COMPUTER GRAPHICS AND IMAGE PROCESSING 10, 289–296 (1979)

NOTE

Linear Transformation of Pictures Represented by Quad Trees[1]

G. M. HUNTER[2] AND K. STEIGLITZ

Department of Electrical Engineering and Computer Science, Princeton University, Princeton, New Jersey 08540

Received June 28, 1978; revised December 8, 1978; accepted February 2, 1979

An algorithm is described for transforming a quad tree encoding for a picture into a quad tree for the same picture after application of a general linear operator. The technique relies on algorithms and data structures described previously for the construction and superposition of quad trees. The transformation algorithm is shown to take time $O(n + sp + mq)$, where n is the number of nodes in the input tree, s is a scale factor, p is the total perimeter of all the regions of the original picture, m is the number of regions, and q is a resolution parameter.

1. INTRODUCTION

In this paper we describe an algorithm to transform a picture represented by a quad tree. In the quad tree representation of an image the root represents an entire square field, and each node has either four children, each representing a quadrant of its parent, or is a leaf representing a quadrant. Figure 1 shows a simple example of a quad tree representation of a picture. The ordering of the children of each node from left to right in this sketch of the tree is SE, SW, NW, NE. Solid dots at the leaves represent shaded quadrants; hollow dots represent blank quadrants.

Klinger and Dyer [1] provide a good bibliography of the history of quad trees. Their paper reports experiments on the degree of compaction of picture representation which may be achieved with tree encoding. Pavlidis [2] reports on approximation of pictures by quad trees, and Horowitz and Pavlidis [3] show how to segment a picture by polygonal boundaries using traversal of a quad tree. Tanimoto [4] discusses distortions which may occur in quad trees for pictures, and he observes in [5, p. 27] that quad tree representation is particularly convenient for scaling a picture by powers of 2. Quad trees are also useful in graphics and animation applications [6, 7, 11], which are oriented toward construction of images from polygons and superposition of images. Encoded pictures are

[1] This work was supported by National Science Foundation Grant GK-42048 and U. S. Army Research Office (Durham, North Carolina) Grant DAAG29-75-G-0192.

[2] Dr. Hunter is now with Decisions and Designs, Inc., McLean, Virginia.

0146-664X/79/070289-08$02.00/0

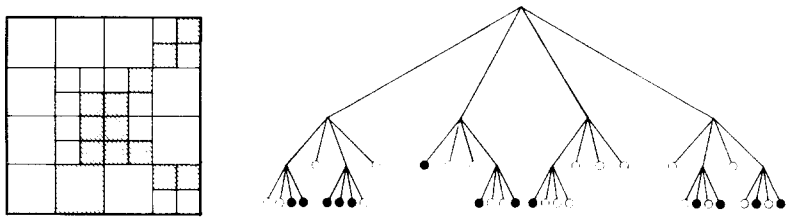

FIG. 1. An example of a picture and a quad tree representing it.

especially useful for display [8] if the encoding lends itself to processing of the image while in encoded form, and it is that sort of problem that we study in this paper.

In particular, we consider the following problem: Suppose we are given a quad tree for a picture. How may we efficiently generate a quad tree for the same picture after application of a general linear transformation T? (This will include translation, scale change, rotation, and combinations thereof.) The input to the algorithm is therefore a quad tree for a picture and a transformation $T(x, y) = (ax + by + c, dx + ey + f)$, where a, b, c, d, e, and f are constants. The output will be a quad tree representing the picture which is obtained when the input picture is mapped, point-by-point, by T.

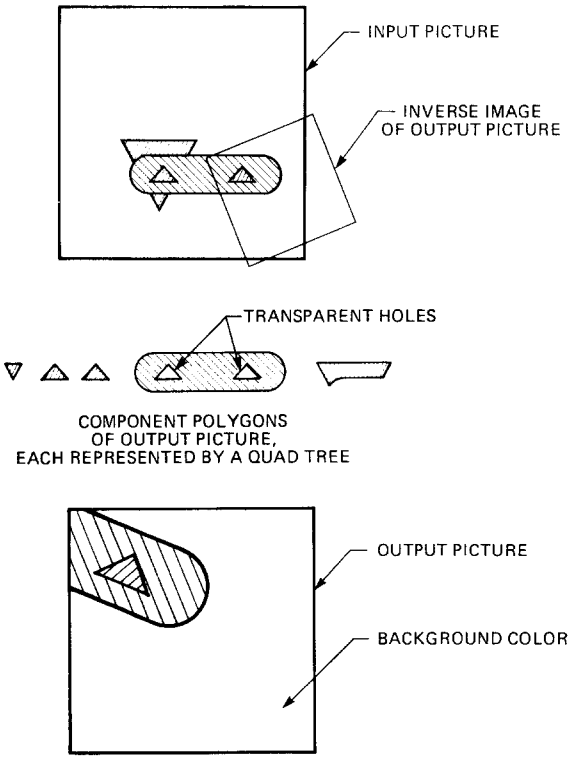

FIG. 2. Pictorial description of the algorithm.

The idea of the algorithm can be described informally as follows. Each region of uniform color in the input picture is in fact a polygon, with sides that are edges of squares in the input picture's quad tree. These polygons are traced from vertex to vertex, and the images of these vertices determine polygons in the output picture. In [9, 10, 12] it is shown how to construct in the output picture a quad tree for each of these polygons. The final picture is obtained by superposing these polygons, with the polygons interior to a given polygon represented by holes—regions of a transparent color. Figure 2 illustrates the process.

To describe the algorithm precisely, we must first repeat the definitions and main results of [9, 10, 12], which we do in the next section.

2. DEFINITIONS AND SUMMARY OF PREVIOUS RESULTS

Pictures. A picture is a square array of colored points. The color of a point may in fact be any information to be associated with a point in a two-dimensional grid.

Quad trees. A quad tree is a tree whose nodes are either leaves or have four children. The children are ordered: 1, 2, 3, and 4.

Quad trees for pictures. A quad tree for, or representing, a picture is a quad tree whose leaves represent areas of the picture. Each leaf is labeled with the color of the area of the picture which it represents. Each node is associated with a square region of the picture, and the root is associated with the entire picture. Besides the root, each other node is associated with one of the four quadrants of its parent's square, the ith child being associated with the ith quadrant of its parent's square. No parent node may have all its descendant leaves the same color. Since a parent and its descendants represent the same region of the picture, if all the descendants have the same color, the picture can be more compactly represented by coloring the parent, and removing all the children.

The superposition algorithm. The superposition algorithm has as input the tree encodings of two pictures. The output is the tree for the picture which is the superposition of one picture over the other. Input specifies which tree represents the occluding picture, and also designates a color to be considered transparent.

The superposition theorem. Superposition of pictures encoded as quad trees can be performed in time linear in the number of input nodes.

Finitely many quad tree pictures may be superposed at once, rather than by sequential superposition of pairs, and the total time is still linear in the total number of input nodes. ("**" Denotes exponentiation.)

Polygons. A polygon is a list of coplanar points, called its vertices, listed as pairs of nonnegative coordinates. Polygons are simple; that is, their edges do not intersect.

Pictures of polygons. A picture $P = (C, A)$ is a picture of polygon G if the following conditions hold. C is a finite set of colors, and A is a $2^{**}q \times 2^{**}q$ square array of square pixels, each associated with an element of C. Coordinates of G are between 0 and $2^{**}q$, inclusive. $A(1, 1)$ represents a pixel with corners (0, 0), (0, 1), (1, 0), and (1, 1); $A(2^{**}q, 2^{**}q)$ is a pixel with corners $(2^{**}q, 2^{**}q)$, $(2^{**}q - 1, 2^{**}q - 1)$, $(2^{**}q - 1, 2^{**}q)$, and $(2^{**}q, 2^{**}q - 1)$. Other pixels in A are mapped into the coordinate system in the natural way. All pixels inside

the polygon have one color, "interior." Pixels outside the polygon are colored "exterior." Pixels intersecting the polygon are colored interior.

Quad trees for polygons. T is a quad tree for polygon G if T is a quad tree for picture P, and P is a picture of polygon G.

In a quad tree for a polygon, nodes are partitioned into three exclusive classes: interior, exterior, and boundary nodes.

Boundary nodes. Nodes intersecting the boundary of the polygon are boundary nodes. It may be that only the boundary of the node intersects the polygon.

Interior nodes. Nodes containing only points interior to the polygon are interior nodes.

Exterior nodes. Nodes wholly outside the polygon are exterior nodes.

Complexity parameters. v is the number of vertices in a polygon. q is the base-2 logarithm of the side of a picture. q limits the depth of the quad tree. If we assume that the number of bits in the coordinates of polygonal vertices is fixed, it is not meaningful to let q increase indefinitely. We will assume that if the number of bits in polygonal coordinates is fixed, so is picture resolution, and so is q, but our analyses will make explicit the effect of varying q. p is the smallest integer no less than the perimeter of a polygon. p is measured in pixel-widths, assuming there is a picture associated with the polygon.

A pair of nodes are *neighboring*, or *bordering*, if their quadrants do not overlap areas, but intersect on an entire side of one of them, not only on a corner. (A quadrant includes the points on its boundary.)

A quad tree may be built in such a way that each side y of each leaf x is connected by a two-way pointer to a circularly linked list of two-way pointers to the neighbors (of the leaf x on the side y), in order along the side y, counterclockwise around x. Call a tree so structured, a *netted quad tree.* This data structure makes it possible to search the neighbors on a side in time linear in their number.

The outline-and-color algorithm. An algorithm which takes as input a polygon and produces as output a netted quad tree for the polygon is an outline-and-color algorithm.

THEOREM. *The Outline-and-Color Algorithm may be done in time* $O(v + p + q)$.

3. THE QUAD TREE TRANSFORMATION ALGORITHM

Input. Input consists of a netted quad tree for a picture; the designation of some color to be called "background"; and T, some transformation which is a linear transform together with a translation. $T(x, y) = (ax + by + c, dx + ey + f)$, where a, b, c, d, e, and f are constants. We will assume that T does not map all points into the same line or point. T defines a transformation of a picture which is the same picture with its coordinates transformed by T. Input specifies the desired resolution of the output tree, which can be different from the resolution of the input tree.

Output. Output consists of a netted quad tree for the picture which is the transformation of the input picture by T; when an output area contains no image of any input area, it is colored background. The following is a precise definition of the output netted quad tree. For every pixel represented by a leaf

B in the output tree, some leaf colored the same as B in the input tree must have one of its points transformed into the pixel. An output leaf which contains no image of any point of any input leaf is colored background.

The edge of visibility is defined as the inverse image of the output picture's root's sides, that is, of the output picture boundary. *Visible* is defined to be true of those things in the input tree containing some points either on the edge of visibility, or interior to the region defined by the edge of visibility. We shall mean the same thing by the *edge* of a leaf as by the *side* of a leaf.

Start by stacking all visible edges of leaves, with two-way pointers between them and the edges in the input quad tree. Note that an edge may be stacked twice and associated with two different leaves. For our purposes an edge shared by two leaves is two edges. If the stack is empty, output is a tree with one node. If the input is a one-node tree colored background, output is the same as input.

Pop the stack until a leaf-side is found with a visible portion which lies between two different colors, or which lies on the edge of the picture and has a non-background-colored leaf. This may require examining all the neighbors of any particular popped side, using the netted data structure.

Follow the boundary of the largest contiguous region of homogeneous color containing this leaf. The outline of this region is a polygon formed of edges of leaves. The edge of visibility, the inverse image of the output picture's root's sides, is an intercolor boundary.

Here is how we trace the polygon. Begin by proceeding along a visible portion of a side which has been popped from the stack, choosing the direction of the path taken so that the interior of the leaf for the side is on the right of the path. The color of the leaf will be called the right-side color. The right-side color will lie to the right of all the edges of the polygon. The edges of the polygon will be sides of leaves and intersections of the edge of visibility with leaves having the right-side color. When the following of the boundary of the polygon intersects a corner or a side of a leaf or the edge of visibility, there may be a choice of edges with which to continue the polygon. The point of intersection may be an endpoint of some sides of leaves. Should one of these sides be followed in continuing the polygon? The edge of visibility may lead out of the point in two directions. Should one of these possible continuations be chosen? First, eliminate from consideration those continuations which would enclose another color in the polygon we are forming by having a color other than the right-side color on the right side at the point of continuation. Of the remaining possible continuations, the next edge is chosen to be the leftmost (in an angular sense) continuation. Since sides may be duplicated if they are common to two leaves, if the continuation taken is a portion of a side, then it should be the side of the leaf whose visible points are interior to the polygon. That is, the leaf should be of the right-side color. If the continuation taken is a portion of a side of a leaf, remove all sides of the leaf from the stack.

As this polygon we shall call *primary* is outlined, each edge is transformed by T and the transformed polygon is processed by the Outline-and-Color Algorithm to form a quad tree for its picture. The interior of the primary polygon is *colored* in the input tree. The coloring is not a true coloring, but only temporary for the

purpose of finding the outside edges of the largest contiguous regions interior to the primary polygon which do not have the primary polygon's right-side color on their boundaries. These are called *interior* polygons and are separated from the boundary of the primary polygon by the right-side color of the primary polygon.

Coloring is defined to be the recursive spreading of color from boundary leaves to neighboring leaves, their neighbors, and so on. The number of times leaves propagate color to smaller neighbors is limited to a constant times the number of leaves which can be neighbors to leaves larger than themselves. The number of propagations of color to non-smaller leaves is also linear in the number of leaves, so the total coloring cost is linear in the number of leaves. The coloring process extends only to the edges of the interior polygons, does not extend inside the interior polygons, and is erased after the interior polygons are discovered.

The interior polygons are outlined and transformed as above to create a quad tree for each. Now the interior polygons are cut out of the primary polygon by superposing them as holes in the quad tree for the primary polygon. In other words, the quad tree for the primary polygon has its background color placed where these interior polygons, and any they contain, are located.

Repeat the process of popping the stack, finding an intercolor edge, and building quad trees with holes for polygons found until the stack is empty and all colored regions are thus represented by quad trees with holes.

Superpose the quad trees for the transformed polygons with holes. Let the color which is in the holes and in the background be transparent in all but the bottom tree, where the background color should be the background color desired in the output of the transformation algorithm. Superposing the quad trees for the transformed polygons gives the output of the quad tree transformation since holes are treated as transparency, as are the backgrounds. The bottom tree may be selected arbitrarily.

No two different non-background colors exist in the same picture locations in different trees being superposed. Possible exceptions to this rule are pixels in different output trees which contain transform images of points on the boundary between two uniformly colored regions in the input tree. Different orderings of superposition may result in different colorings of these pixels. With these exceptions, the holes let all significant information show through, and the order of the trees in superposition does not matter.

4. COMPLEXITY ANALYSIS OF THE TRANSFORMATION ALGORITHM

In essence, the above algorithm uses the Outline-and-Color Algorithm for polygons with a total perimeter which we shall designate p. That is, p is the total perimeter of polygons surrounding the uniformly colored, non-background, visible regions of the input picture, including that portion of their perimeter lying along the picture boundary.

Let m be the number of polygons in the input tree.

Let n be the number of nodes in the input tree.

Let s be the scaling factor: the maximum distance between $T(x, y)$ and $T(w, z)$ when (x, y) and (w, z) are chosen one unit apart.

THEOREM. *Time and space of* $O(n + sp + mq)$ *are sufficient for the quad tree transformation algorithm.*

Proof. The total number of vertices for both holes and primary polygons is $O(n)$. The total perimeter of primary polygons is p, and the total perimeter of holes is no more than p. The total number of primary polygons is m, and the total number of holes is no more than m. Outlining and coloring a polygon into a quad tree is $O(v + p + q)$. Therefore, constructing all the trees requires time

$$O(\sum (v_i + p_i + q_i)),$$

with the v_i's, p_i's, and q_i's from all the trees. Since $\sum v_i$ is $O(n)$, $\sum p_i$ is $O(sp)$, and $\sum q_i$ is $O(mq)$, the total requirement is $O(n + sp + mq)$.

The superposition step is linear in the number of nodes in all the trees to be superposed, so this step does not change the asymptotic complexity.

Checking sides popped from the stack for whether they have a neighbor of different color may require checking all the neighboring nodes on the side. The number of sides is $O(n)$. Therefore, checking the sides which have non-smaller neighbors is $O(n)$. Checking the sides which have smaller neighbors is no worse than reading all their smaller neighbors which is no worse than reading all the leaves in the tree times the constant representing how many sides to which a leaf can be a smaller neighbor: $O(n)$.

In the application of the algorithm for coloring the input trees, the boundary leaves of the primary polygon are colored as the polygon is outlined. Color spreads recursively to all neighboring, interior, uncolored leaves which do not belong to interior polygons. The primary polygons, minus their interior polygon holes, are disjoint sets of nodes. Therefore, coloring them is $O(n)$. Q.E.D.

COROLLARY. *Since* m *is no larger than* n, *time and space of* $O(nq + sp)$ *are sufficient, or* $O(n + p)$ *if the resolution* q *and the scale factor* s *of* T *are fixed.*

REFERENCES

1. A. Klinger and C. R. Dyer, Experiments on picture representation using regular decomposition, *Computer Graphics and Image Processing*, **5**, 1976, 68–105.
2. T. Pavlidis, The use of algorithms of piecewise approximations for picture processing applications, *Assoc. Comput. Mach. Trans. Math. Software* **2**, No. 4, 1976, 305–321.
3. S. L. Horowitz and T. Pavlidis, Picture segmentation by a tree traversal algorithm, *J. Assoc. Comput. Mach.* **23**, No. 2, 1976, 368–388.
4. S. L. Tanimoto, Pictorial feature distortion in a pyramid, *Computer Graphics and Image Processing*, **5**, 1976, 333–352.
5. S. L. Tanimoto, A pyramid model for binary picture complexity, in *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing (77CH1208-9C), June 6–8, 1977, Rensselaer Polytechnic Institute,* pp. 25–28.
6. G. M. Hunter, Computer animation survey, *Computers and Graphics* **2**, 1977, 225–229.
7. N. Negroponte, Raster scan approaches to computer graphics, *Computers and Graphics* **2**, 1977, 179–193.
8. G. M. Hunter, *Full-Color Television, From the Computer, Refreshed by Run-Length Codes in Main Memory,* Technical Report No. 182, Computer Science Laboratory, Princeton University, April 21, 1975.

9. G. M. Hunter, *Efficient Computation and Data Structures for Graphics*, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Princeton University, June 1978.
10. G. M. Hunter and K. Steiglitz, Operations on images using quad trees, *IEEE Trans. Pattern Recognition and Machine Intelligence*, **PAM-1**, No. 2, 1979, 145–153.
11. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, 1973.
12. G. M. Hunter and K. Steiglitz, The construction and use of quad-tree-encoded two-dimensional arrays, 1978 IEEE International Symposium on Circuits and Systems, New York, May 17–19, 1978.