한국 CAD/CAM 학회 논문집
제 14권 제 4호 2009년 8월 pp. 217-224

배열기반 데이터 구조를 이용한 간략한 divide-and-conquer 삼각화 알고리즘

양상욱*, 최 영**

A Compact Divide-and-conquer Algorithm for Delaunay Triangulation with an Array-based Data Structure

Sangwook Yang* and Young Choi**

ABSTRACT

Most divide-and-conquer implementations for Delaunay triangulation utilize quad-edge or winged-edge data structure since triangles are frequently deleted and created during the merge process. However, the proposed divide-and-conquer algorithm utilizes the array based data structure that is much simpler than the quad-edge data structure and requires less memory allocation. The proposed algorithm has two important features. Firstly, the information of space partitioning is represented as a permutation vector sequence in a vertices array, thus no additional data is required for the space partitioning. The permutation vector represents adaptively divided regions in two dimensions. The two-dimensional partitioning of the space is more efficient than one-dimensional partitioning in the merge process. Secondly, there is no deletion of edge in merge process and thus no bookkeeping of complex intermediate state for topology change is necessary. The algorithm is described in a compact manner with the proposed data structures and operators so that it can be easily implemented with computational efficiency.

Key words : Delaunay triangulation, divide-and-conquer, compact data structure

1. 서 론

점 집합으로부터 들로네 삼각화(Delaunay triangulation)를 구하기 위한 방법으로써의 divide-and-conquer (이하 D&C) 방법은 사이트 집합 $\boldsymbol{P}$에 대해서 $|\boldsymbol{P}|$가 1이나 적절히 작은 수가 될 때까지 부분 집합들로 분할하고, 부분 집합들에 대해서 삼각화를 수행한 후 역순으로 병합하여 전체 집합 $\boldsymbol{P}$에 대한 삼각화를 구성하는 방법이다. 이 방법은 사이트 분포에 상관없이 좋은 성능을 보여주나 일반적으로 구현하기가 까다롭다고 알려져 있다[1].

D&C 방법을 이용한 보로노이 다이어그램의 구성은 Shamos와 Hoey[2]가 처음으로 제시하였으며 이차원 평면에서 그 시간 복잡도(time complexity)가 $O(n\log n)$ 임을 보였다. 보로노이 다이어그램을 구하지 않고 사이트 집합으로부터 들로네 삼각화를 직접 구성하는 방법은 Lee와 Schachter[3]가 처음으로 소개하였으며 이후 Guibas와 Stolfi[4]는 quad-edge 기반의 데이터 구조를 정리하고, 데이터 구조와 관련된 기본 연산을 이용한 정형화된 방법을 이용한 알고리즘을 제안하였다. 초기의 알고리즘들은 좌표계의 한 축의 방향으로 공간을 분할했지만, 이후 병합시 계산의 안정성을 높이고 중간 단계에서의 에지의 생성 및 삭제의 회수를 줄이기 위하여 다차원 공간 분할에 대한 연구가 이루어졌다. Ohya 등[5]과 Maus[6]는 버켓(bucket)을 이용한 이차원 공간 분할로 평균(expected running time) $O(n)$, 최대 $O(n^2)$의 알고리즘을 제안하였으며 Dwyer[7]는 Guibas와 Stolfi의 방법을 수정하여 행과 열로 영역을 분등하게 분할하여 이차원 균등 분포 점들에 대한 평균 $O(n\log(\log n))$, 최대 $O(n\log n)$ 알고리즘을 소개하였다.

*정회원, 중앙대학교 미래신기술연구소
**종신회원, 교신저자, 중앙대학교 기계공학부
- 논문투고일: 2009. 03. 04
- 논문수정일: 2009. 04. 15
- 심사완료일: 2009. 04. 29

Katajainen과 Koppinen[11]은 이차원 공간을 사분 트리(quad-tree[8])를 이용하여 분할하였으며, Adam 등[9]은 적응적 이차원 트리를 이용하여 공간 분할의 효율을 높였다. 비교적 최근 연구로는 Lemaire[10]가 *kd*-트리를 이용하여 공간을 분할할 때 개연론에 근거한 수행 시간 예측을 *k* 차원에서 정리하고, 이차원 삼각화에 적용하는 연구를 수행한 바 있다.

제안하는 알고리즘은 이차원 공간의 사이트 집합 V로부터 들로네 삼각화 $T(V)$를 구성하기 위하여 이차원 공간분할을 통해서 D&C 방법을 수행한다. 제안하는 알고리즘은 크게 두 가지 새로운 특징을 가진다. 첫번째로, 공간을 효율적으로 분할하기 위해서 Lemaire[10]의 방법과 마찬가지로 *kd*-트리와 같은 공간 분할을 사용한다. 그러나 실제로 사이트들을 위한 트리를 구성하지 않고도 배열과 인덱스를 사용하여 빠르게 분할된 공간 객체들을 다룰 수 있다. 또한 트리 데이터 구조를 위한 별도의 정보를 저장하지 않음으로써 메모리를 사용을 최소화한다. 둘째로는, 병합 과정이 에지나 삼각형의 삭제 없이 이루어진다. 전통적인 방법의 D&C 알고리즘의 병합 과정에서는 두 삼각화를 병합하여 새로운 삼각화를 구성할 때 기존 삼각화에서 에지들이 제거되어야 하는데, 제안하는 방법에서는 기존 삼각형이나 에지가 제거되지 않고 플립 연산의 전파에 의해서 들로네 특성을 유지한다. 이로써 성능의 저하없이 D&C 삼각화의 구현을 매우 간단하게 할 수 있고, 계산과 구현 코드의 안정성을 기대할 수 있다.

2. 배열기반 데이터 구조와 연산자

2.1 배열기반 데이터 구조

일반적으로 삼각화 알고리즘의 구현을 위해서 그에 적합한 데이터 구조를 채택하여 사용한다. 예를 들어, D&C 알고리즘의 병합 과정에서는 에지의 생성과 삭제가 빈번하게 일어나는데, 이를 위하여 대부분의 알고리즘에서는 에지 데이터를 중심으로 인접 토폴로지의 정보를 검색할 수 있는 quad-edge 데이터 구조를 사용하고 있다[4]. 또한 점진적 구성 방법에서는 에지 보다는 삼각형의 분할과 플립 오퍼레이션으로 삼각화를 진행하기 때문에 구성된 삼각형들의 트리를 표현할 수 있는 데이터 구조를 사용한다[12].

본 논문에서는 배열 기반의 매우 간단하고 효율적인 데이터 구조를 이용하여 직접 삼각화를 수행한다. 제안하는 데이터 구조는 색인된 삼각형(indexed triangle)의 형태로 버텍스와 삼각형의 배열만을 저장하지만 삼각화 과정에서 필요한 인접 정보의 검색과 순회(traversal)가 가능하다.

Kim[13] 등은 보로노이 다이어그램의 듀얼로 정의될 수 있는 3차원 공간의 구들의 삼각화(quasi-triangulation)를 위한 배열 기반의 interworld 데이터 구조를 소개하였으며 [14]에서는 interworld 데이터 구조를 보다 간단히 변형하여 경계기반의 삼각화(CDT; Constrained Delaunay triangulation)를 이차원 매개변수 공간에 활용하기 위한 자료구조를 정의하였다. 여기서는 CDT가 아닌 일반적인 들로네 삼각화를 다루므로 구속조건 정보가 없는 간단한 삼각화 데이터 구조를 설명한다.

이차원 공간상에서 하나의 버텍스를 v라고 하면, n개의 버텍스 집합 V와 V에 의해 생성되는 들로네 삼각화 $T(V)$는 다음과 같이 표현될 수 있다.

$$V = \{v\} = \{v_0, v_1, \ldots, v_{n-1}\} \tag{1}$$

$$T(V) = \{t\} = \{t_0, t_1, \ldots, t_{m-1}\} \tag{2}$$

여기서 m은 삼각형의 개수이다. v와 t는 각각 버텍스 배열과 삼각형 배열의 요소이며, 각 요소는 다음과 같이 표현될 수 있다.

$$v = (x, y, L) \tag{3}$$

$$t = (M_0, M_1, M_2, N_0, N_1, N_2) \tag{4}$$

식 (3)의 x와 y는 버텍스의 직교 좌표 값을 나타내며, L은 v에 연결된 여러 삼각형들 중 임의의 한 삼각형의 인덱스를 나타낸다. v를 참조하고 있는 삼각형은 동시에 여러 개가 될 수 있지만 v의 측면에서는 최소한 하나의 인접한 삼각형만 알 수 있다면 v를 참조하는 모든 삼각형을 순회할 수 있다. 식 (4)와 M_i는 삼각형 t를 구성하는 세 버텍스의 인덱스를 나타내며 그 버텍스들은 집합 V의 요소이다. M_0와 M_1, M_2에 의해 참조되는 점들의 좌표는 이차원 공간에서 반시계 방향(CCW)으로 삼각형을 이룬다. N_i는 삼각형 t에 인접한 세 삼각형의 인덱스를 나타낸다. N_i에 의해 참조되는 삼각형은 t와 에지를 공유하는 인접 삼각형이면서 M_i에 의해 참조되는 버텍스와 연결되지 않는 삼각형이다. Fig. 1의 예를 보면, 색칠된 삼각형은 $t=(M_0=2, M_1=6, M_2=5, N_0=6, N_1=7, N_2=3)$으로 표현될 수 있다. N_0가 가리키는 삼각형 t_6는 t의 인접 삼각형이며 M_0가 가리키는 버텍스 v_2에 연결되어 있지 않다. $i=1$의 경우 $M_1 \rightarrow v_6$, $N_1 \rightarrow t_7$로 v_6와 t_7이 연결되지 않으며 $i=2$의 경우도 마찬가지이다. 직관적으로 이야기하면, 한 삼각형에 있어서 한 버텍스의 인덱스에 대응하는 삼각형은 해당 버텍스의 반대

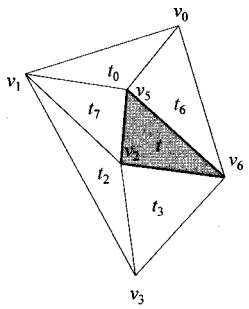

Fig. 1. An example of triangles adjacency.

편에 있다고 할 수 있다.

Fig. 2는 버텍스 집합 V에 대한 들로네 삼각화 $T(V)$의 한 예를 보여 주고 있으며, $V = \{v_0, v_1, ..., v_8\}$와 $T(V) = \{t_0, t_1, ..., t_{10}\}$로 이루어져 있다. 각 버텍스의 화살표는 식 (3)에서 나타난 삼각형에 대한 레퍼런스를 의미한다.

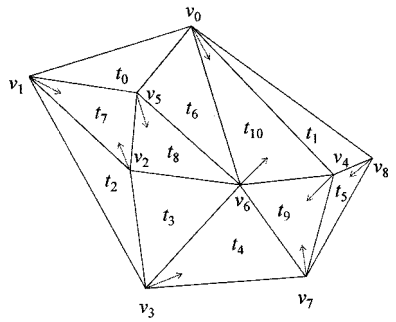

Fig. 2. An example of Delaunay triangulation.

Table 1은 Fig. 2에 대한 삼각형 배열의 각 요소 값들을 보여주고 있다.

Table 1. Array data for the triangles in Fig. 2

t	M_0	M_1	M_2	N_0	N_1	N_2
t_0	5	0	1	ø	7	6
t_1	0	4	8	5	ø	10
t_2	3	2	1	7	ø	3
t_3	6	2	3	2	4	8
t_4	7	6	3	3	ø	9
t_5	8	4	7	9	ø	1
t_6	6	0	5	0	8	10
t_7	5	1	2	2	8	0
t_8	5	2	6	3	6	7
t_9	6	7	5	5	10	4
t_{10}	6	4	0	1	6	9

테이블과 그림에서 모든 삼각형은 각 점들이 CCW 방향이 되도록 참조하고 있으며, 해당 버텍스에 대응하는 인접 삼각형은 기준 삼각형의 반대편에 있음을 확인할 수 있다. 전체 삼각화의 컨벡스 폴리곤을 구성하는 최외곽의 삼각형들의 경우 인접 삼각형이 없을 수 있는데, 이 때 인덱스로써 ø 혹은 nil을 이용하여 표현한다. Fig. 2를 보면 t_4의 경우, (v_7, v_6, v_3)으로 이루어져 있고, 각각의 대응 삼각형이 (t_3, ø, t_9)이며 이는 Table 1의 다섯 번째 항목에 나타나 있다.

2.2 삼각화를 위한 연산자

D&C 삼각화 연구에서 널리 인용되고 있는 Guibas[4]의 연구는 보로노이 다이어그램 계산을 위한 형상 및 위상 정보를 위한 데이터 구조로서 quad-edge 데이터 구조를 소개하고 기본적 연산(manipulation)을 이용하여 알고리즘을 설명하였다. 이렇게 연산자에 기반한 알고리즘 기술은 알고리즘을 구조적(hierarchical)으로 분리하여 기술할 수 있게 하므로 간략하면서 이해하기 쉬운 서술을 가능하게 하고, 구현 또한 용이하게 해준다. 이 절에서는 2.1절에서 정의된 데이터 구조를 기반으로 삼각화 데이터의 정보를 검색하고 수정할 수 있는 기본 연산자들을 정의한다. 정의된 연산자들은 3절의 삼각화 알고리즘에서 사용된다.

2.2.1 인접 정보 조회와 순회

아래에서 정의되는 *vert*와 *tri* 연산자는 각각 버텍스와 삼각형을 조회(retrieval)하기 위한 연산자이다. 이 연산자를 이용하여 Fig. 2와 Table 3의 예를 살펴보면, $vert(t_3, 2)$은 v_3이 되고, $tri(t_3, 2)$은 t_8이 된다. 참고로 M_i와 N_j는 별다른 표기가 없는 한, t의 버텍스 인덱스와 인접 삼각형 인덱스를 의미한다. $M_i(t_s)$로 표기된다면 t_s의 M_i 인덱스를 의미하며 $N_j(t_s)$의 경우 t_s의 N_j 인덱스를 의미한다. 두 번째 *tri* 연산자는 Fig. 2에 나타난 화살표와 같이 식 (3)으로부터 v가 참조하는 삼각형을 얻는다. 얻어진 삼각형은 v를 공유하는 여러 삼각형 들 중 하나이다.

$vert(t, i_0) = v_{Mi}$, where $v_{Mi} \in V$ and $i = i_0 \bmod 3$
$tri(t, j_0) = t_{Nj}$, where $t_{Nj} \in T$ and $j = j_0 \bmod 3$
$tri(v) = t_L$, where $t_L \in T$

*vert*와 *tri*의 입력 파라메터로 오는 인덱스가 어떤 정수이든지 3으로 나눈 나머지를 취하므로 M_i와 N_j의 인덱스 값은 항상 0, 1, 2가 된다. 다음의 연산자는 버텍스와 삼각형의 인덱스를 조회한다.

$idx(t, v_a) = i$, where, $v_{Mi} = v_a \in V$ and $i = 0, 1, 2$
$idx(t, t_a) = j$, where, $t_{Nj} = t_a \in T$ and $j = 0, 1, 2$

제안하는 삼각화 데이터 구조에는 에지를 위한 데이터가 정의되어 있지 않다. 그러나 에지는 다음과 같이 삼각형과 그 에지를 공유하는 인접 삼각형의 인덱스의 쌍(pair)으로 기술할 수 있다.

$edge(t, k)$ = the edge shared by t and $tri(t, k)$

Fig. 2와 Table 1의 예에서 $edge(t_8, 1)$은 t_8과 t_6 사이에 있는 에지를 가리킨다. 테이블로부터 t_8의 인접 삼각형 인덱스를 찾아보면 $tri(t_8, 1) = t_6$임을 알 수 있다.

식 (3)에서 버텍스는 인접한 삼각형의 참조를 단 하나 가지는 것으로 정의되어 있다. 따라서 하나의 버텍스로부터 인접 삼각형들을 얻어내기 위해서는 순회(traversal)가 필요하다. 연산자 *cwt*와 *ccwt*는 다음과 같이 정의된다.

$cwt(v, t) = tri(t, idx(t, v) + 2)$,
where t is an adjacent triangle of v
$ccwt(v, t) = tri(t, idx(t, v) + 1)$,
where t is an adjacent triangle of v

Fig. 2에서 v_5의 경우 t_8을 참조하고 있는데, v_5의 기준으로 t_8의 CCW 방향의 삼각형을 찾으려면 *ccwt* 연산자를 사용할 수 있다. 즉, $ccwt(v_5, t_8) = tri(t_8, idx(t_8, v_5) + 1) = t_6$이 된다. 마찬가지로 CW의 경우에는 *cwt* 연산자를 이용하면 $cwt(v_5, t_8) = tri(t_8, idx(t_8, v_5) + 2) = t_7$이 된다. *ccwt*나 *cwt* 연산자를 반복 사용하면 버텍스 v에 인접해 있는 모든 삼각형을 순회할 수 있다.

삼각형 t를 기준으로 v의 선행 버텍스와 v의 후행 버텍스를 얻는 *tprev*와 *tnext* 연산자는 다음과 같이 정의된다.

$tprev(t, v) = vert(t, idx(t, v) + 2)$
$tnext(t, v) = vert(t, idx(t, v) + 1)$

컨벡스 폴리곤을 이루는 외곽의 버텍스 v에서 컨벡스의 경계를 따라 선행 버텍스와 후행 버텍스를 얻기 위해서는 경계상의 삼각형들을 먼저 검색해야 하는데, 이를 위한 연산자 *mcwt*와 *mccwt*는 다음과 같이 재귀적으로 정의된다.

$mcwt(v,t) = \exists cwt(v, t)$, then $mcwt(v, cwt(t))$, else t
$mccwt(v,t) = \exists ccwt(v, t)$, then $mccwt(v, ccwt(t))$, else t

mcwt 연산자는 v를 기준으로 하고 t를 시작으로 해서 CW 방향으로 더 순회할 수 없는 끝 삼각형이며 *mccw* 연산자는 CCW 방향으로 더 순회할 수 없는 끝 삼각형이다.

외곽 컨벡스 폴리곤 상의 버텍스 v로부터 CW 방향의 외곽 버텍스를 얻는 연산자 *prev*와 CCW 방향의 외곽 버텍스를 얻는 연산자 *nexv*는 다음과 같다.

$prev(v) = tprev(mccwt(v, tri(v)), v)$
$nexv(v) = tnexv(mcwt(v, tri(v)), v)$

Fig. 2에서 v_3의 경우 식 $prev(v_3)$와 $next(v_3)$는 다음과 같이 전개된다.

$$\begin{aligned} prev(v_3) &= tprev(mccwt(v, tri(v_3)), v_3)) \\ &= tprev(mccwt(v_3, t_4), v_3)) \\ &= tprev(t_2, v_3) \\ &= v_1 \\ nexv(v_3) &= tnexv(mcwt(v_3, tri(v_3)), v_3)) \\ &= tnexv(mcwt(v_3, t_4), v_3)) \\ &= tnexv(t_4, v_3) \\ &= v_7 \end{aligned}$$

2.2.2 삼각형 수정 연산

다음의 연산자 *ctneighbor*는 존재하는 삼각형 t와 t 외부의 점 v로부터 t에 인접한 새 삼각형을 만든다.

$ctneighbor(t, v, i)$ = a new triangle t_n,
where $tri(t, i) = t_n$, $tri(t_n, idx(v)) = t$
if $tri(v) = \emptyset$ then set $tri(v) = t_n$

Fig. 3(b)와 (c)는 각각 Fig. 3(a)의 예로부터 $ctneighbor(t, v, 1)$와 $ctneighbor(t, v, 0)$ 연산이 수행된 결과를 보여주고 있다.

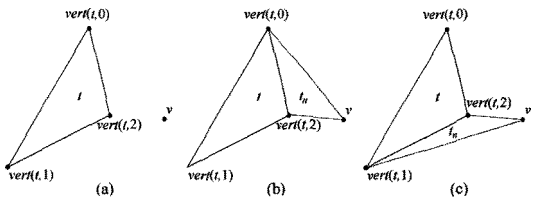

Fig. 3. Examples of applying *ctneighbor* operator.

다음의 연산자 *mate*는 실제로 두 버텍스를 공유하지만 토폴로지의 연결이 되어있지 않은 두 삼각형 t_1, t_2가 있을 경우, 서로 이웃하는 정보를 가질 수 있도록 토폴로지를 수정한다. 연산자 *flip*은 $edge(t, i)$를 플립하여 t와 $tri(t, i)$의 토폴로지를 갱신한다.

mate(t_1, t_2):
  set *tri*(t_1, i_j) = t_2 and *tri*(t_2, i_k) = t_1,
  where *vert*(t_1, i_j) ∉ t_2 and *vert*(t_2, i_k) ∉ t_1

flip(t, i):
  let t_1 = *tri*(t, i), i_1 = *idx*(t_1, t), t_2 = *tri*(t, i+2),
    t_3 = *tri*(t_1, i_1+2).
  set *vert*(t, i+1) = *vert*(t_1, i_1), *vert*(t_1, i_1+1) = *vert*(t, i).
  mate(t, t_3), *mate*(t_1, t_2) *mate*(t, t_1).
  if *tri*(*vert*(t_1, i_1+2)) = t then set *tri*(*vert*(t_1, i_1+2)) = t_1.
  if *tri*(*vert*(t, i+2)) = t_1 then set *tri*(*vert*(t, i+2)) = t.

Fig. 4는 *flip* 연산의 시작과 끝 시점에서 토폴로지를 보여주고 있다. 연산 중에 삼각형이나 버텍스 객체의 생성과 삭제는 없으며, 기존 삼각형들의 토폴로지를 변경함으로써 *flip* 연산을 수행한다.

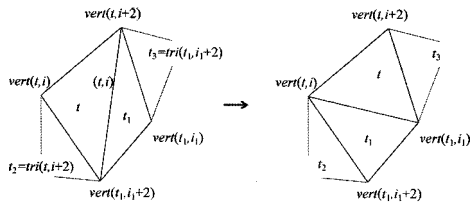

Fig. 4. Topology change in a *flip* operation.

3. 배열 데이터를 이용한 D&C 알고리즘

3.1 제안 알고리즘의 특징

제안하는 알고리즘은 *kd*-트리에 의한 이차원 공간 분할을 표현하기 위하여 실제로 트리를 구성하지 않고 배열 요소를 정렬한다. 이렇게 트리 데이터 구조를 위한 별도의 정보를 저장하지 않음으로써 메모리를 사용을 최소화한다. 또한 병합 과정이 병합 대상 삼각화의 삼각형이나 에지의 삭제 없이 플립 연산의 전파에 의해서 들로네 특성을 유지하며 진행된다. 이로써 성능의 저하 없이 D&C 삼각화의 구현을 매우 간단하게 할 수 있고, 계산과 구현 코드의 안정성을 기대할 수 있다. 제안하는 삼각화 알고리즘은 플립 기반의 D&C 방법이라는 의미에서 FBD&C(Flip Based D&C)라고 명명되었다[15].

3.2 공간 분할

Fig. 5의 (b)와 (c), (d)는 (a)에 나타난 이차원 공간의 점 집합 V = {v_0, v_1,..., v_8}에 대하여 공간을 분할하여 그룹을 나누는 과정을 보여주고 있다. 공간의 분할은 영역에 두 개 이하의 점이 존재할 때까지 재귀적으로 반복된다.

FBD&C에서 공간을 분할하는 과정은 최적화된 (optimized 혹은 balanced) *kd*-트리를 구성하는 알고리즘[16]과 동일하다. 그러나 실제로 트리를 구성하는 것이 아니라, 배열로 저장된 점들을 최적화된 *kd*-트리의 순서로 정렬하여 permutation vector[17] 형태로 사용한다. 이로써 공간 분할 트리를 위해 별도로 추가되는 데이터 구조 없이 점들의 원래 배열 안에서 순서만으로 공간의 분할 정보가 구성된다.

Fig. 5(d)의 이차원 공간에서 최종 분할된 정보는 정렬된 점 배열 V_a = {v_2, v_5, v_3, v_1, v_0, v_6, v_7, v_4, v_8}의 형태로 정리되며, 배열 요소의 순서는 이들 점 집합을 최적화된 *kd*-트리로 구성했을 때, 중위 운행 (in-order traversal)하는 순서와 동일하다.

정렬된 배열 V_a의 요소들을 반씩 나누어 살펴보면 각각의 깊이(depth)에 해당하는 분할을 나타냄을 알 수 있으며 이 분할은 Fig. 5의 각 단계의 공간 분할과 일치함을 확인할 수 있다. 예를 들어 V_a를 좌우 방향으로 이등분한 V_{a1L}과 V_{a1R}은 각각 V_{a1L} = {v_2, v_5, v_3, v_1, v_0}, V_{a1R} = {v_6, v_7, v_4, v_8}이며 이는 Fig. 5(b)의 분할과 일치한다. 또한 V_{a1L}을 상하 방향으로 반으로 나눈 V_{a1L2B}과 V_{a1R2T}은 각각 V_{a1L2B} = {v_2, v_5, v_3}, V_{a1L2T} = {v_1, v_0}이며 이는 각각 Fig. 5(c)의 좌측의 아래, 위 영역과 일치한다. 즉, 정렬된 별도의 데이터 구조를 준비하지 않고 배열의 요소 순서만으로 최적화된 *kd*-트리와 같은 공간 분할정보를 나타낼 수 있다.

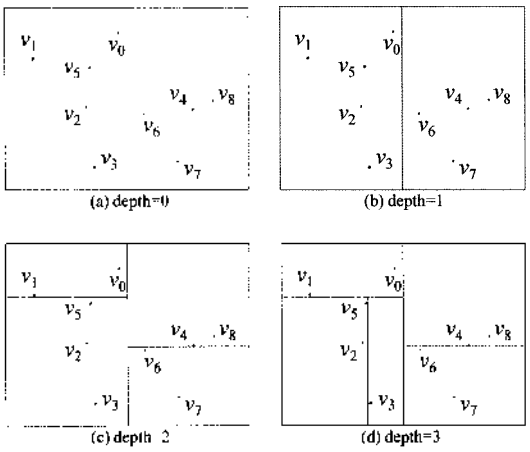

Fig. 5. Space partitioning with balanced *kd*-tree.

3.3 구성 및 병합

D&C 방법은 기본적으로 분할된 영역에 대해서 삼각화를 수행하고, 각 영역을 합하면서 삼각화를 병합한다. Fig. 6은 FBD&C의 병합과정을 보여준다. 분할

된 각각의 영역에서 삼각화를 수행한 후 영역을 통합하면서 삼각화를 병합하는 과정을 보여주고 있다.

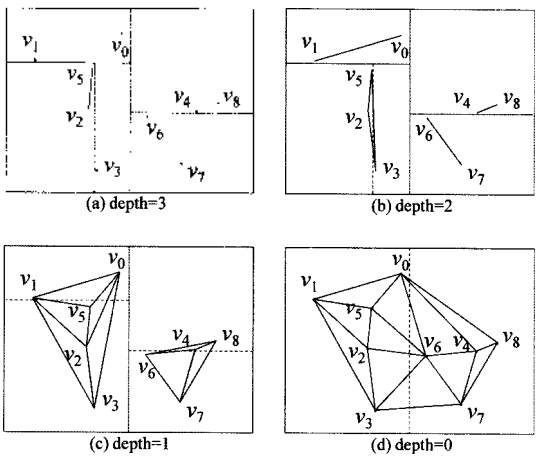

Fig. 6. Merging triangulations.

병합 과정은 두 삼각화 $T_L(V_L)$과 $T_R(V_R)$를 병합하여 $T(V)$를 구성하는 과정이다. $T_L(V_L)$과 $T_R(V_R)$은 들로네 삼각화이며 $T(V)$ 또한 들로네 삼각화이다. 기존의 D&C 알고리즘은 Fig. 7(a)와 같이 $T_L(V_L)$과 $T_R(V_R)$의 삼각형들이 병합 후에도 들로네 삼각형 특성을 만족할 수 있을지 검사한 후 만족시키지 못하는 삼각형들을 제거하고 두 삼각화를 병합한다. Fig. 7(b)는 $T_L(V_L)$과 $T_R(V_R)$의 일부 에지가 제거된 후 병합하는 과정을 보여주고 있다. 이러한 동작의 구현을 위해서 에지 단위의 연산이 가능한 데이터 구조가 필요하며 따라서 대부분의 D&C 알고리즘들에서 quad-edge 기반의 데이터 구조를 사용하고 있다.

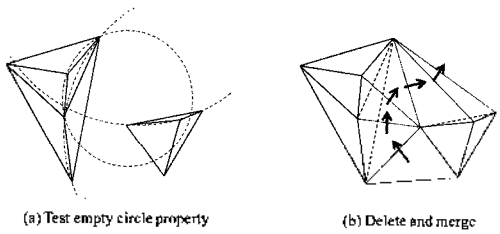

Fig. 7. Conventional merge process.

FBD&C 알고리즘은 2절에서 제안한 배열 기반의 간단한 데이터 구조를 이용하여 삼각화를 수행한다. 이때 데이터 구조 내에서 에지 정보를 가지고 있지 않기 때문의 기존의 방법과는 다른 방식으로 병합을 수행한다. 전통적인 방식[3,4]과는 달리 $T_L(V_L)$과 $T_R(V_R)$ 간의 empty circle 특성을 미리 검사하지 않고 일단 병합을 수행한다. Fig. 7의 예와 비교하면 Fig. 7(a)의 단계가 생략되고 (b)의 단계에서 바로 병합을 시작하는 것이다.

기존 삼각형의 삭제없이 일단 병합이 바로 시작되고 두 $T_L(V_L)$과 $T_R(V_R)$ 사이에 새로운 삼각형이 생성되면, 새 삼각형이 $T_L(V_L)$ 혹은 $T_R(V_R)$과 인접한 에지를 대상으로 empty circle 특성을 조사하고, 이를 만족하지 못하면 그 에지를 시작으로 기존 삼각화 $T_L(V_L)$ 혹은 $T_R(V_R)$ 중 한 방향이나 양쪽 방향으로 플립의 전파가 수행된다. 이는 점진적 구성(incremental construction) 방법에서 한 점이 삼각형의 내부에 추가되었을 때 수행되는 플립의 전파[1]와 유사하며 보다 자세한 내용은 [15]에 기술되어 있다.

4. 성능 시험

FBD&C 의 평균 수행시간 분석을 위하여 랜덤 생성 포인트의 수를 변화시키면서 삼각화의 수행 속도를 측정하였다. 데이터 구조 및 알고리즘은 C++ 언어로 구현하고, 매번 같은 랜덤 포인트 셋을 얻기 위하여 랜덤 시드를 고정하여 랜덤 포인트를 생성하였다. Fig. 8은 2.4 GHz의 Intel Core2 Quad CPU와 3 GB 메모리를 가진 Windows XP SP 3 운영체제의 데스크탑 PC 상에서 수행한 결과이다.

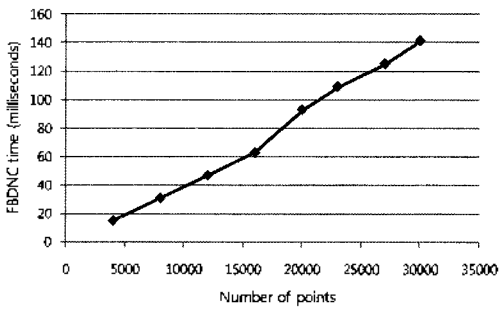

Fig. 8. Running time of FBD&C on 2.4 GHz CPU.

Fig. 8의 실행 결과를 보면 사이트의 수 n이 늘어남에 따라서 수행 시간 $T(n)$은 선형으로 증가함을 볼 수 있다. 수행 시간을 보면 20,000 포인트에 대한 삼각화 수행 시간이 93 ms, 즉 0.093초이며, 30,000 포인트에 대한 삼각화 시간이 0.141초이다. 예전의 연구 결과들이 수행된 컴퓨팅 환경과는 다르고, 다른 삼각화 방법을 같은 컴퓨터에 구현 및 수행하지 않아서 공정한 비교는 어렵다. 다만 사이트 수에 대해서 선형 증가 함을 고려하여, 보다 많은 사이트에 대해서 2000년에 Convex C3 슈퍼 컴퓨터에서 수행한 Lemarie의 결과에 비교해 본다면 느리지 않거나 더 빠르다고 판

단되는데, 구현의 계산의 안정성을 좀 더 보완해서 100만 사이트 이상의 수행 결과를 가지고 비교해 볼 필요가 있을 것으로 사료된다.

Fig. 9는 n의 수를 100에서 1,000까지 100씩 변화시키면서 랜덤 분포사이트에 대한 일차원 분할 및 이차원 분할의 D&C를 수행함에 있어서 플립연산이 일어나는 총 회수를 세어본 결과이다. 가로 축은 사이트의 개수를 나타내고 세로 축은 해당 사이트에 대한 삼각화를 구성할 때의 플립 연산이 일어나는 회수이며 이차원 공간분할의 경우 플립 회수가 월등히 적음을 보여주고 있다.

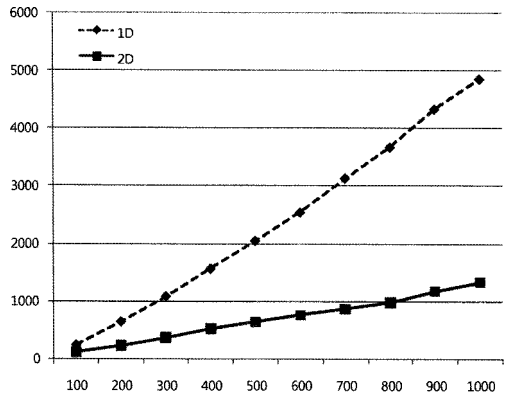

Fig. 9. Counting flip occurrences.

5. 결 론

배열 기반의 자료 구조는 선행 연구[14]에서 CDT를 위한 점진적 구성 방법에 사용하기 위하여 제안되었으며 본 논문에서는 이러한 자료구조를 이용하여 D&C 방법으로 삼각화를 수행하는 과정을 설명하였다. 배열 기반 자료 구조는 가장 기본적인 요소인 점과 삼각형의 최소한의 연결 관계만을 기술하기 때문에 매우 컴팩트하게 정의된다. 자료 구조 자체는 기본적이고 간결하지만 삼각화에 필요한 모든 정보를 제공할 수 있도록 토폴로지의 조회와 순회, 삼각화의 수정 등에 필요한 연산자들이 정의된다.

FBD&C는 일반적으로 수행 효율은 좋지만 구현이 복잡하다고 알려져 있는 D&C의 단점을 보완하여 쉬운 구현이 가능하도록 하는 알고리즘이다[15]. 제안 알고리즘이 기존의 D&C방법과 다른 점은 크게 두 가지로 요약할 수 있다. 첫 번째, 공간을 적응적으로 이차원 분할하며 분할된 공간 정보는 순열벡터 안에서 버텍스의 순서만으로 표현된다. 따라서 공간 분할을 위한 별도의 정보를 따로 가지지 않으면서 kd-트리가 가지는 모든 관계 검색의 기능을 수행할 수 있다. 이를 이용하여 병합시 공통 컨벡스를 찾는 과정에서 효율을 향상 시키는 데도 도움이 된다. 또한 적응적 이차원 공간 분할은 D&C의 병합 과정에서 일차원 분할에 비해서 보다 적은 수의 토폴로지 변화를 가져오므로 수행 속도 향상에도 도움이 된다. 두 번째, 병합 과정에서 에지의 삭제 추가가 일어나지 않고 플립에 의해서 들로네 특성을 유지한다. 이로 인해서 수행 속도를 떨어뜨리지 않고 간단하고 쉬운 구현이 가능하게 되었다.

FBD&C는 수행 속도 분석에 있어서 아직 lower-bound의 정교한 분석이 이루어지지 않았고, 다른 삼각화 알고리즘과의 실험을 통한 비교가 이루어지지 않았는데 향후 연구에서는 이러한 점들이 보완되어야 할 것이다. 또한 현재 구속조건 들로네 삼각화(Constrained Delaunay Triangulation)가 아닌 일반적인 들로네 삼각화를 수행하는데, 향후 연구를 통해서 FBD&C에서 CDT를 지원할 수 있도록 확장될 것이다.

후 기

이 논문은 2007년도 중앙대학교 학술연구비 지원에 의해서 수행되었습니다.

참고문헌

1. O'Rouke, J., *Computational Geometry in C 2nd Ed.*, Cambridge University press, 1998.
2. Shamos, M. I. and Hoey, D., "Closest-point Problems", *Proceedings of the 16th Annual IEEE Symposium on FOCS*, pp. 151-162, 1975.
3. Lee, D. T. and Schachter, B. J., "Two Algorithms for Constructing a Delaunay Triangulation", *Int. J. Comput. Inf. Sci.*, Vol. 9, pp. 219-242, 1980.
4. Guibas, L. and Stolfi, J., "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", *ACM Transactions on Graphics*, Vol. 4, No. 2, pp. 75-123, April 1985.
5. Ohya, T., Iri, M. and Murota, K., "A Fast Voronoi-diagram Algorithm with Quaternary Tree Bucketing", *Information Processing Letters*, Vol. 18, pp. 227-31, 1984.
6. Maus, A., "Delaunay Triangulations and the Convex Hull of n Points in Expected Linear Time", *BIT 24*, pp. 151-163, 1984.
7. Dwyer, R. A., "A Simple Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations in O(n log log n) Expected Time", *ACM Symposium*

on Computational Geometry, pp. 276-284, 1986.
8. Finkel, R. A. and Bentley, J. L., "Quad Trees: A Data Structure for Retrieval on Composite Key", *Acta Informatica*, Vol. 4, No. 1, pp. 1-9, 1974.
9. Adam, B., Elbaz, M. and Spehner, J. C., "Construction du diagramme de Delaunay dans le plan en utilisant les mélanges de tris", *Quatrièmes Journées de l'AFIG, Dijon*, pp. 215-223, 1996.
10. Lemaire, C. and Moreau, J. M., "A Probabilistic Result on Multi-dimensional Delaunay Triangulations, and Its Application to the 2D Case", *Computational Geometry*, Vol. 17, No. 1-2, pp. 69-96, October 2000.
11. Katajainen, J. and Koppinen, M., "Constructing Delaunay Triangulations by Merging Buckets in Quadtree Order", *Fundamenta Informaticae*, Vol. 11, No. 3, pp. 275-288, 1988.
12. Boissonnat, J. D. and Teillaud, M., "On the Randomized Construction of the Delaunay Tree", *Theoretical Computer Science*, Vol. 112, pp. 339-54, 1993.
13. Kim, D. S., Kim, D., Cho, Y. and Sugihara, K., "Quasi-triangulation and Interworld Data Structure in Three Dimensions", *Computer-Aided Design*, Vol. 38, No. 7, pp. 808-819, 2006.
14. Yang, S. W., Choi, Y. and Lee, H. C., "CAD Data Visualization on Mobile Devices Using Sequential Constrained Delaunay Triangulation", *Computer-Aided Design*, Vol. 41, No. 5, pp. 375-384, 2009.
15. 양상욱, "들로네 삼각화를 위한 간결한 알고리즘 및 CAD 가시화 적용 연구", 박사학위논문, 중앙대학교, 2008. 12.
16. Bentley, J. L., "Multidimensional Binary Search Trees Used for Associative Searching", *Communications of ACM*, Vol. 18, pp. 509-17, 1975.
17. Bentley, J. L., "Multidimensional Binary Search Tree in Database Applications", *IEEE Transactions on Software Engineering*, Vol. se-5, No. 4, pp. 333-340, 1979.
18. Ohya, T., Iri, M. and Murota, M., "Improvements of the Incremental Method for the Voronoi Diagram with Computational Comparison of Various Algorithms", *J. Oper. Res. Soc. Japan*, Vol. 27, pp. 306-336, 1984.
19. Anglada, M. V., "An Improved Incremental Algorithm for Constructing Restricted Delaunay Triangulations", *Computers & Graphics*, Vol. 21, No. 2, pp. 215-23, 1997.

양 상 욱

1998년 중앙대학교 기계설계학과 학사
2000년 중앙대학교 기계설계학과 석사
2000년~2003년 삼성SDS 솔루션개발센터 대리
2003년~2004년 이다이액솔루션즈 차장
2009년 중앙대학교 기계공학부 박사
2009년~현재 중앙대학교 미래신기술연구소 전임연구원
관심분야: 형상모델링, 모바일3D, 들로네 삼각화, CAD 데이터 교환, 웹기반 협력설계

최 영

1979년 서울대학교 기계설계학과 학사
1981년 KAIST 생산공학과 석사
1989년 Carnegie Mellon University 박사
1990년~1991년 KIST CAD/CAM 연구실 선임연구원
2001년~2002년 미국 국립표준연구소(NIST) 객원연구원
1992년~현재 중앙대학교 기계공학부 교수
관심분야: 형상모델링, Physically based modeling, 웹기반 협력설계, 설계방법론, 보로노이 다이어그램