창 립
40주년학술대회
논문 87-1-20-6

BF 선형 4진트리로 표현된 화상에서 둘레 길이 연산

°한 상호, 이 극, 김 민환, 황 희웅
서울대학교 공과대학 전자계산기공학과

Computing Perimeters of Regions for Images Represented by BF Linear Quadtree

Han Sang Ho, Lee Geuk, Kim Min Hwan, Hwang Hee Yeung
SEOUL NATIONAL UNIVERSITY, DEPT. COMPUTER ENGINEERING

<ABSTRACT>

An algorithm is presented which computes the perimeter of regions in images represented by BF(breadth first) linear quadtree. In order to compute the perimeter, we determine adjacent neighbor nodes in the horizontal and vertical directions. It is the method which directly find the associated nodes in order to know whether it exists in BF linear quadtree or not.

Average execution time of the algorithm is proportional to the number of black nodes in the quadtree.

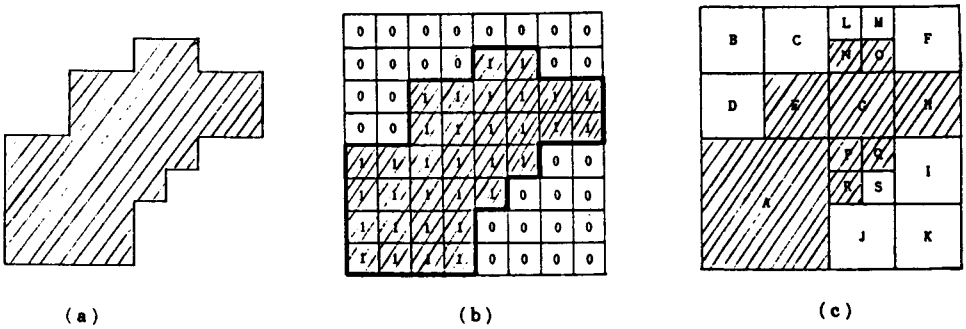
1. 서론

계층적 데이타 구조(hierachical data structure)인 영역 4진트리 (region Quadtree)는 화상처리, 콤퓨터 그래픽스, 패턴 인식 , 로보틱스, 지도 제작등과 같은 여러 분야에서 중요한 표현 기법으로 인식되고있다[1]. 이러한 범주내에 있는 전형적인 연산들에 대한 많은 알고리즘들이 각 단말 노드에서 인접한 이웃 노드 탐색과 관련된 트리 순회(tree traversal)로서 구현되었다[2,4]. 이웃 노드 탐색을 필요로 하는 전형적인 연산중의 하나로써, 임의의 물체의 둘레 길이(perimeter)계산은 화상처리 분야에서 기본적인 연산 중의 하나로 인식되어 이에 대한 연구들이 진행되어 왔다[3,5]. 그러나 영역 4진트리 (region Quadtree) 표현 방법은 각 노드의 자식과 부모 노드를 나타내기 위해 포인터가 필요하고 자료 저장 방법이 낭비적이기 때문에 새로운 표현 방법으로서 선형 4진트리 (linear quadtree) 표현이 제시되었다[6,7]. 선형 4진트리는 4진트리의 각 노드에서 4개의 자식 노드와 부모 노드에 대한 포인터를 사용하지 않고 단말 노드인 흰 노드와 검은 노드를 정렬시킴으로써 효율적으로 화상을 표현할 수 있는 장점이 있다.

본 논문은 물체 영역의 둘레 길이를 계산하기 위하여 특별히 검은 노드들로만 구성된 BF 선형 4진트리로 표현된 화상에서 BF 탐색 방법으로 처리하였다[8]. 그 방법은 4진트리에서 트리순회하여 공통 조상 노드(ancestor node)를 찾아내어 그 공통 조상 노드를 중심으로해서 인접한 이웃 노드(neighbor node)를 구하는 방법을 사용하지 않고 이웃 노드의 위치를 직접 계산하여[9] BF 선형 4진트리상에서 해당 노드가 있는가를 검사한다. 이때 물체의 둘레 길이를 구하기 위한 알고리즘의 수행 시간은 평균적으로 단말 노드 중에서 검은 노드의 수에 비례한다. 물체 영역의 둘레 길이(perimeter)를 계산할 때 이웃 노드는 수평 방향과 수직 방향, 즉 모두 4방향으로 갈거나 큰 노드이다.

2. 정의

4진 트리(quadtree)는 화상을 같은 크기로 4등분씩 계속 분할하여 표현하는 방법으로서 만약 분할된 구간이 모두 같은 특징을 나타내면 더 이상 분할하지 않고, 만약 분할된 구간이 서로 다른 특징, 즉 전부 1 이거나 0 인 블럭(block)이 아니면 계속 4등분하여 해상도가 허용하는 단계까지 분할한다. 예로써 그림 1-(b)와 같이 $2^3 * 2^3$ 이진 화상을 영역(region)으로 표시한 것이 그림 1-(a)이다. 그림 1-(b)를 블럭으로 표시한 것이 그림 1-(c)이고 4진트리로 표현 한 것이 그림1-(d)이다. 일반적으로 루리 노드(root node)를 레벨 0라 하고 level i에 있는 노드는 트리의 루리 노드로 부터 i거리에 있다고 한다. 화소(pixel)에 해당하는 노드들은 level n 이다. 또한 어느 한 노드가 트리의 level s에서 존재한다면 변(side)의 크기는 2^{n-s}이다.



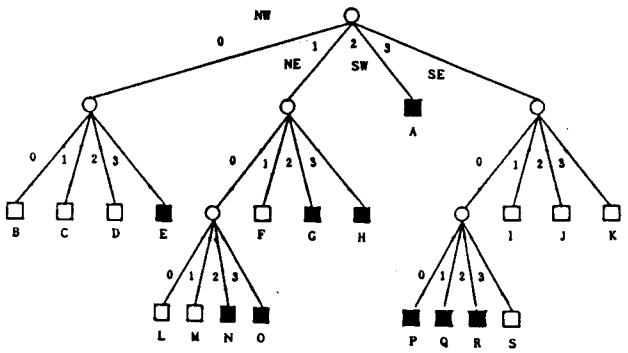


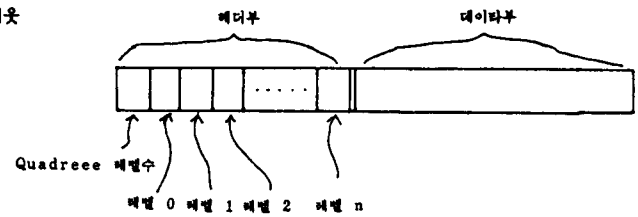
그림 1. (a) 영역(region) (b) 이진 화상 (c) (a)의 영역을 블럭으로 분해
(d) (c)에서 블럭들의 4진트리 표현

4진트리의 각 노드는 주어진 이진 화상의 블럭에 해당한다. 각 블럭은 4개의 변(side)과 4개의 모서리(corner)를 갖는다. 그림 2와 같이 한 블럭의 4개의 변에 대한 방향은 N, E, S, W 방향을 갖는다. 이때, D는 N, E, S, W 방향을 가진다고 한다. 주어진 두개의 노드 P와 Q에서 P의 D 방향 쪽의 중첩되지 않은 이웃이 있으면 P는 D방향 쪽으로 인접한 이웃 Q를 갖는다고 하고 adjacent(P,D) = Q라는 이웃 함수(neighbor fuction)로 정의한다. 이웃 함수는 전단사 함수가 아니고 대칭성이 없다.

NW (0)	NE (1)
SW (2)	SE (3)

그림 2. 화상 분할시 위치의 정보

BF(Breadth First) 선형 4진트리는 4진트리에서 검은 노드들로만 Breadth- First방식에 의해서 순서적으로 저장하는 표현기법이다. 이것은 데이타를 저장할 때 각 레벨의 갯수를 나타내는 헤더부와 루리로부터 각 검은 노드에 대한 경로명(path name)을 breadth-first탐색 순서로 나열한 데이타부로서 표현한다. 예를 들어 그림 1-(d)에서 노드 R의 경로명은 302이다. 그림 3-(a)가 BF 선형 4진트리의 데이타 구조이며, 그림 3-(b)는 그림 1-(b)를 BF 선형 4진트리로 표현한 것이다.



(a)

4	0	1	3	5	2	03,12,13	102,103,300,301,302

(b)

그림 3. (a) BF 선형 4진트리의 데이타 구조
(b) 그림 1-(b)를 선형 4진트리로 표현

3. 인접한 이웃 노드 찾기 연산

일반적으로 $2^n * 2^n$ 이진 화상을 4진트리로 표현했을 때, 각 단말 노드의 경로명을 구할 수 있다.

예를 들면 $2^3 * 2^3$ 이진 화상인 그림 1-(b)를 각 단말 노드의 경로명을 구하여 표현한것이 그림 4이다. 일반적으로 어느 한 노드는 변(side)방향인 N,W,S,E 4 방향으로 인접한 이웃 노드가 가능하다. 그러나, 여기서는 변 방향중에서 서쪽(W) 방향의 이웃에 대해서 예로써 설명한다.

3.1 서쪽(W) 방향의 이웃 찾기

검은색 노드인 A에서 서쪽으로 인접한 이웃 노드를 W라 하고, A 배열의 크기는 n이라 한다. 이때, A의 서쪽 이웃 노드 W는 A와 같은 크기인 배열이다. 그리고 A와 W 배열의 크기는 1 에서부터 가장 큰 레벨값, 즉 화소의 경로명 길이까지 가질 수 있다. 그림 1-(c)에서 노드 N의 경로명은 102 인데, A[0] = 2, A[1] = 0, A[2] = 1 인 관계를 가진다.

A의 서쪽 이웃 W는 다음과 같은 특성을 가진다.

1] A[0]가 1이나 3이면, W는 같은 사분면 안에 존재한다.

2] A[0]가 0이나 2이면, W는 다른 사분면 안에 존재한다.

알고리즘은 아래와 같다.

```
node procedure FIND_ADJACENT(A,W)
      begin
        integer A[],W[];
        integer i,j,n;
        n = length(A);
        for i = 0 step 1 until n-1
            begin
              if (A[i] == n - 1) then
                 W[i] = A[i] - 1;
              if (A[i] == 1 or A[i] == 3) then
                 begin
                 W[i] = A[i] - 1;
                 for j = i + 1 step 1 until n - 1
                     W[j] = A[j];
                     return(W);
                 end;
              else if (A[i] == 0 or A[i] == 2) then
                   W[i] = A[i] + 1;
            end;
        return(W);
      end.
```

3.2 BF linear quadtree 에서 이웃 노드 탐색

다음에 Q가 BF 선형 4진트리에 있는가를 검사하여야만 한다. 우리는 어느 한 노드 P에 D방향으로 인접한 노드 Q를 구할때, D방향으로 P와 같은 크기의 노드 Y를 구하고, Y를 이용하여 검은 노드들만 저장한 BF 선형 4진트리상에서 탐색을 하여 Q가 검은 노드인지 흰 노드인지를 결정할 수 있다. 이때, P의 이웃 노드 Q는 P보다 같거나 큰 노드이다. 그 방법은 BF 선형 4진트리를 각 노드가 경로명에서 맨 처음에 시작하는 코드값, 즉 0, 1, 2, 3 중의 하나로 시작되므로, 한 코드값으로 시작되는 경로명들을 하나의 BF 선형 4진트리로 만들 수 있다. 예를 들면 그림 2-(b)를 그림 5로 바꿀 수 있다. 이때 Q를 검사하기 위해 그림 5에서 4개의 BF

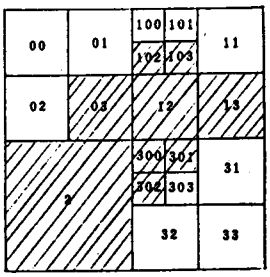


그림 4. 그림 1-(c)의 블럭들을 경로명으로 표현한 예

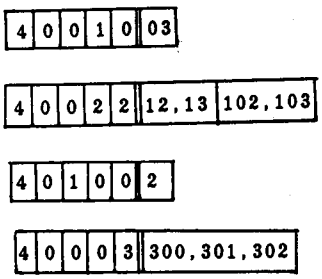


그림 5. 그림 2-(b)에서 각각의 코드값에 대한 BF 선형 4진트리

선형 4진트리중에서 Q의 처음 코드값으로 표현된 것만 검사한다. 만약 Q가 존재하면 검은 노드이고,그렇지 않으면 Q는 흰 노드이다. 이때 Y와 Q의 관계는 다음과 같다.

1] Y가 BF 선형 4진트리상에서 존재하고, size(Y) <= size(Q)면, Q는 검은 색 노드이고 Y의 이웃 노드;

2] Y가 BF 선형 4진트리상에서 존재하고, size(Y) > size(Q)면, Q는 검은 색 노드이나 Y의 이웃 노드는 아니다.;

3] 1],2]가 아니면 Q는 흰색 노드이다.

4. 물체의 둘레 길이(perimeter) 연산.

4진트리로 표현된 $2^n * 2^n$ 이진 화상에서, 물체의 둘레 길이 계산은 어느 한 노드 P에서 4 방향의 인접한 이웃 노드 Q를 검사함으로서 가능하다. P가 검은 노드이고, Q가 흰 노드인 경우에만 물체 둘레 길이의 값에 포함된다. 이때, BF 선형 4진 트리가 검은 노드들로만 저장되어 있으므로 BF 선형 4진트리에서 검은 노드들인 각 노드 P에서 4 방향으로 같거나 큰 인접한 이웃 노드 Q를 구하여, Q가 흰 노드이면 물체 둘레 길이의 값에 포함시키고, 그렇지 않으면 포함시키지 않는다. 그리고 Q가 P보다 같거나 크며 검은 노드이면, P의 이웃 노드라고하고, Q가 P보다 크기가 작거나 또는 같거나 크더라도 흰 노드이면, P의 이웃 노드가 아니라고 가정한다.

물체의 둘레 길이를 구하는 방법은 BF 선형 4진트리에서 크기가 가장 작은 검은 노드부터 시작하여 4 방향으로 이웃 노드를 검사한다. 이웃 노드가 없는 방향은 P의 변 길이만큼 물체의 둘레 길이 값에 더하고, 큰 노드와 이웃한 경우는 그 방향에 대해 P의 변 길이만큼 뺀다. 그리고 이 과정을 가장 큰 검은 노드까지 모든 검은 노드에 대해 검사한다. 모든 검은 노드를 검사한 후에, 물체의 둘레 길이는 계산된다. 만약 화상에서 물체가 여러개 있는 경우는 모든 물체 길이의 총합이 구해진다. 예로써, 그림 1-(c)의 화상에서 물체의 둘레 길이 계산은 다음과 같다. 먼저 가장 작은 노드 N, O, P, Q, R에 대해서 수행하고 다음에 E, G, H를 수행하고 마지막으로 A를 수행한다. 노드 N에서 이웃이 없는 방향은 서쪽과 북쪽이고, 동쪽은 이웃 노드가 존재하지 않고 이웃 노드가 있는 방향은 남쪽이다. 그래서 노드 N의 값은 1 + 1 + 0 + (-1) = 1 이다. 이와 같은 방법을 적용하면 O, P, Q, R, E, G, H, A는 각각 1, -2, 1, 1, 2, 4, 6, 16인 값을 갖는다. 이것을 전부 더하면 그림 1-(c)의 둘레 길이가 30이 된다. 이때 길이의 기본 단위는 화소의 한 변의 길이이다.

알고리즘은 아래와 같다.

```
integer procedure PERIMETER(Q)
/* compute perimeter of BF linear quadtree BLQ */
begin
  Quadtree Q;
  node P,Q;
  integer PER,level;
  Direction D;
  PER <- 0;

  for each node P in BLQ
     begin
        for each D in ("N,""E,""S,""W,") do
           begin
              Q = GTEQUAL_ADJ_NEIGHBOR(P,D);
              if NOT(Q = NULL) then
                 begin
                   level = length(P);
                   PER = PER + 2^(n-level)
                 end;
           end;
     end;
     return(PER);
end.

node procedure GTEQUAL_ADJ_NEIGHBOR(P,D)
/* Return the neighbor of node P in horizontal
or vertical direction D which is greater than
or equal in size to P */

begin
   node P,Y,Q;
   direction D;
   Y = FIND_ADJACENT(A,D);
   Q = SEARCH_LQ(Y);
   if NOT(Q == NULL) then
      return(Q);
   else return(NULL);
end.
```

5. 성능 평가.

각 상수를 정의하면 다음과 같다.

TN : 총 노드 수

BN : 총 검은 노드 수

WN : 총 흰 노드 수

TN = BN + WN

각 방법에 대한 평균 실행 시간은 다음과 같다.

* tree traversal에 의한 둘레 길이 계산 : O(TN)
* BF linear quadtree에 의한 둘레 길이 계산 : O(BN)

6. 결론.

물체의 둘레 길이를 계산할 때, 수평 방향 또는 수직 방향, 즉 모두 4 방향에 대해 같거나 큰 이웃 노드를 구하였다. 또한 물체 둘레 길이를 구하는 알고리즘은 4진트리에서 단말 노드 전체에 대해서 적용시키지 않고 검은 노드들에 대해서만 적용시켰다. 이때 이 알고리즘의 수행 시간을 단말 노드중에서 검은 노드 수에 비례하게 하여, 물체 둘레 길이를 구할 때 평균 수행 시간을 줄였다.

7. 참고 문헌

[1] H.Samet, "The Quadtree and Related Hierarchical Data Structures," Computing Surveys.ACM,vol.10,No.2,pp.187-260,June, 1894

[2] C.R.Dyer, "Computing the Euler number of an image from its quadtree," Comput. Graphics Image Processing, vol. 13, no.3, pp.270-276, July 1980

[3] H.Samet, "Computing Geometric Properties of Images Represented by Linear Quadrees," IEEE trans. PAMI-7, No. 2, pp.229-239, Mar., 1985

[4] -, "Conntected component labeling using quadtrees," J. Ass. Comput. Mach., vol28, pp.487-501, July 1981

[5] -, "Computing Perimeters of Regions in Images Represented by Quadtrees," IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-3, pp.683-687, Nov., 1981

[6] E.Kawaguchi, T.Endo, digital picture processing," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-5, pp373-384, July, 1983

[7] E.Kawaguchi, and its application to data compression," IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-2, pp25-35, Jan., 1980

[8] 이 민규,김 민환,황 희웅, "2진 영상의 효과적인 표현법:BF Linear Quad- tree, "대한 전기 학회 전자계산기 연구회 추계 학술 연구 발표회 논문집, pp.23-28, 1985

[9]I.Gargantini, "An effective Way to represent quadtrees," Comm. ACM, vol.25, pp.905-910, Dec. 1982