한국공작기계학회 2000년도 추계학술대회 논문집 pp.151~156

Quad-Subdivision을 이용한 Delaunay 삼각화 알고리즘 개발

Development of Delaunay triangulation algorithm using quad subdivision

박시형*, 이성수**

* 건국대학교 대학원 기계설계학과 (E-mail : parksh@konkuk.ac.kr)
** 건국대학교 공과대학 기계설계학과 (E-mail : sslee@konkuk.ac.kr)

ABSTRACT

Delaunay triangulation is well balanced in the sense that the triangles tend toward equiangularity. And so, Delaunay triangulation hasn't some slivers triangle. It's commonly used in various field of CAD applications, such as shape reconstruction, solid modeling and volume rendering. In this paper, an improved Delaunay triangulation is proposed in 2-dimensions. The suggested algorithm subdivides a uniform grids into sub-quad grids, and so efficient where points are non-uniform distribution. To get the mate from quad-subdivision algorithm, the area where triangulation-patch will be most likely created should be searched first.

Key Words : quad subdivision, Delaunay triangulation, incremental construction algorithm, triangle patch, uniform grid

1. 서 론

평면 또는 삼차원 상에 존재하는 불규칙한 점들을 이용하여 삼각분할을 생성하는 방법은 현재 CAD/CAM, Computer Graphics분야에서 중요한 연구과제 중 하나로 인식되고 있다.

공간상의 점들을 연결하는 문제에 대한 접근은 19세기 Gauss와 Dirichlet에 의해 처음 시도되었으며 20세기 초 Voronoi와 Delaunay에 의해 Delaunay triangulation이 정의되었다.

이러한 Delaunay triangulation은 Dual인 Voronoi diagram과 함께 연구되어져 왔는데 Delaunay triangulation의 특징은 분할된 삼각패치들을 내각이 등각에 가까운 정삼각형 형태로 개선함으로써 전체적으로 삼각망을 균일하게 하는 장점이 있다.

최근 Delaunay triangulation을 이용한 다양한 응용분야에는 solid modeling, volume rendering, 컴퓨터를 이용한 시각화 등이 있다. 최근 volume품질의 향상, 시각화 개선을 위해 더 많은 데이터가 추가적으로 입력되어지고 있어 고성능 컴퓨터가 등장하는 요즘에도 효율적인 알고리즘 개발이 필수적이다.

본 논문에서 제안하는 알고리즘은 현재까지 개발되었던 여러 Delaunay triangulation 알고리즘들이 uniform grid를 사용함으로써[6],[7] point의 분포가 비균일한 점집합일 경우, 균일한 점집합에서 보다 수행속도가 느려지는 단점을 보완하기 위해서 sub-quad grid division을 수행한다.

또, incremental construction algorithm에서 mate를 찾는 방법으로, 기존의 방법인 외접원(3차원상에서는 외접구)의 반지름을 임의의 크기만큼 증가시켜가면서 외접원을 포함하는 grid 내부의 점집합을 찾는 것이 아니라, 정삼각형망이 생성될 가능성이 높은 지역부터 우선 탐색하는 방법을 사용함으로써 탐색효율을 높인다.

2. 기존의 Delaunay triangulation algorithm

2.1 Voronoi diagram을 이용한 방법

이 방법은 먼저 필요한 점들의 집합에 대한 Voronoi diagram을 생성한 뒤 common edge를 가지는 Voronoi site들을 직선연결하여 Delaunay triangulation을 생성하는 방법이다. 이미 Voronoi diagram을 생성하였을 경우에 $O(n)$의 수행시간만을 가지면 되지만, 이 방법은 먼저 Voronoi diagram을 구해야 하고, Voronoi cell들에 대한 위상정보가 필요하므로 데이터가 방대해 지는 단점이 있다.

2.2 Divide & Conquer 알고리즘

Divide & Conquer 알고리즘[3],[5]은 먼저 점집합들을 분할(Divide)하고, 각각의 부분들을 Delaunay triangulation(Conquer)하는 방법으로 triangulation 후에 다시 각각의 부분들을 병합하는 과정이 필요하다.

이 algorithm은 병합의 어려움 때문에 쉽게 병렬화 할 수 있는 장점에도 불구하고, 널리 이용되는 방법은 아니다.

2.3 Flipping

임의의 삼각화를 먼저 생성한 후에 부분적인 변경을 통해서 Delaunay triangulation을 생성하는 기법이다. 2차원인 경우에 먼저 4점으로 이루어진 두 개의 삼각을 고려하여 삼각형내의 최대각을 전체 삼각형들에 대해 최소화하는 방향으로 Flipping을 진행한다.(Fig.1)

이 방법은 algorithm이 매우 단순하여 구현하기에는 쉬우나 속도 면에서 Divide & Conquer 알고리즘이나 최적화된 incremental construction 알고리즘에 비해 느리다.

2.4 incremental construction 알고리즘

이 알고리즘은 Delaunay triangulation를 생성하는데 있어서 그 기본정의인 empty circle(3차원에서는 sphere)를 이용하는 방법이다.

먼저 empty circle 성질을 만족하는 Delaunay 삼각형을 하나 만들고 만들어진 이 Delaunay 삼각형과 이웃하는 Delaunay 삼각형은 empty circle 성질을 이용하여 추가하는 방법이다.

이 방법은 모든 점집합을 고려하여 triangulation 하기 때문에 이론적인 최악수행시간은 $O(n^2)$ (2차원인 경우)로 매우 나쁘지만 검색하는 점집합의 수를 제한하면 성능을 크게 향상시킬 수 있다.

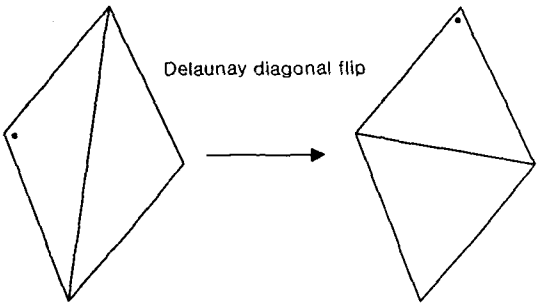

Fig.1 Delaunay triangulation flip

3. 본 논문에서 제안하는 Delaunay triangulation 알고리즘

본 논문에서 제안하는 알고리즘은 Fang과 Piegl이 제안[6],[7]하는 uniform grid를 이용한 Delaunay triangulation에 근간을 두고 있다. uniform grid를 사용하면 특정 영역에 포함되는 기하요소를 짧은 시간에 찾을 수 있는 장점이 있다. 따라서, Delaunay triangulation과정에서 이를 적용시키면 한 외접원 내부에 존재하는 점을 찾을 때, 모든 점집합을 찾을 필요가 없어 전체 수행속도를 크게 향상시킬 수 있다. 특히 점집합의 수가 매우 크면 더욱 효과적이다.

그러나, uniform grid는 그 점집합이 균일 할 때는 매우 효과적이나 점집합이 불 균일 할 때는 grid 내에 점이 하나도 없거나 또는 너무 많은 점들이 존재하기 때문에 효율성에 문제가 생긴다. 따라서, quad subdivision을 시행하여 하나의 grid내에 4개이상의 점들이 존재하는 경우 동일한 크기의 sub-quad를 생성한다.

3.1 data의 전처리

(x_i, y_i) $i=0,\ldots,n$를 2차원상의 점군이라고 한다면, 최단 시간 안에 임의의 선분 $\overline{ab}$의 mate인

점 c를 찾아 Delaunay triangulation을 생성하고, Delaunay triangulation에 의해 생성된 새로운 edge를 이용하여 연속적인 Delaunay triangulation을 만든다.

mate를 빨리 찾기 위한 방법으로 먼저, 2차원 평면상에 uniform grid를 생성한다. uniform grid의 간격은 점군이 균일할 때 grid가 대략 하나의 점을 포함할 수 있도록 다음과 같이 처리한다.

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

$x_{\max}$, $x_{\min}$, $y_{\max}$, $y_{\min}$: x, y의 최대, 최소값
N : 점의 개수

$$grid\_x = \frac{x_i - x_{\min}}{size} \quad (2)$$

$$grid\_y = \frac{y_i - y_{\min}}{size} \quad (3)$$

$$i\_cell = int(grid\_x) \quad (4)$$

$$j\_cell = int(grid\_y) \quad (5)$$

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

3.2 quad subdivision

위의 방법으로 uniform grid를 생성하고 난 후, grid 내부에 4개이상의 점이 존재할 때(즉, 동일한 index 번호를 가진 점이 4개 이상일 때) 동일한 크기의 4개 grid로 나눈다. (Fig.2)

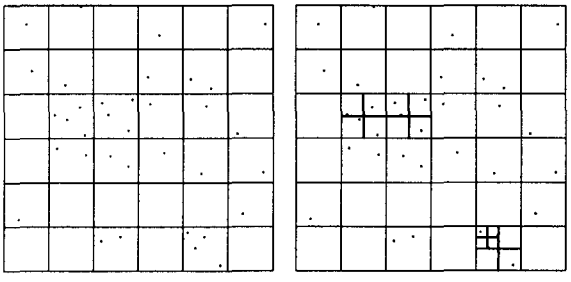

Fig.2 Divide uniform grid into sub-quad grids

3.3 Delaunay triangulation의 시작 선분 찾기

incremental construction 알고리즘은 어떠한 선분에서부터 출발하든지 문제가 없다. 그러나, 효율성을 위해 grid의 중앙에서부터 시작하는 것이 일반적이다. 중앙의 grid index를 (i_{middle}, j_{middle})이라 한다면,

$$(i_{middle}, j_{middle}) = \left(\frac{grid_{x_{\max}} - grid_{x_{\min}}}{2}, \frac{grid_{y_{\max}} - grid_{y_{\min}}}{2}\right) \quad (6)$$

$grid_{x_{\min}}$, $grid_{x_{\max}}$, $grid_{y_{\min}}$, $grid_{y_{\max}}$: x,y grid index의 최소, 최대값

3.4 Delaunay triangulation 생성

1. 먼저 선분 $\overline{P_1P_2}$의 우측부분을 탐색한다.
2. Fig.3에서 선분 $\overline{P_1P_2}$의 연장선을 그어 grid의 밑면 또는 측면부분과 만나면 선분 $\overline{P_1P_2}$의 연장선을 포함하는 영역을 만든다. 이 부분이 선분 $\overline{P_1P_2}$의 mate가 존재할 확률이 제일 높은 부분이 되므로 이 영역을 우선 검색한다.
3. Delaunay triangulation은 삼각형의 내각이 등각에 가까운 정삼각형 형태여야 하므로 영역의 정점부분에서부터 대각방향으로 검색한다. 즉 정점부분의 index가 (i, j)라면 (i, j), $(i-1, j)$, $(i, j-1)$...의 순서로 검색해 나간다. 특히, 검색을 해 나가는 중간에 sub-quad grid를 만나게 되면, 선분 $\overline{P_1P_2}$와 가까운 sub-grid부터 우선 검색한다(Fig.3).
4. 위와 같은 검색을 실시하여 검색 영역 안에 점이 발견되면, P_1, P_2와 새로운 점 P_3를 지나는 원을 그리고 이 원을 포함하는 최소의 정사각형을 만든다.

그리고, 이 정사각형이 지나는 grid 내부의 점들을 검색하여 점 P_1, P_2, P_3이외에 다른 점들이 circle 내부에 존재하는지 판정한다(Fig.4). 만약, 내부에 포함하는 점이 없으면 P_3는 선분 $\overline{P_1P_2}$의 mate가 되어 새로운 Delaunay triangulation이 완성된다.

5. 그러나, circle 내부에 점들이 존재하면, 그 중 circle 중심으로부터 가장 가까운 점을 선택하여 위와 같은 작업을 empty circle을 만족할 때까지 반복한다.

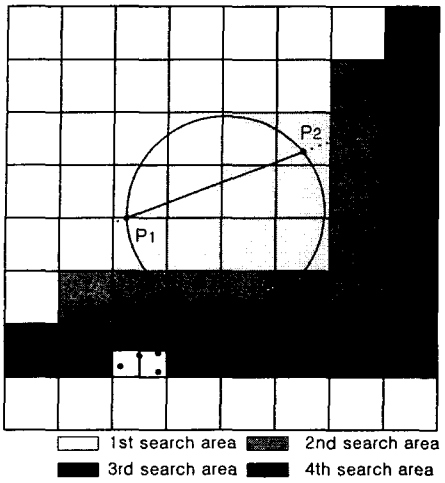

Fig.3 Process to make a triangle, given an edge

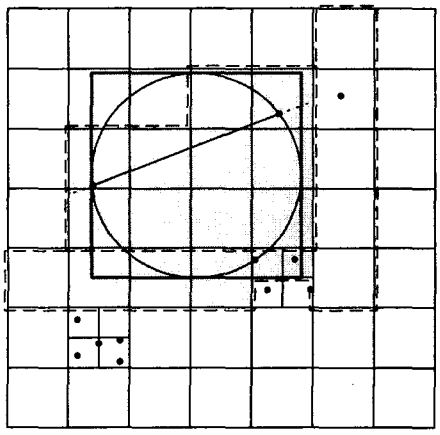

Fig.4 Search grids within bound box

4. 개발 환경

본 연구에서는 Visual C++ 6.0을 이용하였고 다음과 같은 환경 하에서 프로그램을 개발하였다.

Computer	Pentium III 450MHz
운영체계	Windows 2000 Professional
language	Visual C++ 6.0

Table 1 System Configuration

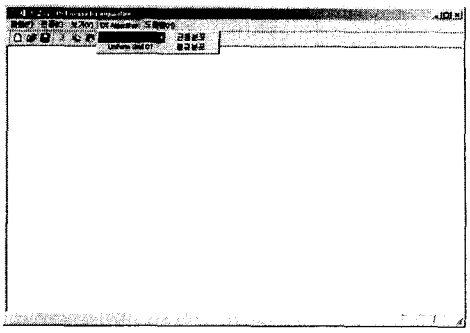

Fig.5 프로그램 인터페이스

Fig.5는 프로그램 인터페이스로서 Delaunay Triangulation을 생성하기 위해서 먼저 임의의 점집합을 발생시킨다. 이 프로그램에서 제공하는 점집합은 전 구간을 균일하게 분포시켜주는 균등분포(Uniform Distribution)와 정규분포(Normal Distribution) 점집합이 있다.

Fig.6과 Fig.7은 각각의 메뉴를 실행시켰을 때의 결과화면이며 Fig.8은 DT를 수행한 후의 화면이다.

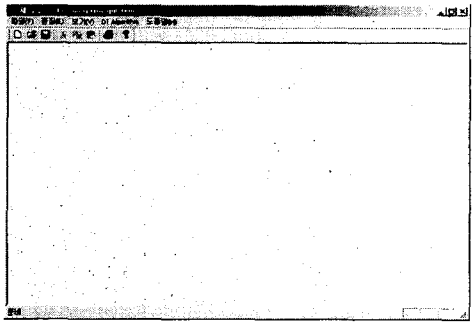

Fig.6 Uniform Distribution시 점집합

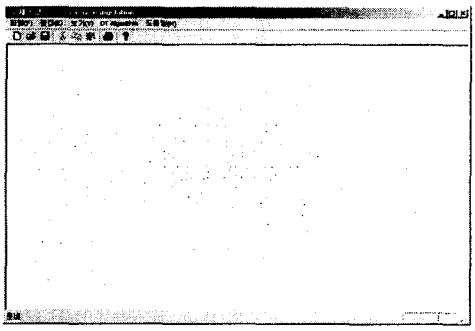

Fig.7 Normal Distribution시 점집합

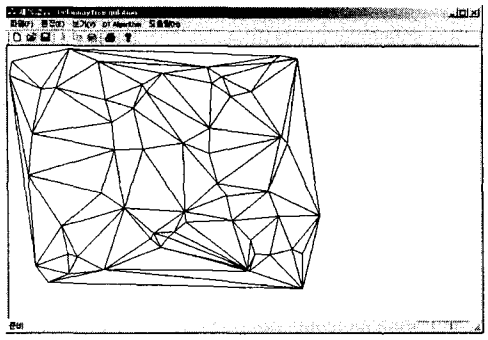

Fig.8 Triangulation Patch결과

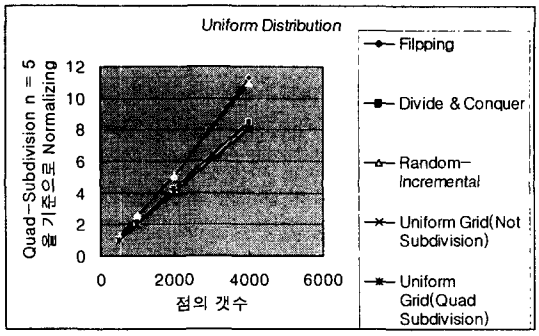

Fig.9 Uniform Distribution 일 때 수행시간

5. 결과

본 논문에서 제시한 알고리즘을 이용한 Delaunay Triangulation과 기존의 Delaunay Triangulation 알고리즘과 비교한 결과 다음과 같은 결과를 얻었다.

	500	1000	2000	4000
Flipping	1.15	2.58	5.02	11.28
Divide and Conquer	1.08	2.14	4.25	8.48
Random-incremental	1.27	2.60	5.08	10.95
Uniform Grid (Not Subdivision)	0.99	2.02	4.03	8.05
Uniform Grid (Quad Subdivision)	1	2.01	4.08	8.11

Table 2 Uniform Distribution일 때

	500	1000	2000	4000
Flipping	1.16	2.60	5.32	11.48
Divide and Conquer	1.08	2.18	4.37	8.65
Random-incremental	1.26	2.60	5.12	11.25
Uniform Grid (Not Subdivision)	1.12	2.20	4.51	9.01
Uniform Grid (Quad Subdivision)	1	2.03	4.10	8.21

Table 3 Normal Distribution일 때

(Quad-Subdivision을 이용한 Delaunay Triangulation Patch의 수행시간이 1(n = 500)일 때 타 algorithm의 수행시간)

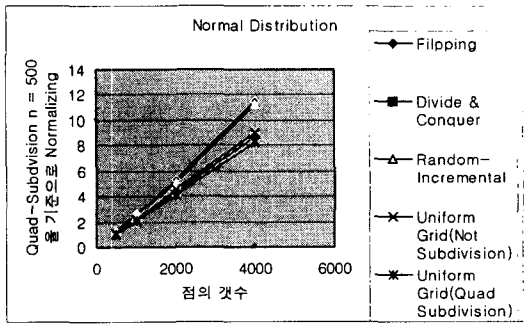

Fig.10 Normal Distribution일 때 수행시간

6. 결 론

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

그러나, uniform grid를 이용한 Delaunay triangulation은 점집합이 정규 분포나 지수분포형태를 가질 경우, 그리드 내부에 점이 밀집되어 있기 때문에 밀집된 점을 모두 검색하여야 한다.

따라서, quad subdivision을 시행하여 검색을 하면 우선 검색 범위를 줄일 수 있기 때문에 mate를 찾는 시간을 감소시킨다. 이 알고리즘의 평균 검색시간은 $O(GST(n) + n * Search\_Mate\_S( ))(GST(n,$: n개의 점집합으로 부터 uniform grid를 생성하고 quad subdivision을 실행하는 시간, *Search_Mate_S*( ) : quad subdivision된 grid에서 mate를 찾는 시간)로서 점집합이 불균일할 경우, 기존의 uniform grid에서의 평균 검색시간 $O(GT(n) + n$: *Search_Mate_U( ))* $(GT(n)$: n개의 점집합으로 부

터 grid를 생성하는 시간, *Search_Mate_U( )* : uniform grid에서 mate를 찾는 시간) 보다 효율성이 높다.
(불균일한 점집합에서 mate를 검색하는 시간 *Search_Mate_S( ) << Search_Mate_U( )*)

참고문헌

[1] Michael J. Laszlo, "Computational Geometry and Computer Graphics in C++", Prentice Hall, 1996.

[2] Ding-Zhu Du and Frank Hwang, "Computing in Euclidean Geometry", World Scientifc, 1995.

[3] Rex A. Dwyer, "A Faster Divide and Conquer Algorithm for Constructing Delaunay Triangulations, Algorithmica, 2, pp. 137-151, 1987.

[4] David P. Dobkin and Michael J. Laszlo, "Primitives for the Manipulation of Three-Dimensional Subdivisions", Algorithmica, 4, pp. 3-32, 1989.

[5] 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.

[6] Tsung-Pao Fang and Les A. Piegl, "Delaunay Triangulation Using a Uniform Grid", IEEE Computer Graphics and Applications, 13(3), pp. 36-47, 1993.

[7] Tsung-Pao Fang and Les A. Piegl, "Delaunay Triangulation in Three Dimensions", IEEE Computer Graphics and Applications, 15(5), pp. 62-69, 1995.

[8] Ding-Zhu Du and Frank Hwang, "Computing In Euclidean Geometry", World Scientific, pp225-263, 1998