THE K-D TREE REPRESENTATION OF EDGE DESCRIPTIONS

Thomas C. Henderson
Department of Computer Science
University of Utah,Salt Lake City, Utah 84112

Ernst Triendl
DFVLR - Oberpfaffenhofen
8031 Wessling, West Germany

1. Abstract

The detection and manipulation of edges in digital images constitue major aspects of many image analysis systems. This edge information is usually kept in another image, the "edge image." Such a representation of the edges is less efficient than the k-d tree representation if the percentage of edges is below a certain level. The tradeoffs between the two representations are presented.

2. Introduction

Several types of intrinsic images (see Barrow and Tennenbaum [1978]) have been shown useful in recovering 3-D scene information. One of the most important and probably the most often used is the edge image. Usually, the edge image is an array registered with the original, and edges are recorded pixelwise. If the edge image is thresholded, the result is a sparse feature image, and in this case a more efficient representation can be found.

We propose to use the 2-D tree which is simply the 2-D version of the kd-tree introduced by Friedman et al [1977] for storing multidimensional keys. The efficacy of this data structure is investigated in the context of the storage of edge features, and in particular, the storage of full edge descriptions including orientation, radius and edge likelihood information. However, the results apply equally well to any set of sparse feature arrays.

3. Edge representation

We assume that edges are detected by some full edge description method: Nevatia, Hueckel, Triendl, etc. For example, an edge crossing a pixel may be represented with subpixel accuracy by a triple (r,a,p) which gives the radius, angle and probability, respectively of that edge with respect to the center of the pixel.

Given such an edge description, then there are multiple channels of output for each channel in the input image. Thus, a four channel LANDSAT image produces a 12 channel edge description image. In most applications, the edge output will be thresholded, and the resulting edge images will be very sparse (e.g., see Henderson et al [1981]).

4. Storage Requirements

Suppose that each input channel is an m by n array, that there are k channels of input, and that there are e storage elements required in the output description. Let $p = mn$. Then, the number of storage elements required for the output description is ekp.

As for the 2-D tree representation, the goal is to organize the features so as to minimize the expected number of records to be examined in a search query. Each (x,y) location of a feature is a 2-D key to be stored. Suppose that there are ap significant responses in each input channel, and that b is the bucket size of each terminal node. Then, there are approximately $(e+2)apk + (4apk/b)$ storage elements required. (The first term is the number of terminals, and the second term is the number of nonterminals.)

Comparing these two, we find that the 2-D tree is more economical whenever:

$$a < e/\{(4/b)+e+2\}.$$

For example, for single channel input, single channel edge description and bucket size 1, the breakeven point is at 14% edge density. This increases quickly with increase in either bucket size or edge description size: e.g., the breakeven point is 25% for either b=4 or e=2.

The cost of operations on the edge description storage structure is also important. Two of the most common operations are channel grouping and spatial grouping of edges. Channel grouping requires kp operations for the array representation, while for the 2-D tree, channel grouping can be accomplished by simple tree traversal requiring $2akp + (4akp)/b$ operations; thus, whenever:

$$a < b/\{2(2+b)\}$$

the 2-D tree is more efficient. This occurs at around 17% edge density for bucket size 1 and 33% for bucket size 4. The breakeven point of spatial grouping is not so easily analyzed, involving a log term for finding nearest neighbors in the 2-D tree; however, arguments can be made that the 2-D tree should outperform the storage array for this operation. First, the spatial grouping is guided by the tree, and consequently only processes where edges occur. Second, since the 2-D tree method is used only if a savings in storage occurs, then the 2-D tree is more likely to fit in core memory than the array structure and less time will be spent in costly disk I/O.

5. Conclusion

The conditions under which the 2-D tree is a more efficient representation of feature images has been described. Moreover, standard grouping operations have been shown to be less costly when performed on the 2-D tree rather than the traditional array representation. The kd-tree has other applications, e.g., as Hough accumulators and for range data organization, and the 2-D tree modules can be useful tools in these other domains. Finally, the 2-D tree allows straightforward access to interesting locations, i.e., exactly where the features occur, and this can be useful in search, display, etc.

6. References

[1] Barrow, H.G. and J.M. Tennenbaum, "Recovering Intrinsic Scene Characteristics from Images," SRI Tech. Rept. 157, April 1978.

[2] Henderson, T.C., E. Triendl and R. Winter, "Edge-Guided Image Registration," 2nd Scand. Conf. On Image Analysis, Helsinki, Finland, June 1981, pp. 106-111.

[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., Vol. 3, No. 3, Sept. 1977, pp. 209-226.