한국정밀공학회지 제 24 권 제 5 호 (2007 년 5 월)
Journal of the Korean Society for Precision Engineering, Vol. 24, No. 5, May 2007

2D Delaunay Triangulation 을 이용한 점군 절단

박형태#, 장민호*, 박상철**

Point Cloud Slicing Based on 2D Delaunay Triangulation

Hyeong-tae Park#, Minho Chang* and Sang-chul Park**

ABSTRACT

Presented in the paper is an algorithm to generate a section curve by slicing a point cloud including tens of thousands of points. Although, there have been previous research results on the slicing problem, they are quite sensitive on the density variations of the point cloud, as well as on the local noise in the point cloud. To relive the difficulties, three technological requirements are identified; 1) dominant point sampling, 2) avoiding local vibration, and 3) robustness on the density changes. To satisfy these requirements, we propose a new slicing algorithm which is based on a node-sphere diagram. The algorithm has been implemented and tested with various examples.

Key Words : Slicing (절단), Point Cloud (점군), Rapid Prototyping (쾌속조형), Reverse Engineering (역공학)

1. 서론

역공학(Reverse Engineering)은 이미 존재하는 형상 모델의 디자인 개념을 추출 하거나 복제품 제조, 또는 re-engineering 을 위해 객체로부터 CAD 모델을 만드는 것을 의미한다.[1] 형상 디자이너가 만든 최초의 파트 모델은 나무나 점토로 나타내어지며 여기서 얻어진 디자인 샘플 정보가 CAD 모델의 형식으로 변환된다. 레이저 스캐너 같은 비 접촉 광학 측정 장치는 이런 형상 정보를 빠르게 획득하는 것을 가능하게 한다. 이것은 보통 Fig. 1 과 같이 불규칙적이며 무수히 많은 점 구름 형태의 데이터로 나타난다. 쾌속조형(Rapid Prototyping)은 입체 형상 개체를 무수한 층으로 나누어 각 층의

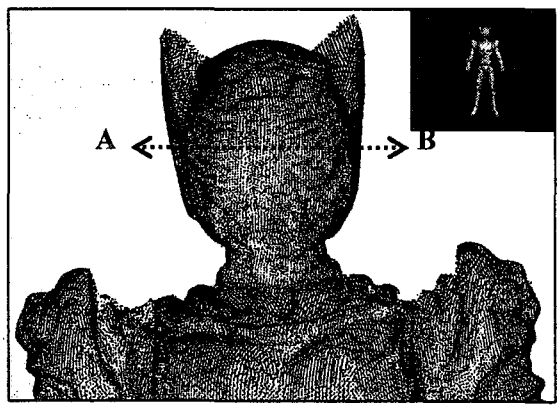

Fig. 1 Measured point cloud

형상을 가공하여 전체적인 모습을 빠르게 나타내

☞ 접수일: 2006 년 11 월 22 일; 게재승인일: 2007 년 3 월 21 일
교신저자: 아주대학교 산업공학과 대학원
E-mail: taiji416@ajou.ac.kr Tel. (031) 219-1763
* ㈜ Solutionix
** 아주대학교 산업공학과

는 제조 방법이다. 복잡한 기하 형상을 가진 파트 모델을 만들기 위해 쾌속조형은 광범위하게 사용된다.[2] 그러므로 쾌속조형을 위한 점군 모델링은 빠른 제품 개발을 위해 필수적이라고 할 수 있다.

일반적으로 쾌속조형을 위한 점군 모델링은 세 가지 접근법으로 이뤄 진다.[3] 첫 번째 방법에서는 먼저 점군으로부터 표면 모델을 재 구성(Surface Reconstruction)하고 이것을 STL 파일형태로 변환한다. 이 STL 파일이 쾌속조형을 위한 층들을 생성하기 위해 잘려진다. 두 번째 접근법은 점군으로부터 곧바로 STL 파일을 생성하여 절단하는 방법이며,[4,5] 세 번째 방법은 점군에서 곧바로 쾌속조형 절단 파일(절단커브)을 생성한다.[6] 첫 번째 접근법에 의해 생성된 평면 모델은 수정할 수 있는 장점이 있다. 그러나 최종 쾌속조형 모델의 형상 오차(쾌속조형모델과 점군 사이)는 세 가지의 원인에 의해 야기된다. (1)점군과 표면 모델 사이의 오차, (2)표면 모델과 STL 모델 사이의 오차, 그리고 (3)STL 모델과 절단커브 사이의 오차. 이런 오차들은 최종 쾌속조형 모델과 점군 사이 오차의 원천 규명과 정확한 제조에 방해가 될 것이다. 두 번째 접근법에 의한 방법도 두 가지의 오차 원인을 가진다. (1)점군과 STL 사이의 오차, (2)STL 모델과 쾌속조형모델 사이의 오차. 첫 번째와 두 번째 접근법은 최종 쾌속조형모델을 생성할 때에 복수의 오차 원인이 영향을 준다. 때문에 오차의 최소화 및 원천 규명이 어려우며 이것은 원 점군 모델이 가진 절단 형상 정보를 정확하게 반영하는데 장애가 될 것이다. 게다가 표면 모델의 재구성과 STL 모델의 생성 모두 과도한 계산 시간이 요구되며 그 알고리즘 자체도 강건하지 않는 것이 사실이다.

반면, 세 번째 접근법의 경우, 점군 모델로부터 곧바로 절단커브를 도출함으로써 오차 원인을 최소화 할 수 있으며 계산 과정도 앞의 두 방법에 비해 훨씬 가볍다. 이에 본 논문은 쾌속조형을 위해 측정된 점군의 임의 층에서 절단 커브를 생성하기 위한 정확하고 효율적인 알고리즘의 고안에 그 목적을 둔다.

2. 기존 연구

본 장에서는 점군에서 직접 절단 커브를 생성하는 기본적인 알고리즘과 Sun[7] 이 제시하는 알고리즘을 살펴 보자.

2.1 기본 알고리즘

우리는 점군으로부터 절단면을 도시하는 다음과 같은 상식적이며 기본적인 알고리즘을 생각 해 볼 수 있다(Fig. 2).

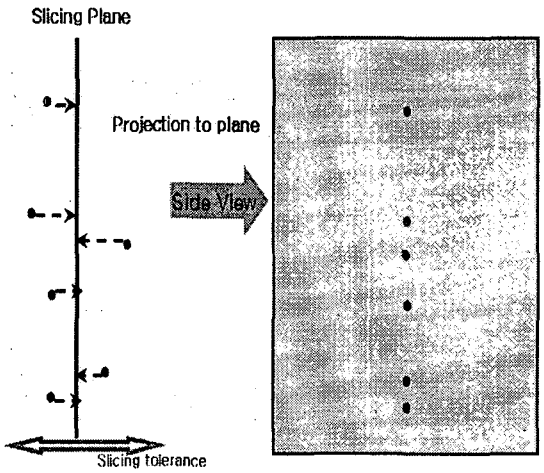

Fig. 2 Basic Slicing algorithm

알고리즘의 첫 번째 단계로, 절단 평면에서 사용자 입력 거리 안의 공차 공간에 놓인 점들을 추출 한다. 여기서 공차의 크기는 측정한 레이저 스캐너의 resolution 에 따라 다르게 설정된다. 너무 작은 양의 점이 공차 공간에 놓일 경우 정확한 절단 커브를 도출하는데 부족한 정보를 제공할 것이며 너무 많은 점이 공차 공간에 놓일 경우 계산 시간의 불필요한 증가 및 커브의 떨림을 초래 할 것이다. 경험적으로 0.2mm 공차의 절단평면 설정은 절단 커브의 생성에 적절하게 작용한다.

알고리즘의 두 번째 단계는 추출될 점을 절단 평면에 투영(projection) 시킨다. 마지막으로 투영시킨 점들을 가까운 순서대로 연결 하여 절단 커브를 완성 한다. 기본 알고리즘을 Fig. 1 점군의 절단면 A-B 에 적용하여 생성한 절단 커브는 Fig. 3 과 같다. 결과에서 볼 수 있듯 절단 커브가 비교적 잘 나온 부분과 그렇지 못한 부분이 있다. 이것은 점 밀도 차이에 기인한다. 점 밀도가 높은 부분의 경우 많은 점들이 투영되고 그 점들을 연결하는 커브는 지그재그 형태의 떨림을 보여준다. 이때 이 떨리는 커브는 형상을 제대로 표현하고 있지 못할 뿐만 아니라 절단 평면과 각 점과의 거리를 고려하지 않고 무작정 투영하고 연결한 결과

로써 정확한 절단커브도 아니다. 따라서 우리는 많은 점이 추출된 부분에서는 절단 커브 생성에 필요한 점을 선택적으로 반영할 필요성을 느낀다. 즉, 많은 점 중에서도 절단 평면에 가까운 점이 실제 절단 커브에 정확한 의미를 부여 할 수 있음으로 그런 점을 비중 있게 반영하여야 한다.

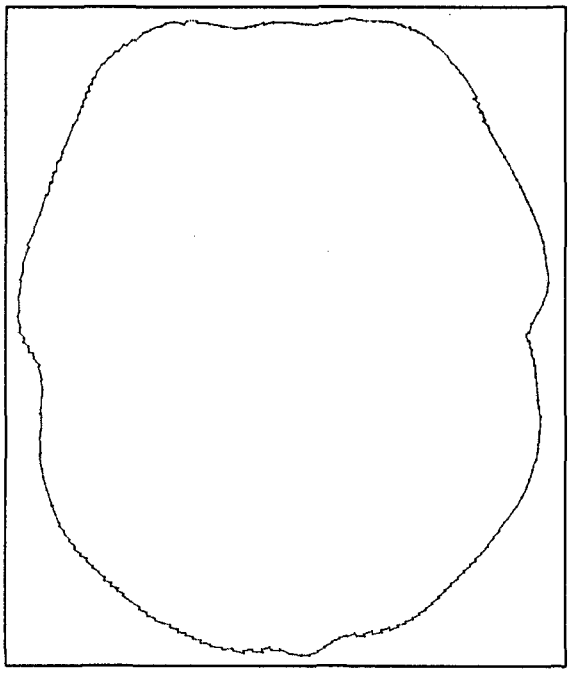

Fig 3 Slicing result of basic algorithm

2.2 Pair-Link 알고리즘

Sun[7] 은 점군으로부터 직접 절단 커브를 생성하는 다음의 알고리즘을 제시한다. 그것의 첫 단계로, 기본 알고리즘과 같이 절단 평면과 그 공차 안에 놓인 점들을 추출한다. 두 번째 단계로 Fig. 4 에 보이는 것처럼 임의의 점과 이 점의 반대편 공차 공간에 놓인 가장 가까운 짝을 찾는다. 그리고 그 두 점을 연결한 선과 절단 평면과의 교점을 구한다. 이런 방법으로 모든 점에 대해 그 짝을 찾은 후 연결한 선분과 절단 평면과의 교점을 구하며 마지막 단계로 구해진 교점들을 가까운 순서대로 연결한다.

Sun 이 제안한 알고리즘의 경우 점 밀도가 조밀하고 비교적 간단한 형상의 예에는 잘 적용될지 모른다. 그러나 점군의 밀도는 모든 부분에 대해

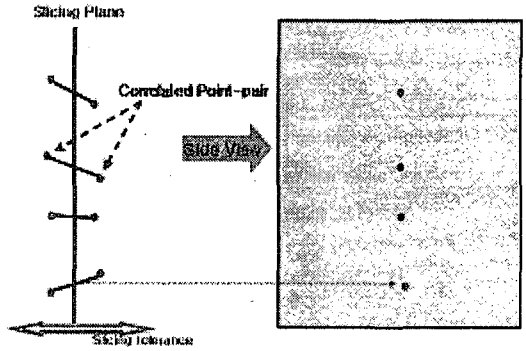

Fig. 4 Pair-link algorithm

조밀성과 균일성이 보장되지 않기 때문에 일반적인 알고리즘과는 거리가 있다. Sun 의 알고리즘을 A-B 절단평면에 적용했을 경우 Fig. 5 에서 보는 것과 같이 반대편 공차공간에서 자신의 짝을 찾지 못한 점이 엉뚱한 곳에서 짝을 이루거나, 혹은 절단 평면을 기준으로 양쪽 두 공간 중에서 한쪽 영역에만 점들이 편중돼 존재할 경우 짝이 없는 부분의 형상이 누락되어 의도하지 않는 결과를 절단 커브로 나타내게 된다.

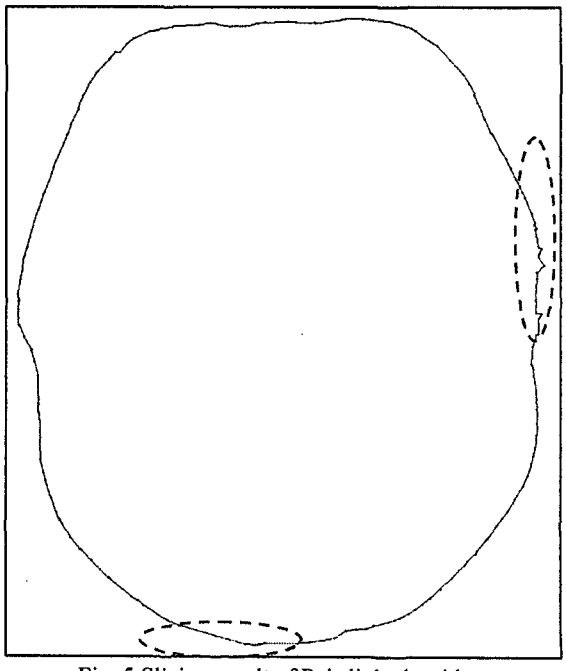

Fig. 5 Slicing result of Pair-link algorithm

3. 새로운 알고리즘의 접근

3.1 기존 알고리즘의 문제점

우리는 기존 연구의 한계점에 대한 고찰과 문제의 본질적 특성을 유심히 관찰함으로써 새로운 알고리즘의 가능성을 찾아볼 수 있다. 절단커브 계산 및 도시에서 제기되는 문제점은 다음과 같다.

첫째, 절단 평면과 교차하는 점군의 점은 절단 커브의 계산에 매우 부족하다. 때문에 사용자 입력의 허용공차 안의 점들을 추출하며 이때 절단커브 도출 시, 보다 정확한 결과를 얻기 위해 절단 평면에서 가까운 점을 더 영향력 있는 점으로 반영해야 한다.

둘째, 절단 커브는 부드러운 곡선 형태로 나타내어져야 한다. 원래 절단 형상이 지그재그 형태일 수도 있으나 곡선이나 직선으로 나타내어질 부분에서는 Fig. 3 과 같은 떨리는 커브가 아닌 매끄러운 곡선 이어야 한다. 따라서 점 밀도가 높은 부분에서 특정조건 없는 추적으로 인해 커브가 떨리는 것을 지양하고 첫째 조건인 가까운 점 위주로 추적하는 동시에 부드러운 방향성 추적의 제약 조건을 가져야 할 것이다.

셋째, 점군에 따라 또는 점군의 특정 부분에 따라 다른 점 밀도에 대해 강건하게 절단 커브를 도시할 수 있어야 한다. Pair-link 알고리즘의 경우 점 밀도가 높은 부분에서는 비교적 안정적인 결과를 보여 주나 밀도가 낮은 부분에서는 반대편 공차 공간에서 자신의 짝을 찾지 못한 점들이 다수 발생하여 절단 커브가 끊기거나 원래의 형상이 생략되는 경우가 있다. 따라서 새로운 알고리즘은 이런 다양한 점 밀도에서도 일관된 절단 커브를 도출해 낼 수 있어야 할 것이다.

3.2 문제 해결을 위한 설계 변수

앞서 언급한 기존 알고리즘의 한계점을 극복하기 위해 우리는 두 가지 설계 변수를 생각할 수 있다.

첫째, 각 점과 절단 평면과의 거리에 따라 가중치를 부여한다. 임의 점이 절단 평면과 가깝다면 이 점은 절단 커브 도출에 정확한 정보로 작용 할 것이다. 이를 위해 각 점을 중심으로 하고 절단공차를 반지름으로 하는 구를 만든다. 각 구와 절단 평면과의 교원의 반지름은 절단 평면과의 거리를 반영할 수 있다.

둘째, 각 구는 자신과 교차하는 구를 자신의 노드로 설정 하는 Node-Sphere diagram 을 구성한다. 다음 절에서 자세히 설명될 이 노드는 다양한 밀도 차이로 분포하는 점군에서 안정적인 절단 커브를 생성할 수 있는 장치로 작용한다. 즉, 점들이 드물게 분포하더라도 반지름을 가진 구는 거리가 떨어진 점들과의 연결고리를 이어 줄 수 있다.

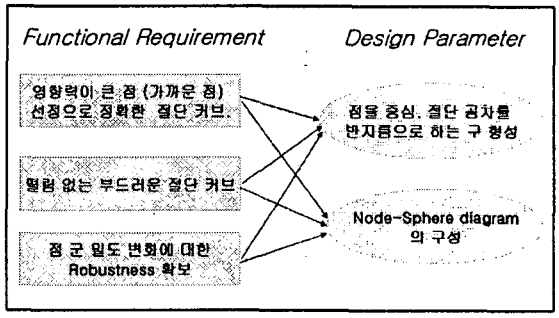

Fig. 6 Design parameter for new algorithm

4. 새로운 알고리즘

4.1 Node-sphere diagram

제안하는 알고리즘의 첫 단계는 각 점을 중심, 절단 공차를 반지름으로 하는 구를 형성하는 것이다. 절단 공차는 점군을 측정한 레이저 스캐너의 resolution 에 따라 다르게 설정 될 수 있으나 실험에 의하면 대부분의 점군은 0.2mm 의 절단 공차에서 안정적인 결과를 나타내 주고 있으며 제시하는 Node-Sphere 알고리즘 또한 이런 resolution 차이의 영향을 최소화하는 것에 도움을 주고 있다. Node-Sphere diagram 은 다음과 같이 구성한다.

Node-Sphere Diagram

// 입력 : 점군 데이터, 절단 평면, 절단 공차

// 출력 : 절단 커브를 연결하는 점 들.

Step 1) 사용자 지정 절단 평면 및 공차(2δ) 입력

Step 2) 절단 평면과 공차 안에 놓인 점들의 추출

distance(*Pi*, slicing plane) <= δ

Step 3) 반지름= 2δ, 각 점이 중심인 구 형성

Step 4) For (각 구에 대해서...)

Step 4-1) 자신과 교차하는 구 들을 노드로 저장

distance(center i, center j) < $Ri + Rj$

Step 4-2) Node 중에서 가장 가까운 구를 선택

Step 4-3) 선택한 구와 중심을 연결

Step 4-4) 연결선과 절단 평면과의 교점 계산
Step 4-5) 교점 저장

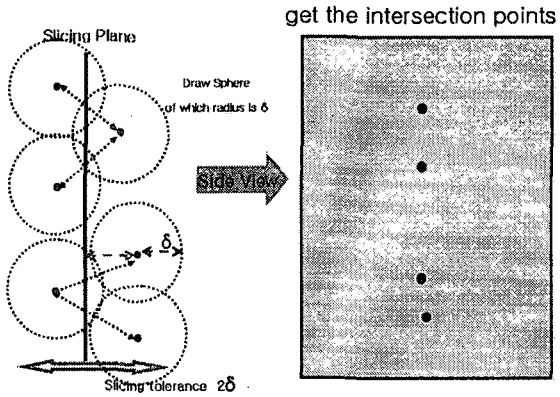

Fig. 7 Intersection Point from Node-sphere diagram

위의 Fig. 7 은 Node-sphere diagram 에 의해 생성된 교점들을 나타낸다. 교차하는 구 들을 연결한 Node-sphere 는 구 중에서 가장 가까운 구를 선택함으로써 앞 장에서 살펴 본 새로운 알고리즘의 조건 중 정확한 절단 커브의 도출을 지원한다. 또 구의 반지름으로 인해 점 사이의 매개가 이뤄짐으로써 점과 점들 사이의 거리가 멀더라도, 즉 점 밀도가 낮다 하더라도 안정적인 절단 커브의 도시를 보장한다.

한편, 절단면과 평행한 평면에 놓이는 매우 규칙적인 점들의 경우, 절단면과 교차하는 구가 한쪽 공간에만 편중되어 교점이 생기지 않을 수도 있다. 이런 경우를 판별하여 국지적으로 절단공차를 늘이는 방법 등의 해결책은 본 알고리즘을 보완하기 위해 향후 연구되어야 할 것이다.

4.2 Delaunay triangulation 을 이용한 연결

위의 Node-sphere diagram 에 의해 도출된 점들은 최종 절단 커브를 잇는 점들을 나타낸다. 이제 남은 과정은 이 점들을 적절히 연결하는 것이다. 기존 연구의 경우 절단 커브를 구성하는 점들을 추출 한 후, 가까운 점 순서로 연결을 하고 있다. 그러나 이 방법은 점의 배열에 따라 때로 원치 않는 방향으로의 연결이나 폐 곡선 생성, 또는 로컬 종결을 야기시킨다. 따라서 본 논문은 추출된 점을 연결 할 때에 최외각 점들을 적절히 연결하기 위해 delaunay 삼각화 방법을 제시한다.[8] 절차는 다음과 같다.

Step 1. 교차점들의 Delaunay triangulation

4.1 절에 의해 생성된 점들이 Fig. 8(a)와 같다고 하자. 절단 커브는 하트 모양을 나타낸다고 할 때 이 점들로 delaunay 삼각화 한 결과는 Fig. 8(b)와 같다.

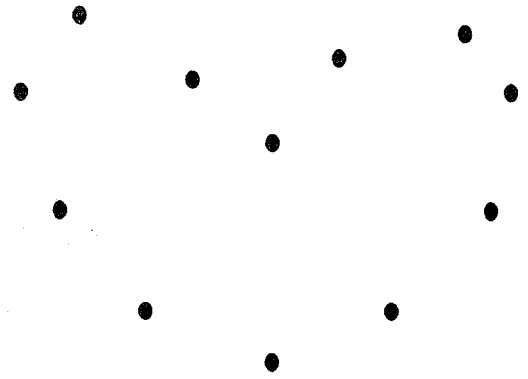

Fig. 8(a) Intersection points from the Node-sphere diagram

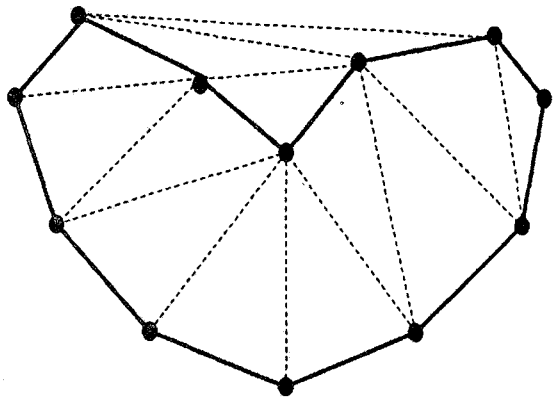

Fig. 8(b) Delaunay triangulation with intersection points

Fig. 8(b)에 나타난 것처럼 교차점들을 Delaunay 삼각화 한 결과는 우리가 원하는 외곽 연결 커브가 아니다.

Step 2. Voronoi 점들을 포함한 delaunay triangulation (삼각화)

우리가 원하는 파란(실선) 선으로 연결된 절단

커브를 얻기 위해서는 위 삼각형들의 외심인 voronoi 점들을 얻은 후(Fig. 8(c)의 붉은 점), 원래 교차점과 생성된 voronoi 점들을 합해서 delaunay 삼각화를 한다. 그 결과 생성된 삼각형의 선분 중에서 양 끝 점이 원래 점(검은 점)으로 이뤄 진다면 그 선분을 나타냄으로써 절단 커브를 연결하는 과정을 마칠 수 있다(Fig. 8(d)).

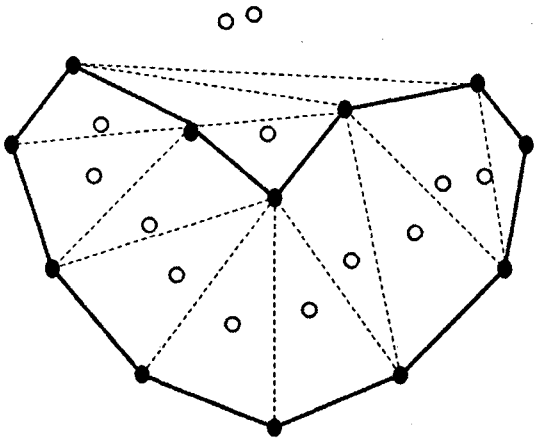

Fig. 8(c) Voronoi vertices of triangles

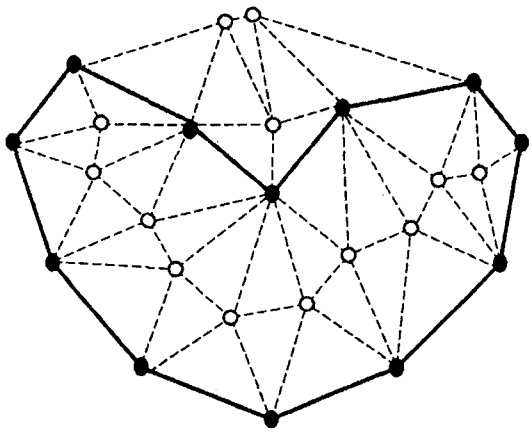

Fig. 8(d) Delaunay triangulatin of total points

지금까지 기술한 새로운 알고리즘을 Fig. 1 의 절단면 A-B 에 적용한 결과는 Fig. 9 와 같다. 그림에서 볼 수 있듯 절단면이 끊기거나 생략되지 않았으며 또한 부드러운 곡선 형태로 매끄럽게 나타난다.

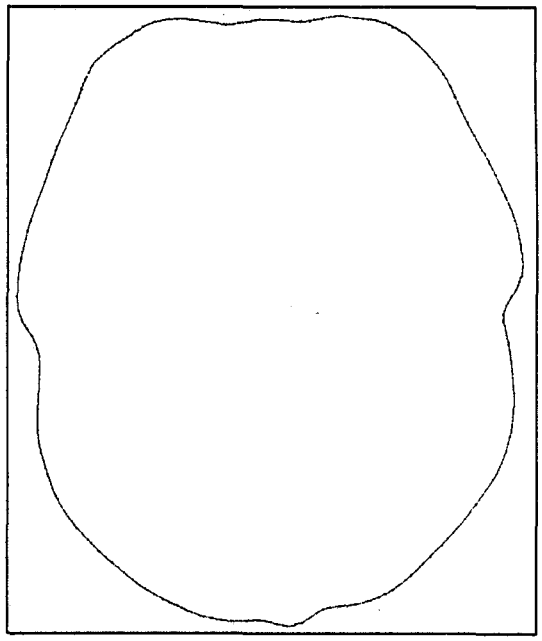

Fig. 9 Slicing result of proposed algorithm

5. 절단 커브의 정확도 측정

생성된 절단 커브가 육안으로 확인해서 부드럽고 그럴 듯 하게 보인다 하더라도 실제 절단면과의 오차가 크다면 의미는 없을 것이다. 따라서 본 장에서는 기존의 알고리즘과 제안되는 알고리즘이 실제 절단면에 얼마나 정확한 커브를 생성하는가를 알아보고자 한다. 점군의 실제 절단면은 알려지지 않은 값이며 우리는 두 가지 방법으로 실제 절단면과 알고리즘이 도출한 절단 커브와의 오차를 계산할 수 있다.[9]

첫째, 절단 평면에 극히 가까운 점과 절단 커브와의 평균 거리를 계산한다. 여기서는 절단 공차 내 412 개의 모든 점들 중에서 절단 평면과의 거리가 0.02mm 이내인 71 개의 점과 각 절단 커브와의 거리를 계산하였다. 절단 평면에 가까운 점일수록 실제 절단면을 반영하는 점임으로 이 평균 거리가 작다는 것은 그만큼 실제 절단면에 가까운 커브임을 의미한다.

두 번째 방법으로 절단 공차 내(여기서는 0.2mm)에 속하는 모든 점(412 개)들과 절단 커브와의 거리 평균을 구하는 방법이다. 한편, 거리는 점과 절단 커브의 선분 중 가장 가까운 선분과의 수직 거리로 정의한다.

이렇게 계산한 절단 커브와 점들과의 평균 거리는 Fig. 10 에 나타나 있다. 여기서 볼 수 있듯 제안하는 알고리즘은 끊기지 않으며 부드러운 연결을 보일 뿐만 아니라 실제 절단면에도 굉장히 근접한다는 것을 알 수 있다.

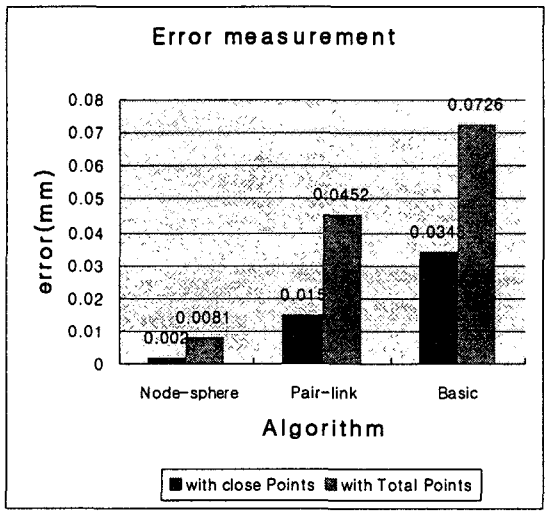

Fig. 10 Measured errors of different algorithms

6. 결 론

본 논문은 쾌속조형 제조를 위해 형상 모델에서 측정된 점군의 절단 커브 계산 및 도시에 관한 내용을 기술한다. 표면 재구성 혹은 STL 파일 변환 후 절단 커브의 도출은 중간 단계의 의한 통제 불가능한 오차의 발생 및 과도한 계산이 요구된다. 따라서 논문은 점군으로부터 곧바로 절단 커브를 생성하는 알고리즘을 제시 함으로써 절단 커브의 정확성과 계산시간의 감소에 기여한다. Node-sphere diagram 에 의해 절단 커브에 영향력 있는 점을 비중 있게 반영하며 밀도 차이에 강건한 절단 커브 점을 생성한다. 이렇게 생성된 점은 Voronoi vertices 와 Delaunay 삼각화를 이용하여 자연스러운 곡선으로 연결되어 최종적인 절단 커브를 완성한다.

한편, 논문의 실험에서 적용한 0.2mm 의 절단 공차는 측정기의 resolution 이나 동일 부분의 반복 측정 횟수에 따라 다르게 설정되며 일반적인 공차 결정에 관한 이슈는 향후 연구되어야 할 것이다.

23 만여 개의 점을 가진 Fig. 1 점군의 여러 층에 대해 논문의 알고리즘을 적용하여 도출한 결과는 Fig. 11 과 같다.

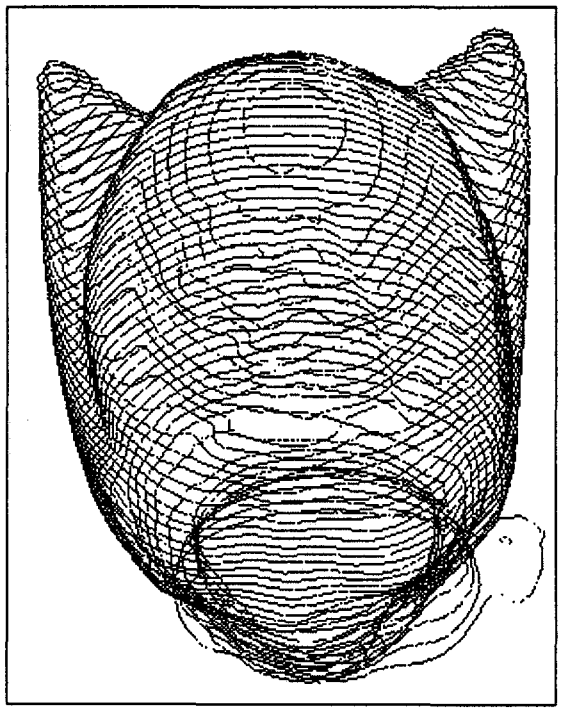

Fig. 11 Sliced multiple layer of a point cloud model

참고문헌

1. Varady, T., Martin, R. R. and Cox, J., "Reverse engineering of geometric models-an introduction," CAD, Vol. 29, No. 4, pp. 255 - 268, 1997.
2. Li, L., Schemenauer, N. and Peng, X., "A reverse engineering system for rapid manufacturing of complex objects," Robotics Comput. Integr. Manufact., Vol. 18, No.1, pp. 53 - 67, 2002.
3. Lee, K. H. and Woo, H., "Direct integration of reverse engineering and rapid prototyping," Comp. Ind. Engng., Vol. 38, No. 1, pp. 21 - 38, 2000.
4. Chen, Y. H., Ng, C. T. and Wang, Y. Z., "Generation of a STL file from 3D measurement data with user-controlled data reduction," Int. J. Adv. Manufact. Technol., Vol. 15, No. 2, pp. 127 - 131, 1999.
5. Lee, S. H., Kim, H. C., Hur, S. M. and Yang, D. Y., "STL file generation from measured point data by segmentation and Delaunay triangulation," CAD, Vol. 34, No. 10, pp. 691 - 704, 2002.

6. Liu, G. H., "Segmentation of cloud data for reverse engineering and direct rapid prototyping," Master of Engineering thesis, National University of Singapore, 2001.
7. Sun, Y., Guo, D. and Jia, Z., "B-spline surface reconstruction and direct slicing from point clouds," Int J. Adv. Manuf. Techonol., Vol. 27, No. 9, pp. 918 - 924, 2006.
8. Joe, B., "Three-dimensional triangulations from local transformations," SLAM Journal on Scientific and Statistical Computing, Vol. 10, No. 4, pp. 718 - 741, 1989.
9. Wu, Y. F., Wong, Y. S., Loh, H. T. and Zhang, Y. F., "Modeling cloud data using an adaptive slicing approach," CAD, Vol. 36, No. 3, pp. 231 - 240, 2004.