Pattern Recognition Letters 7 (1988) 45–49
North-Holland

January 1988

A formula for computing the number of quadtree node fragments created by a shift

Clifford A. SHAFFER
Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

Received 1 June 1987

Abstract: A formula is provided to compute the number of new blocks resulting from the decomposition induced by a shift of a single quadtree node of arbitrary size by an arbitrary amount. A precise calculation is also presented for the average number of BLACK nodes required to represent a square of width 2^m in a region quadtree.

Key words: Quadtrees, hierarchical data structures, image translation.

1. Introduction

The quadtree (Samet, 1984a) is a hierarchical data structure which has proven useful in many domains, including computer graphics and digital cartography. In its most general formulation, the quadtree decomposes a data set that does not meet some criterion into four quadrants; each quadrant is then examined in turn to see if it meets the criterion, with further subdivisions occurring as necessary. Figure 1 illustrates the *region quadtree*, which is used to represent region data. Here the criterion is simply to split a block into four equal quadrants whenever it is not homogeneous.

This note investigates the effects of a shift operation on a square represented by a single node of the region quadtree. When an image is shifted (i.e., moved some number of pixels parallel to the x and/or y axes), the quadtree blocks may be subdivided as illustrated by Figure 2. These sub-blocks will be referred to as *fragments*. For quadtree representations of 'typical' images, a shift may change the positions and sizes of the blocks; however, the total number of nodes normally remains about the same (for an empirical study, see (Samet, 1984b)). This is due to the fact that the fragments of a block decomposed as in Figure 2b will often merge with the fragments created by a neighboring block of the same color. Nonetheless, it is worthwhile to examine the number of fragments produced by shifting an individual block to understand the amount of work that goes on behind the scenes during shifting operations.

This note presents a formula to determine the number of fragments created from a single region quadtree block of width 2^m (before merging with neighboring blocks of the same color) if it is shifted by specified distances parallel to the x and y axes. Dyer (1982) has presented an analysis for the best, worst, and average number of quadtree nodes required to represent a square of width 2^m in an image of width 2^n. However, his calculation for the average case cost is only a close estimate. From our formula to calculate the number of fragments produced by a shift operation, we can derive an exact calculation for the average number of BLACK nodes required to store the square.

2. The shift formula

We begin by defining a function $F(X, Y, W)$ that expresses the number of fragments produced when a block of width W is shifted X units parallel to the

0167-8655/88/$3.50 © 1988, Elsevier Science Publishers B.V. (North-Holland)

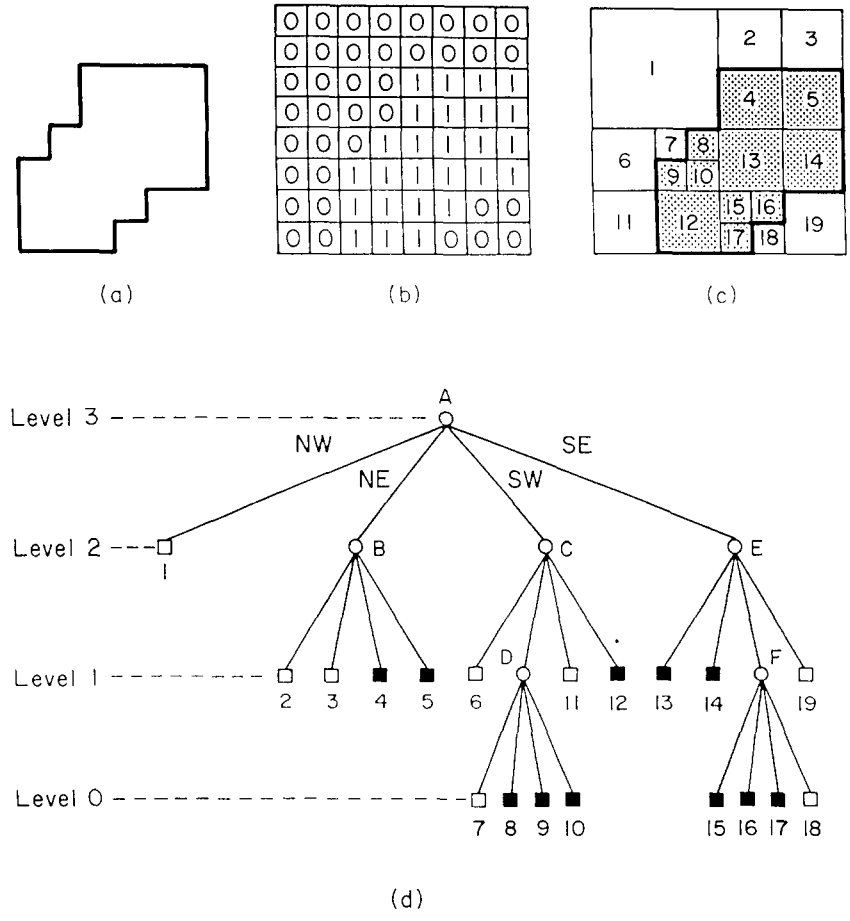

Figure 1. A region, its binary array, its maximal blocks, and the corresponding quadtree. (a) Region. (b) Binary array. (c) Block decomposition of the region in (a). Blocks in the region are shaded. (d) Quadtree representation of the blocks in (c).

x axis, and Y units parallel to the y axis, for X, Y, and W positive integers (X and Y are absolute distances). Note that the number of resulting fragments is the same whether the shift is to the left or the right. An important property of the quadtree is that all shifts by any multiple of the largest power of two that is a factor of the shift distance will result in the same number of fragments. In other words, shifts of 1, 3, 5, ... pixels, whose largest power of 2 factor is 2^0, will each result in the same number of fragments. In the same way, shifts of 2, 6, 10, ... pixels, whose largest power of 2 factor is 2^1, each result in the same number of fragments. See (Li, 1982) for a quadtree node minimization algorithm that takes advantage of this property.

We can state this property formally by

$$F(X, Y, W) = F(C_1 \cdot 2^x, C_2 \cdot 2^y, 2^i) = F(2^x, 2^y, 2^i)$$

where x, y, i, C_1, and C_2 are positive integers, with x and y being as large as possible. Note that, since W is the width of a quadtree block, $W = 2^i$ for some integer i. For convenience, we will define x (or y) to be i when the block is shifted 0 pixels parallel to the x (or y) axis, since this is the same as shifting the block by a multiple of 2^i pixels. We now define a new function f as

$$f(x, y, i) = F(2^x, 2^y, 2^i).$$

Thus we have

$$F(X, Y, W) = f(x, y, i)$$

where x, y, and i are defined as above. The values x and y in function f can be viewed as the number of consecutive zeros on the least significant (rightmost) end of the binary representations for X and Y.

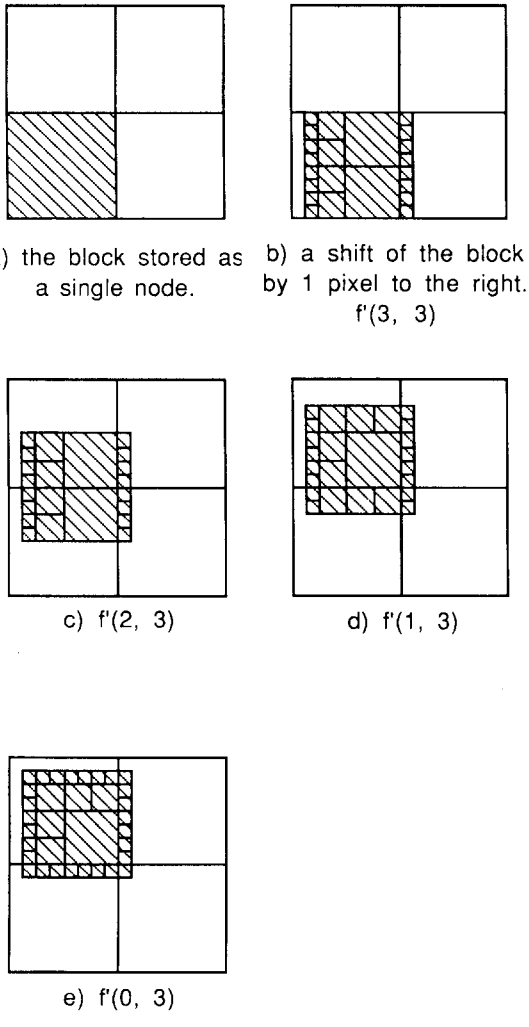

Figure 2. Decompositions for a block of size 8 by 8 stored in a quadtree of size 16 by 16.

Function f has the following three properties, which can be easily verified by inspection of shifted blocks.

(1) A block shifted by (X, Y) pixels will be represented by the same number of fragments as a block shifted by (Y, X) pixels, i.e., $f(x, y, i) = f(y, x, i)$.

(2) If the size of the block is doubled, and the shift amounts along each axis are also doubled, then the number of resulting fragments remains the same. Generalized, this means that $f(x, y, i) = f(x + m, y + m, i + m)$.

(3) If a block of width 2^i is shifted horizontally by a distance of $c \cdot 2^i$ for any integer c, that block will not be decomposed. In other words, when $x \geq i$, $f(x, y, i) = f(i, y, i)$.

Using these equalities, function f can always be rewritten such that

$$f(x, y, i) = f(0, s, d)$$

where $s = \min(i, \max(x, y)) - \min(x, y, i)$, and $d = i - \min(x, y, i)$. We will now define a two-parameter function f' as follows:

$$f'(s, d) = f(0, s, d).$$

f' formalizes the effects of (a) reducing x and y to be at most i, (b) scaling the shift by reducing x, y, and i by $\min(x, y, i)$, and (c) realizing that either x or y will now be 0. Thus, $f'(s, d)$ represents a shift by 2^0 pixels along one axis and by 2^s pixels along the other axis on a block scaled to width 2^d.

We now derive the closed form solution for function $f'(s, d)$ as follows. The shift producing the greatest number of fragments occurs when $s = 0$ (i.e., a shift by an odd number of pixels along each axis). From (Dyer, 1982) we see that the number of fragments in this case is $3(2^{d+1} - d) - 5$. Now, look at Figure 2b which shows an example for the minimum number of fragments as computed by f' for $d = 3$ (i.e., when the block is shifted by an odd number of pixels parallel to one axis, and by a multiple of the block's width parallel to the other axis). This best case occurs when $s = d$. For each reduction of s by 1, we see that a row of blocks of size 2^{s+1} is decomposed. For example, when shifting from Figures 2b to 2c, the BLACK block can be viewed as having its bottom 2^2 rows removed, with these rows reappearing at the top of the block. Note that $f'(d - 1, d)$, illustrated by Figure 2c, is a special case in that it always requires the same number of fragments as $f'(d, d)$, since there can never be a fragment of width 2^d. Since the smallest fragment in Figure 2b is 4×4 pixels, no decomposition results from this shift. When shifting from Figure 2c to 2d, the bottom 2^1 rows of the block are moved to the top of the block; the bottom 4×4 fragment of Figure 2c is decomposed into 2×2 sized blocks in Figure 2d. When shifting from Figures 2d to 2e, the bottom 2^0 rows of the block are moved to the top; the bottom row of 2×2 fragments in Figure 2d is again split into quarters in Figure 2e. Thus, for $s = j$, a row of $2^{d-j} - 1$ fragments is decomposed into four fragments each for a total increase of $g(j) = 3(2^{d-j} - 1)$ fragments as compared to $s = j - 1$.

To compute the value of $f'(s, d)$, we calculate.

$$\begin{aligned} f'(s,d) &= \text{Worstcase} - \sum_{j=1}^{s} g(j) \\ &= 3(2^{d+1} - d) - 5 - \sum_{j=1}^{s} 3(2^{d-j} - 1) \\ &= 3(2^{d+1} - d) - 5 - 3(2^{d} - 2^{d-s} - s) \\ &= 3(2^{d} + 2^{d-s} - d + s) - 5. \end{aligned}$$

3. Computing the average number of blocks needed to represent a square

Given a square of width 2^m in an image represented by a quadtree, we will now compute the average number of blocks required for random placement of the square. Dyer (1982) attempted to calculate this average; however, his approach may overestimate by as much as $8m$ blocks. Thus, his final result is the approximation $O(2^{m+2} - m)$. Given the formula derived in the previous section for the number of fragments required to represent the block at a given position, we can generate the average number of fragments required by simply iterating this equation over all possible positions of the block.

Since we wish only to compute the number of BLACK quadtree nodes required to represent the block, we need only consider shifts such that the upper-left corner of the 2^m block falls on some pixel of the original 2^m block's space. A shift of X pixels for $X > 2^m$ will be identical to a shift of X MODULO 2^m. Thus, to determine the average number of blocks, we need simply evaluate f over all shifts within a square of width 2^m. We will call this function AVG (m) for a square of width 2^m.

Consider that for the set of all possible shifts, half of the vertical shifts will be by an odd number of pixels, and half by an even number of pixels. Likewise, for the horizontal shifts, half will be by an odd number of pixels and half by an even number of pixels. We see that

$$\text{AVG}(m) = \tfrac{1}{4}|OO| + \tfrac{1}{4}|OE| + \tfrac{1}{4}|EO| + \tfrac{1}{4}|EE|$$

where $|OO|$, $|OE|$, $|EO|$, and $|EE|$ stand for the average number of fragments required by an odd vertical and odd horizontal shift, odd vertical and even horizontal shift, even vertical and odd horizontal shift, and even vertical and even horizontal shift, respectively.

Recall from the previous section that, given a specified shift parallel to one axis, a shift by any odd number of pixels parallel to the other axis will always yield the same number of fragments. Thus, if we calculate the average number of fragments over all possible shifts along an entire odd row (or column), this average will be the same regardless of which odd row (or column) we are measuring. Furthermore, this value is also the average number of fragments resulting from all cases involving an odd y shift. For each odd row, we can derive the equation to describe the average number of fragments as follows. One half of the horizontal shifts will be odd (i.e., $x = 0$). Since the value of y is also 0, the fragment count for these cases will be $f(0, m)$. One quarter of the horizontal shifts will have an x value of 1, counting $f(1, m)$ fragments, and so on. Finally, there is one pixel with horizontal shift of 0, for a fragment count of $f(m, m)$. The row equation is then

$$\text{ODDROW} = \sum_{i=0}^{m-1}\left(\frac{1}{2^{i+1}} \cdot f(i,m)\right) + \frac{1}{2^m} \cdot f(m,n).$$

The average number of fragments for all shifts involving an odd y shift can also be written as $\frac{1}{2}|OE| + \frac{1}{2}|OO|$. In a like manner, one half of the column shifts will also be odd. An identical argument yields $\frac{1}{2}|EO| + \frac{1}{2}|OO|$ as the contribution to the weighted average for the odd columns. Note that the contribution for the cases in which the shifts are both odd has been counted twice. This occurs one quarter of the time, and should be subtracted from the weighted sum of the contributions for odd rows and odd columns. We still need an expression for the case where both shifts are by an even distance. However, by property (2) of the previous section, this case is the same as calculating the average number of fragments required to represent a square of width 2^{m-1}. Thus, we get the recurrence

$$\begin{aligned} \text{AVG}(m) &= \tfrac{1}{2} \cdot \text{ODDROW} + \tfrac{1}{2} \cdot \text{ODDCOL} \\ &\quad - \tfrac{1}{4}|OO| + \tfrac{1}{4}\text{AVG}(m-1) \\ &= 2 \cdot \tfrac{1}{2}\left(\sum_{i=0}^{m-1}\left(\frac{1}{2^{i+1}} \cdot f(i,m)\right) + \frac{1}{2^m} \cdot f(m,m)\right) \\ &\quad - \tfrac{1}{4} f(0,m) + \tfrac{1}{4}\text{AVG}(m-1). \end{aligned}$$

The factor 2 indicates that the expression is used for both rows and columns; the factor $\frac{1}{2}$ indicates that half of the rows (columns) use this equation; the next factor is the row equation; the term $\frac{1}{4}f'(0,m)$ corresponds to the contribution to the average for the cases in which both shifts are odd (which is subtracted because it was counted twice in the previous term); and finally, the last term takes care of the cases in which both shifts are even. Changed to summations, the equation for the average number of BLACK blocks is

$$\begin{aligned}
\text{AVG}(m) &= \sum_{d=1}^{m} \frac{1}{4^{m-d}} \left(\sum_{i=0}^{d-1} \left[\frac{1}{2^{i+1}} \cdot f'(i,d) \right] \right. \\
&\qquad + \frac{1}{2^d} \cdot f'(d,d) \\
&\qquad \left. - \tfrac{1}{4} \cdot f'(0,d) \right) + \frac{1}{4^m} \\
&= \sum_{d=1}^{m} \frac{1}{4^{m-d}} \left((5 \cdot 2^d - 3d - 5) \right. \\
&\qquad + \frac{1}{2^d} \cdot f'(d,d) \\
&\qquad \left. - \tfrac{1}{4} \cdot f'(0,d) \right) + \frac{1}{4^m} \\
&= \sum_{d=1}^{m} \frac{1}{4^{m-d}} \left(\frac{7 \cdot 2^d}{2} - \frac{2}{2^d} - \frac{9d}{4} - \frac{3}{4} \right) + \frac{1}{4^m} \\
&= 2^{m+2} - 3m - \frac{4}{2^m} + \frac{1}{4^m}.
\end{aligned}$$

4. Conclusions

We have derived a formula that yields the number of blocks required to store a node of size 2^m when shifted by (X, Y) pixels. We have also made precise the average case expectancy for the number of blocks required to store a square of size 2^m. Our calculation of this average agrees with the value of $O(2^{m+2} - m)$ calculated in (Dyer, 1982).

Acknowledgements

Many thanks to Todd Kushner for helping me grind these equations through Macsyma. I have also benefited greatly from discussions with Randal Nelson and Hanan Samet. The support of the National Science Foundation under Grant DCR-86-05557 is gratefully acknowledged.

References

Dyer, C.R. (1982). The space efficiency of quadtrees. *Computer Graphics and Image Processing* 19(4), 335–348.

Li, M., W.I. Grosky, and R. Jain (1982). Normalized quadtrees with respect to translations. *Computer Graphics and Image Processing* 20(1), 72–81.

Samet, H. (1984a). The quadtree and related hierarchical data structures. *ACM Computing Surveys* 16(2), 187–260.

Samet, H., A. Rosenfeld, C.A. Shaffer, R.C. Nelson, and Y.G. Huang (1984b). Application of hierarchical data structures to geographical information systems Phase III, Computer Science TR-1457, University of Maryland, College Park, MD, November.