한국 CAD/CAM 학회 논문집
제 10권 제 3호 2005년 6월 pp. 168-178

기술논문

3차원 8분할 Delaunay 삼각화 알고리즘 개발

박시형*, 이성수**

Development of Delaunay Triangulation Algorithm Using Oct-subdivision in Three Dimensions

S.H. Park* and S.S. Lee**

ABSTRACT

The Delaunay triangular net is primarily characterized by a balance of the whole by improving divided triangular patches into a regular triangle, which closely resembles an equiangular triangle. A triangular net occurring in certain, point-clustered, data is unique and can always create the same triangular net. Due to such unique characteristics, Delaunay triangulation is used in various fields., such as shape reconstruction, solid modeling and volume rendering. There are many algorithms available for Delaunay triangulation but, efficient sequential algorithms are rare. When these grids involve a set of points whose distribution are not well proportioned, the execution speed becomes slower than in a well-proportioned grid. In order to make up for this weakness, the grids are divided into sub-grids when the sets are integrated inside the grid. A method for finding a mate in an incremental construction algorithm is to first search the area with a higher possibility of forming a regular triangular net, while the existing method is to find a set of points inside the grid that includes the circumscribed sphere, increasing the radius of the circumscribed sphere to a certain extent. Therefore, due to its more efficient searching performance, it takes a shorter time to form a triangular net than general incremental algorithms.

Key words : Delaunay triangulation, Incremental construction algorithm, Triangle patch, Uniform grid, Subdivision algorithm

1. 서 론

삼각망은 컴퓨터 하드웨어 기술발전에 따라 다양한 분야에서 확장되고 응용되어 사용되고 있다. 주로 계산 기하학(computational geometry), 야금학, 지리정보 시스템, 가상현실, 컴퓨터를 이용한 시각화, 역공학(reverse engineering), 유한요소법, 솔리드 모델링, 볼륨 렌더링 등에서 이용된다. 평면을 규정하는데 있어 삼각형을 가장 많이 사용하는 이유는 세 개의 점이면 평면 결정이 가능하며 기하학적으로 가장 안정된 형태이기 때문이다.

삼각망을 생성하는 방법에 따라 Delaunay 삼각망 알고리즘을 이용한 것과 그렇지 않은 것으로 나눌 수 있는데, Delaunay 삼각망은 그 정의 상, 생성된 삼각망의 요소들이 대부분 내각이 등각에 가까운 정삼각형으로 생성되므로 삼각망을 구성하는 가장 기본적인 알고리즘으로 이용되고 있다. 3차원 Delaunay 삼각망은 2차원에 비해서 동일한 개수의 흩어진 점 데이터(scattered point)라도 더 많은 수행시간을 요구한다. 그러므로 효율적이지 못한 알고리즘을 사용하였을 경우, 3차원 Delaunay 삼각망을 생성하는 시간은 데이터의 양에 비례하여 기하급수적으로 증가한다. 따라서 고성능 컴퓨터가 등장하는 요즘에도 효율적인 알고리즘 개발이 필수적이다. 본 논문에서 제안하는 알고리즘은 현재까지 개발되었던 여러 Delaunay 삼각망 알고리즘들이 균등격자를 사용함으로써, 점의 분포가 균일하지 않은 점집합일 경우, 균일한 점집합보다 수행속도가 느려지는 단점을 보완하기 위해서 격자 내부에 점집합이 밀집해 있을 경우 격자를 좀 더 작은 하부격자로 분할(8분할)한다. 또, Incremental Construction 알고리즘에서 메이트를 찾는 방법으로, 기존의 방법인 외접원 또는 외접구의 반지름을 임의의 크기만큼 증가시켜 가면서 외접원을 포함하는 격자 내부의 점집합을 찾는 것이 아니라, 정삼각형 망이 생

*학생회원, 건국대학교 대학원 기계설계학과
**종신회원, 건국대학교 기계설계학과
- 논문투고일: 2004. 10. 18
- 심사완료일: 2004. 11. 26

성될 가능성이 높은 지역부터 우선 탐색하는 방법을 사용한다. 따라서, 일반적인 Incremental Construction 알고리즘 보다 높은 탐색효율로 삼각망의 생성시간을 단축할 수 있다.

2. 기존의 Delaunay 삼각망 생성 알고리즘

이전의 논문[1]에서 언급한 바와 같이 기존의 알고리즘으로는 Divide & Conquer[6,8] 알고리즘, Flipping을 이용한 알고리즘, Incremental Construction 알고리즘 등이 있으며 이외에도 다음에서 설명하는 알고리즘이 있다. 또한, 최근에 병렬컴퓨터의 발전으로 많은 병렬 알고리즘이 개발되고 있다.

2.1 Plane-sweep 알고리즘

산개된 데이터 점들이 놓인 평면위로 sweepline이 지나면서 Delaunay 삼각망을 생성하는 방법이다. 먼저, sweepline이 지나면서 해당 점들을 만나게 되면 그 점과 sweepline과 거리가 같은 곡선을 그리고 이미 생성된 곡선과 교점을 경계로 해서 sweepline을 여러 개의 구간으로 분할한 후에 분할된 sweepline상의 구간 중에 서로 이웃하는 구간을 찾아 선분을 만들면 Delaunay 삼각망이 완성된다. 2차원에 국한되어 사용된다.

2.2 병렬 알고리즘

최근에 많이 연구되고 있는 방법이다. 이 알고리즘은 순차적인 것이 아니라 병렬화할 수 있고 다점 시작이 가능하다는 장점이 있다. 알고리즘의 특성상 가장 좋은 효율을 낼 수 있으며 위에 언급된 다양한 알고리즘을 이용할 수 있다. 그러나 현재까지 개발된 병렬 알고리즘은 대부분이 특수한 경우에서만 수행 가능하고, 후에 병합과정을 통해 각각의 부분망들을 연결하여야 하며 부분망을 연결하는 병합과정 중 문제가 발생할 수 있기 때문에 실제로 효용성 있는 알고리즘은 많지 않아 추후에 많은 연구가 필요하다.

3. 본 논문에서 제안하는 Delaunay 삼각망 알고리즘

본 논문에서 제안하는 알고리즘은 Fang과 Piegl이 제안[2,3]한 균등격자를 이용한 Delaunay 삼각망에 기초를 두고 있다. 균등격자를 사용하면 특정 영역에 포함되는 기하요소를 짧은 시간에 찾을 수 있는 장점이 있다. 따라서, Delaunay 삼각망 생성 과정에서 이를 적용시키면 한 외접구 내부에 존재하는 점을 찾을 때 모든 점집합을 찾을 필요가 없어 Delaunay 삼각망의 구성속도를 크게 향상시킬 수 있다. 그러나, 균등격자를 이용한 알고리즘은 그 점집합이 균일할 때는 매우 효과적이나 점집합이 부분적으로 불 균일 할 때는 격자 내에 점이 하나도 없거나 또는 너무 많은 점들이 한 격자 안에 존재하기 때문에 효율성에 문제가 있다. 불규칙한 점군 데이터 안에서 이러한 문제가 더욱 심하게 발생되어 필연적으로 수행시간의 저하로 나타난다. 따라서, 본 논문에서는 8분할(Oct-Subdivision)을 시행하여 하나의 격자 내에 4개 이상의 점들이 존재하는 경우 동일한 크기의 8개의 하부격자를 생성하여 각각의 격자를 기존의 격자의 하부격자로 추가함으로써 비어있는 구(empty sphere)를 만들거나 구 내부에 다른 점 데이터가 없는지 탐색할 때 이러한 하부격자를 이용하게 된다. 또한, 탐색영역을 모든 공간에서 순차적으로 찾는 것이 아니라 다양한 기존의 Delaunay 삼각망 패치들을 분석하여 영역을 분할하고 삼각망이 가장 많이 생성된 영역부터 우선 탐색하여 수행시간의 단축을 꾀한다. 특히 점집합의 수가 증가하면 더욱 효과적이다.

3.1 3차원 방법

3차원의 불규칙 점군에서 부터 신속하게 Delaunay 삼각형을 생성하는 문제는 2차원에서 보다 복잡하고 소요시간도 매우 길게 된다. 탐색하려는 부분이 평면이 아닌 공간이기 때문에 Incremental Construction 알고리즘과 같은 경우 수행 성능이 크게 저하하게 된다. 따라서, 삼각망 생성속도를 높이기 위해 3차원상의 삼각망을 생성하는 방법으로 이전의 논문에서 제시한 2차원에서 Delaunay 삼각망을 생성하는 방법을 확장하여 3차원에서 적용 가능한 알고리즘을 제안한다.

3.2 데이터 확장

3차원상의 점군을 (x_i, y_i, z_i) $i = 0, ..., n$라고 한다면, 메이트를 빨리 찾기 위한 방법으로 3차원 평면상에 균등격자를 생성한다. 균등격자의 간격은 다음과 같은 방법으로 구한다.

$$size = \sqrt[3]{\frac{(x_{max}-x_{min})(y_{max}-y_{min})(z_{max}-z_{min})}{N}} \quad (1)$$

x_{max}, x_{min}, y_{max}, y_{min}, z_{min}, z_{max} : x, y, z의 최대, 최소값

N : 점의 개수

size 변수를 위와 같이 정하게 되면 각각의 격자가 대략 하나의 점 데이터를 포함하게 된다.

임의의 점을 포함하고 있는 격자의 인덱스 번호를 구하는 방법은 다음과 같다.

$$grid_x = \frac{x_i - x_{\min}}{size}$$

$$grid_y = \frac{y_i - y_{\min}}{size}$$

$$grid_z = \frac{z_i - z_{\min}}{size}$$

$$i_cell = |grid_x + 1| \quad (2)$$

$$j_cell = |grid_y + 1| \quad (3)$$

$$k_cell = |grid_z + 1| \quad (4)$$

i_cell, j_cell, k_cell : grid의 *i, j, k*번째 index number

3.3 8분할

먼저 위의 수식을 이용하여 균등격자를 생성하고 나면 다음과 같이 전처리를 행한다.

격자 내부에 4개 이상의 점이 존재할 때(즉, 동일한 인덱스 번호를 가진 점이 4개 이상일 때) 동일한 크기의 8개 상자로 나눈다. 그리고 각각의 상자를 균등격자의 하부 격자로 등록한다. 이는 재귀함수를 통하여 격자 내부에 점집합의 개수가 4이하가 될 때까지 반복한다.

3.4 Delaunay 삼각망의 시작 선분 찾기

본 논문에서 제안하는 알고리즘은 어떠한 선분에서부터 출발하든지 문제가 없다. 그러나, 효율성을 위해 격자의 중앙에서부터 시작하면 보다 효과적이다. 중앙의 격자 인덱스를 $(i_{middle}, j_{middle}, k_{middle})$이라 한다면 다음과 같이 계산할 수 있다.

$$(i_{middle}, j_{middle}, k_{middle}) = \left(\frac{grid_{x_{\max}} - grid_{x_{\min}}}{2}, \frac{grid_{y_{\max}} - grid_{y_{\min}}}{2}, \frac{grid_{z_{\max}} - grid_{z_{\min}}}{2}\right) \quad (5)$$

$grid_{x_{\min}}, grid_{x_{\max}}, grid_{y_{\min}}, grid_{y_{\max}}, grid_{z_{\min}}, grid_{z_{\max}}$: *x, y, z* 격자 인덱스의 최소, 최대값

중앙의 격자 인덱스에 점 요소가 없다면 시작점 $(i_{middle} - n,\ j_{middle} - n,\ k_{middle} - n)$에서 종점, $(i_{middle} + n,\ j_{middle} + n,\ k_{middle} + n)$, ($n$은 1, 2, 3 ...)까지 영역을 확장해가면서 내부 점 요소를 찾는다.

첫 점 요소를 P_1이라 한다면 P_1을 찾고 이 점과 최단거리에 있는 다른 점을 찾아 시작 선분을 만든다. P_1을 포함하고 있는 격자 내부에 다른 점이 있다면 이 점을 가장 가까운 점 P_2로 하고 다른 점이 없다면 격자영역을 확장하여 최단 거리에 있는 P_2를 찾는다.

3.5 3차원 Delaunay 삼각망 생성

1. 중심이 되는 점을 구하고 (두 점의 좌표가 $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$라 한다면 이 중심점에서 선분을 이끌어 내어 길이가 선분 $\overline{P_1P_2}$ 의 $\sqrt{3}/2$ 이 되며 기울기가 선분 $\overline{P_1P_2}$ 와 수직인 선분을 임의로 하나 생성한다.

2. 1과 같이 선분을 생성한 후에 선분 $\overline{P_1P_2}$ 를 축으로 하여 위에서 생성한 선분을 1회전하여 이 선분이 통과하는 모든 영역을 제 1 탐색영역으로 선택한다. 만약 이 부분에 원하는 점 P_3가 존재하지 않을 경우 이 영역의 *x, y, z*방향으로 1씩 증가한 영역으로 확대하여 탐색한다.

3. 만일, 격자 내부에 하부격자가 존재할 경우엔 우선 탐색지역에서 가장 가까운 격자 부분을 먼저 탐색한다.

4. 위와 같은 검색을 실시하여 검색 영역 안에 점이 발견되면, P_1, P_2와 새로운 점 P_3를 대원으로 하는 구를 그리고 이 구를 포함하는 최소의 정육면체를 만든다.

그리고, 이 정육면체가 지나는 격자 내부의 점들을 검색하여 점 P_1, P_2, P_3 이외에 다른 점들이 Sphere 내부에 존재하는지 판정한다.

만약, 내부에 포함하는 점이 없으면 P_3는 선분 $\overline{P_1P_2}$ 의 메이트가 되어 첫 Delaunay 삼각형이 완성된다.

5. 만약 위 정육면체가 지나가는 격자에 다른 점이 발견되면 이점이 위에서 생성된 구 내부에 존재하는지 확인하고 만약 이 점이 구 내부에 존재한다면 즉시 그 점을 새로운 메이트로 갱신하고 새 메이트를 기준으로 하여 위와 같은 작업을 비어있는 구(Empty Sphere)를 만족할 때까지 반복한다.

6. Empty Sphere를 만족하면 최초의 Delaunay 삼각형이 생성된 것이다.

4. 사면체 생성 알고리즘

3.5절에서 설명한 것과 같은 방법으로 최초의 Delaunay 삼각형을 생성한 후에 이 삼각형을 기본으로 하여 Delaunay 사면체를 생성해야 한다.

본 논문에서 제안하는 방법은 다양한 점 분포를 가진 점 데이터에서 Delaunay 사면체를 생성해 본 후 Delaunay 사면체가 생성될 확률이 가장 높은 지역을 선별하여 이 지역을 우선적으로 탐색하는 방법이다.

Fig. 1은 Delaunay 사면체가 생성될 때 삼각평면 모서리의 평균 모서리 길이와 삼각형 중심부터 메이트가 되는 점 P_4의 거리관계를 나타낸 것이다. 균일분포, 정규분포의 2가지 분포를 가진 10만개의 점집합 중 10000개를 sampling한 것으로 각각 10번씩 테스트 하여 평균을 낸 것이다.

이에 따르면 삼각형의 모서리 평균길이와 삼각형 중심에서 사면체가 생성될 확률이 가장 높은 거리 비율은 삼각형의 평균모서리 길이가 L이라 하였을 때

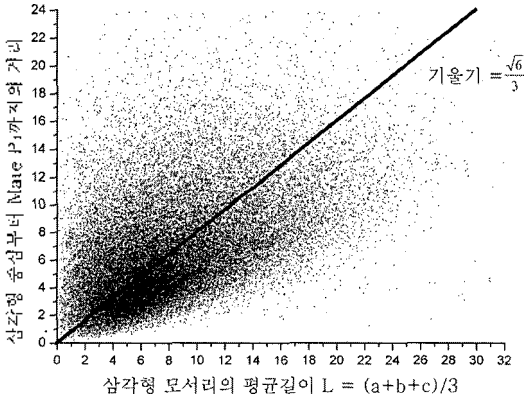


Fig. 1. Delaunay삼각화가 완성되었을 때 삼각평면 모서리의 평균길이와 삼각평면의 중심부터 메이트까지 거리관계.

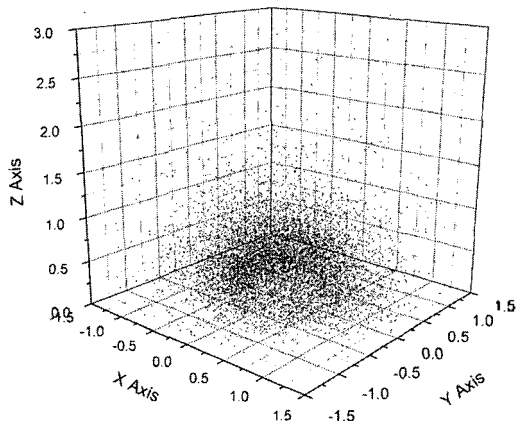


Fig. 2. Delaunay삼각화가 완성되었을 때 삼각평면 모서리의 평균길이와 삼각평면의 중심부터 메이트가 되는 점의 위치관계.

정사면체의 높이 $\frac{\sqrt{6}}{3}L$에 해당하는 영역이 높다(실제로는 이 거리의 80% 정도에 가장 밀도가 높으며 어떠한 점집합군도 이지역의 밀도가 타 지역 보다 높다). 따라서, 삼각형의 중심에서부터 $\frac{\sqrt{6}}{3}\times\frac{4}{5}L$의 거리만큼 떨어져 있는 격자영역을 우선 탐색한다. Fig. 2는 메이트를 구하려는 Delaunay삼각형의 모서리 길이가 1인 정삼각형으로 가정했을 때 이 삼각형의 무게중심에서 메이트까지의 위치관계를 나타낸 것으로 10만개의 점 집합 중 10000개를 random sampling한 것으로 각각 10번씩 테스트 하여 평균을 내었으며 무게 중심의 좌표는 (0, 0, 0)이 되도록 하였다.

4.1 사면체 탐색 알고리즘

위에서 살펴본 것처럼 메이트의 분포는 Delaunay 삼각형 중심에서 $\frac{\sqrt{6}}{3}\times\frac{4}{5}L$ 떨어진 지역을 중심으로 집중 분포하고 있으며(이 지점을 중심으로 하는 반경이 2L인 구 안에 메이트의 80%가 밀집되어 있다.) 이 지역에서 멀어질 수 록 삼각형 의 중심에 집중적으로 존재함을 알 수 있다. 따라서, 이 지역을 우선적으로 탐색하는 알고리즘을 작성하여 탐색효율을 높인다.

이 알고리즘에서 가장 먼저 수행해야 할 것은 현재 메이트를 구하고자 하는 Delaunay 삼각형의 법선벡터를 구하고 이 법선벡터가 x, y, z축 중 어느 축에 더 가까운지 판단한다. 예를 들어 법선 벡터값이 (1, −2, 7)이라면 z축 방향으로 가장 가깝다. 이런 경우에는 z축에 수직인 방향으로 탐색한다. 다음은 법선벡터가 z축 방향으로 기울어져 있을 때 메이트를 탐색하는 방법을 설명하였다(x, y축 방향은 각각 x축, y축에 수직인 방향으로 탐색하되 탐색순서는 z축 방향과 같다.).

탐색 순서

1. 삼각평면의 중심을 계산한다.
2. 삼각평면과 직각이 되는 법선을 구한다.
3. 삼각평면의 각각의 모서리 길이의 평균 $l_{ave}=a+b+c/3$을 구한 후 삼각평면의 중심에서부터 법선 방향으로 $\frac{4}{5}\times\frac{\sqrt{6}}{3}l_{ave}$만큼 떨어진 거리에 있는 격자를 계산한다.
4. 이 격자에 점 데이터가 있는지 탐색한다.
5. 이 격자에 점 데이터가 있으면 9번으로 간다.
6. 이 격자에 점 데이터가 없으면 Fig. 3과 Fig. 4에 설명한 것과 같이 탐색 우선순위를 정한다. 두 번째 탐색영역은 첫 번째 탐색영역의 주위이며 (Fig. 3에서는 앞쪽 영역도 탐색한다.) 이 부분에 점 데이터가 없으면 격자를 축방향으로 확대하며 탐색한다.

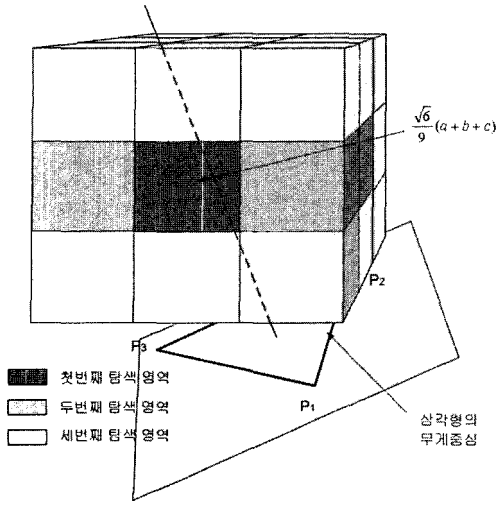


Fig. 3. 사면체 탐색 알고리즘 z축에 수직인 경우.

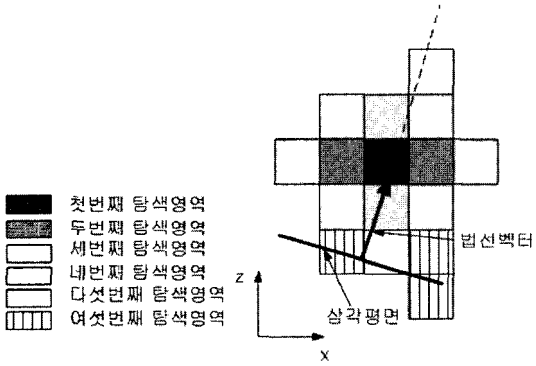


Fig. 4. 사면체 탐색 알고리즘 z축에 수직인 경우(수직 단면).

7. 위와 같은 방법으로 탐색하되 무게중심을 포함하는 격자가 탐색되면 Fig. 4에서 아래쪽 영역은 더 이상 탐색하지 않고 삼각형을 포함하는 경계 영역으로 탐색을 수행한다.
8. 6. 7.항에 해당하는 탐색을 수행하면 삼각평면의 바깥부분에 해당되는 영역을 모두 탐색하게 된다.
9. 탐색영역 내에 점 데이터가 존재하면 삼각평면의 꼭지점과 발견한 점을 포함하는 구를 생성한다. 구를 포함하는 최소격자상자를 계산하고 이 안에 존재하는 데이터 점들이 위에서 계산한 구 내부에 존재하는 지 판별한다. 격자 내부에 8분할되어 있는 영역이 있을 경우 축 부분에서 가까운 분할된 영역을 먼저 탐색한다.
10. 만약 최소격자상자 안에 존재하는 데이터 점들 중 구 내부에 존재하는 점이 있으면 이 점을 가지고 다시 사면체를 만들고 외접하는 구를 생성한 다음 9번을 반복한다.
11. Empty Sphere를 만족하는 점을 탐색할 때까지 계속한다.

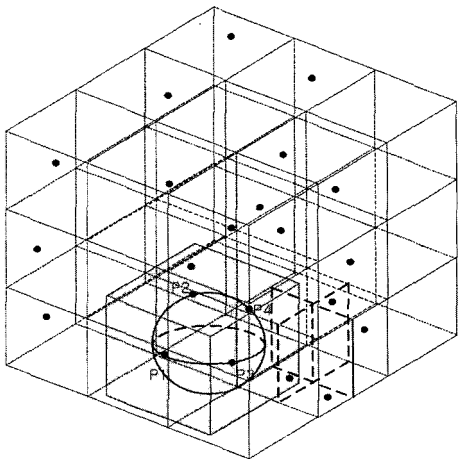


Fig. 5. 경계상자 안에 있는 격자 탐색(8분할 이용).

4.2 사면체 구성방법

3차원 Delaunay 삼각망 생성에서 가장 중요한 문제는 Delaunay사면체가 오류 없이 완벽한 형상으로 생성되어야 한다.

오류 없는 완전한 삼각망을 생성하기 위한 좋은 방법은 3차원 격자 내부에 최초의 데이터 점을 선택하고 이를 기준으로 하여 최초의 선분을 생성하며 선분을 모서리로 하는 최초의 삼각형을 만들고 이를 하나의 면으로 하는 사면체를 생성하는 것이 그 시작이다. 또한 이 사면체의 면을 기준으로 새로운 사면체를 쉽게 탐색하고 연결하여 추가할 수 있어야 하며 내부에 구멍이 생성되는 경우처럼 어떠한 기하학적인 오류도 발생해서는 안 된다.

최초의 면 리스트는 최초로 생성된 사면체로부터 시작된다. 최초로 생성된 사면체가 $P_1P_2P_3P_4$라면 다음과 같은 4개의 삼각형으로부터 면 리스트가 시작된다.

$\Delta P_3P_2P_1 \quad \Delta P_1P_2P_4 \quad \Delta P_2P_3P_4 \quad \Delta P_3P_1P_4$

모든 면은 오른손 법칙에 따라서 사면체의 바깥쪽으로 법선벡터가 향하도록 삼각형의 꼭지점 순서를 배치한다. 사면체의 면이 서로에게 영향을 주므로 면 리스트는 이중 원형 연결리스트로 만드는 것이 접근과 유지에 편리하다.

최초의 면 리스트가 생성되면 사면체구성은 다음과 같은 단계를 따른다.

1. 위의 면 리스트로부터 가장 우측의 면을 선택하여 Delaunay 사면체를 구성하는 4번째 점을 탐색한다.

2. 만약 점이 발견되면 새로 형성된 사면체가 이전에 만들어진 사면체와 접촉이 있는지 검사한다. 만약에 접촉이 있다면 몇 개의 면이 접촉했는지 판단한다.

3. 하나의 면이 접촉했다면 리스트에서 접촉한 하나의 면을 삭제한 후 2개의 새로운 면을 면 리스트에 추가한다. 두개의 면이 접촉했다면 리스트에서 접촉한 두개의 면을 삭제한 후 1개의 새로운 면을 면 리스트에 추가한다. 세 개의 면이 동시에 접촉했다면 리스트에서 접촉한 세 개의 면을 모두 찾아서 삭제한다. 면 접촉이 아닌 점접촉이나 선분접촉은 본 알고리즘에 영향을 주지 않으므로 면 리스트로부터 현재면을 삭제한 후 새로운 3개의 면을 리스트에 추가하면 된다.

3 접촉하는 부분이 없다면 면 리스트로부터 현재면을 삭제한 후에 새로운 3개의 면을 면 리스트에 추가한다.

3. 만약 점이 발견되지 않으면 현재면은 최 외곽의 경계면에 도달한 것이므로 현재면을 면 리스트에서 삭제한다.

4. 이러한 순서를 면 리스트에 면 데이터가 완전히 삭제될 때까지 반복한다. 예를 들어 최초의 면 리스트가

$\Delta P_1P_2P_3$ $\Delta P_1P_2P_4$ $\Delta P_2P_3P_4$ $\Delta P_3P_1P_4$

라고 한다면 현재면은 $\Delta P_3P_1P_4$이다.

이 현재면을 기준으로 새로운 점 P_5가 발견되어서 Delaunay 사면체가 완성된다면 다음과 같은 면 리스트로 갱신된다.

$\Delta P_1P_2P_3$ $\Delta P_1P_2P_4$ $\Delta P_2P_3P_4$ $\Delta P_1P_4P_5$
$\Delta P_4P_3P_5$ $\Delta P_3P_1P_5$

따라서 현재면은 면 리스트의 맨 뒤에 있는 $\Delta P_3P_1P_5$이 되며 위와 같은 과정을 면 리스트가 완전히 삭제될 때까지 반복한다.

다음은 이와 같은 과정을 pseudo code화 한 것이다.

첫 번째 점요소 P_1을 중앙격자에서 찾는다.

점요소 P_1과 가장 가까운 점 P_2를 찾는다.

P_1과 P_2 P_3를 대원으로 하였을 때 구 내부에 점요소를 포함하지 않는 점 P_3를 찾는다. 이 세 점이 최초의 삼각형이 된다.

현재의 삼각형으로부터 첫 번째 사면체를 찾는다.

(P_4 탐색)

면 리스트를 $P_3P_2P_1$, $P_1P_2P_4$, $P_2P_3P_4$, $P_3P_1P_4$로 초기화한다.

```
while(면 리스트에 값이 존재하는 동안){
 현재 면 리스트의 가장 마지막 값을 현재면으로 변경
  if(P4가 존재하면){
   if(발견한 P4가 이전에 사용된 값이면){
   접촉상태 평가;
              1. 한 개면 접촉이면 :
              현재 면을 리스트에서 삭제
              리스트에서 접촉된 면 리스트 값 삭제
              접촉하지 않은 두 개의 면추가
              2. 두 개면 접촉이면 :
              현재 면을 리스트에서 삭제
              리스트에서 접촉된 면 리스트 값 삭제
              접촉하지 않은 한 개의 면추가
              3. 세 개면 접촉이면
              현재 면을 리스트에서 삭제
              리스트에서 접촉된 면 리스트 값 삭제
              4. 점접촉 또는 선분접촉이면
              현재 면을 리스트에서 삭제
              새로운 세 개의 면 리스트 추가
}else{
현재 면 리스트 삭제
새로운 세 개의 면 리스트 추가
P4에 사용됨 플래그 표시
 }
 }else{ // 경계면 도달
현재 면 리스트 삭제
 }
}
```

4.3 테스트 환경

본 알고리즘을 테스트 하기 위하여 다음과 같은 환경 하에서 프로그램을 개발하고 테스트를 진행하였다.

Table 1. 시스템 환경

Hardware	Pentium 4 2.4 GHz
운영체계	Windows2000 Professional
Language	Visual C++ 6.0
Graphic library	OpenGL 1.2

5. 수행 결과

본 논문에서 제시한 알고리즘을 이용한 Delaunay

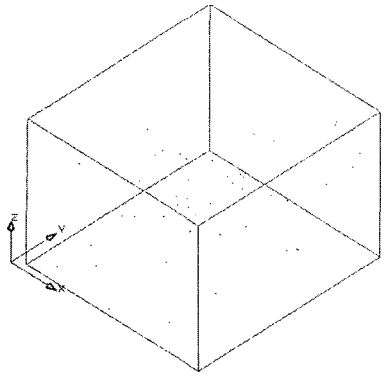


Fig. 6. 50개의 점군으로 이루어진 집합.

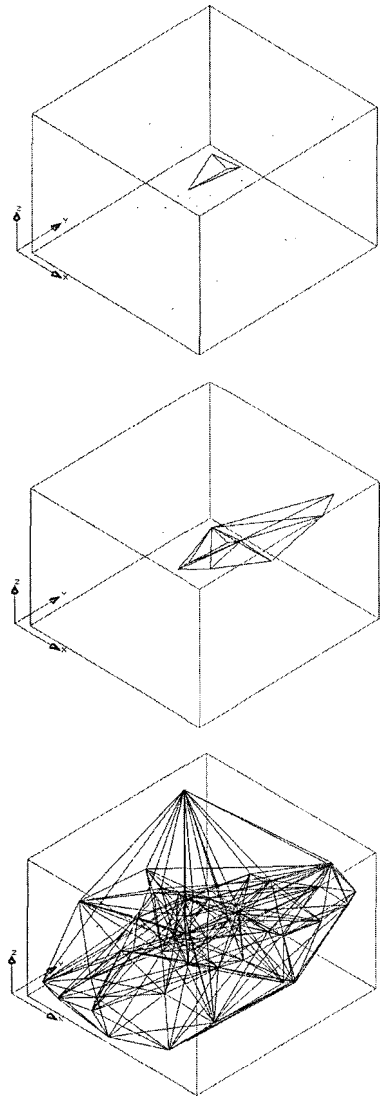


Fig. 7. Fig. 6의 점집합을 삼각화 하는 과정.

삼각망과 기존의 Delaunay 삼각망 알고리즘과 비교한 결과 다음과 같은 결과를 얻었다.

Fig. 6은 50개의 점군으로 이루어진 집합을 표현하였으며 Fig. 7은 Fig. 6의 점집합을 이용하여 Delaunay 삼각화 하는 과정을 단계별로 표현하였다. 첫 번째 그림은 최초의 사면체를 생성해 낸 것이며 두 번째 그림은 최초의 사면체에서 Delaunay 삼각망을 5단계 진행한 모습이다. 그리고 마지막은 Delaunay 삼각망을 완성한 것을 보여주고 있다.

Fig. 9는 점집합이 균일분포일 때 패치를 수행한 것이며 Fig. 8은 초기 입력 값이다. Fig. 11, Fig. 13는 x, y, z축 상에 독립적인 정규분포, x, y, z축 상에 종속적인 정규분포일 때 패치를 수행한 것으로 점집합의 개수는 모두 5000개이다. Fig. 14, Fig. 15, Fig. 16에서는 점집합의 수가 각각 5000, 10000, 20000, 40000개로 이루어진 임의의 점 데이터를 삼각화 했을 때 수행성능을 평가한 것으로 본 논문에서 제안한 알고리즘(8분할방법 + 최우선탐색법)을 이용한 탐색시간(점집합 5000개)을 1이라고 하였을 때 균일격자법

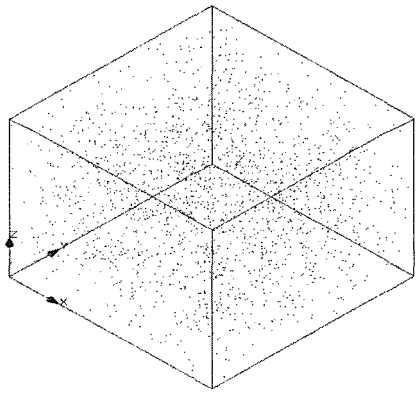


Fig. 8. 균일분포된 점집합(n = 5000).

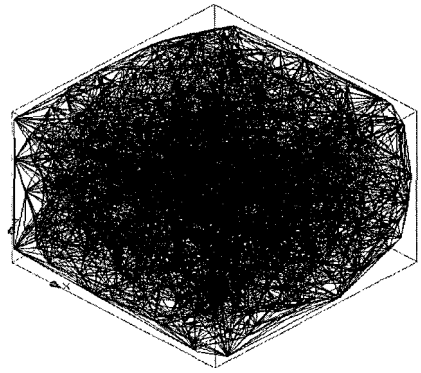


Fig. 9. 3차원 Delaunay삼각망(균일분포).

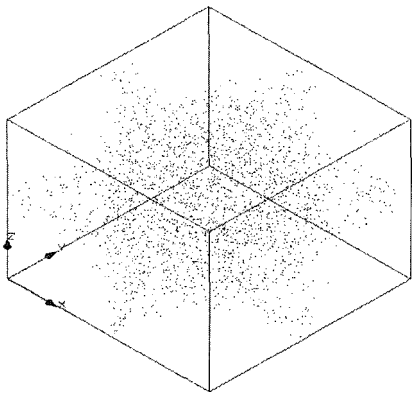


Fig. 10. x, y, z축에 서로 독립적인 정규분포 적용($n = 5000$).

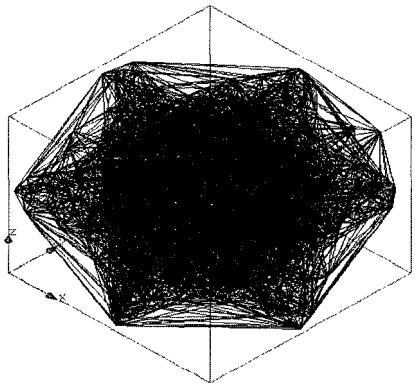


Fig. 11. 3차원 Delaunay삼각망(x, y, z축에 독립된 정규분포).

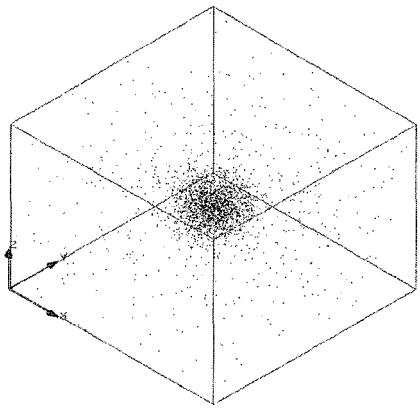


Fig. 12. 정규분포 된 점집합($n = 5000$).

+ 최우선 탐색법을 이용한 것과 Fang이 제안한 균일격자법 + 터널 탐색법, 균일격자법 + Box 기초탐색법의 삼각망 생성속도의 상대 비교를 나타내었다. 터널 탐색법은 전형적인 Delaunay 사면체에서 메이트가 이전에 생성된 삼각형 평면의 중심을 지날 확률이 높다는 관찰에서 시작한 방법으로 삼각형 중심에서 시작하여 $+x$, $-x$, $+y$, $-y$, $+z$, $-z$의 탐색방향으로 값을 증가시키면서 탐색한다. 그리고 축방향의 영역은 다른 영역보다 더 빠르게 확장시키는 방식이다.

Box 기초 탐색법은 삼각형 면의 경계상자를 각각의 방향으로 확장하면서 Delaunay점을 찾는 방식이다.

점 데이터가 균등분포를 나타낼 때 생성속도는 균일격자법을 이용한 방법이 8분할격자를 이용한 방법보다 3~4% 더 뛰어난 성능을 보였다(Fig. 14 참조). 이러한 이유는 균일격자 내에서 8분할을 위한 전처리 작업을 수행하였으나 점 데이터가 고르게 분포하였기 때문에 8분할의 이점을 살리지 못하였기 때문이다. 이에 비해 불균일한 점집합(Fig. 15, Fig. 16)을 입력받았을 때에는 8분할을 이용한 최우선 탐색방법이 균일격자법 보다 평균 25% 더 효율이 뛰어났다. 이는 점 데이터가 집중되어 있는 격자일 경우 8분할을 하게 되면 탐색범위를 효율적으로 줄일 수 있을 뿐 아니라 최우선탐색법의 효율적인 탐색영역을 이용할 수 있기 때문이다. 이는 점집합의 분포가 일정하지 않은 점군에서 균일 격자법보다 더 뛰어난 성능을 보임을 알 수 있다. 또한, Fig. 14, Fig. 15, Fig. 16의 결과에서 보는 것과 같이 점집합의 수가 증가하더라도 8분할 + 최우선 탐색법을 사용하여 삼각망을 생성하는 생성시간이 선형적으로 비례함을 알 수 있다(실제로 100만개 이상의 대단위 점집합에 대해서도 선형성을 확인하였다.). 일반적인 Random Incremental Construction 알고리즘이나 Flipping 알고리즘 등의 실제 생성시간이 비선형적인 것을 고려해 보면(최악 수행시간이 아닌 평균적인 생성시간은 2차원에 근접한다.) 점집합의 개수가 크다면 이 생성시간의 차이는 크다. 따라서, 점집합의 개수가 큰 삼각망 형성에 효과적이다.

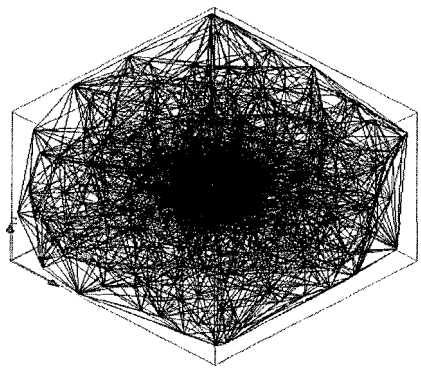


Fig. 13. 3차원 Delaunay삼각망(정규분포).

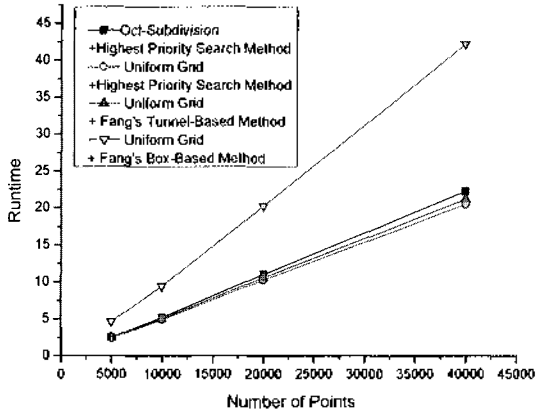


Fig. 14. 3차원 Delaunay 삼각화 알고리즘의 속도(Uniform Data).

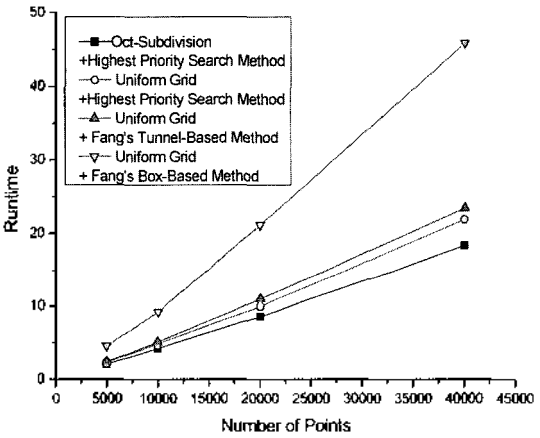


Fig. 15. 3차원 Delaunay 삼각화 알고리즘의 속도(Normal Distribution Data(x, y mutual independence)).

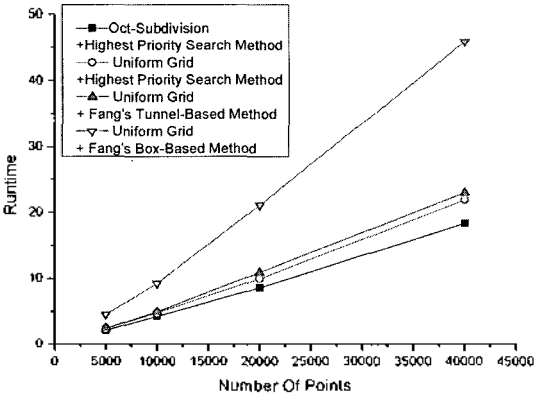


Fig. 16. 차원 Delaunay 삼각화 알고리즘의 속도 (Normal Distribution Data(x, y mutual dependence)).

6. 결 론

균일격자를 사용하면 특정 영역에 포함되는 기하요소를 짧은 시간 안에 찾을 수 있는 장점이 있다.

본 논문에서 제시한 알고리즘을 이용하여 3D Delaunay 삼각망을 적용할 경우 Delaunay 사면체의 정의에 따른 삼각망을 생성하기 때문에 비 Delaunay 삼각망은 발생하지 않는다. 또한, 삼각망의 생성방법으로 볼 때 항상 Convex Hull 조건을 만족하며 구멍과 같은 패치 오류가 발생하지 않는다.

그러나, 3D Delaunay 삼각망의 경우 2D Delaunay 삼각망 보다 탐색해야 하는 범위가 평면격자가 아닌 공간의 큐브이므로 타 알고리즘과 마찬가지로 수행시간의 저하를 피할 수 없다.

본 논문에서는 2차원과 3차원 상에서 점집합이 주어질 경우 균일격자를 생성하고 각각의 균일격자 마다 특정 수 이상의 점 데이터가 존재할 경우 하부격자로 분할함으로써 주어진 점집합을 빠르게 패치하여 삼각화 하는 Delaunay 삼각망 알고리즘을 제안하였다. 기존의 균일격자를 사용하면 특정 영역에 균일하게 분포한 점집합을 짧은 시간 안에 찾을 수 있는 장점이 있다. 그러나, 균일격자를 이용한 Delaunay 삼각망은 점집합 중 일부가 특정 위치마다 집중으로 모여 있어 불규칙하면 특정 격자 내부에 점 데이터가 밀집되므로 밀집된 점을 모두 검색하여야 한다. 따라서, 격자를 분할하여 얻는 탐색범위의 축소이점을 제대로 얻지 못하게 된다.

기존 균일격자의 평균검색시간은

$O(GT(n) + n \times SearchMateU())$이며 여기서 $GT(n)$은 n개의 점집합에서 격자를 생성하는 전처리 시간이고 $SearchMateU()$는 균일격자에서 메이트를 찾는 시간으로 x, y, z 방향의 우선탐색 범위의 평균 모서리 길이를 각각 a, b, c라고 하면 $n^3 \times \frac{a}{x_{res}} \times \frac{b}{y_{res}} \times \frac{c}{z_{res}}$가 된다.

본 논문에서 제안한 8분할을 시행하면 데이터가 집중된 격자 내부를 하부격자로 다시 세분화하기 때문에 검색할 때에 우선 검색 범위를 그만큼 줄일 수 있어 메이트를 찾는 시간을 감소시킨다. 이 알고리즘의 평균검색시간은 $O(GST(n) + n \times SearchMateS())$로 표현할 수 있다.

여기서, $GST(n)$는 n개의 점집합에서 균일격자를 생성하고, 8분할을 실행하는 시간이며 $SearchMateS()$는 8분할된 격자에서 메이트를 찾는 시간으로 x, y, z 방향의 우선탐색 범위의 평균 모서리 길이를 각각 a, b, c라고 하고 하부격자로 분할함으로써 탐색범위 한쪽

방향에서 제외되는 모서리 길이를 각각 a', b', c' 각각의 모서리방향에서 제외되는 전집합의 개수를 n'이라 한다면 탐색 시간은 $n^3 \times \frac{a}{x_{res}} \times \frac{b}{y_{res}} \times \frac{c}{z_{res}} - n'^3 \times \frac{a'}{x_{res}} \times \frac{b'}{y_{res}} \times \frac{c'}{z_{res}}$ 가 된다.

불균일한 점집합에서 메이트를 검색하는 시간은 *SearchMateS*()가 *SearchMateU*()보다 뛰어나므로 전처리 과정에서 $GT(n)$보다 상대적으로 많은 시간을 요하는 $GST(n)$의 수행속도를 고려하더라도 전체 성능은 8분할이 균일격자법보다 2차원인 경우 평균 19%의 성능향상이 3차원인 경우 25%의 성능향상이 있다.

또한, 탐색범위를 Fang이 제안한 터널 탐색법 대신에 본 논문에서 제안한 최우선 탐색법을 이용함으로써 Delaunay삼각형 또는 사면체가 발생할 확률이 높은 지역을 최우선 탐색하여 더욱 효과적으로 Delaunay 삼각망을 생성할 수 있다.

따라서, 본 논문에서 제안한 삼각화 알고리즘을 적용할 경우 삼각망 생성시간은 속도향상기법을 적용하지 않은 Incremental Construction 알고리즘(수행시간 이차원인 경우 $O(n^2)$, 삼차원인 경우 $O(n^3)$)과 비교해 큰 시간향상이 있을 뿐 아니라 일반적인 균일격자와 비교할 경우에도 삼각망 생성시간의 향상을 가져올 수 있다.

감사의 글

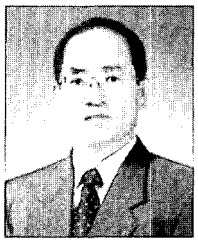
이 논문은 2002학년도 건국대학교 학술진흥연구비 지원에 의한 논문임.

참고문헌

1. 박시형, 이성수, "분할 Delaunay 삼각화 알고리즘 개발", CAD/CAM학회, Vol. 7, No. 4, pp. 248-253, 2002.
2. Tsung-Pao Fang and Les A. Piegl, "Delaunay Triangulation Using a Uniform Grid", *IEEE Computer Graphics and Applications*, Vol. 13, No. 3, pp. 36-47, 1993.
3. Tsung-Pao Fang and Les A. Piegl, "Delaunay Triangulation in Three Dimensions", *IEEE Computer Graphics and Applications*, Vol. 15, No. 5, pp. 62-69, 1995.
4. Michael J. Laszlo, "Computational Geometry and Computer Graphics in C++", Prentice Hall, 1996.
5. Ding-Zhu Du and Frank Hwang, "Computing in Euclidean Geometry", World Scientifc, 1995.
6. Rex A. Dwyer, "A Faster Divide and Conquer Algorithm for Constructing Delaunay Triangulations", *Algorithmica*, Vol. 2, pp. 137-151, 1987.
7. David P. Dobkin and Michael J. Laszlo, "Primitives for the Manipulation of Three Dimensional Subdivisions", *Algorithmica*, Vol. 4, pp. 3-32, 1989.
8. P Cignoni, C Montani and R Scopigno, "DeWall : A Fast Divide and Conquer Delaunay triangulation algorithm in E^b", *Computer-Aided Design*, Vol. 30, No. 5, pp. 333-341, 1998.
9. Ding-Zhu Du and Frank Hwang, "Computing In Euclidean Geometry", *World Scientific*, pp. 225-263, 1998.
10. Bowyer, A., "Computing Dirichlet Tessellations", *The Computer J.* , Vol. 24, No. 2, pp. 162-166, 1981
11. 박형준, "측정데이터로부터 형상 재구성을 위한 곡면 모델링에 관한 연구", 한국과학기술원, 박사학위 논문, 1996.
12. 조송래, "개선된 병렬 3차원 딜러니 삼각화 알고리즘", 포항공대, 석사학위 논문, 1997.
13. Lee, S., Park, C. I. and Park, C. M., An Improved Parallel Algorithm for Delaunay Triangulation on Distributed Memory Parallel Computers, In Proceedings of the 1997 Advanced in Parallel and Distributed Computing, Shanghai, China, May, 1997.
14. 김동욱, 김덕수, 조동수, 조영송, Kokichi sugihara., "점 집합의 보로노이 다이어그램을 이용한 원집합의 보로노이 다이어그램의 계산: I. 위상학적 측면", CAD/CAM학회, Vol. 6, No. 1, pp. 24-30, 2001.
15. 김동욱, 김덕수, 조동수, 조영송, Kokichi sugihara., "점 집합의 보로노이 다이어그램을 이용한 원집합의 보로노이 다이어그램의 계산 : II. 기하학적 측면", CAD/CAM학회, Vol. 6, No. 1, pp. 31-39, 2001.

박 시 형

1996년 건국대학교 기계설계학과 졸업
1998년 건국대학교 기계설계학과 공학석사
2005년 건국대학교 기계설계학과 공학박사
관심분야: CAD/CAM, Delaunay Triangulation, Surface Reconstruction, 전문가 시스템, 자동차 브레이크 시스템



이 성 수

1981년 한양대학교 정밀기계공학과 졸업
1987년 일본 NAGOYA(名古屋)대학 대학원(기계공학과) 공학석사
1991년 일본 OSAKA(大阪)府立대학 대학원(기계공학과) 공학박사
1991년~1993년 삼성전자 정보통신 종합연구소(CAD팀) 수석연구원
1995년~ 건국대학교 공과대학 기계항공공학부 교수
관심분야: 설계검증시스템, CAD데이터의 활용 및 이용, CAD/CAM, 기계시스템의 지능화, 설계 지원 S/W의 연구