query sets, which is not only dependent on the quantity of the redundancy. Note that the BMS exhibits an elastic condition and the property of the BMS is not included in that of the CBS from a viewpoint of QA systems, though the system redundancy of the former is lower than that of the latter.

IV. Discussion

Especially in the information retrieval system, the BMS and its conjunctive system are simple but flexible, since these conditions are analogous to those of the indexed sequential access method (ISAM) and basic sequential access method (BSAM) system [8], where the query set consists of queries that specify all collections of k records out of m, i.e., a record-collection combinatorial query set of order k. This suggests the possibility of designing the information retrieval system in elastic condition.

References

[1] J. Pearl, "On the storage economy of inferential question-answering systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-5, no. 6, pp. 595–602, Nov. 1975.
[2] ——, "Memory versus error characteristics for inexact representations of linear orders," *IEEE Trans. Comput.*, vol. C-25, no. 9, pp. 922–928, Sept. 1976.
[3] ——, "On coding precedence relations with a pair-ordering fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 1, pp. 118–120, Jan. 1976.
[4] A. Crolotte and J. Pearl, "Bounds on memory versus error trade-offs in question-answering systems," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 2, pp. 193–202, Mar. 1979.
[5] ——, "Elasticity conditions for storage versus error exchange in question-answering systems," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 6, pp. 656–664, Nov. 1979.
[6] P. Elias and R. A. Flower, "The complexity of some simple retrieval problems," *J. ACM*, vol. 22, no. 3, pp. 367–379, July, 1975.
[7] T. Berger, *Rate Distortion Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
[8] S. P. Ghosh, "Consecutive storage of relevant records with redundancy," *Communications of the ACM*, vol. 18, no. 8, pp. 464–471, Aug. 1975.

MKS: A Multisensor Kernel System

THOMAS C. HENDERSON, WU SO FAI, AND CHARLES HANSEN

***Abstract*—The multisensor kernel system (MKS) is presented as a means for multisensor integration and data acquisition. This system has been developed in the context of a robot work station equipped with various types of sensors utilizing three-dimensional laser range finder data and two-dimensional camera data. Specific goals that have been achieved include 1) Developing a suitable low-level representation of raw data and/or features extracted from the raw data of the various sensors; 2) Providing a method for efficient reconfiguration of the sensor system in terms of "logical" sensors which map onto physical sensors and computation; and 3) Providing a basis for high-level object modeling techniques.**

I. Introduction

Current pattern recognition systems tend to operate on a single sensor, e.g., a camera, however, the need is now evident for pattern recognition systems which can operate in multisensor environments. For example, a robotics workstation may use range finders, cameras, tactile pads, etc. The multisensor kernel system (MKS) provides an efficient and coherent approach to the specification, recovery, and analysis of patterns in the data sensed by such a diverse set of sensors. We demonstrate how such a system can be used to support both feature-based object models as well as structural models. Moreover, MKS allows rapid reconfiguration of the available sensors and the high-level models.

Manuscript received April 29, 1983; revised September 1983 and April 1984. This work was supported in part by the System Development Foundation and in part by NSF under Grants ECS-8307483 and MCS-82-21750.
The authors are with the Department of Computer Science, The University of Utah, Salt Lake City, Utah 84112.

Multiprocessor and multisensor systems are being proposed to solve a wide range of problems. In particular, distributed sensing systems and general robot workstations require real-time processing of information from visual, auditory, tactile, and other types of sensors. Three major issues must be addressed:

1) low-level representation of the sensory data;
2) high-level specification and organization of the sensor systems; and
3) high-level control of processors and sensors.

The first major goal is to provide a mechanism for the integration of data available from different sensors into a coherent low-level representation of the three-dimensional world. Such a representation is crucial to the successful application of multisensor systems, and in particular, robot technology. Shneier *et al.* [21] argue

> ...the use of easily acquired information from a number of sources can lead more easily to understanding a scene than can exhaustive analysis of an image from a single source.

Although their work dealt only with visual information, we heartily concur in principle and propose the spatial proximity graph (see Section II-A) as a structuring mechanism for the integration of data from different sensors. We propose the *spatial proximity graph* as a low-level representation of sensory data from diverse sources and use this as the basis for high-level organization and control over the acquisition of data.

The second major goal is to provide a simple, yet complete method for reconfiguring a multisensor system. We propose the "logical" sensor as a key notion toward this end. A logical sensor maps either directly onto a physical sensor, or provides a description of how data from one or more logical sensors is combined to produce the desired data. (See Foley and Van Dam [4], Pfaff *et al.* [18], or Rosenthal *et al.* [20] for a similar approach to the specification of computer graphics systems' interactive interfaces.) Ultimately such logical sensors could be implemented in special hardware (a "sensor engine"). The notion of "logical" (or abstract) sensor allows for flexible hardware/software mix in terms of a multisensor system and permits a simple method of reconfiguration whenever logical or physical sensors are added to or removed from the system.

The third major goal is to provide a context in which constraints, both physical and logical, can be brought to bear to reduce the amount of computation required to solve problems. A prominent example of a multisensor system is the *distributed sensing system* for situation assessment [23]. Distributed sensing systems consist of several independent stations interacting to produce an assessment of the activity being monitored collectively by the stations. Most research in this area is directed toward organizing the information flow between stations so as to achieve an efficient and successful interpretation of the sensed data (see also Smith [22]). Usually the stations transmit reports or evaluations of their own data rather than the raw data itself. Thus, there is a need for a high-level model to provide an interpretation of the various patterns of information provided by the sensors, and a mechanism for controlling the acquisition of data.

Many methods exist for modeling three-dimensional objects, and the multisensor kernel system supports a wide range of three-dimensional object modeling techniques. Feature-based models are supported directly by MKS, whereas structural types of descriptions, e.g., those requiring the derivation of surface patches and their spatial relations, can use MKS as a first step to

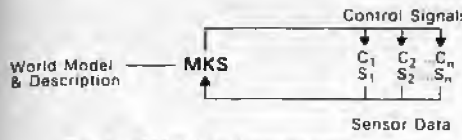

Fig. 1. **MKS**: Multisensor kernel system.

organize three-dimensional data from several sources. Several high-level modeling methods have been investigated, including standard feature models and the Hough shape transform [11], [12]. The use of high-level models provides a mechanism to limit the amount of sensor data acquired, and thus reduces the amount of computation necessary to identify objects. Although the methods proposed here are developed in the context of a robot workstation, the result will be applicable to any multisensor system, including distributed sensing systems, situation assessment systems, etc.

II. Acquisition and Organization of Three-Dimensional Data

The multisensor kernel system (MKS) must coordinate the active control of several sensors, e.g., turn a sensor off or on, aim a camera, etc., and integrate the data from the various sensors into a coherent and useful description of the world. Fig. 1 shows the flow of data and control in such a system, where C_1 to C_n are the controllers or actuators for the sensor systems S_1 to S_n, respectively. In this section, we describe the organization of incoming data into the low-level representation.

Each sensor system, S_i, in Fig. 1 has an associated controller, C_i; for example, a camera may be aimed, focused, or have the shutter speed changed. A sensor system may have several components:

a camera system:	a camera and a light source;
a laser range finder:	a laser, mirror system, diode arrays, and optics.

That is, a sensor system consists of all the sensor components and the associated controller.

The prototype system consists of two sensors: a camera and a laser range finder; these provide visual and range data, respectively. Visual information arrives as digital images which must be processed, whereas the range data was acquired by presenting the object to the laser range finder from several different orientations.

In general, any set of sensors can be used, and the system is organized such that each sensor contributes information independently of the other sensors. However, as will be described later, a high-level model is used to control the acquisition of data so that as time goes on, less data is demanded from the sensors. Constraints from the already processed data control the sensors' acquisition of new data.

A. Spatial Proximity Graphs

In the context of digital image analysis, various schemes have been proposed for organizing properties or features recovered from two-dimensional images, e.g., Marr's primal sketch [15], the intrinsic images of Barrow and Tennenbaum [1], and in a more limited context, the region adjacency graph of Pavlidis [17]. However, all of these representations were developed with the intent of mapping from two-dimensional images, and we propose a more general three-dimensional organization called the *spatial proximity graph* as the initial data organizational tool. From this one can then proceed to derive other levels, e.g., the hierarchical three-dimensional models proposed by Marr. However, if one has only two-dimensional camera data, then MKS should not be used to organize the intensity data directly, but we will show how MKS can be efficiently used to organize feature vectors extracted from the camera data.

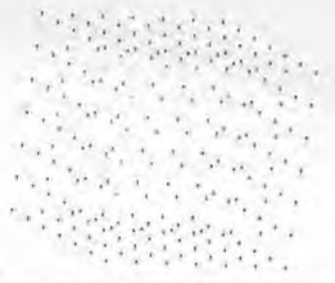

Fig. 2. Points on the surface of a synthetic cube.

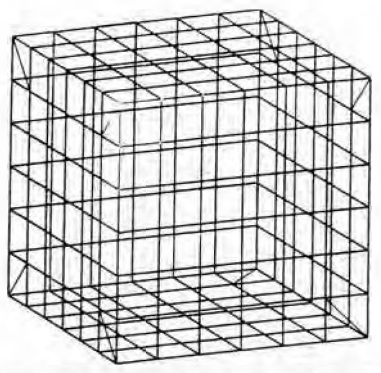

Fig. 3. Spatial proximity graph for points in Fig. 2.

The spatial proximity graph provides a means for organizing information about the three-dimensional world. In particular, the approach is as follows:

1) to obtain raw sensory data;
2) to extract features from the data and the three-dimensional locations of these features; and
3) to determine the spatial relationships between the features.

The nodes of the spatial proximity graph correspond to the positions in three-space of the features extracted from the raw sensory data. Nodes are linked by an edge if they are within some prespecified distance. This then provides a means for organizing information from different sources. Moreover, analysis can be performed on this graph, e.g., fitting planar faces which can in turn be used to derive high-level models [6].

Thus, given a set of sensors, we assume that each sensor provides raw data in the form of two pieces of information. Namely, each datum from a sensor consists of a feature and a location (in three-space) of that feature. In this manner, data from various sensors can be treated uniformly. This data protocol places an additional burden on some types of sensors, e.g., cameras, but for most sensors, techniques are available to determine the required information, and for many sensors, e.g, laser range finders, tactile sensors, etc., the information is directly available.

The spatial proximity graph has been studied in the context of three-dimensional range data [8]. For example, consider the surface points of the synthetic cube shown in Fig. 2. The spatial proximity graph (SPG) for those points is given in Fig. 3. Figs. 4 and 5 show two sets of points obtained with a laser range finder used to scan an industrial object. There are about 2000 point samples. Two views of the SPG for the merged data are shown in Figs. 6 and 7. The original object is shown in Fig. 8.

The *spatial proximity graph* is a graph G, having a distinct node for each distinct feature location. An edge exists between two nodes if either of the two nodes has the other as one of its m-nearest neighbors, for some small m. Note that the SPG works equally well when the sensors sample data at the same resolution or if different sensors provide information with different spatial resolutions, so long as the largest sampling distance is used to set

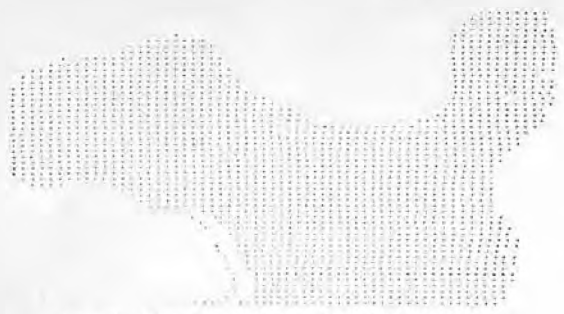
Fig. 4. Points on the surface of an industrial piece.

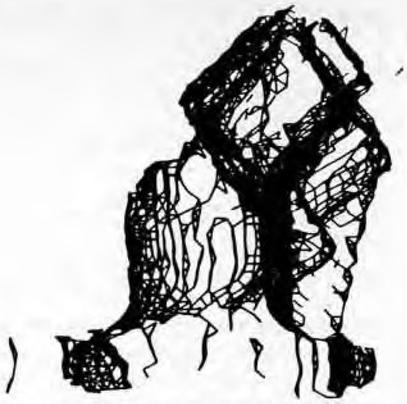
Fig. 7. End view of spatial proximity graph.

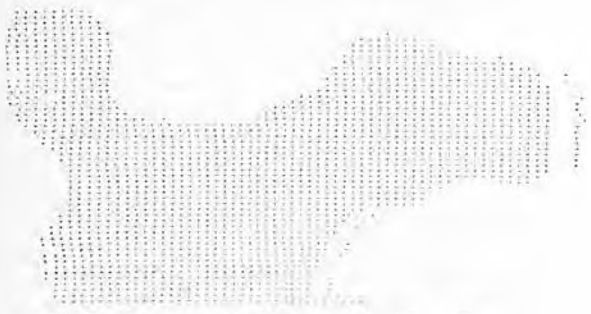
Fig. 5. Opposite view of points on the industrial piece.

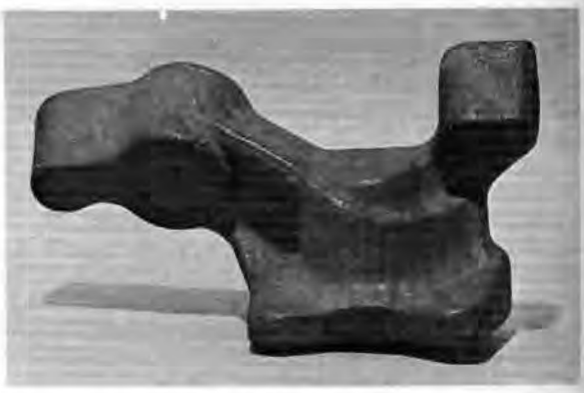
Fig. 8. Original object.

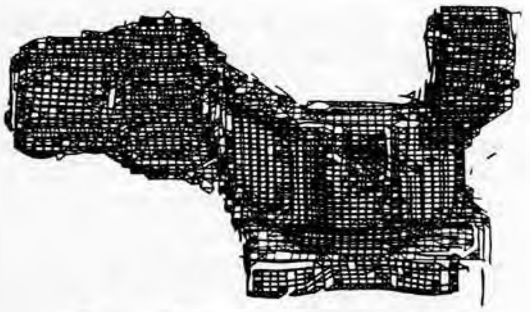
Fig. 6. Side view of spatial proximity graph.

$$\begin{tikzcd} & & & x_1 & \\ m_2 & & m_1 \arrow[r, dash] \arrow[ur, dash] \arrow[dr, dash] & x_2 \arrow[u, dash] \arrow[d, dash] \arrow[r, dash] & x_4 \\ & & & x_3 & \end{tikzcd}$$

Fig. 9. Feature based matching.

the nearest neighbors distance threshold. The nearest m neighbors will not change just because the neighborhood within which they are sought is enlarged.

There are two ways of using the SPG. One takes as input a set of three-space points and outputs a description of the spatial neighborhood relations between those points. The SPG is just a list giving the m nearest neighbors for each point. (Note that the actual vectors output by the sensors are never stored "in" the *k-d* tree or the SPG—only indexes into the data are stored.) For example, the data in Figs. 4 and 5 was obtained from two logical range finder sensors located on opposite sides of the object. The SPG shown in Figs. 6 and 7 was produced by MKS and represents the starting point for further structural analysis. Such a graph can be used to efficiently recover planar faces approximating the data points (see [6] for examples). The other way of using the SPG is to input a set of feature vectors which characterize the objects in a scene. For example, the model reference vector (8,1,700) might represent (number of vertices, number of holes, volume elements). If logical sensors provide a description vector of this kind for each detected object, then in the SPG produced from the set of description vectors combined with the model vector, any node connected to the model vector's node indicates a detected object whose features match the model (see Fig. 9). An edge between a model node and an object node indicates a match, i.e., the object is an instance of the model. In Fig. 9, model m_1, matches objects x_1, x_2, and x_3, while no match was found for model m_2 or object x_4.

The combinatorics of any matching technique must always be closely controlled. Otherwise, inefficient matching makes the analysis too slow. MKS offers the tools by which efficient matching can be accomplished. First, feature-based models are matched to the data directly by building the SPG; if faster processing is required, then the models are organized in a *k-d* tree and the detected objects queried directly on the model tree for the nearest match. On the other hand, when organizing three-dimensional surface data, the kind of matching which is performed is usually some form of surface approximation, and as has been shown elsewhere (see [6]), this is done very efficiently in terms of the SPG.

This method of representation is well-suited to organizing multisensor data. Intuitively the spatial proximity graph makes explicit the neighborhood relations of selected features extracted from the data.

B. Feature Selection

Feature extraction plays a prominent role in image analysis, and there is every indication that it will do so for tactile sensors, also. Features range from the intrinsic characteristics found in images (edges, reflectance, depth, etc.) to physical characteristics of a surface (temperature, smoothness, compressibility).

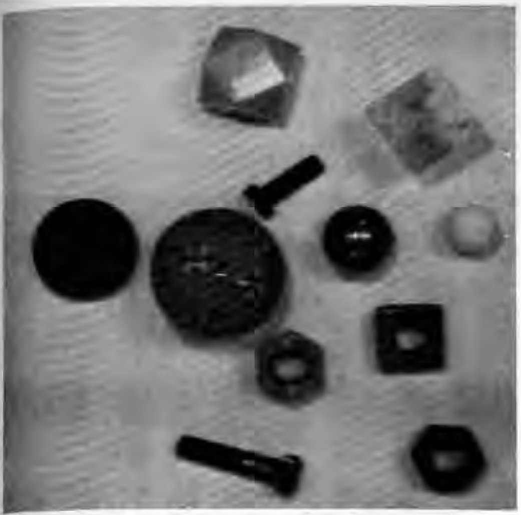
Fig. 10. Scene 1, a two-dimensional scene.

Features are often used to characterize objects, and as time efficiency is of utmost importance, features are usually chosen so as to provide an adequate description and which can be obtained cheaply and reliably. Discovering useful features will no doubt be an outcome of further research, but such features as edges, surface texture, and surface shape are used currently. We view feature extraction as a distinct step performed on the raw sensor data, but obviously a "smart" sensor might provide such features directly.

C. Feature Organization

The cost is prohibitive to try and form the spatial proximity graph directly from the sensor data. Therefore, as a first step, the feature-location pairs are organized into a special tree structure (called the *k-d* tree) that can be built in Order($n \log n$) time for n keys, and that allows the m-nearest neighbors of any given key to be found in Order($\log n$) time complexity. See [5] for a detailed explanation of *k-d* trees. Basically a *k-d* tree is a binary tree of k-dimensional keys that is organized such that at each subdivision step, the data is split at the median along the axis having greatest spread in vector element values along that axis. In our application the feature–location pairs are used as the keys of the tree, and the spatial proximity graph is built by finding for each node, the m-nearest neighbors. This approach has already been studied in the context of feature encoding for satellite imagery [9], [10].

D. Feature Models

Within the multisensor framework, we define each feature model directly in terms of a logical sensor. Every logical sensor has a characteristic output vector that contains all the features detected by the sensor. However, sometimes only a subset of the possible features is used for matching. The *k-d* tree builder only organizes sets of these k features on the *k-d* tree. For matching, we incorporate the model reference vectors into the detected data to build the spatial proximity graph. In the spatial proximity graph, if a detected vector matches a certain reference vector, they will be neighbors. Therefore, in building the spatial proximity graph, it is merely necessary to specify the threshold dissimilarity associated with the model, and a large number of neighbors (this is due to the lack of advance knowledge of the number of detected objects that match the model).

The following three examples illustrate the feature-based object modeling technique.

Circle Model: We use the logical sensor called "circle_detector" to analyze scene 1 (see Fig. 10). The output of "circle_detector" is vectors of the form (x_i, y_i, r_i) where x_i, y_i are the coordinates of the centroid of the ith detected object, and r_i is the radius of the detected object. The model reference vector is

("don't care," "don't care,"2).

"Don't care" indicates that the field it appears in is not be used as part of the model definition. We include the model reference vector at the beginning of the file containing the detected vectors. Table I gives a set of vectors input to the *k-d* tree builder.

Table II gives the resultant spatial proximity graph. The integers on the ith row are the indexes of the neighbors of the ith object. The number of nearest neighbors chosen is 12, and the distance threshold is 0.1.

Since the 5th detected vector is the only one with radius 2, the reference vector and this vector are neighbors.

Nut Model: We use the logical sensor called "corner_holes_detector" to analyze scene 1. The output of "corner_holes_detector" is vectors of the form (x_i, y_i, c_i, h_i) where the x_i, y_i are the coordinates of the centroid of the ith detected object, c_i

TABLE I
VECTORS PRODUCED BY "CIRCLE_DETECTOR"

Object #	(x-location,	y-location,	radius)
0	"don't care"	"don't care"	2
1	256	86	0
2	371	137	0
3	269	171	0
4	397	223	1
5	320	224	2
6	128	240	3
7	230	257	5
8	358	309	0.5
9	333	343	0.5
10	256	429	0
11	384	429	0.5

TABLE II
SPATIAL PROXIMITY GRAPH FOR THE CIRCLE RADII

Object #	Neighbors	
0	5	
1	2	3
2	1	3
3	1	2
4	–	
5	0	
6	–	
7	–	
8	9	10
9	8	10
10	8	9
11	–	

TABLE III
VECTORS PRODUCED BY "CORNERS_HOLES_DETECTOR"

Object #	(x-location,	y-location,	corners,	holes)
0	"don't care"	"don't care"	4	1
1	256	86.	6	0
2	371	137	4	0
3	269	171	8	0
4	397	223	0	0
5	320	224	0	0
6	128	240	0	0
7	230	257	0	0
8	358	309	4	1
9	333	343	6	1
10	256	429	8	0
11	384	429	6	1

TABLE IV
SPATIAL PROXIMITY GRAPH OF NUT DATA

Object #	Neighbors
0	8
1	-
2	-
3	10
4	5 6 7
5	4 6 7
6	4 5 7
7	4 5 6
8	0
9	11
10	3
11	9

is the number of corners detected in the object, and h_i is the number of holes detected. The model reference vector is

$$(\text{"don't care," "don't care,"} \ 4,1).$$

Table III gives a set of vectors input to the *k-d* tree builder.

Table IV gives the resultant spatial proximity graph. The integers on the *i*th row are the indexes of the neighbors of the *i*th object. The number of nearest neighbors chosen is 12.

Since the 8th detected vector is the only one with four corners and one hole, the reference vector and this vector are neighbors.

III. CONFIGURING THE MULTISENSOR KERNEL SYSTEM

The multisensor kernel system (MKS) permits the specification of: 1) both physical and logical sensors; 2) the meaning of the low-level representation in terms of any particular high-level representation; and 3) high-level models.

We will consider the requirements of each of these capabilities. Fig. 11 shows how physical and logical sensors are specified.

Physical sensors are defined by parameters associated with the individual sensor of some known class, e.g, CCD array, TV camera, tactile sensor, etc., Moreover, some indication of operationality of the sensor should be provided. Logical (or abstract) sensors are defined in terms of physical devices and algorithms on their data, e.g., an "edge image" sensor or "surface normal" sensor. It is possible for logical sensors to be defined in terms of other logical sensors. The compilation process involves producing a process that carries out the required computation on the data from the desired physical sensors.

The low-level model must be specified in that meanings must be provided for the elements of the *k*-dimensional vectors stored in the *k-d* tree. This basically amounts to formatting instructions (see Fig. 12). Moreover, the number of neighbors and distance thresholds in the spatial proximity graph must be defined in terms of these meanings. For example, if one sensor returns (x,y,z) location and a measure of the "edgeness" at that location, while another sensor gives a measure of surface curvature at a point, then position in the vector must be assigned for the various features measured. Another use of the *k-d* tree data structure is simply to organize locations where features are detected, i.e., (x,y,z) positions, and associate features measured at those locations with the position vectors. User defined constants and functions necessary to build the *k-d* tree must be specified, too, e.g., bucket size of the terminals.

Finally, the high-level models must be specified, along with some mechanism for matching the models to descriptions derived from the sensor data (see Fig. 13). In principle, many high-level modeling methods can be used, but it is reasonable to choose methods that can better exploit the low-level representation. For example, feature models can be matched to sensor-derived descriptions in a straightforward way. On the other hand, MKS can also be used to organize data for the recovery of primitive shape elements for use in high-level structural analysis.

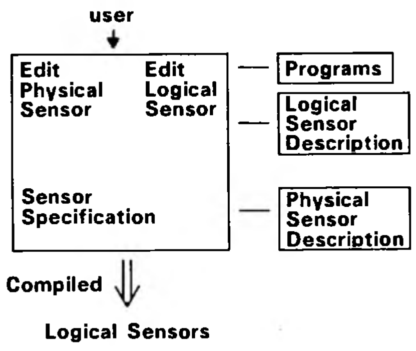

Fig. 11. Sensor specification.

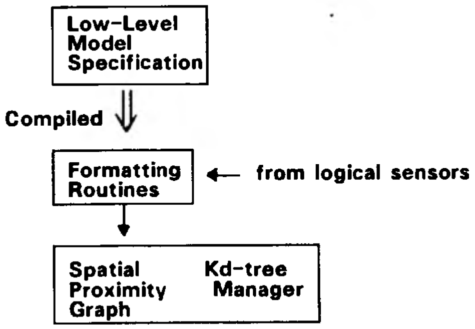

Fig. 12. Low-level model specification.

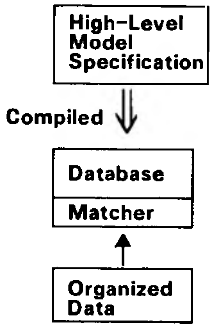

Fig. 13. High-level model specification.

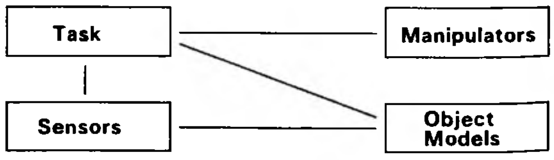

Fig. 14. Relation of task to multisensor kernel system.

Thus, the system is configured by defining the sensors, the low-level, and the high-level representations, and the preceeding paragraphs have given the compile time view of the system. The goal, though, is the performance of some task. Our view is that the task ties the system together as shown in Fig. 14. Thus, the given task description defines when new sensor data is required and how it should be obtained. Matching models to descriptions can take place in the task or can be incorporated directly into the sensor system. Based on the results of the analysis of the data, objects (or the environment) can be manipulated.

Although the development of task definition languages and high-level model techniques are important research topics, the main goal of MKS is to provide a coherent and efficient method

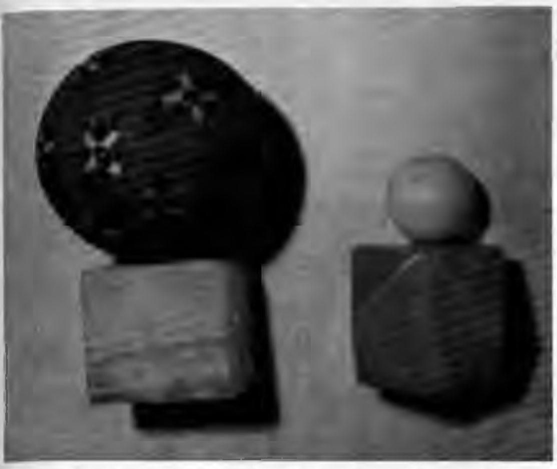
Fig. 15. Scene 2, a set of simple three-dimensional objects.

for obtaining a useful, low-level representation that can serve as the basis for a wide variety of such high-level systems.

IV. High-Level Models

A wide range of high-level modeling techniques are available for use, and these divide naturally into two classes: feature models and structural models. Feature models involve mapping sensed data (or perhaps restricted portions of the data) into a single number or a vector that then represents the data, while structural methods provide a description of the parts of an object and the relations (usually spatial) between the parts (see Henderson [7]).

Most of the current industrial vision systems model objects in terms of global features of objects (or regions), e.g., area, number of holes, hole area, etc. An object model is simply a set of feature values, and an unknown object is identified by how similar its feature measurements are to the reference values. This form of object modeling is supported quite easily by the MKS approach, namely, a logical sensor returns the location and feature vector of any object detected in the image (this generalizes to nonimage-type sensors, too). Matching can be performed by standard methods, or the reference vectors can be stored as a *k-d* tree and then matching merely requires a query on the tree.

We use the logical sensor called "surface-curvature" to analyze the sphere-cube pair in scene 2. (We assume that the cube and the sphere atop it have been separated out from the other objects in the scene, e.g., by the connectivity of the spatial proximity graph based on the three-dimensional surface points.) The output of "surface curvature" is vectors of the form: (x_i, y_i, z_i, n_i) where (x_i, y_i, z_i) are the coordinates of the ith detected object, and n_i is the encoded curvature at the ith surface point. For a curved surface, n_i is encoded as "1." For a planar surface, n_i is encoded as "-1." The planar surface model reference vector is:

$$(\text{"don't care," "don't care," "don't care,"} -1).$$

The curved surface reference vector is

$$(\text{"don't care," "don't care," "don't care,"} 1).$$

Table V gives a small sample of the set of vectors input to the *k-d* tree. The first two entries in the table are the reference vectors, then the next few lines give some sample points from the sphere, and finally, the last few lines give some sample points from the cube.

Table VI shows the corresponding spatial proximity graph for the above input vectors. The neighbors lists for objects 0 and 1 give the planar surface points and the curved surface points, respectively. The spatial proximity graph (SPG) actually divides the set of detected vectors into one set with planar surface and the other set with curved surface. This information is then used to

TABLE V
VECTORS PRODUCED BY "SURFACE CURVATURE"

Point #	(x-location,	y-location,	z-location,	curvature)	
0	"don't care"	"don't care"	"don't care"	−1	Planar Model
1	"don't care"	"don't care"	"don't care"	+1	Curved Model
2	0.0	0.0	3.00	+1	
3	−1.76	0.0	2.43	+1	
4	2.31	1.68	0.93	+1	Sphere Points
5	−2.31	−1.68	0.93	+1	
6	2.31	1.68	−0.93	+1	
		⋮			
7	−3.0	−6.0	−3.0	−1	
8	−3.0	−6.0	3.0	−1	
9	3.0	−6.0	0.0	−1	Cube Points
10	0.0	−6.0	0.0	−1	
11	3.0	−6.0	3.0	−1	
		⋮			

TABLE VI
SPATIAL PROXIMITY GRAPH FOR PLANAR AND CURVED SURFACE POINTS

Point #	Neighbors	
0	7,8,9,10,11	Planar Model
1	2,3,4,5,6	Curved Model
2	1,3,4,5,6	
3	1,2,4,5,6	
4	1,2,3,5,6	Sphere Points
5	1,2,3,4,6	
6	1,2,3,4,5	
	⋮	
7	0,8,9,10,11	
8	0,7,9,10,11	
9	0,7,8,10,11	Cube Points
10	0,7,8,9,11	
11	0,7,8,9,10	
	⋮	

build separate spatial proximity graphs from the two distinct sets of points. We can then perform a higher level analysis based on the surfaces and surface structure (e.g., see Bhanu [2]).

Obviously MKS also provides the basis for structural modeling. Features are detected and organized locally; they can then be analyzed directly (as suggested by Bolles and Cain [3]), or they can be further grouped (say from surface points to faces) and then analyzed. We have investigated several high-level modeling techniques in terms of the MKS, and in particular, feature models and Hough shape models, both with great success [12], [24]. We are now exploring the use of constraint mechanisms for high-level modeling in the context of the MKS system. Also, we hope to use MKS to process tactile data provided by a multi-fingered dexterous hand currently under development by the University of Utah Center for Biomedical Design in conjunction with the MIT AI laboratory. This is a four-finger dexterous hand that includes touch sensors on palmer and finger surfaces. The contact sensors are based on the use of birefringent materials in conjunction with optical fibers [14]. However, other contact sensors such as that described by Hillis [13] or Raibert and Tanner [19] may also be used as they become available.

V. Conclusions and Further Research

The multisensor integration and data acquisition system that is of interest to us is configured by defining the sensors, the low-level representation, and the high-level modeling techniques. The specified task description dictates when new sensor data is required and how it should be obtained. Matching models to descriptions can take place in the task. Based upon the results of the analysis on the data, the objects or the environment can be manipulated.

The primary goal of this research was to develop a system capable of integrating and analyzing data from several sensors in a coherent and efficient way. Along the way we discovered that the specification of the sensors played a large role in the usefulness and success of the system. Moreover, the low-level representation must adequately support several radically different types and formats of data.

We extended previous researchers' two-dimensional low-level representations, such as the region adjacency graph, to a realistic and flexible three-dimensional low-level representation, the spatial proximity graph. We can handle not only three-dimensional visual information, but we can also manage three-dimensional nonvisual information, such as tactile information from a robot hand.

Concerning the issue of object localization, we must take into consideration treatment of errors in detected data, strategies that are best for acquiring new information for object determination, and measures to disambiguate situations such as multiple objects that are similar in one view but different in actuality. In matching we define a dissimilarity measure between the model and the detected object. The distance function for every vector dimension, and the overall tolerance in the matching function should be tailored to take into account errors in the detected data. In gathering new information to complete object determination for partially recognized objects, the strategy is to activate the appropriate sensors for missing features. In distinguishing objects that are similar in one view but different in actuality, information should be gathered from more than one view so as to capture fully the three dimensionality of the target environment.

Based upon case studies with our framework for a multisensor system in a simulated environment, we observe the following in relation to computation speeds for real-time hand manipulation. Before manipulation can occur, objects must be localized. Object localization involves organization of sensor systems through controllers and actuators to achieve a smooth flow of data in a multisensor environment, organization of sensor data into their corresponding spatial proximity graphs, and matching between the world descriptions and the high-level models. Since the spatial proximity graph can be efficiently constructed by an algorithm of order($n \log n$) time complexity, the possibility of real-time hand manipulation is constrained by the efficiency both of the sensors in supplying features of the detected environment, and of the high-level modeling technique used in matching. Another way of considering this issue of real-time manipulation in connection with the tactile sensors is as follows. The robot hand and other sensors, such as a zooming device, which are mechanical devices, consume time when being positioned from one physical location to another. Consequently the amount of sensor data output in a small time interval will most likely be of a manageable quantity that can be organized into a coherent low-level representation by our system. Some of the most important areas for further research are presented in the following paragraphs.

Of crucial importance to building up-to-date spatial proximity graphs to organize a continuous flow of a massive amount of sensor data is the ability to dynamically insert and delete data on a *k-d* tree or any equivalent database storage structure that allows efficient query and searching to be performed on the data. Overmars and van Leeuwen [16] have presented some initial work on dynamic multidimensional data structures, and the usefulness of their results to our application must be investigated.

Another important area for research concerns the logical sensor system. Physical sensors are defined by parameters associated with the individual sensor of some known class, e.g., TV cameras, tactile pads, etc. Logical sensors are defined in terms of physical devices, and algorithms on their data, e.g., an "edge finder" can be defined in terms of a tactile pad on a robot hand, and a pressure analysis program. Associated with each sensor is a characteristic output vector that defines exactly the name and the allowed range of data. With these characteristic output vectors, we can combine vector elements of same name but different allowed range by some appropriate coercion, such as forcing a higher precision range to become a lower one. With this mechanism of treating the "same" kind of data from different precision sensors, in addition to the uniform sensor output, namely a feature and its three-dimensional location, we can integrate sensor data from different sensors with the logical sensor system. An easily reconfigurable sensor system certainly facilitates the acquisition and derivation of appropriate features needed for the construction of a low-level world description which is defined in terms of a high-level model description.

A third area of further research pertains to the investigation of structural modeling techniques which allow the automatic derivation and exploitation of constraints that can then be used to control the acquisition of data, in terms of limiting the amount of data to acquire and specifying the type of data to acquire. It would be interesting to know what kinds of constraints can be derived from such representation methods as generalized cylinders, shape grammars, and relational networks.

In order to have a versatile and robust pattern recognition system, the class of objects to model should include, besides polyhedra, objects with curved surfaces. The inclusion of such objects with curved surfaces naturally opens up avenues for research in determining relevant models that in turn control the kind of features appropriate to be acquired through the low-level representation for object recognition. This may offer some exciting research in conjunction with current research on representing curved surfaces analytically using splines for instance.

Since we hope to operate our multisensor framework in the context of a robot workstation, the list of possible research areas will be incomplete without including research on manipulation of objects. The idea in abstract is that once object recognition and localization are achieved, how should manipulation be performed? The current system provides the necessary first step, namely object recognition and localization, for performing such manipulation tasks.

Acknowledgment

The original range data for the object shown in Fig. 8 was obtained with a laser range finder developed by Francois Germain at INRIA (L'Institut National de Recherche en Informatique et en Automatique) in Roquencourt, France.

References

[1] Harry Barrow, and Jay Tennenbaum, Recovering Intrinsic Scene Characteristics from Images, Tech. Rep. 157, SRI International, April 1978.

[2] Bir Bhanu, Shape Matching of Two Dimensional Objects, IEEE Trans Pattern Analysis and Machine Intelligence, vol. PAMI-6, pp. 137–155, 1984.

[3] R. C. Bolles and R. A. Cain, Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method, *Robotics Res.*, vol. 1, no. 3, pp. 57–82, 1982.

[4] J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesley, 1982.

[5] J. H. Friedman, J. L. Bentley, and R. A. Finkel, An Algorithm for Finding Best Matches in Logarithmic Expected Time, *ACM Trans Math. Soft.*, 3(3):209-226, Sept. 1977.

[6] T. C. Henderson, An Efficient Segmentation Method for Range Data, in *Proc. SPIE Conf. on Robot Vision*, pp. 46–47, Arlington, VA, May, 1982.

[7] ——, Feature-based Shape Models, in *Fundamentals of Computer Vision*, O. D. Faugeras, Ed. Cambridge, MA: Cambridge Univ. Press, 1983, pp. 263–272.

[8] T. C. Henderson and B. Bhanu, Three-Point Seed Method for the Extraction of Planar Faces from Range Data, in *Proc. IEEE Workshop on Industrial Applications of Machine Vision*, pp. 181–186, Triangle Park, NC, May, 1982.

[9] T. Henderson and E. Triendl, Storing Feature Tree Descriptions as 2-D Trees. in *Proc. Pattern Recognition and Image Processing Conf.*, pp. 555–556, June, 1982.

[10] T. Henderson and E. Triendl, "The *k-d* Representation of Edge Descriptions," in *Proc. Int. Conf. Pattern Recognition*, Oct. 1982.

[11] Thomas C. Henderson and Wu So Fai, "A multi-sensor Integration and Data Acquisition System," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 274–280, June, 1983.

[12] T. C. Henderson and Wu So Fai, Pattern Recognition in a Multi-sensor Environment, University of Utah, UUCS 001, July, 1983.

[13] D. Hillis, A High-Resolution Image Touch Sensor. *Robotics Res.*, vol. 1, no. 2, pp. 33–44, Summer, 1982.

[14] S. Jacobsen, Personal Communication, 1982.

[15] D. Marr, Early Processing of Visual Information, MIT, Cambridge, MA, AI Memo 450, Dec. 1975.

[16] M. H. Overmars and Jan van Leeuwen, Dynamic Multi-Dimensional Data Structures Based on Quad- and *K-D* Trees. *Acta Informatica*, vol. 17, pp. 267–285, 1982.

[17] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

[18] G. Pfaff, H. Kuhlmann, and H. Hanusa, Constructing User Interfaces Based on Logical Input Devices, *Comput.*, vol. 15, no. 11, pp. 62–69, Nov. 1982.

[19] M. Raibert and R. Tanner, VLSI Implementation of Tactile Sensing. in *Proc. 12th Int. Conf. on Industrial Robot Techn.*, pp. 417–426. June, 1982.

[20] D. S. Rosenthal, J. C. Michener, G. Pfaff, R. Kessener, and M. Sabin, "The Detailed Semantics of Graphics Input Devices," Computer Graphics, vo. 16, no. 3, pp. 33–38, July, 1982.

[21] M. Shneier, S. Nagalia, J. Albus, and R. Haar, "Visual Feedback for Robot Control," in *IEEE Workshop on Industrial Applications of Industrial Vision*, pp. 232–236, May, 1982.

[22] R. G. Smith, *A Framework for Distributed Problem Solving*. Ann Arbor, MI: UMI Research, 1981.

[23] R. Wesson, F. Hayes-Roth, J. W. Burge, C. Stasz, and C. Sunshine, "Network Structures for Distributed Situation Assessment," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 1, pp. 5–23, Jan. 1981.

[24] Wu So Fai, "A multi-sensor Integration and Data Acquisition System," M.A. thesis, University of Utah, Salt Lake City, June, 1983.