A MULTI-SENSOR INTEGRATION AND DATA ACQUISITION SYSTEM

Thomas C. Henderson and Wu So Fai

Department of Computer Science
University of Utah, Salt Lake City, Utah 84112

1. Abstract

The Multi-sensor Kernel System (MKS) is proposed as a means for multi-sensor integration and data acquisition. This system is being developed in the context of a robot workstation equipped with various types of sensors, including tactile sensors mounted on a dexterous hand and cameras. Specific goals are to:

1. Develop a suitable low-level representation of raw data and/or features extracted from the raw data of the various sensors,

2. Provide a method for efficient reconfiguration of the sensor system in terms of "logical" sensors which map onto physical sensors and computation, and

3. Provide a basis for 3-D object modeling techniques which allow the derivation of constraints useful in controlling and directing the acquisition of data for object recognition.

2. Introduction

The long-term goal is the development of a flexible and programmable robot workstation involving several sensors and several robot arms and hands. The object of such a robot workstation is the automatic inspection or assembly of parts. Clearly such a system requires a capacity for planning actions, modeling objects, and integrating vast amounts of sensor data to those ends. Therefore, a prerequisite to intelligent action is a method for representing and integrating data from several different types of sensors.

Various 2-D low-level representations have been proposed by workers in the image analysis community, however, we propose the spatial proximity graph as a means of describing 3-D spatial relations between features, and a sensory data protocol which allows for efficient integration of non-visual information and features. The significance of such a representation is that it allows uniform handling of data from diverse types of sensor systems, and that given an efficient representational scheme, it allows more rapid exploitation of the sensor data for object identification and manipulation.

Such a system should be easily reconfigurable (perhaps even automatically). Thus, a mechanism will be provided for defining "logical" sensors which may involve several physical sensors and some amount of computation; e.g., a "range finder" can be defined in terms of two cameras and a "stereo" program. Then the sensor system can be "compiled" once all the logical and physical sensors have been defined.

Many methods exist for modeling 3-D objects, and the Multi-sensor Kernel System supports a wide range of 3-D object modeling techniques. Such high level models allow the automatic derivation of constraints which can then be used to control the acquisition of data. This 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 results will be applicable to any multisensor system, including distributed sensing systems, situation assessment systems, etc.

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.

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 notion of "logical" (or abstract) sensor allows for flexible hardware/software mix in terms of a multi-sensor system and permits a simple method of reconfiguation whenever logical or physical sensors are added to or removed from the system.

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 3-D world. Such a representation is crucial to the successful application of multisensor systems, and in particular, robot technology. Shneier et al [14] 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 2.1) as a structuring mechanism for the integration of data from different sensors.

The second major goal is to provide a simple, yet complete, method for (re)configuring a multi-sensor 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 physical sensors is combined to produce the desired data. (See Pfaff et al [11] and Rosenthal et al [13] for a similar approach to computer graphics systems.) Ultimately, such logical sensors could be implemented in special hardware (a "sensor engine").

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 [16]. 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 [15]). 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. Several high-level modeling methods will be investigated, including standard feature models and structural models. Although the system organization and modeling capabilities proposed here are generally applicable to multisensor systems, the focus of this proposal is the design of a multisensor robot workstation. In particular, the system is presented in terms of the integration of visual and tactile data toward the goal of forming a model of the 3-D objects within the range of a robot arm.

3. Acquisition and Organization of 3-D Data

The Multi-sensor 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. Figure 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 Figure 1 has an asociated 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.

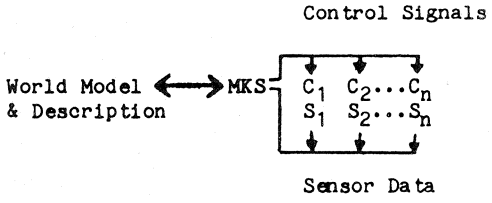

Figure 1. MKS: Multi-sensor Kernel System

The prototype system will consist of two sensors: a camera and a dexterous hand; these will provide visual and tactile data, respectively. Visual information will arrive as digital images which must be processed, whereas the tactile information will be 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 4-finger dexterous hand which includes touch sensors on palmer and finger surfaces. The contact sensors are based on the use of birefringent materials in conjunction with optical fibers [9]. However, other contact sensors such as that described by Hillis [8] or Raibert and Tanner [12] may also be used as they become available.

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.

3.1. Spatial Proximity Graphs

In the context of digital image analysis, various schemes have been proposed for organizing properties or features recovered from 2-D images, e.g., Marr's primal sketch [10], the intrinsic images of Barrow and Tennenbaum [1], and in a more limited context, the region adjacency graph. However, all of these representations were developed with 2-D images in mind, and we propose a more general 3-D organization called the spatial proximity graph.

The spatial proximity graph provides a means for organizing information about the 3-D world. In particular, the approach is to:

1. obtain raw sensory data,
2. extract features from the data and the 3-D locations of these features, and
3. determine the spatial relationships between the features.

The nodes of the spatial proximity graph correspond to the positions in 3-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, high-level analysis can be performed on this graph [4].

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 3-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 3-D range data [5]. For example, consider the surface points of the synthetic cube shown in Figure 2. The spatial proximity graph (SPG) for those points is given in Figure 3. Figures 4 and 5 show the same process for a set of points obtained with a laser range finder used to scan an industrial object. There are about 2000 point samples. Various views of the object points and the spatial proximity graph are shown in Figures 6 through 13. The original object is shown in Figure 14.

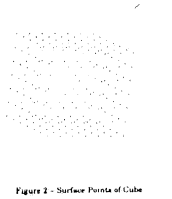

Figure 2 - Surface Points of Cube

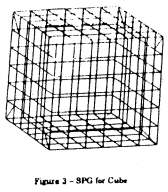

Figure 3 - SPG for Cube

Figure 4 - 0 Degree View of Surface Points

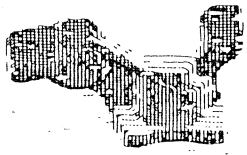

Figure 5 - 0 Degree View of SPG

Figure 6 - 85 Degree View of Surface Points

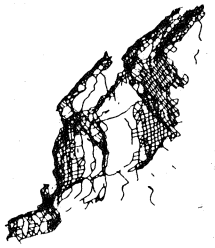

Figure 7 - 85 Degree View of SPG

Figure 8 - 180 Degree View of Surface Points

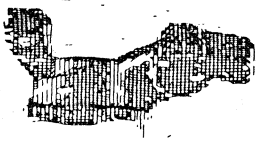

Figure 9 - 180 Degree View of SPG

Figure 10 - Bottom View of Surface Points

Figure 11 - Bottom View of SPG

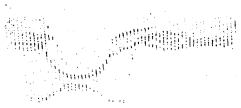

Figure 12 - Top View of Surface Points

Figure 13 - Top View of SPG

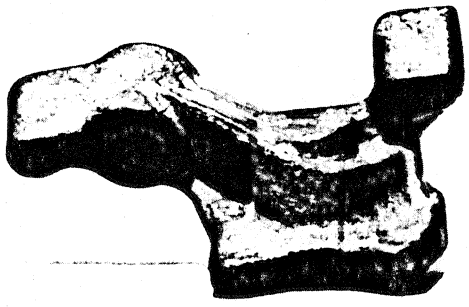

Figure 14 - Original Object

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. If the features are not used in forming the key, then the spatial proximity graph imposes a direct Euclidean nearest neighbors on the features; for example, such a graph can be used to recover planar faces approximating the data when the the features are simply surface points [4].

On the other hand, if the features are encoded as part of the key, then an appropriate choice of the feature values in the feature space dimension can lead to tremendous gains in object recognition efficiency. For example, if linear edges and flat surfaces are features assigned a large positive value in the first key dimension, whereas curved edges and surfaces are assigned a large negative value, then the spatial proximity graph of a scene containing a sphere and a cube, for example, will be disconnected. Obviously, one would like to take avantage of this whenever possible.

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

3.2. 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, compressability).

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 this project, but such features as edges, surface texture, and surface shape will be used initially. We view feature extraction as a distinct step performed on the raw sensor data, but obviously a "smart" sensor might provide such features directly.

3.3. 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 kd-tree) which can be built in Order(nlogn) time for n keys, and which allows the m-nearest neighbors of any given key to be found in Order(logn) time complexity. See [3] for a detailed explanation of kd-trees. Basically, a kd-tree is a binary tree of k-dimensional keys which 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 [6, 7].

4. Configuring the Multi-sensor Kernel System

The Multi-sensor 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. Figure 15 shows how physical and logical sensors are specified.

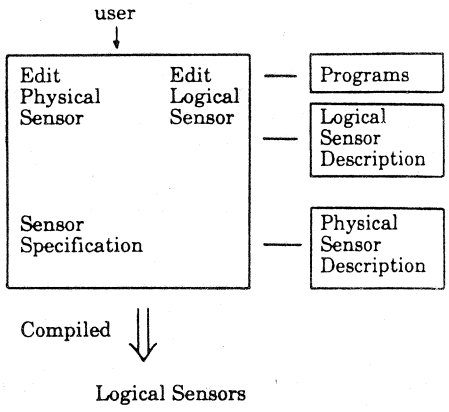

Figure 15 - Sensor Specification

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 may be possible that logical sensors can be defined in terms of other logical sensors. The compilation process involves producing a process which 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 kd-tree. This basically amounts to formatting instructions (see

Figure 16). 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 positions in the vector must be assigned for the various features measured. Another use of the kd-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 kd-tree must be specified, too, e.g., bucket size of the terminals.

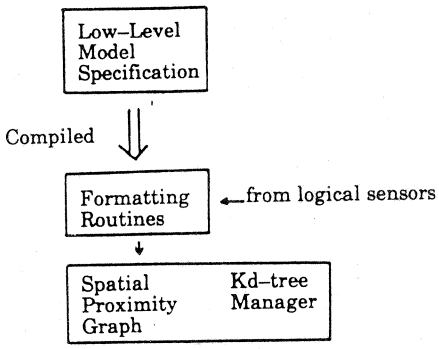

Figure 16 - Low-Level Model Specification

Finally, the high-level models must be specified, along with some mechanism for matching the models to descriptions derived from the sensor data (see Figure 17).

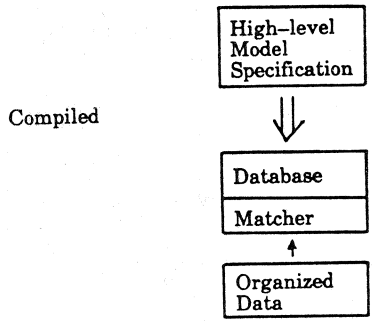

Figure 17 - High-Level Model Specification

In principle any high-level modeling method could be used, but it is reasonable to choose methods which can better exploit the low-level representation. For example, any method based on the graphical representation of spatial relations between the features could correspond directly to the spatial proximity graph.

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 Figure 18. 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.

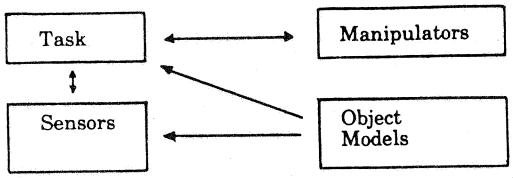

Figure 18 - Relation of Task to MKS

Although the development of task definition languages and high-level model techniques are important research topics, the main goal of this project is to provide a coherent and efficient method for obtaining a useful, low-level representation which can serve as the basis for a wide variety of such high-level systems.

5. 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 which then represents the data, while structural methods provide a description of the parts of an object and the relations (usually spatial) between the parts.

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 non-image type sensors, too). Matching can be performed by standard methods, or the reference vectors can be stored as a kd-tree and then matching merely requires a query on the tree.

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 [2]), or they can be further grouped (say from surface points to faces) and then analyzed. We plan to investigate several high-level modeling techniques in terms of the MKS, and in particular, graph models of feature organization, Hough shape models, and syntactic (or grammatical) methods.

6. Acknowledgements

This work was supported in part by the System Development Foundation. The original range data for the object shown in Figure 14 was obtained with a laser range finder developed by Francois Germain at INRIA (l'Institut National de Recherche en Informatique et en Automatique) located in Roquencourt, France.

7. Bibliography

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

[2] Bolles, R.C. and R.A. Cain.
Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method.
Robotics Research 1(3):57-82, 1982.

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

[4] Henderson, T.C.
An Efficient Segmentation Method for Range Data.
In SPIE Conference on Robot Vision, pages 46-47. Arlington, VA, May, 1982.

[5] Henderson, T.C. and B. Bhanu.
Three-Point Seed Method for the Extraction of Planer Faces from Range Data.
In Proc. of IEEE Workshop on Industrial Applications of Machine Vision, pages 181-186. IEEE, research Triangle Park, NC, May, 1982.

[6] Henderson, T. and E. Triendl.
Storing Feature Tree Descriptions as 2-D Trees.
In Proceedings of Pattern Recognition and Image Processing Conference, pages 555-556. June, 1982.

[7] Henderson, T. and E. Triendl.
The k-d Tree Representation of Edge Descriptions.
In Proceedings of International Conference on Pattern Recognition. October, 1982.

[8] Hillis, D.
A High-Resolution Image Touch Sensor.
Robotics Research 1(2):33-44, Summer, 1982.

[9] Jacobsen, S.
personal communication.
Technical Report, Center for Biomedical Design, 1982.

[10] Marr, D.
Early Processing of Visual Information.
AI Memo 450, MIT, Cambridge Mass, December, 1975.

[11] Pfaff, G., H. Kuhlmann and H. Hanusa.
Constructing User Interfaces Based on Logical Input Devices.
Computer 15(11):62-69, November, 1982.

[12] Raibert, M. and R. Tanner.
VLSI Implementation of Tactile Sensing.
In Proceedings of the 12th International Conference on Industrial Robot Technology Conference, pages 417-426. June, 1982.

[13] Rosenthal, D.S., J.C. Michener, G. Pfaff, R. Kessener and M. Sabin.
The Detailed Semantics of Graphics Input Devices.
Computer Graphics 16(3):33-38, July, 1982.

[14] Shneier, M., S. Nagalia, J. Albus, and R. Haar.
Visual Feedback for Robot Control.
In Workshop on Industrial Applications of Industrial Vision, pages 232-236. IEEE, May, 1982.

[15] Smith, R.G.
A Framework for Distributed Problem Solving.
UMI Research Press, 1981.

[16] Wesson, R., F. Hayes-Roth, J. W. Burge, C. Stasz and C. Sunshine.
Network Structures for Distributed Situation Assessment.
IEEE Trans. on Systems, Man, and Cybernetics SMC-11(1):5-23, JAN, 1981.