한국 CAD/CAM 학회 논문집
제9권 제3호 2004년 9월 pp. 203-211

학술논문

옵셋팅을 위한 정규 삼각망 추출

정원형*, 정춘석*, 신하용*, 최병규*

Extracting a Regular Triangular Net for Offsetting

Jung, W.H.*, Jeong, C.S.*, Shin, H.Y.* and Choi, B.K.*

ABSTRACT

In this paper, we present a method of extracting a regular 2-manifold triangular net from a triangular net including degenerate and self-intersected triangles. This method can be applied to obtaining an offset model without degenerate and self-intersected triangles. Then this offset model can be used to generate CL curves and extract machining features for CAPP. The robust and efficient algorithm to detect valid triangles by growing regions from an initial valid triangle is presented. The main advantage of the algorithm is that detection of valid triangles is performed only in valid regions and their adjacent self-intersections, and omitted in the rest regions (invalid regions). This advantage increases robustness of the algorithm. As well as a k-d tree bucketing method is used to detect self-intersections efficiently.

Key words : Regular triangular net, Regular triangular net extraction, Offset, Self-intersection, k-d tree

1. 서 론

본 연구에서는 offset 등에 의해서 기하학적으로 변형된 삼각망으로부터 퇴화 삼각형과 self-intersection에 의해 형성된 무효 영역이 제거된 2-mainfold 경계를 구하는 정규 삼각망 추출(regular triangular net extraction) 방법을 제안한다. 이 논문에서는 정규 삼각망을 퇴화 삼각형과 self-inter-section이 존재하지 않는 2-manifold 경계의 삼각망으로 정의한다. 또한 단일 부피를 정의하는 2-manifold 경계가 기하학적으로 변형된 삼각망에 대한 정규 삼각망 추출로 연구 범위를 한정한다.

정규 삼각망 추출(regular triangular net extraction) 방법은 삼각망 모델 기반의 CAM 시스템에서 CL (cutter location) 곡선을 생성하거나 CAPP를 위한 machining feature 정의(Choi *et al.*[1])를 위해서 삼각망 마스터 모델로부터 CL 곡면 모델인 offset 모델을 계산하는데 적용할 수 있다.

퇴화 삼각형과 self-intersection을 제거하는 정규 삼각망 추출은 intersection을 계산하고 그것으로부터 필요한 영역을 추출해낸다는 점에서 Boolean operation (Cardan and Perrin[2])과 유사하다. 하지만 Boolean operation은 2개의 solid 개체를 대상으로 하기 때문에 intersection의 계산과 필요한 영역의 추출은 본 연구의 정규 삼각망 추출보다 용이하게 수행될 수 있다.

의류 시뮬레이션과 같은 변형 가능 물체(deformable object)의 시뮬레이션[3-6]은 self-collision 탐색과 collision response 문제를 다룬다. 이것은 self-collision을 효과적으로 탐색해야 한다는 점에서 본 연구의 문제와 유사하다. 하지만 self-collision을 발생시키는 부분을 제거하는 것이 아니고 그 위치를 재정의 해주는 문제를 다루며, 시뮬레이션을 위한 다중 프레임 계산 과정에서 인접 프레임간에 self-collision 계산은 프레임들 간의 유사성을 이용하여 계산의 효율성을 높일 수 있다는 점에서 차이가 있다.

정규 삼각망 추출은 먼저 무효 삼각형을 탐색해서 제거하는 방식을 생각할 수 있다. Offset 등의 변형에 의해 self-intersection이 발생하는 영역에서는 삼각형들이 매우 복잡하게 얽혀있는 부분이 많고 이러한 영역 내부의 self-intersection segment들은 유효 삼각형과 제거되어야 할 무효 삼각형의 경계 역할을 하지 않고 무효 삼각형에 의해서만 형성된 것들이 대부분이기 때문에 무효 삼각형의 탐색과 판단은 쉽지 않다. 따라서 본 연구에서는 먼저 유효 삼각형들을 탐색한 후에 탐색되지 않은 삼각형들을 제거하는 방식을 사

*KAIST 산업공학과
- 논문투고일: 2003. 11. 08
- 심사완료일: 2004. 01. 05

용하여 무효 영역 내에서는 불필요한 유효, 무효 삼각형 판단을 수행하지 않으면서 단순화된 유효, 무효 삼각형의 판단이 가능한 방법을 제안하고자 한다.

2. 전체 수행 절차

Fig. 1에서와 같이 다음의 5단계를 거쳐서 정규 삼각망을 추출한다.

Step 1. 퇴화 삼각형 제거

퇴화 삼각형은 면적이 zero tolerance 이내인 삼각형으로 삼각망의 변형으로 인해 발생할 수 있다. 퇴화 삼각형의 제거는 이후의 각 단계의 알고리즘 수행을 위해 우선적으로 수행되어야 한다. 퇴화 삼각형 t_{zero}는 최소 길이 edge $l < \varepsilon_1$(Zero length tolerance)인 삼각형과 최소각 $\alpha < \varepsilon_2$(Zero angle tolerance)인 삼각형으로 정의한다. 최소 길이 edge $l < \varepsilon_1$인 삼각형은 Fig. 2(a)와 같이 edge collapse에 의해 제거되고, 최소각 $\alpha < \varepsilon_2$(Zero angle tolerance)인 삼각형은 Fig. 2(b)와 같이 최대 길이의 edge를 swapping하여 제거한다. 삼각망의 모든 삼각형에 대해서 퇴화 삼각형 테스트를 수행한다. 각 퇴화 삼각형의 제거 후에는 주변 삼각형들의 위상(topology) 정보가 수정되어야 한다.

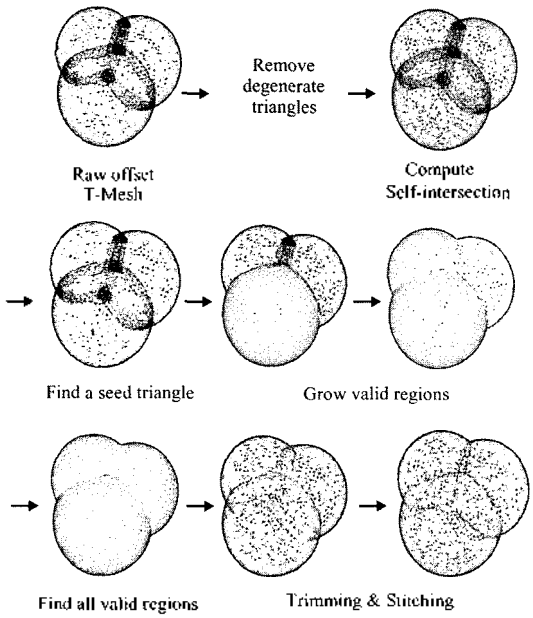


Fig. 1. An explanatory example of overall procedure.

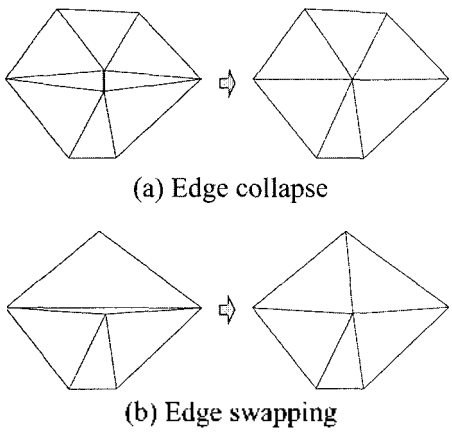


Fig. 2. Deletion of degenerate triangles.

Step 2. Self-intersection 계산

두 삼각형간의 intersection 테스트를 통해서 intersection segments들을 계산하고 리스트에 저장한다. 이때 intersection segment는 양 끝점에 대한 위치(position)와 intersection을 만든 두 삼각형 정보를 저장한다. 상세한 계산 방법은 3장에서 설명한다.

Step 3. 초기 유효 삼각형 탐색

정규 삼각망 추출을 통해서 남게 될 유효 영역의 삼각형 중에 하나를 탐색하는 과정이다. 유효 영역이 항상 최외곽에 존재한다는 것이 보장되는 경우에는

1. x, y, z의 최대 또는 최소 값을 가지는 vertex 중 임의의 한 vertex v를 선택
2. v에 연결된 삼각형 중에서 임의의 삼각형 t 선택
3. t가 다른 삼각형과 intersection이 없는 경우에는 t를 초기 유효 삼각형으로 선택하고, t가 다른 삼각형과 intersection이 있는 경우에는 t를 sub-triangulation한 후 v에 연결된 sub-triangle 중 임의의 한 삼각형 t_{sub}를 유효 삼각형으로 선택

한다. 유효 영역이 최외곽에 존재하지 않는 경우에는 사용자의 선택에 의해서 초기 유효 삼각형을 결정한다.

Step 4. 유효 영역 탐색

하나의 유효 삼각형에 인접한 self-intersection이 없는 삼각형도 역시 유효 삼각형이다. 유효 영역의 탐색은 Fig. 1에서 보는 것처럼 Step 3에서 선택된 초기 유효 삼각형에서부터 self-inter-section segment를 가진 삼각형을 만날 때까지 인접 삼각형으로 유효 삼각형들의 영역을 확장해 나가는 방식이다.

단일 부피를 정의하는 2-*manifold* 경계가 기하학적으로 변형된 삼각망의 경우 하나의 유효 영역은 항상 self-intersection segment들로 그 경계가 한정된다. 따

라서 Fig. 1의 유효 영역 탐색 과정에서 보는 것처럼 유효 영역을 한정하는 self-intersection segment를 통해서 인접한 다른 유효 영역으로 넘어갈 수 있다. 이러한 과정을 반복해서 유효 영역을 한정하는 모든 self-intersection segment에서 유효 삼각형이 탐색되면 유효 영역 탐색이 완료된다.

이러한 방식의 유효 영역 탐색은 무효 영역에 속하는 self-intersection segment들에서는 유효/무효 삼각형 판단 및 탐색을 전혀 수행하지 않는다. 유효 영역 탐색에 대해서는 4장에서 자세하게 설명한다.

Step 5. Trimming & Stitching

Step 4에서 얻어진 유효 영역의 경계를 한정하는 self-intersection들에서 sub-triangulation을 통한 trimming과 self-intersection segment를 기준으로 segment에 인접한 sub-triangle들의 인접 관계를 설정해주는 stitching을 수행하고 탐색되지 않은 무효 삼각형과 무효 sub-tri-angle들을 제거하는 과정을 수행하면 된다.

Step 4의 유효 영역 탐색 과정에서는 4.2절과 4.3절에서 설명할 것처럼 유효 영역의 경계를 한정하는 self-intersection segment를 포함하는 삼각형들에서 sub-triangulation과 이를 통해 생성된 sub-triangle들에 대한 유효 삼각형 탐색이 수행되기 때문에, 이 단계에서는 단순히 이러한 self-intersection segment를 기준으로 seg-ment에 인접한 sub-triangle들의 인접 관계를 설정해 주고 탐색되지 않은 무효 삼각형과 무효 sub-triangle들을 제거하는 과정만을 수행함으로써 정규화된 삼각망을 얻을 수가 있다.

3. Self-intersection 계산

삼각망 모델의 self-intersection을 계산하기 위해서는 삼각형들간에 intersection 테스트가 필요한데 삼각형들의 모든 쌍에 대해서 intersection을 테스트한다면 $O(N^2)$의 complexity가 요구된다. 따라서 지역화(localization)하여 테스트할 삼각형들의 쌍을 제한할 필요가 있다. 이와 관련하여 삼각망 모델을 Self-intersection이 존재하지 않는 영역으로 clustering하여 계산의 효율성을 높인 연구(Volino *et al.*[3])가 있다. 본 연구에서는 k-d tree[7]를 사용하여 삼각형들을 bucketing한 후에 k-d tree의 leaf node내에 있는 삼각형들간에만 intersection 테스트를 수행하는 방법을 사용하였다. 아래의 단계를 거쳐 삼각망 모델을 k-d tree로 bucketing한다(Fig. 3).

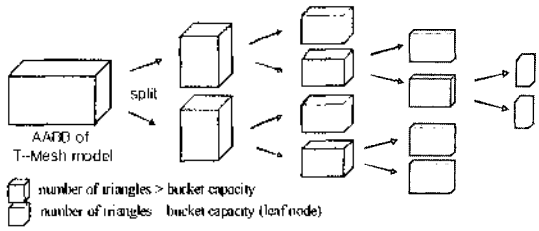


Fig. 3. k-d tree bucketing.

1. x, y, z 축에 평행하도록 삼각망 모델의 bounding box를 계산하고, 모든 삼각형을 bounding box의 삼각형 리스트에 저장한다.
2. bounding box의 각 모서리의 길이를 비교, 가장 긴 방향으로 bounding box를 2등분하여 2개의 하위 bounding box를 생성한다.
3. 상위 bounding box의 삼각형 리스트에 저장된 모든 삼각형에 대해서 각 하위 bounding box와 intersection이 발생하는 삼각형을 해당 bounding box의 삼각형 리스트에 삽입한다.
4. 하위 bounding box를 자신에 속한 삼각형들을 포함하는 최소 크기로 조정한다.
5. 자신에 속한 삼각형의 개수가 bucket capacity 이상인 하위 bounding box에 대해 2,3,4,5번 과정을 반복한다.

삼각망 모델을 k-d tree로 bucketing한 후에는 k-d tree의 leaf node별로 그것에 속한 모든 삼각형간에 inter-section 테스트(Moller[8])를 수행한다.

Fig. 4는 k-d tree를 사용하는 경우의 계산 시간을 그래프로 표현한 것인데 거의 선형적으로 증가한다는 것을 보여주고 있다. 하지만 Fig. 5에서 볼 수 있는 것처럼 k-d tree의 bucket capacity에 따라서 동일 모델에 대한 self-intersection 계산 시간이 달라진다. 최

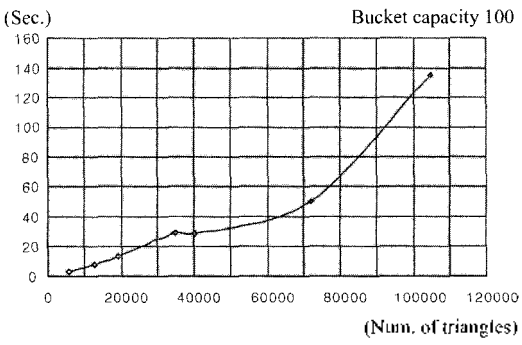


Fig. 4. Analysis of computation time (model size).

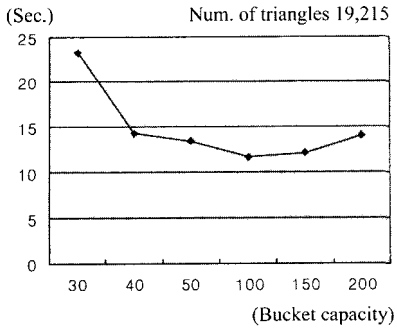


Fig. 5. Analysis of computation time (bucket capacity).

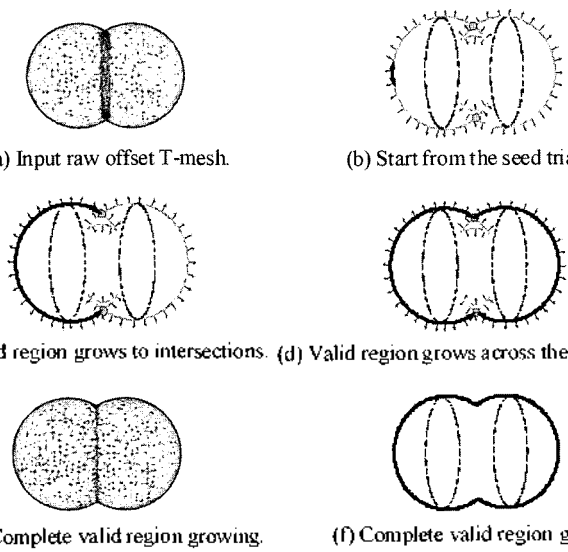
적의 bucket capacity는 모델의 크기 및 형상에 따라 다양하고 최적값의 결정은 어려운 일이다. 또한 삼각형의 크기가 불균일한 경우 크기가 큰 삼각형은 여러 개의 leaf node에 포함되기 때문에 균일한 크기의 모델보다 계산 시간이 증가하게 된다.

4. 유효 영역 탐색

4.1 유효 영역 탐색 방법

단일 부피를 정의하는 2-manifold 경계가 기하학적으로 변형된 삼각망의 경우 하나의 유효 영역은 항상 self-intersection segment들로 그 경계가 한정된다. 따라서 모든 self-intersection segment가 얻어진 상태에서 초기 유효 삼각형에서부터 self-intersection을 가진 삼각형을 만날 때까지 인접 삼각형으로 유효 삼각형들의 영역을 확장해 나가면 self-intersection segment에 의해서 한정되는 하나의 유효 영역을 탐색할 수 있다. 이때 유효 영역의 경계를 이루는 self-intersection segment는 또 다른 유효 영역과 접해있는 부분이 될 수 있다. 따라서 이러한 self-intersection segment를 통해서 인접한 다른 유효 영역으로 넘어갈 수 있다. 이러한 과정을 반복해서 유효 영역을 한정하는 모든 self-intersection segment에서 유효 삼각형이 탐색되면 유효 영역 탐색이 완료된다. 이 과정에서 탐색되지 않은 삼각형들은 모두 무효 영역에 속하는 삼각형들이 된다.

이 과정은 Fig. 6에서와 같이 단순화된 형태로 설명될 수 있다. (a)는 offset에 의해 변형된 삼각망 모델이고 (f)는 추출된 정규 삼각망이다. 가운데 부분에서 self-intersection에 의해 생긴 무효 영역이 제거되었음을 알 수 있다. (b)~(e)는 유효 영역의 탐색과정을 (a)에 대한 2D 단면 형태로 단순화하여 설명한 그림이다. (b)에서 단면 경계면에 수직한 화살표들은 삼각형의 법선 방향, 회색점은 self-intersection segment를 나타내며, 경계면 상의 검은색의 굵은 화살표는 초기 유효 삼각형을 나타내고 있다. (c)는 (b)의 유효 삼각형에서부터 intersection을 가진 삼각형을 만날 때까지 인접 삼각형으로 유효 삼각형들의 영역을 확장해 가서 self-intersection segment에 의해서 경계가 한정되는 하나의 유효 영역을 탐색한 결과이다. (d)는 (c)에서 탐색된 유효 영역의 경계를 이루는 한 self-intersection segment에서부터 인접한 다른 유효 영역



(a) Input raw offset T-mesh. (b) Start from the seed triangle.

(c) Valid region grows to intersections. (d) Valid region grows across the intersection.

(e) Complete valid region growing. (f) Complete valid region growing.

Fig. 6. Procedures of exploring valid regions.

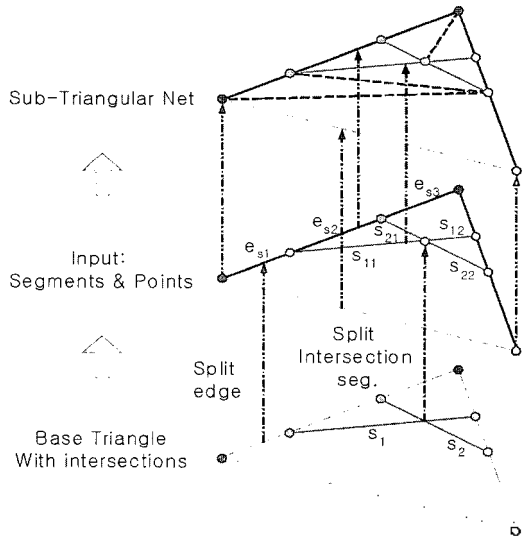


Fig. 7. Sub-triangulation.

을 탐색한 결과이다. 이 과정은 4.2절과 4.3절에서 자세히 설명된다. (d)의 과정이 반복되어 모든 유효 영역의 삼각형이 탐색되면 (e)에서와 같이 무효 영역을 제거된 정규 삼각망 모델을 얻게된다.

4.2 Sub-triangulation

유효 영역 탐색 과정에서 유효 영역의 경계를 이루는 self-intersection segment를 포함하는 삼각형들은 Fig. 7과 같이 삼각형의 세 edge와 intersection segment들을 가지고 sub-triangulation을 수행한다. 본 연구에서는 2D constrained Delaunay triangulation (Ruppert[9]) 방법을 사용 하였다.

이러한 sub-triangulation 결과 탐색된 유효 삼각형에 접해있는 sub-triangle은 역시 유효 삼각형이 되며 4.1에서의 유효 영역 탐색과 마찬가지로 이 삼각형에서부터 intersection segment를 만날때까지 인접한 sub-triangle로 유효 영역을 확장함으로써 sub-triangulation영역에서 유효 삼각형을 탐색한다.

Intersection segment를 가지는 입력 모델의 삼각형인 base triangle은 각각 sub-triangulation 과정을 통해서 자신의 sub-triangular net을 가지게 되는데 이 과정은 다음과 같다(Fig. 7).

1. Base triangle T_p의 intersection segment $\{s_i | i = 1, ..., n\}$에 대해서 다음을 수행한다.
   (1) intersection segment s_i, s_j와의 교점을 계산, 분할, s_{i1}, s_{i2}, s_{j1}, s_{j2}, p_{ij}(intersection point) 생성
   (2) s_i, s_j를 공유하는$\{t_k\}$에 s_{i1}, s_{i2}, s_{j1}, s_{j2}, p_{ij}를 할당
2. t_p의 edge $\{e_i | i = 1, ..., 3\}$에 대해서 다음을 수행한다.
   (1) e_i상의 intersection point $\{p_j | j = 1, ..., m\}$들로 e_i를 분할, $\{e_{sk} | k = 1, ..., m+1\}$ 생성
   (2) e_i의 인접 삼각형에 $\{e_{sk}\}$를 할당
3. T_p의 sub-triangular net 생성한다.
   (1) 입력: base triangle vertex $\{v_i\}$, non split base triangle edge $\{e_i\}$, split edge $\{e_{sk}\}$, intersection segment $\{s_m\}$, intersection point $\{p_n\}$
   (2) 2D constrained Delaunay triangulation
4. Non split base triangle edge $\{e_i\}$, split edge $\{e_{sk}\}$, inter-section segment $\{s_m\}$와 sub-triangular net의 edge $\{e_{subi}\}$와의 대응 관계를 설정한다.

4에서 생성된 정보는 sub-triangulation영역에서 유효 삼각형 탐색과 함께 유효 영역 탐색이 완료된 후에 trimming & stitching단계에서 사용하게 된다.

4.3 Self-intersection segment에서 인접 유효 영역의 탐색

4.2절에서 설명한 것처럼 하나의 유효 영역이 탐색될 때 intersection segment를 포함하는 삼각형들은 sub-triangulation된 후 이 영역에서도 sub-triangle에 대한 유효 영역 탐색이 수행되어 self-inter-section segment들이 유효 영역의 경계가 된다. 이러한 self-intersection segment에서 인접한 다른 유효 영역을 탐색하게 된다.

Fig. 8의 (a)는 하나의 유효 영역 탐색이 완료된 상태이다. self-intersection segment s는 T_p과 T_c에 의해 생성되었다. (b)~(d)는 이 영역에 인접한 다른 유효 영역을 탐색하는 과정을 설명하고 있다. s를 포함하는 다른 삼각형 T_c가 sub-triangulation되면 s에는 T_p와 sub-triangle t_1, t_2와 T_c의 sub-triangle t_3, t_4가 접하게 된다.

t_1은 유효 삼각형으로 탐색된 상태이기 때문에 s에 접한 T_c의 sub-triangle t_3, t_4중 하나가 다른 유효 영역의 삼각형 후보가 된다. 이때 삼각망 모델의 외부를 나 타내는 sub-triangle의 법선 방향이 일정하게 유지되도록 하기 위해서 (b)에서와 같이 t_4를 유효 삼각형으로 선택한다. 그러면 다시 t_4에서부터 인접 sub-triangle들로 유효 영역을 확장해 나가면서 또 하나의 유효 영역을 탐색하게 된다. 이 과정은 4.1, 4.2절의

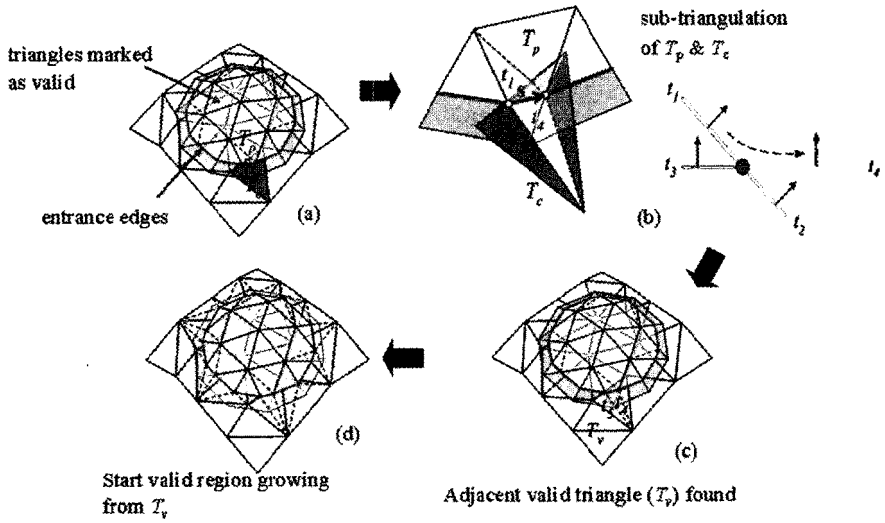
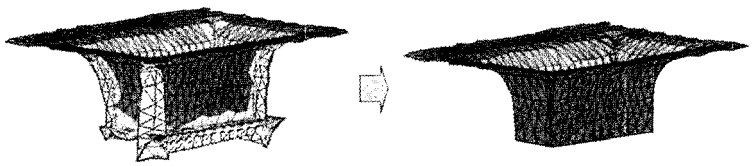


Fig. 8. Exploring an adjacent valid region.

유효 영역 탐색 과정이 동일하게 적용된다. (c)~(d)는 t_4에서 t_5, T_v로 유효 삼각형을 탐색해 나가는 과정을 보여주고 있다.

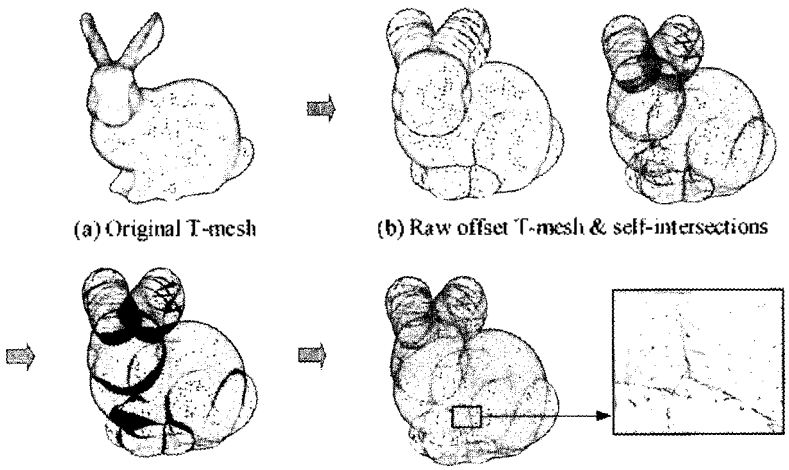
5. 적용 결과

Fig. 9는 유효 영역을 탐색한 후에 탐색되지 않은



(a) After valid region growing (b) Trimming & Stitching

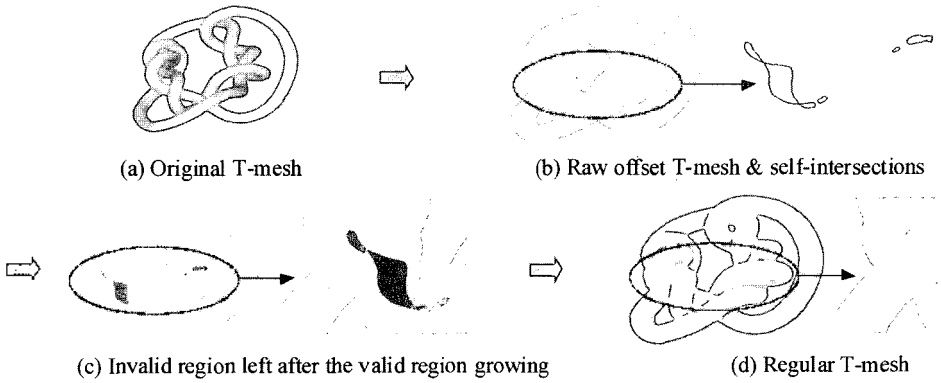
Fig. 9. An example of regular triangular net extraction.



(a) Original T-mesh (b) Raw offset T-mesh & self-intersections

(c) Invalid region left after the valid region growing (d) Regular T-mesh

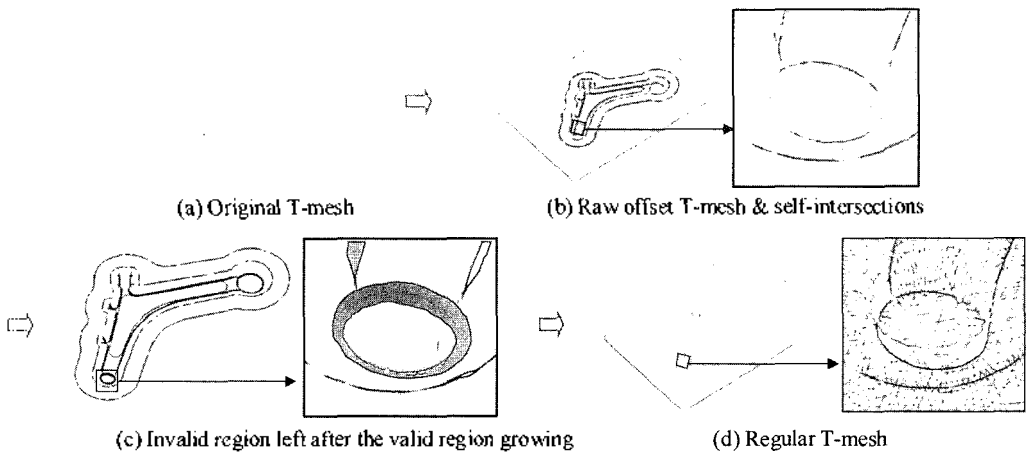
Fig. 10. Bunny.



(a) Original T-mesh (b) Raw offset T-mesh & self-intersections

(c) Invalid region left after the valid region growing (d) Regular T-mesh

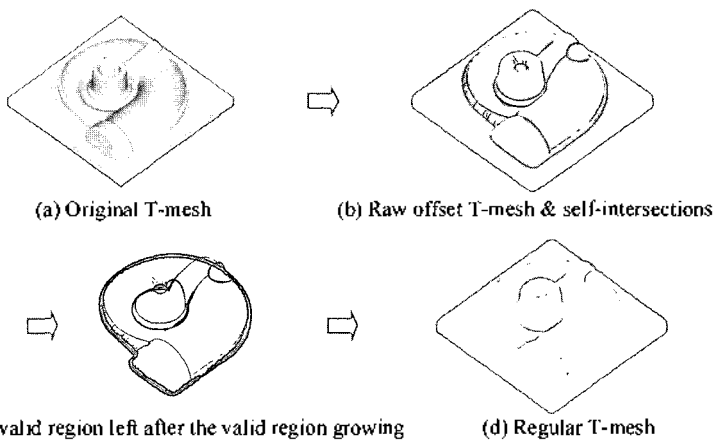
Fig. 11. Kont.



(a) Original T-mesh (b) Raw offset T-mesh & self-intersections

(c) Invalid region left after the valid region growing (d) Regular T-mesh

Fig. 12. C-arm lower.



(a) Original T-mesh (b) Raw offset T-mesh & self-intersections

(c) Invalid region left after the valid region growing (d) Regular T-mesh

Fig. 13. Pump.

무효 삼각형들을 제거하는 Trimming & stitching 예를 보여주고 있다.

Fig. 10~13은 초기 삼각망 모델과 이를 row offset 한 모델에서 self-intersection를 계산한 후 유효 영역을 탐색한 후에 남겨진 무효 영역과 무효영역을 제거한 후에 Trimming & stitching을 하여 얻어진 정규

Table 1. The experimental results

Name	# of Triangles	# of Intersection Segments	Time (sec)		
			Self-intersection Calculation	Region growing & Stitching	Total
Bunny	15,224	4,253	2	0.454	2.454
Knot	150,776	44,211	11.766	0.656	12.422
C-arm lower	173,880	14,141	19.266	2.591	21.857
Pump	256,672	35,401	50.781	3.140	53.921

삼각망을 생성한 예들을 보여준다.

Table 1에서의 적용 예제에 대한 수행 시간을 보여주고 있다. Table 1에서 볼 수 있는 것처럼 self-intersection 계산 시간이 수행 시간의 대부분을 차지함을 알 수 있다. 펜티엄4 1.8GB, 1GB의 PC에서 테스트하였다.

6. 결론 및 추후 과제

단일 부피를 정의하는 2-manifold 경계가 가하학적으로 변형된 삼각망으로부터 퇴화 삼각형과 self-intersection에 의해 형성된 무효 영역이 제거된 2-manifold의 삼각망을 추출하기 위해서 하나의 유효 삼각형으로부터 영역을 확장해나가는 방법을 제안하였다. 이 방법은 무효 영역 내에서는 불필요한 유효, 무효 삼각형 판단을 수행하지 않으면서 단순화된 유효, 무효 삼각형의 판단이 가능하기 때문에 효율성과 함께 robustness를 높이고 구현이 용이하다는 장점이 있다.

모델에 따라서 하나의 초기 유효 삼각형으로 모든 유효 영역을 탐색하지 못할 수 있다. 초기 유효 삼각형 탐색 방법과 함께 이러한 점을 고려한 보다 일반화된 유효 영역 탐색 방법의 연구가 필요하다. 이와 함께 점, 면이 접하는 경우등의 특이 상황을 고려한 연구가 실제 응용 측면에서 중요하게 요구되기 때문에 이에 대해서 다양한 테스트를 통한 연구를 추가로 진행할 계획이다. 또한 5장의 적용 결과에서 볼 수 있는 것처럼 수행 시간의 대부분이 self-intersection 계산에 소요된다. 따라서 대용량 모델에 적용하기 위해서는 무효 영역으로 판단되는 부분에서는 가능한 한 self-intersection 계산 자체를 생략할 수 있는 방법의 개발이 필요하다. 그리고 이와 함께 CAD 모델에 대한 STL 파일처럼 삼각형의 크기의 불균일 정도가 큰 경우에는 계산 시간이 크게 증가하는데 이를 함께 고려한 self-intersection 계산 방법을 연구할 계획이다.

감사의 글

본 연구는 과학기술부 국가지정연구실 사업의 지원으로 수행되었다.

참고문헌

1. Choi, B. K., Kim, D. H. and Robert B Jerard, "C-space approach to tool-path generation for die and mould machining," *Computer-Aided Design*, Vol. 29, No. 9, pp. 657-669, 1997.
2. Cardan, Y. and Perrin, E., "An algorithm reducing 3D Boolean operatoins to a 2D problem: concepts and results," *Computer-Aided Design*, Vol. 28, No. 4, pp. 277-287, 1996.
3. Volino, P. and Thalmann, N. M., "Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity," *Computer Graphics Forum (EuroGraphics Proc.)*, Vol. 13, pp. 155-166, 1994.
4. Mezger, J., Kimmerle, S. and Etzmub, O., "Progress in collision detection and response techniques for closth animation", Proc. of the 10th Pacific Conference on Computer Graphics and Application, 2002.
5. Lau, R. W. H., Chan, O., Luk, M. and Li, F. W. B., "A collision detection framework for deformable objects," ACM VRST'02, November 11-13, 2002.
6. Provot, X., "Collision and self-collision handling in cloth model dedicated to design garments," *Graphics Interface*, pp. 177-189, May 1997.
7. Samet, H., "Applications of spatial data structures - computer grapichs, image processing and GIS," Addison-Wesley, 1990.
8. Moller, T., "A fasst triangle-triangle intersection test," *Journal of Graphics Tools*, Vol. 2, No. 2, pp. 25-30, 1997.
9. Ruppert, J., "A delaunay refinement algorithm for quality 2-Dimensional mesh generation," *Journal of Algorithms*, Vol. 18, No. 3, pp. 548-585, May 1995.
10. Choi, B. K. and Jerard, R. B., "Sculptured surface Machining - theory and applications", Kluwer Academic Publishers, 1998.

정 원 형

1998년 숭실대학교 산업공학 학사
2000년 KAIST 산업공학 석사
2000년~현재 KAIST 산업공학 박사과정 재학
관심분야: 형상모델링, CAD/CAM, CG, Reverse engineering, Mesh generation

정 춘 석

2001년 한양대학교 산업공학 학사
2003년 KAIST 산업공학 석사
2003년~현재 (주)DMS 연구소 연구원
관심분야: 형상모델링, CAD/CAM, CG, Reverse engineering

신 하 용

1985년 서울대학교 산업공학 학사
1987년 KAIST 산업공학 석사
1991년 KAIST 산업공학 박사
1991~1993년 LG 생산기술원 선임연구원
1993~1997년 (주)큐빅테크 이사
1997~2001년 DaimlerChrysler Senior specialist
2001년~현재 KAIST 산업공학과 교수
관심분야: 형상모델링, CAD/CAM, CG, DMU, VMS, Reverse engineering, BPM

최 병 규

1973년 서울대학교 산업공학 학사
1975년 KAIST 산업공학 석사
1982년 미국 Purdue대 생산공학 박사
1982년~현재 KAIST 산업공학과 교수
관심분야: 형상모델링, CAD/CAM, CAPP, 자동화 제조시스템 모델링 및 시뮬레이션, VMS, e-Business, BPM