2D 삼각화를 통한 3D 점집합으로부터의 메쉬 생성

최지훈°, 윤종현, 박종승
인천대학교 컴퓨터공학과
{jongbang,jhyoon,jong}@incheon.ac.kr

3D Mesh Creation using 2D Triangulation of 3D Point Clouds

Ji-Hoon Choi°, Jong-Hyun Yoon, Jong-Seung Park
Department of Computer Science and Engineering, University of Incheon

요 약

본 논문에서는 3D 점집합으로부터 3D 메쉬를 생성하는 효율적인 기법을 소개한다. 대표적인 3D 삼각화 방법으로 3D Delaunay 삼각화 기법이 있으나 물체의 표면만을 고려한 메쉬 생성을 위한 방법으로 비효율적인 측면이 있다. 본 논문에서는 적은 계산량으로 물체의 표면 메쉬를 생성하는 기법을 소개한다. 물체의 각 영역을 분할하고 각 영역에 대해서 2D Delaunay 삼각화를 적용하여 3D 메쉬 구조를 얻는다. 3차원 점 집합에 대해 OBB를 계산하고 이를 기준으로 점집합을 다양한 각도에서 자르고 각 부분 점집합에 대해서 2D Delaunay 삼각화를 실시한다. 절단하는 각도의 간격이나 폭은 원래의 3D 점집합에서의 가장 가까운 이웃점들까지의 평균 거리를 이용하여 결정하도록 하였다. 후처리 과정으로 삼각 분할 과정에서 잘못된 에지의 연결을 제거함으로써 객체의 삼각 분할 결과를 향상시킨다. 제안된 메쉬 생성 기법은 다양한 영상 기반 모델링 응용에서 효과적으로 적용될 수 있다.

1. 서론

최근 실 영상들로부터 3D 점들을 획득하는 연구가 활발히 진행되고 있다. 3D 점들의 좌표값들이 추정되면 이들로부터 실제의 형상 메쉬를 생성해야 3D 렌더링이 가능하다. 3D 형상에 대한 메쉬 모델이 주어지면 다양한 기법으로 화려한 입체 표현이 가능해진다. 3D 그래픽의 응용에는 정교한 3D 객체들이 필요하고 이들 3D 객체들의 생성 절차는 디자이너들의 고된 수작업의 결과를 통해서 완성된다. 물체의 표현은 다각형이 결합한 메쉬 형태가 일반적이다. 다각형의 형태는 일반적으로 삼각형이 사용된다. 하나의 물체는 적게는 수백 개에서 많게는 수십만 개 이상의 다각형을 포함한다. 다각형의 수는 정교한 표현과 비례하므로 가급적이면 많은 다각형으로의 구성이 바람직하다. 그러나 최근의 3D 그래픽 분야의 추세인 실시간 렌더링에서는 렌더링의 고속화가 중요하므로 비슷한 정교함을 가지면서 최소한의 다각형을 사용하는 접근이 가장 바람직하다. 본 연구에서의 목표는 주어진 3D 점 집합(point cloud)으로부터 물체를 자연스럽게 표현하는 메쉬 구조를 생성하는 기법의 개발이다.

점 집합으로부터의 메쉬 구조 생성은 삼각화(triangulation) 기법을 통해 수행된다. 대표적인 삼각분할의 방식에는 Greedy 방식과 Delaunay 방식이 있다[1]. Greedy 방식은 요구된 수의 선분이 더해질 때까지 에지의 길이가 짧은 것부터 순서대로 검사하면서 삼각 분할을 구성하는 방식이다. Delaunay 삼각 분할 방법은 주어진 절점에 대해 등각 조건을 만족하는 삼각형, 변형이 적은 삼각형, 즉 정삼각형에 가까운 형상을 갖는 요소를 얻을 수 있다. 삼각 분할이 형성된 삼각형의 외접원 내부에 어떠한 점도 포함되지 않는 경우를 말한다.

현재 Greedy 방식과 Delaunay 방법 또는 이외의 방법으로 최적의 삼각 분할을 만족하기 위한 여러 가지 연구가 진행되고 있다[2][3]. 삼각 분할 된 상태에서 정삼각형에 가까운 형태를 만들기 위해 에지를 교환하는 방법, 에지를 추가하는 방법 등이 있고 연구가 진행됨에 따라 더욱더 최적화된 결과를 얻을 수 있다[4].

본 논문에서는 2차원 삼각분할의 개념을 조금 더 확장시켜서 3차원 객체에 대한 효율적인 삼각분할 방법을 제안한다. 2D Delaunay 삼각화에 기반하여 3D 점 집합을 분할하고 각 부분 점 집합들에 삼각화를 적용한다. 각 부분 점 집합들에 대한 삼각화 결과들을 합쳐서 전체적인 메쉬를 생성한다. 각 삼각화 결과에 대해서 잘못된 에지를 검사 및 배제시키는 후처리 절차를 통해서 보다 정확한 메쉬 구조를 생성한다.

2장에서는 관련 연구, 3장에서는 제안된 3차원 삼각 분할 방법에 대해 설명하고, 4장에서는 삼각 분할 된 실험과정 및 결과에 대해 설명하며, 마지막으로 결론 및 향후 보완과제를 5장에서 설명한다.

2. 관련 연구

Delaunay 삼각 분할 방법은 점의 배치에 관한 몇 가지 접근 방법을 가진다. 하나는 점들의 경계 영역에서 모든 점들을 포함하는 Steiner 점들을 추가시키는 방법이다. Delaunay 삼각 분할 방법은 대표적으로 제한된 삼각분할기법, 점진적(incremental) 방법, 분할정복기법으로 분류할 수 있다. 제한된 삼각 분할 방법은 삼각형의 세 각 중 가장 큰 각을 최소화하는 삼각분할은 다각형의 외곽을 이루는 선분이 미리 명시되고 삼각형의 세 정점 모두에서 볼 수 있는 점을 외접원 내부에 포함되지 않도록 하는 방법이다. 점진적 방법[5]은 초기에 모든 점들을 포함하는 초기 삼각형을 지정한다. 새로운 점 p가 입력되었을 때 그 점을 포함하는 삼각형의 예전의 에지는 제거되고 새로운 점과 새로운 에지를 만들어 나간다. 이렇게 삼각 분할을 하게 되면 $O(n^2)$의 수행 시간이 걸린다. 분할 정복기법[6]은 하나의 점 집합을 재귀적으로 2개의 부분으로 쪼개면서 선을 그리는 방법이다.

Edge flipping 방법은 사변형 들의 최대 각을 최소화 하기 위해 에지를 교환하는 방법이다[7]. 이 방법은 항상 정확하게 최적의 에지를 발견하지 못한다. 더 향상된 방법인 에지 삽입 방법이 있다[8]. 일반적으로 최악의 삼각 분할된 삼각형의 최악의 점은 흔한 에지 삽입의 후보이다. 이런 후보 점들이 충분히 좋은 형태로 이동하면서 구멍 난 부분에 에지를 추가하는 방법이다.

3. 효과적인 3차원 삼각 분할을 통한 메쉬 생성

본 논문에서는 3차원 점 집합 데이터를 입력으로 OBB를 계산하고 이를 기준으로 삼각분할을 수행하는 방법을 제안한다. 삼각분할 과정에서 각 점마다 가장 가까이 있는 점까지의 거리들 중 가장 큰 값을 임계값으로 지정하고 삼각분할 과정에서 잘못 연결된 긴 에지를 제거하여 결과를 개선한다.

3.1. 입력 데이터의 정렬

3차원 데이터에 대한 삼각분할의 입력 데이터로써 객체에 대한 3차원 점 집합 S를 사용한다. 먼저 입력된 점 집합 S에 대해서 OBB(Oriented Bounding Box)를 계산한다. 한 물체의 OBB는 3차원 바운딩 박스로 각 면에 대한 방향 벡터와 길이를 가진다. 즉 OBB의 중심점의 위치 벡터 (bc), 상자의 각 직교축의 방향 벡터 (bu, bv, bw), 상자의 각 직교축으로의 절반 길이 (lu, lv, lw)의 인자로 표현된다. OBB의 각 축은 Principal component analysis (PCA)를 통해서 세 component axis를 찾아서 설정한다. 첫번째 축인 u축은 첫번째 component axis로 설정하고 그 두번째 축인 v축은 두번째 component axis로 설정한다. 마지막 w축은 u축과 v축의 외적으로 설정한다.

OBB가 구해지면 OBB의 (u,v,w) 좌표계가 3차원 (x,y,z) 좌표계와 일치하도록 변화를 계산하고 이를 적용시켜서 (u,v,w)축이 (x,y,z)축과 일치하도록 한다.

3.2. 삼각 분할과 병합

본 논문에서는 3차원 객체의 점 집합에 대한 principal axis를 계산하고 이를 기준으로 삼각분할을 하는 방법으로 3차원 폴리곤을 생성한다. 이는 unstructured 메쉬 타입으로서 보다 복잡한 형태를 처리할 수 있다[9].

이전 단계에서 오리엔테이션의 일치를 통한 결과로써 3차원 객체가 월드 좌표계로 정렬되었다. 정렬된 3차원 점 집합은 principal axis의 회전에 의해 삼각 분할이 이루어 진다. 좌표계의 6DOF 중 한 방향을 레퍼런스 방향으로 설정한다. 설정된 레퍼런스 방향을 초기 방향으로 앞, 뒤 면의 부분 점 집합을 구성한다. 이때 구성된 부분 점 집합들은 다시 여러 개의 하위 부분 집합으로 나뉘게 된다. 하위 부분 집합을 나눌 때, 초기화 과정에서 계산된 임계값이 사용된다. 임계값의 사용을 통해 근접해 있는 점들간의 하위 부분 집합을 형성할 수 있다. 삼각 분할은 하위 부분 집합 단위로 진행된다. 이를 통해 멀리 있는 점들 사이에 폴리곤이 형성되는 문제를 해결할 수 있다.

3.3. 잘못된 에지의 제거

삼각화 수행으로 잘못된 에지들이 발생될 수 있다. 이러한 에지들은 모두 제거되어야 하므로 유효 에지의 기준을 설정해주도록 해야 한다. 우선 초기 점 집합으로부터 계산한 가장 가까운 이웃점까지의 거리를 사용한다. 가장 가까운 이웃점까지의 최대 허용 거리를 계산할 수 있으며, 이 거리보다 더 큰 거리를 가지는 삼각형들은 모두 제어하도록 한다.

삼각형 제거 시에 사용될 임계값을 초기화 단계에서 계산한다. 입력 점 집합에서 각 점으로부터 최단거리에 위치한 점까지의 거리를 계산한다. 이때, n개의 점에 대해 점 p_i에서 가장 가까운 점까지의 거리를 d_i라고 하자. 임계값은 다음과 같이 계산한다:

$$d_t = \max_i(d_i)$$

임계값은 삼각형의 에지가 잘못 연결되는 것을 제거하는데 사용된다.

3.4. 전체적인 삼각 분할 방법

그림 1은 제안한 분할을 통한 삼각화 방법에 대한 전체적인 절차에 대한 개념도를 나타낸다. 많은 3D 점 집합이 국부적으로 2D 평면으로 투사할 경우에 겹치지 않도록 할 수 있으므로 분할을 통한 2D 삼각화 기법이 유효함을 알 수 있다. 3차원 객체를 6개의 부분데이터로 나누고, 각각에 대하여 삼각 분할한 후 병합하는 과정을 가진다. 우선 3차원 객체의 표면을 6분할 할 수 있는 직육면체의 박스를 설정한다. 설정된 박스는 객체의 좌표축과 같은 좌표축을 갖는다고 가정한다. 박스의 6면에 대한 방향은 {-x, +x, -y, +y, -z, +z}과 같이 표현할 수 있다. 그리고 나서 분할 박스를 사용하여 3차원 객체를 6부분으로 분할한다. 분할된 6개의 부분 데이터들에 대하여 분할

박스의 6면 중 한 면의 방향을 레퍼런스 방향으로 지정한다. 레퍼런스 방향이 지정되면 나머지 5면에 대하여 레퍼런스 방향과 같은 방향을 갖도록 회전한다. 회전을 통해 모두 같은 방향을 갖는 부분 데이터들을 삼각 분할을 위한 입력 데이터로 사용한다.

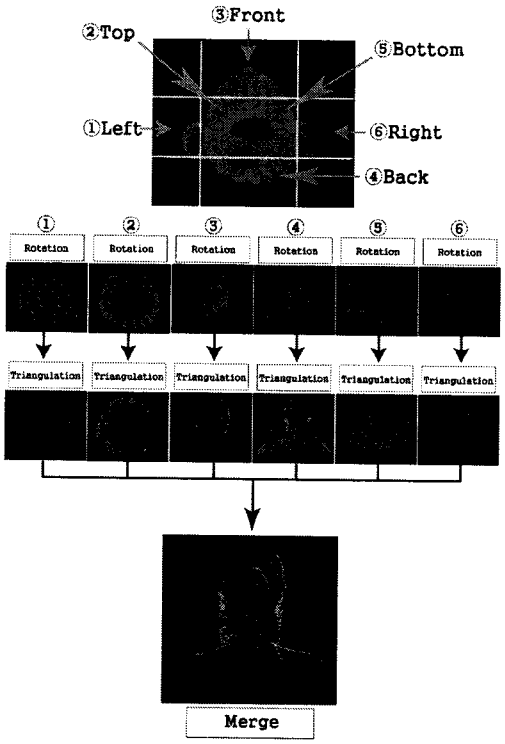

그림 1. 분할을 통한 삼각화의 개념도.

단계 1 (초기화 단계)
1. 3차원 좌표를 입력 받아 점 집합 S를 생성.
2. 입력된 좌표들의 데이터 방향에 대해 principal axis를 구하고 그에 수직인 minor axis를 구함.
3. 두 개의 축을 원점 기준으로 위치시킴.
4. 집합 S의 원소를 { S_1, S_2, · · · · , S_n } 이라고 할 때, 각 원소에서 가장 가까운 에지를 구하고, 그 중에 최대값을 구함.

단계 2 (점 집합 S에 대한 분할)
1. 집합 S에 대해 두 개의 축을 기준으로 앞면과 뒷면을 가정하고 앞면과 뒷면 사이를 4조각으로 나눈다. 앞면과 앞면에 가까운 1/4조각 사이에서, 뒷면과 뒷면에 가까운 1/4조각 사이에서 포함된 점들로 자름

단계 3 (점 집합 S의 그룹화, 삼각 분할)
1. Step 5에서 분할된 점 집합 S에 대해 임의로 한 개의 점을 입력으로 넣고, Step 4에서 구한 최대값을 활용한다. 만약 임의로 입력한 한 개의 점이 에지의 최대값보다 작은 범위 내에 존재하면 에지들을 이음.
2. 범위 내에 있던 다른 점들에 대해서도 동일한 방법으로 실시하여 이어진 모든 에지를 그룹화한다. 만약 Step 5의 잘려진 면에서 에지의 최대값보다 큰 거리를 사이에 두고 점의 집합이 둘로 나뉘어진 형태가 되면 그룹도 2개가 된다. 그 점들에 대해 삼각 분할을 실시.
3. 삼각 분할을 실시한 점들에 대해서는 입력으로 들어온 점의 개수에서 제거해준다. 그룹화된 영역을 { S_{G0}, S_{G1}, S_{G2} ,…,S_{Gn}} 라고 할 때, n(S)가 0이 아닐 동안에 n(S) = n(S) - S_G 실시.
4. x축을 기준으로 90도 회전하여 회전된 면에 대해서 Step 5, 6을 실시하고 45도 회전하여 4 모서리에 대해서도 동일하게 실시.
5. 만약 Step 7까지 진행되도록 n(S)가 0이 아니라면 각 면에 대해 1/2 지점까지 잘라서 삼각 분할.

그림 2. 분할을 통한 삼각화 기법의 수행 과정.

그림 2는 전체적인 삼각화 기법의 수행 절차를 나타낸다. 삼각 분할 과정 시 부분 데이터로 나누는 부분에서 초기 분할 위치를 객체의 표면으로부터 객체의 깊이 값의 1/4까지로 지정하고 principal axis를 축으로 90도와 45도씩 회전하는 방식을 사용한다. 그러나 입력데이터의 형상이 임의의 모양을 가지고 있기 때문에 1/4의 깊이를 통해 삼각 분할 시 모든 점이 포함될 수 없는 예외가 발생한다. 그러므로 삼각 분할에 대한 한 루틴이 실행된 후 점 집합 S를 검사한다. 이때, S의 모든 점이 삼각 분할의 입력으로 포함되었을 경우는 한번의 삼각분할을 통해 결과를 얻을 수 있으나, 1개 이상의 점이 제외 되었을 경우 분할 깊이를 더 깊게 함으로써 모든 점을 포함시킬 수 있다. 삼각 분할의 반복은 분할 깊이가 객체 깊이값의 1/2을 넘지 않는 동안 반복된다.

4. 실험결과

제안한 알고리즘의 유효성 검증을 위해 다양한 3D 점 집합을 사용하여 실험하였다. 그림 3은 프로그램으로 구 표면의 임의의 점들을 생성하고 이를 삼각화한 결과이다. 분할 간격 및 임계값에 영향을 받지 않으면서 올바른 메쉬가 생성되었다. 구 형태의 물체는 가장 전형적인 형태로 6번의 분할로도 정확한 메쉬가 생성된다.

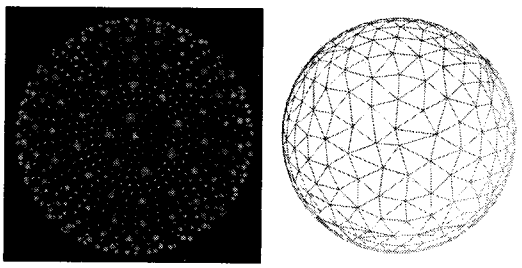

그림 3. 구 표면의 점들에 대한 삼각화.

일반적인 실물 점 집합에 대해 삼각화를 적용하였다. 그림 4는 뇌 구조를 나타내는 점 집합이다. 이를

입력으로 그림 5의 메쉬를 형성하였다. 잘못 연결된 에지가 없이 삼각 분할이 된 결과를 그림으로부터 확인할 수 있다.

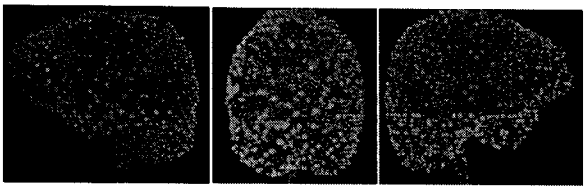

그림 4. 뇌 형상에 대한 점 집합.

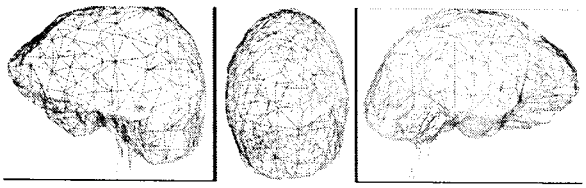

그림 5. 분할을 통한 삼각화 기법을 사용하여 생성한 메쉬의 모습.

5. 결론

본 논문에서는 3차원 객체를 표현하는 점 집합에 대하여 메쉬 구조를 형성할 수 있는 삼각 분할 기법을 제안하였다. 한 객체에 대하여 principal axis를 계산하고 이를 기준으로 부분적으로 삼각 분할 하여 3차원 구조를 형성하였다. 부분 데이터에 대한 삼각 분할 기법으로는 Delaunay 삼각 분할 기법을 확장한 방법을 사용하였다. 물체의 각 영역을 분할하고 각 영역에 대해서 2D Delaunay 삼각화를 적용하여 3D 메쉬 구조를 얻는다. 3차원 점 집합에 대해 OBB를 계산하고 이를 기준으로 점집합을 다양한 각도에서 자르고 각 부분 점집합에 대해서 2D Delaunay 삼각화를 실시한다. 절단하는 각도의 간격이나 폭은 원래의 3D 점집합에서의 가장 가까운 이웃점들까지의 평균 거리를 이용하여 결정하도록 하였다. 후처리 과정으로 삼각 분할 과정에서 잘못된 에지의 연결을 제거함으로써 객체의 삼각 분할 결과를 향상시킨다. 제안된 메쉬 생성 기법은 다양한 영상 기반 모델링 응용에서 효과적으로 적용될 수 있다.

제안한 방법은 영상으로부터 3차원 모델을 얻는 영상기반 모델링에 적용될 수 있다. 향후 연구과제로, 삼각 분할 후 메워지지 않은 영역에 대한 삼각화 방법에 대해 연구할 계획이다.

참고 문헌

[1] S. A. Goldman, "A space efficient greedy triangulation algorithm," Inform. Process. Lett. 31, pp. 191-196, 1989.

[2] D. G. Kirkpatrick, "A note on Delaunay and optimal triangulations," Inform. Process. Lett. 10, pp. 127-128, 1990.

[3] Marshall Bern and Paul Plassmann, "Mesh generation," Chapter 6 In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry. Elsevier Science, 1999.

[4] Marshall Bern and David Eppstein, "Mesh Generation and Optimal Triangulation," Computing and Euclidean Geometry 2nd, pp. 47-123, 1995.

[5] Dani Lischinski, "Incremental Delaunay triangulation," Graphics Gems IV, pp. 47-59, 1994.

[6] R. A. Dwyer, "A simple divide-and-conquer algorithm for computing Delaunay triangulations is O(nloglogn) expected time," Symposium on Computational Geometry, pp. 276-284, 1986.

[7] S. Hanke, T. Ottmann, S. Schuierer, "The edge-flipping distance of triangulations," J. Universal Comput. Sci. 2, pp. 570-579, 1996.

[8] Marshall Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell and T.-S. Tan, "Edge-insertion for optimal triangulations," Computational Geometry 10, pp. 47-65, 1993.

[9] D. D. Morris and T. Kanade, "Image-consistent surface triangulation," Computer Vision and Pattern Recognition Vol. 1, pp. 332-338, 2000.