A Differential Code for Shape Representation in Image Database Applications

Claudio Esperança *
COPPE, Prog. Eng. Sistemas
Universidade Federal do Rio de Janeiro
Cidade Universitária, C.T., Sala H319
Rio de Janeiro, RJ, 21945-970, Brazil
E-mail: esperanc@lcg.ufrj.br

Hanan Samet †
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Science
University of Maryland at College Park
College Park, Maryland 20742
E-mail: hjs@umiacs.umd.edu

Abstract

A new method termed the vertex representation *is presented for approximating the shapes of objects of arbitrary dimensionality d (e.g., 2D, 3D, etc.) with orthogonal $(d-1)$-dimensional faces using a variable number of vertices. The vertex representation can be viewed as a generalization of boundary codes (i.e., chain codes) to higher dimensions. Techniques are described for using the vertex representation to efficiently perform many operations commonly performed on rasters. The utility of these techniques in image database applications is discussed.*

1 Introduction

The description of shape is an important issue in image databases. In particular, we need to be able to match objects not just by their color or texture, but also by their shape. This requires the ability to index the shapes. Shapes are not often used as the basis of the index due to their sheer volume and the fact that a variable resolution representation (in dimensions higher than 2) is difficult to obtain (e.g., QBIC [7]).

Shapes of objects can be described either by their interiors or by their boundaries (see, e.g., [8, 9]). The classical interior-based shape description method uses rasters or voxels, but these usually take too much space to be of practical use even when aided by a compression scheme such as runlength encoding. Alternatively, the classical boundary-based shape description is the chain code [3]. Unfortunately, it is difficult to extend the chain code to dimensions higher than two. The problem, in part, lies in capturing connectivity, as well as the varying resolution.

In this paper, we present a new method termed the *vertex representation* for describing shape that can be viewed as a generalization of the chain code to higher dimensions. It was first introduced by Shechtman [10] who applied it in the context of VLSI design as a model for representing collections of isothetic rectangles in two dimensions. The vertex representation is similar to a differential code in the sense that it enables the user to only record the changes in the values rather than the values themselves.

2 Vertex Representation

The vertex representation is primarily designed to represent orthogonal scalar fields such as digital images. A scalar field is simply a function that maps points of the underlying domain which is a d-dimensional space to scalar values. An orthogonal scalar field is a scalar field where regions of the domain space mapped to the same value are delimited by faces orthogonal to the coordinate axes.

The null element of the vertex representation corresponds to the null scalar field – a function that maps all points of space to zero. The value of the field is altered by adding structures called $vertices$[1] to the representation. A vertex v is defined as a pair (p, α) where p is the location of v (i.e., it is a point) and α is a non-zero integer value known as the *weight* of v. The weight α is used to determine the value of the scalar field anchored at location p. The physical interpretation of a vertex is one of serving as the tip of an infinite cone.

Let $v = (p, \alpha)$ be a vertex. Therefore, v corresponds to a scalar field Q_v that maps all points in the *cone* of v to α and all other points to 0. A point q with coordinates $(q_1, q_2 \cdots q_n)$ is said to be in the cone of v if $q_i \geq p_i$ for all $1 \leq i \leq d$. Figure 1 illustrates the scalar field induced by a single vertex in $\mathcal{R}^2$ with weight $+1$.

*The support of Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Projeto GEOTEC / PROTEM II is gratefully acknowledged.

†The support of the National Science Foundation under Grant IRI-92-16970 is gratefully acknowledged.

[1] Note that we use the term "vertex" to refer to a mathematical entity that does not necessarily correspond to the common definition of "vertex" in Geometry.

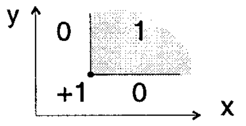

Figure 1: Scalar field in $\mathcal{R}^2$ corresponding to a single vertex with weight $+1$.

Similarly, a set of vertices $V = \{v_1, v_2 \cdots v_n\}$ induces a scalar field $Q_V = \sum_{i=1}^{n} Q_{v_i}$. In other words, the scalar field induced by a set of vertices is the sum of the scalar field induced by each vertex separately. This means that the operation of addition is well-defined for vertices.

The vertex representation can also be interpreted as a differential code for rasters. Consider four adjacent cells with values A, B, C, and D of a 2D raster as depicted in Figure 2. The cell with value A is influenced by vertices lying in regions R_1 and R_2, the cell with value B by vertices lying in region R_2, and the cell with value C by vertices lying in regions R_2 and R_3. Thus, if we want the upper right cell to have a value of D, then a vertex should be placed at point p with weight equal to the difference between D and the weights of all vertices in regions R_1, R_2 and R_3. These weight sums can be rewritten in terms of A, B, C and D so that, after a bit of algebraic manipulation, we find that the weight for the vertex at p is given by D+B−A−C.

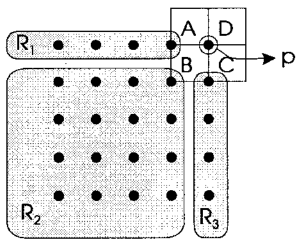

Figure 2: Differential coding scheme: the vertex at p should have a weight equal to D+B−A−C.

3 Vertex Lists

In the previous section we saw how to represent certain kinds of scalar fields by means of sets of structures called vertices. An interesting issue is how to organize these sets so that the processing of the vertices can be done systematically. Shechtman's approach [10] for 2D vertex sets is to store them in lists where the vertices are sorted first by the values of their y coordinate and then by the values of their x coordinate. For instance, vertices a, b, c and d as shown in Figure 3 would be stored in a vertex list in the order c d a b. The utility of this organization derives from the fact that the vertices are stored in an order which is compatible with the sweeping of a line perpendicular to the y axis in the direction of increasing y coordinate values. This permits the development of many algorithms that use the plane-sweep paradigm (or line-sweep in 2D).

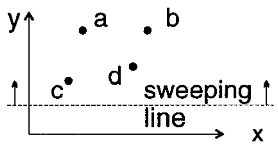

Figure 3: Vertices a, b, c and d would be visited by the sweeping line in the order c d a b.

4 Algorithms on Vertex Lists

The ordering of vertices within a vertex list, as discussed in the previous section, makes it possible to apply the plane-sweep paradigm in the development of many algorithms. In fact, most of these algorithms are capable of handling vertex lists representing scalar fields defined in any dimension $d \geq 1$, by using what we call *recursive* plane sweep. The main idea is to compute the solution of a problem involving vertex lists defined in d dimensions by combining partial results involving vertex lists defined in $d-1$ dimensions. We now proceed to describe informally algorithms[2] on vertex lists which implement some operations commonly performed both on rasters and on features extracted from rasters (i.e., orthogonal polygons).

4.1 Converting rasters to vertex lists

The interpretation of the vertex representation as a differential code makes it easy to obtain the vertices of a 2D raster by performing a simple calculation with the four pixel values surrounding every pixel corner. However, a more elegant way of doing this makes use of the recursive nature of vertex lists and applies a plane-sweep approach. In particular, the conversion algorithm uses induction where the base case corresponds to encoding rows of pixels into one-dimensional vertex lists and the general case corresponds to building d-dimensional vertex lists by subtracting consecutive $d-1$-dimensional vertex lists (see Figure 4).

4.2 Transformations and set operations

A common operation on scalar fields composes them with a transformation function $f : \mathcal{Z} \rightarrow \mathcal{Z}$. That is, if L is a vertex list representing a particular scalar field Q_L, then applying f on L is equivalent to evaluating $f(Q_L)$.

One major application of transformations is to perform set-theoretic operations. For instance, consider two orthogonal polygons represented by their vertex lists A and B. In the scalar field represented by the list $L = A + B$, those areas originally covered by either A or B are mapped to 1,

[2] Full descriptions can be found in [1].

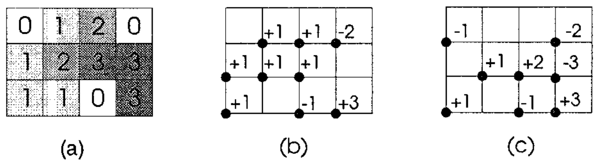

Figure 4: (a) A sample 2D raster; in (b) each row was processed to yield 1D vertex lists, and in (c) consecutive 1D vertex lists are subtracted to yield a 2D vertex list for the raster.

while those areas covered by both A and B are mapped to 2. It is possible to obtain the *union* of A and B by applying to list L the transformation

$$f_{\cup}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Transformations that yield other set operations can be similarly defined.

Performing a transformation f on a scalar field represented by a d-dimensional vertex list follows the same lines as the raster-to-vertex-list algorithm by transforming and combining vertex lists of increasing dimensions.

4.3 Coarsening

This is a shape simplification operation which takes as input an orthogonal polygon containing N vertices and produces as output another polygon containing no more than $M < N$ vertices [2]. The algorithm is quite involved and is not explained here. Figure 5 illustrates this operation as performed on an example orthogonal polygon for varying values of M.

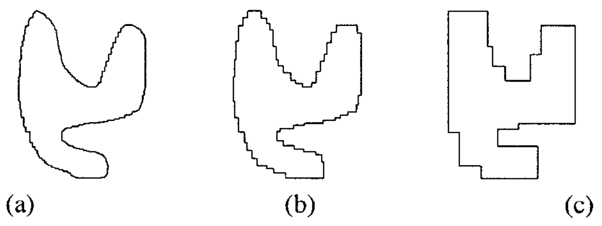

Figure 5: Coarsening example: (a) Orthogonal polygon with 326 vertices is coarsened to (b) 100 and (c) 25 vertices

5 Applications in Image Databases

The vertex representation and its implementation as a vertex list provides a means for representing axes-aligned data that combines the strong points of traditional representations based on interiors (e.g., rasters) and boundaries (e.g., chain codes). This duality enables the vertex representation to replace or complement those more traditional representations in many applications.

5.1 Image Processing and Object Recognition

The central operation in image databases is retrieval of images by content. This operation presupposes the existence of some mechanism whereby image characteristics are extracted and classified. Such mechanisms treat images either as a whole unit in order to obtain information on color or texture, or as containing a collection of individual objects that are identified using image processing and pattern recognition techniques. Some image database systems (e.g. TRADEMARK and ART MUSEUM [6]) employ mechanisms of the former kind, while others (e.g., DDIH [4]) concentrate on mechanisms of the latter kind; still others combine both approaches (e.g., QBIC [7]).

When images are treated as a whole unit, we can use the vertex representation to facilitate the extraction of global characteristics. For example, it is possible to compute the average color of an image by performing a single traversal of the corresponding vertex list. The texture coarseness of a region of the image also has a direct relationship with the number of vertices that appear in that region.

When image processing techniques are applied to an image in order to identify the various objects lying therein (a process known as image segmentation), we want to obtain separate representations of each object. It is possible to decompose a vertex representation of a whole image into separate vertex representations which represent each object in the scene. Figure 6 demonstrates this concept. In this case, the vertex representation for the whole image

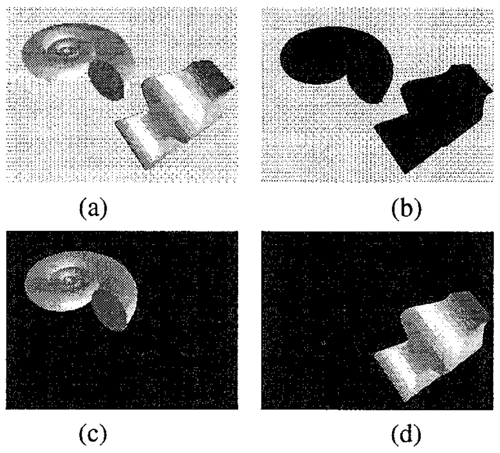

Figure 6: Decomposition of an image into several vertex lists. The image shown in (a) can be represented a vertex list and decomposed into three separate vertex lists corresponding to the background (b), and the two object features (c) and (d).

contains 168304 vertices (Figure 6a). It corresponds to a scene composed of a background and two objects which can be separated using image processing techniques into three vertex representations containing 124689, 20149, and 25502 vertices, respectively.

5.2 Image Summaries

In many image database applications there is frequently a need for a compact representation of the pictorial information. For example, image databases frequently employ reduced versions of the images that will compose the database (called *thumbnails*).

Thumbnails, however, reduce the original image without taking into account small details that may be fundamental for the recognition of key features. On the other hand, since most systems perform some form of image segmentation where key features are identified, it is possible to take advantage of this previous processing in order to prepare summarized images. The vertex representation could be used for such a non-uniform reduction of detail by retaining vertices where there are many changes in the boundary. For example, the two objects shown in Figure 6 are clearly more "interesting" than the background, even though their vertex representations are considerably smaller than that of the background. Hence, using the vertex representation it would be possible to store these in roughly one-third of the space for the whole image.

Image summaries may also be prepared for inspection by the users of the image database. This inspection process usually entails a quick perusal of the summaries of those images that were selected by the system based on user input parameters. The user then selects a subset of these images which will be retrieved in greater detail. This task may be speeded up by what is known as *progressive* or *incremental* image encoding. The idea is to encode and transmit images in increasing levels of detail by expanding rougher versions of the same image that were already transmitted. This topic has been a subject of active research in recent years with techniques based on wavelet theory (e.g., [5]) finding particular promise. Vertex representations can also be used to achieve this effect: the vertices that comprise a vertex representation of a given image need not be transmitted in the same order as they would appear in the final form presented to the user (e.g., as a vertex list), but rather in some order which would reveal image details incrementally. Determining a suitable order for the transmission of vertices is still an open problem, but an approach based on coarsening may lead to a practical method.

6 Summary and Conclusions

We presented a coding scheme for data commonly represented as rasters. Being a differential code, it has a direct relationship with feature boundaries and is usually more compact than uncompressed rasters. We described a data structure for organizing this code in lists and demonstrated how several useful operations can be performed directly on objects thus represented. This representation also finds use in many applications in fields such as image processing and image databases.

References

[1] C. Esperança. Orthogonal objects and their application in spatial databases. Technical Report CS-TR-3566, Computer Vision Laboratory, Center for Automation Research, University of Maryland, December 1995.

[2] C. Esperança and H. Samet. Orthogonal polygons as bounding structures in filter-refine query processing strategies. In *Proceedings of the 5th International Symposium on Spatial Databases*, Berlin, Germany, July 1997. Springer Verlag.

[3] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, March 1974.

[4] W. I. Grosky and R. Mehrotra. Index-based object recognition in pictorial data management. *Computer Vision, Graphics, and Image Processing*, 52(3):416–436, December 1990.

[5] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of the SIGGRAPH'95 Conference*, pages 277–286, Los Angeles, CA, August 1995.

[6] T. Kurita and T. Kato. Learning of personal visual impression for image database system. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 547–552, Tsukuba Science City, Japan, October 1993.

[7] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The qbic project: querying images by content using color, texture and shape. In *Proceedings SPIE Storage and Retrieval of Image and Video Databases*, volume 1908, pages 173–187, San Francisco, February 1993.

[8] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, Reading, MA, 1990.

[9] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[10] J. Shechtman. Processamento geométrico de máscaras VLSI. Master's thesis, Eng. Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, April 1991.