Dynamically Quantized Pyramids

Kenneth R.Sloan, Jr.

Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

Dynamically Quantized (DO) spaces have been developed [O'Rourke] in response to the need for high-precision, high-dimensional Hough-like transforms. [Ballard; Sloan and Ballard) Their purpose is to cover a parameter space with a limited number of accumulators in such a way that fine precision is maintained *where it is needed.*

O'Rourke's solution to this problem is to maintain a binary tree of cells. Each cell covers an n-dimensional rectangular region of the space. Under certain conditions, the cell may be *split* along a particular dimension, and two *sons* created. Under complementary conditions, sets of cells may *merge.* Cell splitting is relatively simple, but the process of cell merging is quite complicated, for reasons which are explored extensively in [O'Rourke].

The solution presented here is based on a *pyramid* data structure, in which the number and connectivity (between *fathers* and *sons)* of cells is fixed. This data structure has the advantage that its resource allocation is fixed, and the cells and their connections may be reduced to a hardware implementation (e.g., in VLSI.) The customary difficulty with the *pyramid* is that the boundaries of the cells (and hence the spatial resolution) are fixed, also. In this *Dynamically Quantized Pyramid* (DQP), the boundaries of the cells are continually modified by means of a hierarchical warping process. Essentially, each cell tries to track the mean position of data points in its part of the space. This estimate of the local mean is used to define the boundaries of the cell's sons.

An experimental implementation has been built and subjected to various distributions (spatial and temporal) of data. The resulting quantizations are shown and discussed.

Introduction

The tradeoff between resource allocation and precision of result is a familiar one. The task of developing a histogram is an example. For a given range of data values, and a given precision with which we need to locate features of the histogram *(e.g.,* a peak), the usual procedure is to quantize the space uniformly, so that each cell covers a small part of the data space. All cells are of the same size, which is small enough to deliver an answer with the required precision.

Usually, the required *precision* determines the amount of *resources* (histogram cells) which are allocated to this task. Sometimes, the precision we want cannot be achieved with the resources available. This is not often the case in one- or two-dimensional histograms. However, techniques which make use of histograms of high-dimensional data (four or more dimensions) are often resource limited.

Hough Techniques

Hough techniques are based on the creation and subsequent analysis of n-dimensional histograms (usually called Accumulator Arrays.) The histogram cells tile a parameter space. Local evidence (usually an edge element) in an image is mapped into this parameter space in a one-to-many fashion, generating a hypersurface in parameter space. Thus, each piece of local evidence *votes* for a large number of possible parameter choices. When a feature of the type being sought *(e.g.,* a line, circle, or shape) is actually present in the image, these hypersurfaces will intersect, forming a peak in the histogram.

This is an attractive paradigm for many low-level visual perception processes [Ballard; Sloan and Ballard; O'Rourke] but is computationally limited when the dimensionality of the parameter space becomes too large. The storage requirements for the histogram (Accumulator Array) become unreasonable, and the time required for generating all of the votes becomes excessive.

Dynamically Quantized Spaces

Dynamically Quantized (DQ) spaces have been developed [O'Rourke] in response to this need for high-precision, high-dimensional Hough-like transforms. Their purpose is to cover a parameter space with a limited number of accumulators in such a way that fine precision is maintained *where it is needed.*

0*Rourke's solution to this problem is to maintain a binary tree of cells. Each cell covers an ndimensional rectangular region of the space. Under certain conditions, the cell may be *split* along a particular dimension, and two *sons* created. Under complementary conditions, sets of cells may *merge.* Cell splitting is relatively simple, but the process of cell merging is quite complicated, for reasons which are explored extensively in [O'Rourke].

Dynamically Quantized Pyramids

The solution presented here is based on a *pyramid* data structure, in which the number and connectivity (between *fathers* and *sons)* of cells is fixed. This data structure has the advantage that its resource allocation is fixed, and the cells and their connections may be reduced to a hardware implementation *(e.g.,* in VLSI.) The customary difficulty with the *pyramid* is that the boundaries of the cells (and hence the spatial resolution) are fixed, also. In this *Dynamically Quantized Pyramid* (DQP), the boundaries of the cells are continually modified by means of a hierarchical warping process. Essentially, each cell tries to track the mean position of data points in its part of the space. This estimate of the local mean is used to define the boundaries of the cell's sons.

A pyramid [Tanimoto and Pavlidis] is a full, balanced tree, in which each internal node has exactly 2^n sons. For the purpose of spanning a parameter space, n is the dimensionality of that space. Each node of the pyramid covers an n-dimensional rectangular region of the space, and divides this region into 2^n sub-regions with an n-dimensional cross-hairs. Typically, this division is fixed at the center of the region. Associated with each node is a *count* of the values entered within its boundaries.

In a *Dynamically Quantized Pyramid* (DQP), the position of the dividing cross-hain is stated in terms of a vector of *percentages,* rather than absolute position. The initial position of all crosshairs is (50,50,...,50). This is the same as the typical configuration of a fixed-boundary pyramid. To determine the absolute boundaries of a node / in a DQP, it is necessary to start at the root of the tree (which covers the entire space, by definition) and follow the path to /, calculating the absolute boundaries of each son node by applying the percentages associated with the father node to the absolute boundaries just calculated for the father node.

The advantage gained is that a DQP may be hierarchically warped. Consider a single node in the pyramid (e.g., the root). When a value is added, the cross-hairs associated with the node is adjusted by moving it a small amount towards this new value. One way to do this is to take the weighted average of the position of the new value (weighted by *a)* and the old cross-hain position (weighted by la). This has the efTect of implicitly changing the boundaries (in absolute terms) of all of the nodes in the sub-tree rooted at this node. The value being inserted is then recursively added to the appropriate son. This will further warp the boundaries of the nodes in that sub-tree.

The effects of this warping process are three-fold. First, the rectangular regions grow smaller near inserted values. Each father node is adjusting its cross-hairs so as to evenly distribute votes among its sons. The resulting histogram tends to have the same number of votes in each bin. The information about the shape of the histogram is largely in the relative sizes of the bins. Second, note that the configuration of the DQP is dependent on the recent insertion history. This is undesirable when the temporal sequence of insertions into the histogram is unimportant, or (worse) when it is based on some regular process (such as a raster scan of an image) which is irrelevant. On the other hand, when temporal sequence is important (as when the world is changing) this property provides an automatic focussing mechanism. Finally, note that the previously inserted values (as represented by counts associated with each node) move around in the space. Thus, the final counts may be wrong. This can be dealt with either by insisting that old votes fade with time,as is appropriate in a changing environment, by separating the warping stage from the actual counting stage, or simply by accepting the inaccuracy.

Experimental Results

Figure 1 shows the response of a two-dimensional DQP to a series of 100 samples taken from a single two-dimensional gaussian distribution. Figure 2 shows the response after 100 samples chosen from a second gaussian. Figure 3 show the response when 100 samples from each of these two distributions were randomly intermingled.

Conclusion

The preliminary experiments shown above indicate that the DQP shows promise as a solution to the space-precision tradeoff problem in multi-dimensional histogramming applications such as Hough techniques. The space-efficiency is not as impressive as that demonstrated by [O'Rourke], but this is offset by a relatively time-efficient insertion procedure. In particular, the DQP cannot ignore irrelevant dimensions of the space, and all nodes representing splits along these dimensions are essentially wasted. However, the fixed resource allocation and tree structure is ideal for hardware implementation, and certainly simplifies software implementation.

The sensitivity of the DQP to the temporal sequence of votes can be controlled by manipulating the value of a, the feed-back parameter. When continuous tracking of moving peaks is desirable, use a constant value for a. When gradual convergence to a stable state is desirable, make α inversely proportional to the number of votes in the histogram so far.. When an initial training phase is to be followed by a final counting phase, leave α constant during the training phase and $\alpha=0$ thereafter.

References

Ballard, D.H Generalizing the Hough Transform to detect arbitrary shapes, *Pattern Recognition,* Vol. 13, No.2, pp. 111-122, 1981.

O'Rourke, Joseph Dynamically Quantized Spaces Applied to Motion Analysis, *Proceedings: IJCAI-81,* Vancouver, Canada, August 1981.

Sloan, K.R., Jr. and D.H. Ballard Experience with the Generalized Hough Transform, *Proceedings: 5th International Conference on Pattern Recognition,* pp. 174-179, December 1980.

Tanimoto, S.L. and T. Pavlidis A hierarchical data structure for picture processing, *Computer Graphics and Image Processing,* vol 4, no. 2, pp. 104-119, 1975.

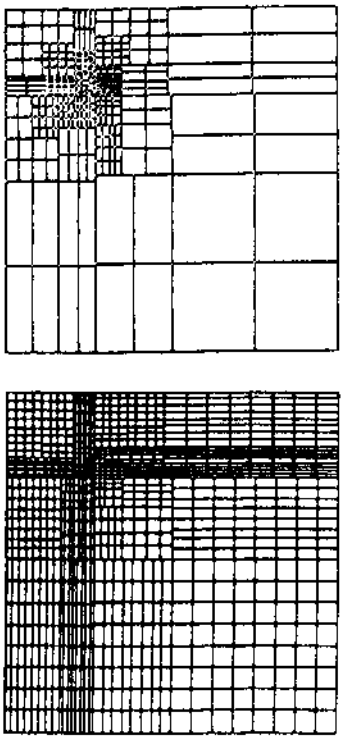

Figure 1

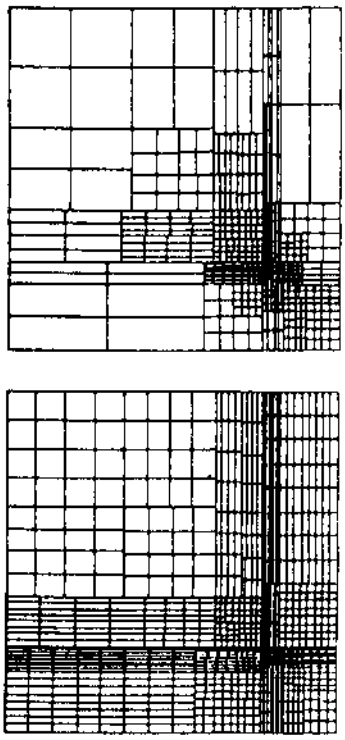

Figure 2

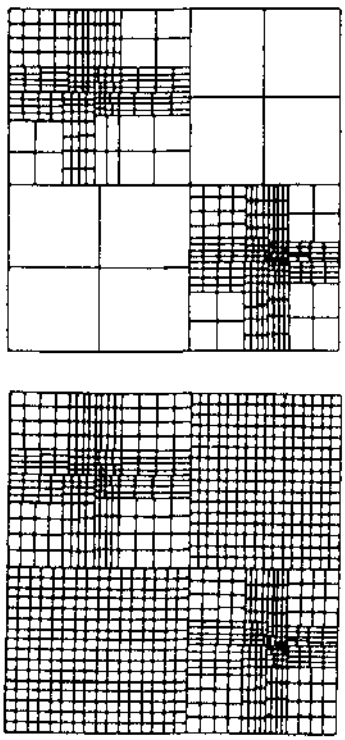

Figure 3

Figures 1-3: Top row shows populated cells; bottom row shows all cells.