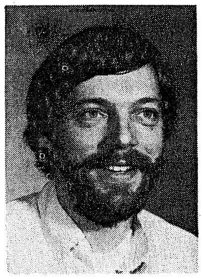
Applications

Organizing Spatial Data for Robotics Systems *

T.C. Henderson **, C.D. Hansen and
Wu So Fai
The Department of Computer Science, The University of Utah, Salt Lake City, Utah 84112, U.S.A.

Successful robotics systems require the organization and analysis of a tremendous amount of sensor data. Moreover, the current computer paradigm of "shape-from" algorithms requires multi-dimensional data from the sensors, e.g. 3-space location and surface normal. Such multi-dimensional vectors must be organized such that spatial searching is efficient and so that spatial proximity is easily determined. Unfortunately, even though methods of computational geometry have been quite successful in dealing with intersection and proximity problems in 2-D, they have not been so successful in dealing with data of dimension greater than two. We define the spatial proximity graph as a low-level organizational structure, and show how it can be built efficiently. Some examples are given.

Keywords: Spatial proximity Graph; *k-d* Trees; Point, Surface and Object Representations, Robotics.



Thomas C. Henderson received the PhD in Computer Science from the University of Texas in 1979. He then spent one year at the Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt near Münich, West Germany as a Research Associate in the Image Analysis Group. The next year was spent as a visiting professor at the Institute National de la Recherche en Informatique et en Automatique near Roquencourt, France. In 1982, he took a regular faculty position at the University of Utah where he is presently an Associate Professor. Professor Henderson's research interests include artificial intelligence, computer vision and robotics, and he has published numerous papers in these areas. He is co-author of *Relaxation Techniques in Computer Vision* to be published by Oxford University Press.

* This work was supported in part by the System Development Foundation and NSF Grants ECS-8307483 and MCS-82-21750.

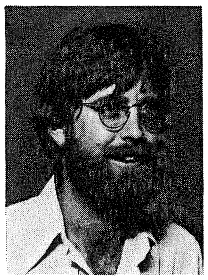
** All correspondence should be addressed to T.C. Henderson.

1. Introduction

Sensed data plays an important role in general robotics systems and will soon play an even bigger role as better sensors appear. The trend is toward higher resolution and shorter time for data delivery. It is becoming very difficult to process in real-time the flood of data delivered by such sensors. We propose the spatial proximity graph as a low-level representation for the organization of *k*-dimensional data from multiple sensors. The solutions to several problems are demonstrated in this context.

Multiple sensors pose several problems for the environment in which they work; it is necessary to coordinate the active control of several sensors and integrate the data from the various sensors into a coherent and useful description of the world. Fig. 1 shows the flow of data in such a system, where S_1 to S_n are the sensor systems.

Each sensor system, S_i, in Fig. 1 has an associ-



Charles Hansen received the B.S. degree from Memphis State University, Tennessee, in 1981. He is currently pursuing a Doctorate degree at the University of Utah where he is the recipient of an ARO fellowship. His interests include computer vision, robotics, and artificial intelligence. He is a member of ACM, SIGART, IEEE, AAAI.

Wu So Fai received an A.B. from Dartmouth College in 1977, and a masters degree in Computer Science from the University of Utah in 1983. At present, she works for Dartmouth College on computer models to display geological formations. She devotes her weekends to designing mylar palm trees for a local theater.

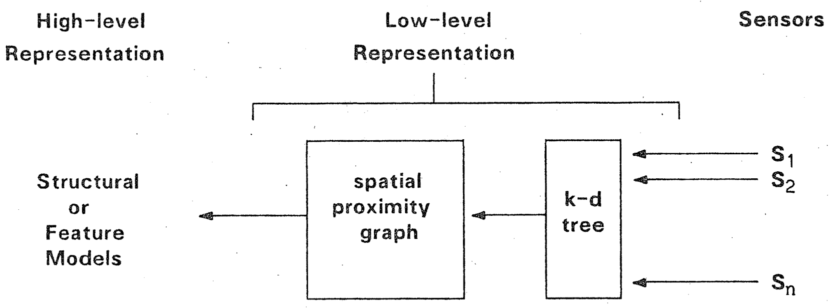


Fig. 1. Framework for Multi-sensor Environment.

ated controller, C_i, for example, a camera may be aimed, focused, or have the aperture setting changed. A sensor system may have several components, e.g.

* *a dexterous hand:* several tactile pads in known relationships due to the finger linkages

* *a stereo range finder:* two cameras with separate controllers for focus, zoom, etc.

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

The second issue that must be addressed is the dimensionalilty of the data. The major result of the last few years of computer vision research is that the sensed data formation process must be inverted to recover the intrinsic characteristics of the real world scene which was sensed. This implies an increased dimensionality of the data in that one or more characteristics may be measured at a given location in space. For example, a vector might have the following elements:

$$(x, y, z, n_1, n_2, n_3)$$

where (x, y, z) is the spatial location of the detected surface element and (n_1, n_2, n_3) is the surface normal at that location. It should be possible to spatially organize the data on any subset of elements of the sensed vector.

The *M*ulti-sensor *K*ernel *S*ystem (MKS) has been proposed as a means for the acquisition, organization and analysis of multi-dimensional data from multiple sensors [7,11]. In general, any set of sensors can be used, and MKS is organized such that each sensor contributes information independently of the other sensors. However, high-level models are used to control the acquisition of data, such that as object recognition and object localization proceed, more explicit data and hence less data will be demanded from the sensors. Specific relations and constraints from the already processed data guide the sensors in further acquisition of new data.

In this paper we discuss the part of MKS in which multi-dimensional data is integrated and spatially organized. It is argued that this representation supports many high-level object modeling techniques, and in particular, feature-based models and certain structural models.

2. Organization of 3-D Data

Upon obtaining raw sensory data, we organize the data as follows:

1. extract features from the data and the 3-D locations of these features, and
2. determine the spatial relationships between the subset of features in their corresponding feature space. The spatial relationships between the data points in the feature space give a measure of the dissimilarity between these data points.

2.1. Feature Selection

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

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 which can be obtained cheaply and reliably. Discovering useful features is an important research area, but such features as

This optimized algorithm minimizes the expected number of records examined during searching for nearest neighbors through choosing both the discriminator key element and partition value for each subfile, and the number of records in each terminal bucket. Two restrictions govern the characteristics of the optimized algorithm.

The first restriction is that the algorithm is too general and is independent of the distribution of queries and only utilizes information contained in the file records. Clearly, such an algorithm works equally well for all possible query distributions and is not optimal for any particular one.

The second restriction is that the possible values for discriminating key and partition at any particular node depend on the subsection represented by that node. This restriction is essential for defining the *k-d* tree recursively and avoiding a general binary tree optimization which is known to be NP-complete and likely of nonpolynomial time complexity [3].

Since information provided to a binary choice is maximal when the two alternatives are equally probable, it is equally likely that a file record will be placed on either side of the partition. Hence, irrespective of which key is selected for the discriminator, the median of the marginal distribution of key values, serves well as the partition.

The search algorithm can stop searching the subsection on the side of the partition opposite the query record if the partition boundary does not intersect the ball centered at the query record with radius equal to the dissimilarity to the m^{th} closest record so far encountered.

With these considerations in mind, the optimized *k-d* tree algorithm chooses at every nonterminal node the key with the largest range in values as the discriminator, and the median of the discriminator key value as the partition. In order to minimize the number of records examined, the terminal buckets should each contain one record.

In searching, the optimized *k-d* tree algorithm provides an efficient scheme for checking only those records closest to the query record, thus greatly reducing the computation required to find the best matches. Hence we will use the *k-d* tree as a means to organize our sensor data to achieve the goal of finding the m nearest neighbors to all target records for the derivation of the neighborhood graph, i.e. the spatial proximity graph.

2.4. Spatial Proximity Graph

The *spatial proximity graph* is a graph, having a distinct node for each distinct vector from a sensor. The nodes of the spatial proximity graph correspond to the positions in feature space of these vectors of features. Nodes are linked by an edge if they are within some specified dissimilarity threshold. 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. The system as currently implemented permits the user to select a subset of the feature vector produced by the sensor. (As a matter of fact, the user can actually choose any non-empty subset of both the location and feature elements.) 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 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 advantage of this whenever possible.

The spatial proximity graph then provides a means for organizing information from different sources. Moreover, high-level analysis in terms of features can be performed on this graph. The *k-d* tree and the spatial proximity graph have already been studied in the context of 3-D range data [4] and feature encoding for satellite imagery [5,6]. This method of representation seems well suited to organizing multi-sensor data. Intuitively, the spatial proximity graph makes explicit the neighborhood relations of selected features extracted from the data.

3. Implementation Considerations

The low-level modeling system consists of three major components:

```
/* build left subtree */
ls_lptr = l_record_ptr;
ls_rptr = median_pos;
leftsubtree_node = kd_tree_builder ( discriminator, partition,
                                     ls_lptr, ls_rptr ) ;

/* build right subtree */
rs_lptr = ls_rptr + 1;
rs_rptr = r_record_ptr;
rightsubtree_node = kd_tree_builder ( discriminator, partition,
                                              rs_lptr, rs_rptr );

        nont_node = make_nonterminal ( discriminator, partition,
                         maxspread, leftsubtree_node, rightsubtree_node );
        return ( nont_node );
}
```

Fig. 3. Code for the *k-d* tree builder.

1. preprocessor to format sensor data into records of *k* real valued keys or attributes as proper input to the *k-d* tree builder,
2. *k-d* tree builder to construct a binary tree for the storage of sensor records of *k* composite keys appropriate for the problem of finding those records in the tree most similar to a query record according to some dissimilarity or distance measure, and
3. spatial proximity graph builder to construct the neighbors graph in which each record in the *k-d* tree is associated with the number of nearest neighbors desired.

3.1. Sensor Data Formatter

Sensor data output by the logical sensors are sets (or streams) of vectors whose components are features of the detected world. Depending on the type of high-level modeling for object recognition and localization, it could be that only a subset of the features is relevant. The sensor data formatter allows the user to choose the relevant subset of features. The formatter establishes a mapping that transforms the full original set of features available for the logical sensor to a chosen subset of *k* relevant features as proper output to the *k-d* tree builder. The original mapping carries with it all the linking information necessary to recover any unused features associated with a particular vector of *k* features.

3.2. k-d Tree Builder

The major procedures to construct an optimized *k-d* tree for best match file search are an adaptation of the algorithm presented by Friedman [3]. The *k-d* tree builder receives as its input the streams of vectors of *k* chosen features from the sensor data formatter. The dissimilarity measure used to estimate the range or spread in feature key values is simply the linear coordinate distance function. In the resultant *k-d* tree, each nonterminal node contains pointers to its left and right successor nodes, discriminating feature coordinate, partition value, and the median index with respect to the discriminating feature coordinate. Each terminal bucket contains a list of indexes of the vectors belonging to it. The indexes can be used to recover the associated unused features of any vector.

The code for the *k-d* tree builder is given in Fig. 3.

3.3. Spatial Proximity Graph

The *k-d* tree search algorithm used to construct the spatial proximity graph is also an adaptation of the one presented by Friedman [3]. The spatial proximity graph results from retrieving for each vector input to the *k-d* tree builder the desired number of vectors closest to this query vector with a given dissimilarity measure. This dissimilarity measure is the same as the one used to build the *k-d* tree. It is possible that the number of nearest neighbors found is less than the desired number.

The code for this is given in Fig. 4. Aside from the general organization outlined above, there are other utilities necessary for debugging and reporting results of computation, and for the develop-

```
                        else
                        {
                                neig_link = NEW_neighbour;
                                neig_ptr->link = neig_link;
                                neig_ptr = neig_link;
                        }
                }

                /* initialize coordinate upper bounds and lower bounds */
                for ( i=0; i<cur_max_dimension; ++i )
                {
                        upper_bound[i] = HUGE;
                        lower_bound[i] = -HUGE;
                }

                /* for each record in file */
                get_query ( j );

                /* find the m nearest neighbours to it */
                done = FALSE;
                search ( node, done );

                /* save neighbours on a link list */
                spg_list ( p );

        }

        return ( spg_ptr );
}
```

and where the search code is:

```
/*
 * search.c - given a file of n records, each of which is described by k
 *            real valued attributes, and a dissimilarity measure, finds
 *            the m records closest to a query record (possibly not in
 *            the file) with specified attribute value.
 *
 * Date: January, 1983.
 * version: a.01
 *
 * The algorithm used here is taken from the following paper:
 * Friedman, J.H., Bentley, J.L., and Finkel, R.A.  An Algorithm for
 * Finding Best Matches in Logorithm Expected Time.  ACM Transaction on
 * Mathematical Software, Vol.3, No. 3, September 1977, Pages 209-226.
 *
 */

#include "kdtree.h"             /* start with definition file of k-d tree */
#include "spg.h"                /* definition file of spg */

/*****************************************************************
 * TAG( search.c )
 *
 *     search
 *
 * argument is the root node of a subtree of a k-d tree, and the
 * status of search.
 *
 */
search( node, done )
treenode * node;                /* root node of a subtree of k-d tree */
boolean done;                   /* status of search */
{
```

```
                        search ( node->right, done );
                }
                lower_bound[discriminator] = prev_lower_bound;
        }

        /* else search the right subtree */
        else
        {
                /* search the right subtree */
                prev_upper_bound = upper_bound[discriminator];
                upper_bound[discriminator] = partition;

                if ( bounds_overlap_ball() )
                {
                search ( node->left, done );
                }
                upper_bound[discriminator] = prev_upper_bound;
        }

        /* see if return or terminate recursive search */
                                if ( ball_within_bounds() )
                                {
                                        done = TRUE;
                                }
                        }       /* end loop to search non-terminal node */
                }       /* end search loop */
        }       /* end function search */
```

Fig. 4.

ment of a multisensor environment in the context of a robot workstation. With the availability of the Evans & Sutherland Picture System 300 (PS 300) which is an interactive graphic system with built-in hardware modeling transformations such as translation, rotation, and scaling, we are able to display the internal data representation whenever appropriate for purposes of debugging and displaying results meaningfully. For example, we display the spatial proximity graph of *m* nearest neighbors by drawing lines connecting each record with its *m* nearest neighbors. With modeling transformations like rotation, we can view the graph from different angles to see if it makes sense. The display procedures are separate from the low-level modeling system. As a result, users can run the separate display modules offline to view the computation results.

4. Examples

We now show some results generated by MKS. The original laser range data is shown in Fig. 5 for a piece used in the construction of a Renault. There are about 2000 data points. A photo of the object is shown in Fig. 6. The spatial proximity graph for that view is shown in Fig. 7. The spatial proximity graph generated from the union of two sets of surface point data obtained from two views of the object is shown in Fig. 8 (with about 4000 data points).

A second example is that of a helicopter, and the original surface point data is shown in Fig. 9. There are about 1300 points. The spatial proximity graph is shown in Fig. 10.

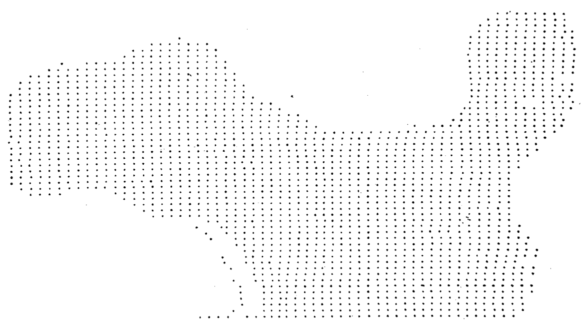


Fig. 5. Original Range Data for Renault Piece.

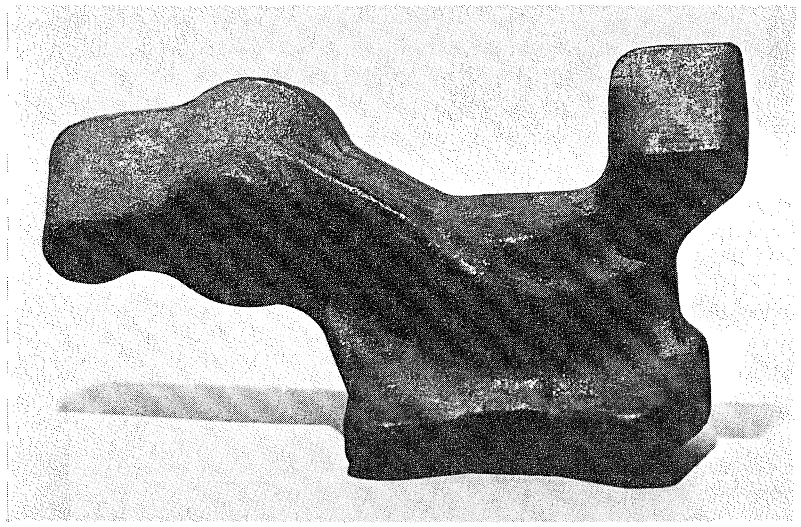
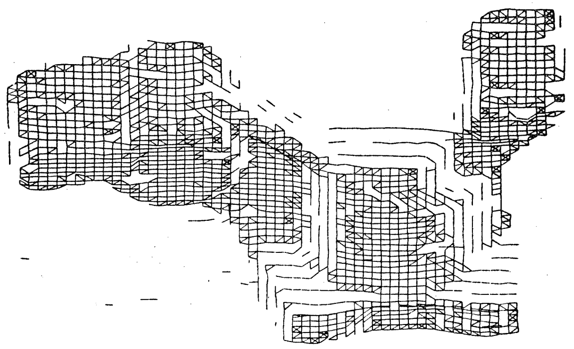


Fig. 6. Renault Piece.



One crucial assumption made in the use of the spatial proximity graph is that the surface to be analyzed is sampled uniformly in all dimensions. Fig. 11 shows points sampled non-uniformly on a cup, and Fig. 12 shows the spatial proximity graph for that data. Clearly, the spatial proximity graph no longer lies on the surface of the object.

Fig. 7. Spatial Proximity Graph of Renault Piece Data.

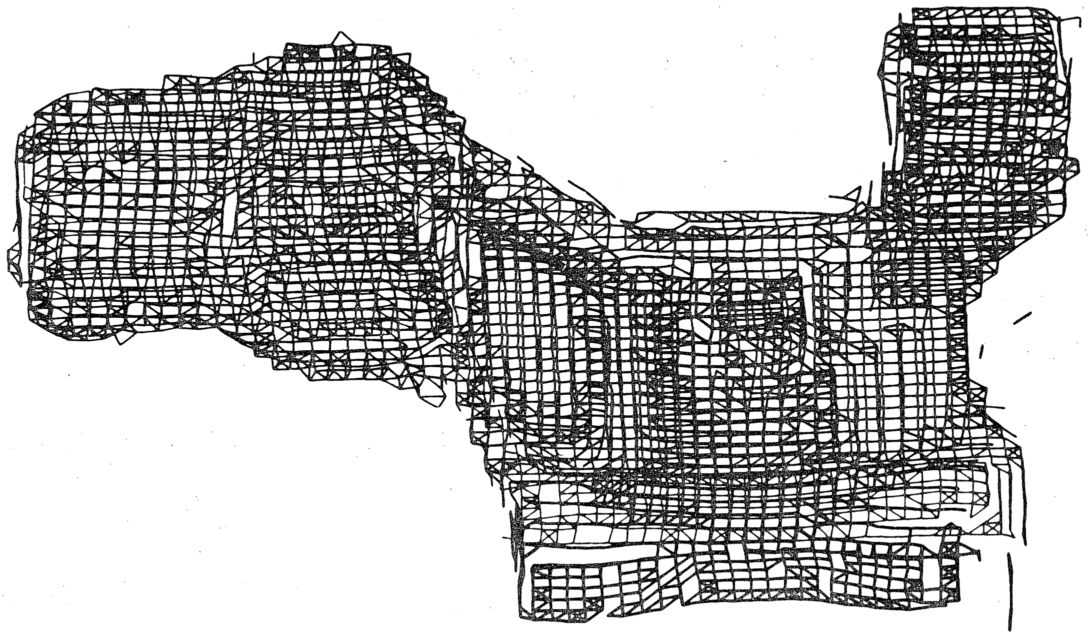


Fig. 8. Spatial Proximity Graph of Merged Data of Renault Piece.

5. Summary

An efficient method for the organization of vector type data has been designed and implemented based on a multi-dimensional divide-and-conquer technique. The spatial proximity graph has been demonstrated as a useful low-level structure for organizing data and has been applied successfully for both 3-D point data analysis and feature based object analysis [11]. It is important that robotics applications take advantage of results in computational geometry if real time processing is to be achieved.

Acknowledgments

We would like to thank Francois Germain at INRIA (Rocquencourt, France) for the range data of the Renault piece, and the α_1 group at the University of Utah for the helicopter and cup data.

References

[1] Ahuja, N.: Dot Pattern Processing Using Voronoi Neighborhoods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4(3): 336–343, May, 1982.

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

[3] Friedman, J.H., J.L. Bentley and R.A. Finkel: An Algorithm for Finding Best Matches ind 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. and E. Triendl: Storing Feature Tree Descriptions as 2-D Trees. In: *Proceedings of Pattern Recognition and Image Processing Conference,* pp. 555–556. June, 1982.

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

[7] Henderson, Thomas C. and Wu So Fai: A Multi-sensor Integration and Data Acquisition System. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 274–280. IEEE, June, 1983.

[8] Marr, D.: *Early Processing of Visual Information.* Al Memo 450, MIT, Cambridge Mass, December, 1975.

[9] Pavlidis, T.: *A Review of Algorithms for Shape Analysis.* Technical Report 218, Princeton U., September 1976.

[10] Toussaint, G.T.: The Relative Neighborhood Graph of a Finite Planar Set. *Pattern Recognition* 12: 261–268, August, 1980.

[11] Wu So Fai: A Multi-sensor Integration and Data Acquisition System. Master's thesis, University of Utah, June, 1983.

[12] Zahn, C.T.: Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers* C-20(1): 68–86, 1971.