Pattern Recognition Letters 3 (1985) 211-214
North-Holland

May 1985

A new data structure for efficient storing of images

David S. SCOTT
Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA

S. Sitharama IYENGAR
Department of Computer Science, Louisiana State University, Baton Rouge, USA

Received 11 September 1984
Revised 22 February 1985

Abstract: The space efficiency of recent results on data structures for quadtrees [2,3,4] may be improved by defining a new data structure called Translation Invariant Data Structures (TID). This paper briefly describes the results of this new data structure for storing images.

Key words: Translation invariant data structures, quadtrees and efficient algorithm.

1. Introduction

Quadtrees are often used to store black and white picture information. They usually lead to a significant savings in storage requirements over the original array of pixels and they facilitate computer processing and interpretation of the pictures. A variety of techniques have been suggested for improving quadtrees including compact quadtrees, hybrid quadtrees, forest quadtrees, linear quadtrees, and QMAT's (quadtree medial axis transform). The major purpose of these improvements is to reduce the storage required without greatly increasing the processing costs. All of these methods start with the quadtree and then eliminate some or all of the white and gray nodes and corresponding pointers. In general the number of black nodes in a quadtree is about the same as the number of white and gray nodes and so the savings achieved usually amounts to a factor of 2 or 3 or so. For overviews of related research on data structures for quadtrees see [2, 3, 4, 5].

This note briefly investigates the following:

1. Sensitivity of the quadtree structure to the placement of the origin (that is, small translations of the black image inside the borders of the frame can make very large changes in the structure of the quadtree).

2. Definition of a translation invariant data structure (which we name TID) for storing and processing images.

We also describe briefly some other manipulation algorithms on TID (such as rotation and union operations).

2. Sensitivity of placement

All the methods of representing images described in Section 1 form sensitivity to the placement of the origin. Two images which are translations to each other can give rise to very different looking structures.

Remark 1. If a $2^{n-1} \times 2^{n-1}$ black square is imbedded in a $2^n \times 2^n$ white frame then the structure

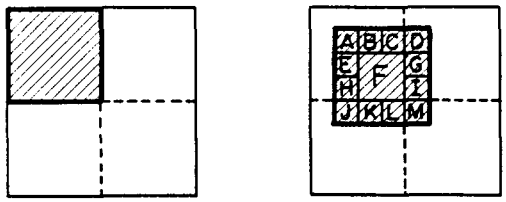

Figure 1. Sample regions.

of the corresponding quadtree depends on the placement of the black square.

The above observation can be illustrated as follows: Consider a region A (black square) in the upper left corner (see Figure 1); then the quadtree has five nodes, one of which is black. If the same image is translated diagonally one pixel (see Figure 2), so that the corner of the black square is in the (2,2) position, then it can be shown that the number of black nodes in the quadtree is equal to $(2^{n+1}+2^n-3^{n-2})$.

Therefore we can state the above observations in the form of a theorem.

Theorem 1. *The quadtree for the image translated diagonally one pixel has*

$$\begin{aligned}\text{GRAY NODES} &= 2^{n+1}-3,\\ \text{WHITE NODES} &= 2^{n+1}+2^n+2^{n-6},\\ \text{BLACK NODES} &= 2^{n+1}+2^n-3^{n-2}.\end{aligned}$$

Proof. To prove the formulas it is necessary to derive recurrence relation among subtrees. A detailed proof for Theorem 1 is given in [1].

2.1. Performance of other data structures

Linear quadtrees, compact quadtrees and forest of quadtrees all are inevitably forced to store $2^{n+1}+2^n-3^{n-2}$ BLACK nodes plus perhaps some others. The only other structure capable of eliminating black nodes is Samet's quadtree medial axis transform. Furthermore, the QMAT for the region described in Figure 2 has an interesting structure. Most of the image is covered by only four nodes but a sequence of decreasing sized nodes is needed to cover the rest. Mathematically, we can state the following theorem.

Theorem 2. *The QMAT structure for the image translated diagonally one pixel has*

$$\begin{aligned}\text{GRAY NODES} &= 2^{n-3},\\ \text{WHITE NODES} &= 4^{n-6},\\ \text{BLACK NODES} &= 2^{n-2}.\end{aligned}$$

Proof. See [1].

Thus we can conclude based on the above observation that sensitivity to the placement of the origin is a serious problem when translating images.

3. New data structure

Scott and Iyengar [1] introduced a new structure for storing the black pixels of a region. We refer to this structure as TID for reasons described in Section 2. Further TID has been shown to save more than two-thirds of the memory locations used by quadtrees. For more one properties and construction of TID see [1]. We now briefly discuss the storing of and searching for a TID structure.

Each maximal square in a TID is characterized by three coordinates (i,j,s). The first two coordinates specify the location and the third specifies the size of the region. Unfortunately, unlike linear

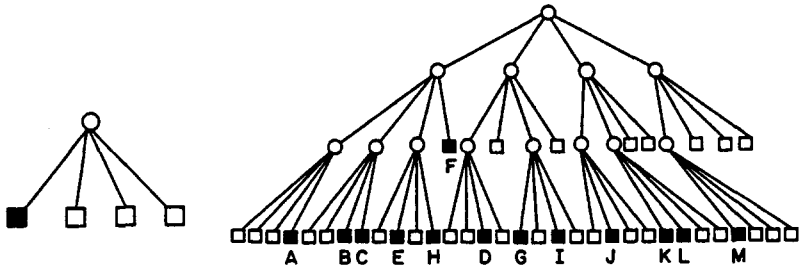

Figure 2. Quadtrees for Case A and Case B shown in Figure 1.

quadtrees, such triples of numbers do not have a natural linear ordering. The best that can be done is to choose some priority order for the coordinator and then order them lexicographically. We will assume that i and j are sorted in increasing order (for reasons described in detail in [1]). We will assume that 's' is sorted in decreasing order. By symmetry we may assume that i is ordered before j. Thus the question is which of the three possible orderings

(a) (i, j, s),

(b) (i, s, j),

(c) (s, i, j)

is the best? There is no definitive answer to this question. It depends on whether storage or access is of primary concern.

Storage is conserved when the primary subdivision are large since the value of the primary variable will only be stored once. On this basis, Plan (a) can be eliminated since there can be at most one maximal square with corner (i, j) and so no savings can be obtained at the second division compared to Plan (b) or Plan (c). Plan (b) may be better since there may be several squares of the same size with the same i value.

For storage purposes the competition is between Plan (b) and Plan (c). For most images Plan (c) is superior since there will be many small squares around and so the subsets of size 1, 2, and 3 should be quite large allowing for a much greater space savings than can be obtained by Plan (b).

For searching it is important to shorten the length of the search whenever possible by skipping to the beginning of the next primary or secondary classification. The following discussion assumes that the purpose of the search is to decide whether pixel (k, m) is black, i.e. whether (k, m) is contained in some square in the TID.

On this basis Plan (c) can be ruled out since it is impossible to determine anything given just s. On the other hand, with Plan (a) or Plan (b) we can stop the search as soon as $k < i$. There remains the question whether Plan (a) or Plan (b) allows more skipping of squares. For fixed i, Plan (a) skips all squares with $j > m$. Plan (b) skips all squares with $s < k - i$. Which set is larger? This obviously depends on the image. For the purpose of analysis we make the following assumptions:

(1) Every value of j is equally likely.

(2) All possible values of s (i.e. 1 to numrow-$i+1$) are equally likely.

Assumption 1 is true for 'random' squares only if they have size 1. For larger squares the distribution is skewed, favoring smaller values of j. Assumption 2 is even less reasonable. In most images there will be many more small squares than large squares.

For fixed i, Plan (a) skips all squares (l, j, s) with $l = i$ and $j > m$. By Assumption 1, this will be about half the squares with $l = i$ for a random (k, m). Plan (b) will skip all squares (l, j, s) with $l = i$ and $l + s < k$. By Assumption 2 this will be about half the squares with $l = i$. Thus by analysis the two plans are about the same. However, both biases in the assumptions favor Plan (b), particularly the second one. Thus Plan (b) is better.

4. Translation, rotation, and union of TID's

A TID is made up of maximal squares. Each square is represented as a triple (i, j, s) where (i, j) is the northwest corner of the square and s is the length of the side. Thus a TID is just a list of such triples. Translations and rotations applied to the image are just simple functions of these triples. To translate a triple (i, j, s) by I units right and J units up yields

$$T_{IJ}(i, j, s) = (i + I, j + J, s).$$

Rotation by $\pi/2$ is only slightly more complicated due to the fact that the NW corner of a square changes upon rotation. The $\pi/2$ rotation around the origin is

$$R(i, j, s) = (-j, i + s, s).$$

Rotations around other coordinates can be obtained by composing R with the appropriate translations.

Superficially unions of TID's are straightforward - simply take the union of the two lists. The resulting list will be the TID of the combined images whenever the images do not overlap. Two problems can occur when the two images overlap. A square in one TID may be contained in a square from the other TID and thus be redundant, or

several squares from both TID's could be combined to form a larger square. The first problem can be easily checked. The second problem is harder - there appears to be no better solution than reforming the array of pixels and recomputing the TID. On the other hand, the combined lists are unlikely to contain many such larger squares (if any) and thus should be an adequate representation of the image.

5. Concluding remarks

The major advantage of TID's over other methods of storing images (regular quadtrees and efficient storage of quadtrees) is that TID's are translation invariant. This is particularly important if several images are being combined into one composite. For example in *CACM*, March 1984, p. 248-249 Markkun Tamminen quotes 5198 black leaves for the quadtree encoding the circle inscribed in a $2^{10} \times 2^{10}$ square. The corresponding TID has 601 black squares, an 88% reduction. The TID structure could be easily extended to deal with objects in three dimensions and is presently under investigation [6].

A software package called 'TIDSOFT' written in PASCAL for VAX 11/780 system, is in preparation at the University of Texas at Austin and LSU-Baton Rouge and will be available on request.

References

[1] Scott, D.S. and S.S. Iyengar. TID - A translation invariant data structure for storing images. Technical Report 84, University of Texas at Austin. Submitted for publication, May 1984.

[2] Samet, H. (1983). A quadtree medial axis transform. *Comm. ACM* 26 (9), 680-692.

[3] Tamminen, M. (1984). Comment on quad- and octtrees. *Comm. ACM* 27 (3), 248-249.

[4] Jones, L.P. and S.S. Iyengar (1984). Space and time efficient virtual quadtrees. *IEEE Trans. Patt. Anal. Mach. Intell.* 6 (2), 244-247.

[5] Gargantini, I. (1982). An efficient way to represent quadtrees. *Comm. ACM* 25 (12), 905-910.

[6] Scott, D.S. and S.S. Iyengar. TID-3D-application of TID structure to three-dimensional objects. In preparation.