MTA SZTAKI Közlemények 32/1985 105-121

# A DATA STRUCTURE FOR QUADTREE CODES AND APPLICATION

*PHAM NGOC KHOI*

Institute of Technical Cybernetics
SAV - Bratislava
Czechoslovakia

## ABSTRACT

This paper describes a data structure for Quadtree code representation and the operators defined on it. Algorithms are presented for finding adjancent blocks, calculating geometric properties of region: area, perimeter and centroid. The efficiency of quadtree representation is evaluated in relation to Run length code and Chain code representation.

## 1. INTRODUCTION

For image data compression, a lot of representation algorithms is based on the various codings for a region: Run length codes, Chain codes, Quadtrees... [1,11]. Especially the tree representation which offers a number of advantages has attracted much attention in recent years [2-14]. A collection of algorithms has been studied for converting between quadtrees and other representations and measuring geometric features of regions represented by quadtrees [4-7, 13-14]. In this paper, we shall be concerned with a data structure for quadtrees as a tool for calculating region features of image. The emphasis is on algorithmic procedures, which are efficient especially from implementation point of view. The last section deals with the comparison of efficiency of image coding based on Quadtrees, Run length codes and Chain codes.

In the following by a binary image, we always mean a $2^n \times 2^n$ array, each element of which has value 0 or 1, respectively to the color of pixel: black or white. The set of

1's pixels in a connected component of array is called a region.

## 2. DATA STRUCTURE FOR QUADTREES

Quadtree codes are an recent region representation method. It is based on successive subdivision of the array into quadrants until we obtain blocks possible simple pixel that are entierly contained in the region or entierly disjoint from it. Note that if the image is an $2^n$ by $2^n$ array of pixels, then after the k-th subdivision, each quadrant has $2^{n-k}$ by $2^{n-k}$ size. For example the region in Fig. 1a corresponds to raffinement process as shown in Fig. 1b. This process can be represented by a tree of degree 4 or a quadtree in which the entier array is a root node, the four sons of a node are its quadrants and the leaf nodes correspond to 1's or 0's pixel blocks and have their color BLACK or WHITE. The no-leaf nodes correspond to those blocks for which the further subdivision is still continued and have their color GRAY. The quadtree representation for Fig. 1b is shown in Fig. 1c. Note that here the blocks must have standard size and positions. Since the array was assumed to be size of $2^n$ by $2^n$, the tree height is at most n. This region representation method was proposed by Klinger [2].

The quadtree can be defined as a tree whose nodes are either leaves or have four sons. A node can be gray, white or black and is, in general, stored as a record with six fields, five of which are pointers to the NW, NE, SW and SE quadrants and the sixth is a colors identifier. In the following, we propose a data structure advantageous to calculate some region features of image.

Assume that a square block has its four quadrants indexed as follows:

| | |
|---|---|
| *0* | *1* |
| *2* | *3* |

correspondently to quadrants NW, SE, SW, SE. Each quadrant in image is associated to a node in quadtree, which has four sons enumered from left to right in the order 0,1,2,3. We present each node of quadtree a integer pair $(L,K)$, where:

- $L$ is level of node, i.e. distance from the root to the node, $0 \leq L \leq n$
- $K$ is defined by

$$K = \sum_{i=o}^{L-1} n_i \, 4^i \qquad 0 \leq n_i \leq 3$$

where $(n_{L-1}, \ldots n_1, n_o)$ is the path from the root to the node $P$.

We denote

$$K = \overline{n_{L-1} \ldots n_1 n_o}$$

$$P = (L,K)$$

$$Level(P) = L$$

$$Code\ (P) = K \quad .$$

We can use the following recurrent formula to determine the pair $(L,K)$ for all nodes of a quadtree:

- The root has presentation $(0,0)$
- If a node $P$ is represented by $(L,K)$ then its four sons have the representations

$$(L+1, \quad 4K+i) \qquad 0 \leq i \leq 3$$

This data structure was introduced by L.P. Jones and S. Iyengan [12] and is called virtual quadtree.

On this data structure we define the following operators:

1- "2's borrow" substraction: *Sub2*

$$Sub2\ P = (L',K')$$

where

$$L' = L$$

$$K' = sign\ \overline{n'_{L-1} \dots n'_1 n'_0}$$

$n'_i$ are recursively computed as follows

- $b_0 := 2$
- $n'_i := (n_i + 4 - b_i) \bmod 4$

$$b_{i+1} := \begin{cases} 0 & \text{if } n_i \geq b_i \\ 2 & \text{otherwise} \end{cases} \qquad i := 0, 1, \dots L-1$$

$$sign := \begin{cases} + & \text{if } b_{L-1} = 0 \\ - & \text{if } b_{L-1} = 2 \end{cases}$$

For example: $Sub2(3,\overline{311}) = (3,\overline{133})$

$Sub2(3,\overline{111}) = (3,-\overline{333})$

2- Clockwise rotation: $Rot^+$

We first define operator Rot on the finite set {0,1,2,3}

$$Rot^+(0) = 1$$
$$Rot^+(1) = 3$$
$$Rot^+(2) = 0$$
$$Rot^+(3) = 2\ .$$

Now, $Rot^+$ is naturally extended on the infinite set $\{0,1,2,3\}^*$

$$Rot^+(xa) = Rot^+(x)Rot^+(a)$$

where $x \in \{0,1,2,3\}^*$, $a \in \{0,1,2,3\}$

Once again, $Rot^+$ is extended on the set of quadtree nodes

$$Rot^+(L,K) = (L,Rot^+(K))\ .$$

$Rot^+(P)$ gives the representation of the same node $P$ after $90^o$-clockwise rotation of the image.

3- Counterclockwise rotation: $Rot^-$

Operator $Rot^-$ is defined in a similar manner to $Rot$, but it manipulates on the set {0,1,2,3} as follows

$$Rot^-(0) = 2$$
$$Rot^-(1) = 0$$
$$Rot^-(2) = 3$$
$$Rot^-(3) = 1\ .$$

In the other word, $Rot^- = (Rot^+)^{-1}$.

$Rot^-(P)$ gives the representation of the same node $P$ after $90^o$-counterclockwise rotation of image.

4- Rounding: *Rnd*

$$Rnd(P) = (L', K')$$

where $L' = L - 1$

$$K' = \overline{n_{L-1} \ldots n_2 n_1}$$

*App(P,i)* determines i-th son of $P$ in Quadtree.

## 3, CALCULATING SOME PROPERTIES OF REGION

### 3.1. ADJANCENCY

In connection with the adjancency of blocks, we have the theorem [13]:

Theorem: Given a variable *side* in the set {Northern, Western, Estern, Southern}, which is encoded respectively by {0,1,2,3}. Let $P$ be a quadrant of image. The quadrant $Q$ is determined by

$$Q = Rot^{-\,side}(Sub2(Rot^{+\,side}(P)))$$

Thus if $sign(Code(Sub2(Rot^{+side}(P)))) > 0$ then $Q$ is the quadrant which is adjancent to $P$ in the *side* direction and which has same size as $P$.

Basing on the theorem, we can jave the procedure finding all quadtree leaves adjancent to a given leaf $P$ in a given direction side

```
procedure JOINBLOCK (P,side,JOIN)
   /* input P: leaf of quadtree
           side: direction
      output  JOIN : set of all leaves adjancent to  P  in
      direction side  */
   node  P,Q
   integer side
   set of node  JOIN
   quadtree  QTREE
   begin  JOIN := ∅ ;
          Q := Rot^{-side}(Sub2(Rot^{+side}(P))));
          SEARCH(Q,QTREE,side,JOIN)
   end / joinblock /

procedure SEARCH(Q,QTREE,side,JOIN)
   node  Q
   integer side
   set of node JOIN
   quadtree  QTREE
   begin if  Code(Q) < 0  then  EXIT  /* not adjancent block */
         else if  Q  not in QTREE  then
                    begin  SEARCH(Rnd(Q),QTREE,side,JOIN1);
                           JOIN := JOIN+JOIN1
                    end
              else if Color(Q) ≠ GRAY then JOIN := Q
                    else
                    begin
                    SEARCH(App(Q,Rot$^{+side}$(2)),QTREE,side,JOIN1);
                    JOIN := JOIN + JOIN1;
                    SEARCH(App(Q,Rot$^{+side}$(3)),QTREE,side,JOIN1);
                    JOIN := JOIN + JOIN1
                    end
end /* search */
```

This algorithm can be efficiently used in some problems such as tracing the boundary of region, calculating the perimeter of region ...

## 3.2. AREA AND PERIMETER OF REGION

In order to calculate the perimeter of region, we visit, say, in postorder all black leaves of the quadtree. For each of them, we find all white leaves adjacent to it and the boundary segments. The sum of boundary segments yields the perimeter of region. The area of region is simply sum of area of all black leaves.

```
procedure GEOM(QTREE,n,PERI,AREA)
   /* image is of size  2↑n × 2↑n */
   node P,Q
   quadtree QTREE
   integer  PERI,AREA
   begin          P := (O,O)
                  /*  find a black or white block in upper-left
                  corner */
                  while  Color(P) = GRAY  do  P := App(P,O);
                  PERI := O;    AREA := O
              repeat  /* cycle calculating perimeter and arca */
                while  P  not in  QTREE  do  P := Rnd(P)
                while  Color(P) = GRAY  do  P := App(P,O)
```

```
            if  Color(P) = BLACK  do
        begin
            AREA  := AREA+(n-Level(P))↑2;
            for i := 0  to  3  do
            begin
              JOINBLOCK (P,i,JOIN) ;
              if JOIN = ∅ then PERI := PERI+(n-Level(P))
              else for all  Q  in  JOIN  do
                  if  Color(Q) = WHITE  then
                            PERI := PERI+min(n-Level(P),
                                          n-Level(Q))
            end
         end
         P := (Level(P),Code(P)+1);
     until  Code(P) = 4↑(n-Level(P))
  end  /* geom */
```

It is easy to see that the average execution time of the algorithm is proportional to the number of leaf nodes in quadtree.

## 3.3. CENTROID

The centroid of a binary image is a point $(\bar{x},\bar{y})$ such that $\bar{x}$ is the average value of the $x$-coordinates of all the black points of the image and $\bar{y}$ is the average of the $y$-coordinates of the black points. In other words, if there are $m$ black points in the image $(x_1,y_1),\ldots(x_m,y_m)$, the centroid is

$$(\bar{x},\bar{y}) = (\Sigma x_i/m, \quad \Sigma y_i/m)$$

```
procedure  CENTROID (QTREE,n,XCENT,YCENT)
   /* calculate the centroid of quadtree for image of size
      2↑n × 2↑n; XCENT, YCENT  are the centroid value  */
```

```
    node P
    integer  n
    quadtree  QTREE
    begin
      P := (O,O);
      XCORD := O; YCORD := O;
      MOMENT (P,n,XCORD,YCORD,X,Y,MASS);
      if  MASS = O   then   begin
                            XCENT := O;
                            YCENT := O
                            end
                    else    begin
                            XCENT := X/MASS;
                            YCENT := Y/MASS
      and  /* centroid */

procedure MOMENT (P,n,XCORD,YCORD,X,Y,MASS)
    /* calculate the moments of order O and 1 for block P
    XCORD, YCORD are are coordinates of upper-left corner of  P
    X  is the moment of order 1  $m_{10}$
    Y  is the moment of order 1  $m_{01}$
    MASS  is the moment of order O  $m_{00}$ */
    node  P
    integer  n,XCORD,YCORD,X,Y,MASS
    begin
      X := O;  Y := O;   MASS := O;
      if Color P = GRAY   then   for  i := O  to  3  do
               begin
               MOMENT(App(P,i),n,XCORD+2↑(n-Level(P)-1)*i mod 2,
                     YCORD+2↑(n-Level(P)-1)*i div 2,X1,Y1,M1);
               X := X+X1;  Y := Y+Y1;  MASS := MASS+M1
               end
               else  if  Color(P) = BLACK  then
               begin
```

```
        X := (2*XCORD+2↑(n-Level(P))-1)*2↑(2*(n-Level(P))-1);
        Y := (2*YCORD+2↑(n-Level(P))-1)*2↑(2*(n-Level(P))-1);
        MASS := 2↑(2*(n-Level(P)))
        end
    end /* moment */
```

It is to see, once again, that in the procedure each black leaf in the tree is visited once and only one. The other moments can be calculated in an analogous way.

## 4. EFFICIENCY OF QUADTREE REPRESENTATION

This section is devoted to the discussion of the efficiency of quadtree representation in relation to the other codings. Basing on quadtree codes we give the evaluations of storage space for run length codes, chain codes. Assume that an image of size $2^n \times 2^n$ is represented by quadtree $Q$ with $Q$ nodes. Each node $P$ of has 3 attributes: *Level(P)*, *Code(P)* and *Color(P)*, which need respectively $log_2 n$ bits, $2n$ bits and 2 bits to store them in memory space. Thus, the amount of storage needed for $Q$ is

$$Q(2+2n+log_2 n)$$

If $2n$ bits for *Code(P)* have the representation

$$Code(P) = y_n x_n \dots y_1 x_1, \quad x_i, y_i = 0,1$$

then the coordinates $(x,y)$ of upper-left corner of P is determined as follows [8,10]

$$x(P) = \sum_{i=1}^{L} x_i \, 2^{n-Level(P)+i-1}$$

$$y\ P = \sum_{i=1}^{L} y_i \, 2^{n-Level(P)+i-1}$$

and the size of $P$ is

$$s(P) = 2^{n-Level(P)}$$

## 4.1. EFFICIENCY OF QUADTREE REPRESENTATION IN RELATION TO RUN LENGTH CODE

Let $\mathcal{B} \subset Q$ be the set of black levels of quadtree. We have the integer set $BY$ defined as follows:

$$BY = \{y(P) \mid P \in \mathcal{B}\} \cup \{y(P) + s(P) + 1 \mid P \in \mathcal{B}\}.$$

With the equivalence relation "=" in the usual sense on $BY$, we have the set $\overline{BY} = BY/_{=}$. Denote by $b$ the cardinality of $\overline{BY}$. Each $\bar{y}_j \in \overline{BY}$ determines the image row where may be arise a new run configuration (Fig. 2.)

In order to determine the number of runs, for each $\bar{y}_j \in \overline{BY}$, the associated set $H_j \subset \mathcal{B}$ and the relation $\gamma_j \subset H_j \times H_j$ are defined as follows

$$H_j = \{P \in \mathcal{B} \mid \bar{y}_j \in [y(P), y(P)+s(P)]\}$$

$$P_m \gamma_j P_1 \iff x(P_m) = x(P_1)+s(P_1)+1 \quad \text{or } x(P_1) = x(P_m)+s(P_m)+1 \qquad P_m, P_1 \in H_j$$

Next, if let $\gamma^*$ be the reflexive, transitive closure of $\gamma_j$ on $H_j$, then we have the set $H_j^* = H_j/\gamma^*$, each element of which contains the blocks successively adjacent in horizontal direction.

The total number of runs in run length representation will be

$$\sum_{j=1}^{b} (\bar{y}_{j+1} - \bar{y}_j)\, card\, H_j^* \quad \text{where} \quad \bar{y}_{b+1} = 2^n$$

Thus, the amount of run length code storage needed for image is

$$2(n+1)\sum_{j=1}^{b}(\bar{y}_{j+1}-\bar{y}_j)\,card\,H_j^* \text{ bits}$$

Now we can conclude that given a region represented by quadtree with $Q$ nodes, run length code will be used efficiently when

$$\sum_{j=1}^{b}(\bar{y}_{j+1}-\bar{y}_j)\,card\,H_j^* < \frac{Q(2+2n+\log_2 n)}{2(n+1)}$$

## 4.2. EFFICIENCY OF QUADTREE CODE REPRESENTATION IN RELATION TO CHAIN CODE

For each $B$ in the set of black leaves $\mathcal{B}$, by the procedure JOINBLOCK we can determine the set

$$N_B = \{P \in \mathcal{B} \mid P \text{ is adjancent to } P\}$$

The boundary part, which belongs to $B$ is calculated by

$$4(n-Level(B)) - \sum_{P\in N_B} min\; n-Level(B), n-Level(P))$$

Therefore the number of directional vectors in chain representation is

$$4\sum_{B\in\mathcal{B}} n-Level(B) - \sum_{B\in\mathcal{B}}\sum_{P\in N_B} min(n-Level(B),\, n-Level(P))$$

We obtain the formula calculating the amount of chain code storage needed

$$3(4\sum_{B\in\mathcal{B}} n-Level(B)) - \sum_{B\in\mathcal{B}}\sum_{P\in N_B} min(n-Level(B),\, n-Level(P)))$$

At this point, we can arrive to the conclusion that given a region represented by quadtree, chain code will be used efficiently when

$$4 \sum_{B\in\mathbf{B}} n\text{-}Level(B) = \sum_{B\in\mathbf{B}} \sum_{P\in N_B} min(n\text{-}Level(B), n\text{-}Level(P)) <$$

$$< \frac{Q(2+2n+log_2 n)}{3}$$

## 5. CONCLUSION

For describing quadtree representation of region, a data structure has been presented with the operators defined on it. This operators can be easily implemented in usual programming languages. Based on the data structure, the algorithms have been described for finding adjancent blocks, calculating geometric features of region: area, perimeter and centroid. These algorithms are useful still in the case where a region may have holes. Basing on the data structure, we can study the algorithms of converting quadtree into chain codes and run length codes [13,14]. The last section deals with the analysis of representation efficiency of run length codes and chain codes in relation to a given quadtree. The explicite formulas have been presented for evaluating storages space memory needed for representations. By simple procedures, the calculation of these formulas can be done efficiently.

## REFERENCES

[1] E.L. Hall: Computer image processing and recognition. New York - Academic Press 1979.

[2] N. Alexandiris and Klinger: A picture decomposition, tree data structure and identifing directional symmetries as node combinations. CGIP Vol8 1978, p. 43-77

[3] A. Rosenfeld: Quadtrees and Pyramids for Pattern Recognition and Image Processing. IEEE 1980 5-th Int. Conference on PR. Miami Beach 1980, p. 802-807.

[4] H. Samet: Region representation: Quadtrees from boundary codes. C ACM Mar 1980 p. 163-170

[5] C.R. Dyer et al.: Region representation: Boundary codes from Quadtrees. C ACM Mar 1980 p. 171-179

[6] M. Shneier: Calculation of Geometric Properties Using Quadtrees. CGIP Vol16 N$^{O}$3 July 1981, p. 296-302

[7] H. Samet: Computing Perimeters of Regions in Image Represented by Quadtrees. IEEE Trans. on PAMI, Vol PAMI-3, N$^{O}$6, Nov 1981 p. 683-687

[8] I. GARGANTINI: An Effective Way to Represent Quadtrees. C ACM Vol 25 N$^{O}$12 Dec 1982, p. 905-910

[9] H. Samet: Neighbour finding Techniques for Images Represented by Quadtrees CGIP 1982, p. 37-57

[10] S.X. Li and M.H. Loew: Quadcodes and their application in Image Processing. George Washington University, Department of EE&CS Rept. N$^{O}$IIST 83-15. Oct. 1983

[11] G. Ram: On the Encoding and Representing of Images. CGIP 26 1984, p. 224-232

[12] L.P. Jones and S.S. Iyengar: Space and Time Efficient Virtual Quadtrees IEEE Trans. on PAMI Vol PAMI-6, N$^{O}$2 Mar 1984, p. 244-248

[13] P.N. Khoi and H. Kiem: Code conversion in image processing. Institute of Technical Cybernetics. SAV-Bratislava. Rept. N$^{O}$14/1984

[14] P.N. Khoi et al: Conversion and Synthesis of Imate Representation. Proceeding of the Conference on Applied Mathematics. Hanoi 1985 (to appear)

## A "négyzetes fa" /quadtree/ kódok adatstrukturája alkalmazásokkal

PHAM NGOC KHOI

Összefoglaló

A szerző a "négyzetes fa"-kódok reprezentációja számára bevezet egy adatstrukturát és operátorokat definiál rajta. Algoritmusokat ad meg a kiegészitő blokkok megkeresésére, valamint a tartományok geometriai jellemzői /pl. terület, kerület, sulypont/ meghatározására.
Összehasonlitva a "futás-hosszuság" és "lánc" kód reprezentációkkal, értékeli a "négyzetes fa" reprezentáció hatékonyságát.

## Структура данных для "квадратического дерева"-кодов с применением

Пхам Нгоц Кхои

Резюме

В статье описывается структура данных для представления кодов типа "квадратических деревьев" и определены операторы на этой структуре. Даются алгоритмы для нахождения дополнительных блоков и для вычисления геометрических свойств /площадь, периметр, центроид/ областей.

Представление типа "квадратических деревьев" сравнивается с представлением типа "длина пробега" и "цепь".