp mode. We plan to combine object information and task requirements in a serial way: first, perform reasoning about grasp modes according to object information; then carry out reordering the grasp modes according to task requirements. We believe that task requirements are dominant in grasp mode selection, whereas object information is less important. Therefore, future research efforts will emphasize the combination of task requirements with object description in order to obtain automatic grasp mode generation from task description with complex objects.

**Acknowledgments**

This research was supported in part by the Defense Advanced Research Projects Agency under Contracts F33615-84-K-1404 and F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, DARPA Order 3119, and in part by the National Institute of Health (Division of Research Resources) under Grant 1P41 RRO1861.

**References**


Gerard Medioni received the Diplome d’Ingenieur civil from the Ecole Nationale Superieure des Telecommunications, Paris, France, in 1977, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1980 and 1983, respectively. He is currently an Assistant Professor in the Departments of Electrical Engineering and Computer Science, University of Southern California. His research interests include computer vision, artificial intelligence, and robotics.

Huan Liu received the B.Eng. degree in electrical engineering and computer science from Shanghai Jiao Tong University, Shanghai, China, in 1983, the M.S. degree in computer science from the University of Southern California in 1985, and is a Ph.D. candidate in the Computer Science Department at the University of Southern California in Los Angeles. His research involves the control of dextrous hands, robot planning, neural networks, and machine learning.

Kashipati Rao received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in April 1983. During Fall 1983, he was with the Department of Electrical and Computer Engineering at Carnegie-Mellon University working toward his M.S. degree as a Graduate Research Assistant and Fellow. Since Spring 1984, he has been with the Department of Electrical Engineering (Computer Engineering) at the University of Southern California, Los Angeles, where he obtained the M.S. and Ph.D. degrees. He has been a Research Assistant in the Institute of Robotics and Intelligent Systems and has been conducting research in the area of shape description for robot vision. His interests are in computer vision, artificial intelligence, and robotics.

George A. Bekey is Professor and Chairman of the Computer Science Department at the University of Southern California (USC) in Los Angeles. His research interests are in the areas of intelligent robotic systems, artificial intelligence applications in biology and medicine, and planning and control of manufacturing systems. From 1983 to 1986, he was Director of the USC Robotics Institute. Currently, he also serves as Director of the USC Center for Manufacturing and Automation Research. In addition to his teaching and university research, Dr. Bekey’s professional experience includes four years as a Research Engineer with the Department of Engineering at the University of California, Los Angeles, where he worked primarily with computers, and three years with Beckman Instruments, Inc., as a Computer Applications Engineer and Manager of the Los Angeles Computation Center, Head of the Analysis and Simulation Section, and Senior Staff Engineer. His last assignment at TRW was to organize a staff group concerned with the guidance and control of manned space vehicles.

Dr. Bekey has published over 140 technical papers in the areas of robotics, computer simulation, control systems, human operator dynamics, and bioengineering. He is the coauthor (with W. J. Karplus) of a text, Hybrid Computation (Wiley, 1968), editor of Mathematical Models of Public Systems (SCS, 1971), and coeditor (with M. D. Schwartz) of Hospital Information Systems (Marcel Dekker, 1972). He is the Editor of the IEEE Journal of Robotics and Automation, and a member of the editorial boards of Mathematics and Computers in Simulation and Transactions of the Society for Computer Simulation.

**Intelligent Control Symposium**

The Fourth IEEE International Symposium on Intelligent Control will be held September 25-27, 1989, at the Albany Hilton Hotel in Albany, New York. The symposium is sponsored by the IEEE Control Systems Society and hosted by the Rensselaer Polytechnic Institute. The General Chairman is A. C. Sanderson, Rensselaer Polytechnic Institute, and the Program Chairmen are A. A. Desrochers, Rensselaer Polytechnic Institute, and K. P. Valavanis, Northeastern University.

Topics of interest include control, machine intelligence, architectures and tools, and applications. Before April 1, 1989, prospective authors should submit extended summaries to Prof. K. P. Valavanis, Robotics Laboratory, Northeastern University, Dept. of ECE, Boston, MA 02115; phone: (617) 437-2164 or 3046. Proposals for tutorials and invited sessions should be mail by May 1, 1989, to Prof. A. A. Desrochers, Dept. of ECSE, Rensselaer Polytechnic Institute, Troy, NY 12180-3590; phone: (518) 276-6718. The deadline for extended summaries is April 1, 1989, and acceptance notification is June 1, 1989, and final paper due on July 1, 1989. Authors are encouraged to submit full papers.