한국 CAD/CAM 학회 논문집
제9권 제4호 2004년 12월 pp. 382-386

학술논문

가시 정보를 이용한 삼각망의 꼬임 찾기

박 상 철*

Finding Self-intersections of a Triangular Mesh by Using Visibility Maps

Park, S. C.*

ABSTRACT

This paper presents an algorithm for the triangular mesh intersection problem. The key aspect of the proposed algorithm is to reduce the number of triangle pairs to be checked for intersection. To this end, it employs two different approaches, the V-group approach and the space partitioning approach. Even though both approaches have the same objective of reducing the number of triangular-triangular intersection (TTI) pairs, their inherent characteristics are quite different. While the V-group approach works by topology (reduces TTI pairs by guaranteeing no intersection among adjacent triangles), the space partitioning approach works by geometry (reduces TTI pairs by guaranteeing no intersection among distant triangles). The complementary nature of the two approaches brings substantial improvement in reducing the number TTI pairs.

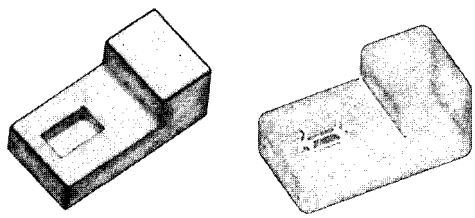
Key words : Triangular mesh, Self-intersections, Visibility map

1. 서 론

삼각망은 전통적으로 컴퓨터 그래픽스, 캐드캠 분야등에서 널리 사용되어 왔으며, 최근에는 게임과 영화에 이르기까지 다양한 분야에서 사용되고 있다[1]. 삼각망의 활용도가 높은 만큼 삼각망을 효율적으로 다루는 알고리즘을 개발하는 것은 매우 중요하고 또한 그 파급효과도 매우 크다. 본 연구의 목표는 주어진 하나의 삼각망에서 꼬임을 찾아내는 알고리즘을 개발하는 것이다. 삼각망의 꼬임을 찾는 알고리즘은 물체들의 충돌 및 공구경로 생성 등에서 유용하게 사용될 수 있다. 예를 들면 공구경로 생성을 위해서는 삼각망을 오프셋(Fig. 1)하는 것이 필요한데, 일반적으로 오프셋 후에 삼각망은 자체 꼬임을 가지게 된다. 이러한 자체 꼬임은 공구경로 생성시 유효하지 않으므로 찾아내어 제거하는 것이 필요한데 이때 삼각망의 꼬임을 찾는 알고리즘을 사용하는 것이 필요하다[1].

주어진 삼각망의 꼬임을 찾는 가장 쉬운 방법은 그 삼각망을 구성하는 각각의 삼각형들에 대해서 나머지 모든 삼각형들과 Triangle-to-Triangle Intersection (TTI)을 적용하는 것이다. 이 방식은 매우 간단하지만 삼각망을 구성하는 삼각형의 개수가 많은 경우에는 적합하지 않다. 왜냐하면 n개의 삼각형을 가지는 삼각망의 경우에 $n*(n-1)/2$번의 TTI를 수행하는 것이 필요하기 때문이다. 한번의 TTI를 수행하는 것에 관한 알고리즘이 아무리 효율적이라 하더라도 TTI 수행 횟수가 너무 많아지면 계산 시간이 폭발적으로 늘어날 수 있다. 따라서 삼각망 꼬임을 효율적으로 찾기 위해서는 TTI 횟수를 최소화 하는 것이 필수적이라 할 수 있다.

기존연구에서는 TTI 횟수를 줄이기 위해서 공간분할(Space Partitioning) 방법을 주로 사용하였다. 공간분할 방법의 예로는 Octree, BSP-tree, Tetrahedral Meshes 그리고 Regular Grid 방법 등을 예로 들 수



(a) 삼각망 (b) 옵셋 삼각망과 자체꼬임

Fig. 1. Triangular mesh offsetting & self-intersections.

*종신회원, 아주대학교 산업정보시스템공학부
- 논문투고일: 2004. 05. 19
- 심사완료일: 2004. 08. 03

있다[6]. 공간분할 방법의 기본 아이디어는 공간을 여러 개의 격자로 분할하고 각 삼각형들이 어떠한 격자에 포함되는지를 미리 계산해 둔다. 그리고 만약 어떠한 두 삼각형이 같은 격자에 포함되지 않는다면 그 두 삼각형은 서로 만나서 교차할 확률이 전혀 없으므로 TTI를 수행하지 않는다. 즉 동일 격자에 속하는 삼각형들에 대해서만 TTI를 수행함으로써 전체 TTI 횟수를 줄이고자 하는 것이다.

본 논문에서는 이러한 공간분할 방법과 더불어 삼각형들의 가시성 정보를 함께 활용함으로써 획기적으로 TTI 계수를 줄이는 방법을 제안하려 한다. 다음의 세가지 조건을 만족시키는 한 그룹의 삼각망을 가정해 보자. 조건 1) 삼각형들이 모두 붙어 있어서 하나의 면을 이루고 있고 또한 하나의 경계곡선 C로 둘러싸여져 있다. 조건 2) 모든 삼각형들을 볼 수 있는 Vector V가 존재한다. (N을 삼각형의 법선 벡터라 할 때 모든 삼각형에 대해서 $N \cdot V>0$를 만족시키는 V가 존재해야 한다). 조건 3) Vector V를 법선 벡터로 가지는 평면에 경계곡선 C를 투영시켰을 때 투영된 곡선이 자체 꼬임이 없다. 위의 세가지 조건을 만족시키는 삼각형 그룹은 자체꼬임이 없다는 것을 직관적으로 알 수 있다[4]. 이러한 사실이 의미하는 바는 만약 우리가 주어진 삼각망에서 이러한 조건을 만족하는 삼각형들의 그룹을 추출 함으로써 TTI 횟수를 줄일 수 있다는 것이다. 왜냐하면 이러한 그룹에 속하는 삼각형들끼리는 TTI를 수행할 필요가 없기 때문이다. 본 논문에서는 이러한 세가지 조건을 모두 만족하는 삼각형 그룹을 V-group이라 부르기로 한다(Fig. 2).

본 논문에서는V-group 방법과 공간분할 방법을 함께 결합하여 사용하려 한다. 그 이유는 이 두 가지 방법이 상호 보완적인 속성을 가지므로 인해서 서로 결합하였을 때 TTI 횟수를 줄이는데 있어서 매우 큰 효과를 얻을 수 있기 때문이다. 공간분할 방식은 멀리 떨어진 삼각형들에 대해서 TTI를 수행하지 않음으로써 TTI 횟수를 줄이는데 반해서 V-group 방식은 가까이 붙어 있는 삼각형들이 꼬임이 없다는 것을 증명함으로써 TTI 횟수를 줄이는 것이다. 다시 설명하면 공간분할 방식은 기하(Geometry) 정보에 기반한 방식이고 V-group 방식은 Topology 정보에 기반한 방식이다. 이렇게 두 방식은 거의 반대의 속성을 가지고 있으면서, 서로를 잘 보완할 수 있다. 본 논문에서는 V-group들을 추출해내기 위해서 가시성 맵의 개념을[4] 사용한다. 다음 2장에서는 삼각망 자체꼬임 문제에 대한 본 논문의 접근 방식을 설명하고, 3장은 V-group들을 추출해내는 알고리즘을 설명한다. 그리고 제안된 알고리즘의 평가는 4장, 결론은 5장에서 설명된다.

2. 접근 방법

본 논문은 V-group을 추출하기 위해서 가시성 맵을 이용한다. 가시성 맵의 개념은 가우스 맵에서부터 출발한다. 하나의 곡면 S에 대한 가우스 맵은 곡면 S의 법선 벡터들과 단위 구(unit sphere)의 교점이라고 할 수 있다. Fig. 3에서 보여주듯이 두 삼각형으로 이루어진 삼각망에 대한 가우스 맵은 그 두 삼각형들의 법선 벡터들과 단위 구와의 교점이다.

가우스 맵과 유사하게 가시성 맵도 단위 구상에 존재하는 영역으로 정의 되지만, 가시성 맵에서 그 영역에 속하는 한 점 은 주어진 곡면 S를 모두 볼 수 있는 방향을 뜻한다. Fig. 4는 두 개의 삼각형으로 이루어지는 삼각망에 대한 가시성 맵을 보여주고 있다. 삼각형 T_1을 볼 수 있는 방향은 T_1의 법선 벡터 N_1으로 정의되는 반구영역이라 할 수 있다. 삼각형 T_2 역시 하나의 반구영역을 정의한다. 이때 그 두 삼각형의 가시성 맵은 이러한 두 반구 영역의 교집합이라 할 수 있으며, 그 의미는 그 영역에 속하는 어떤 점이라도 그 방향에서는 주어진 그 두 삼각형을 모두 볼 수 있다는 것이다(Fig. 4).

본 논문에서는 주어진 삼각망으로부터 V-group들을 추함으로써 TTI 횟수를 줄이려 하며, V-group의 추출

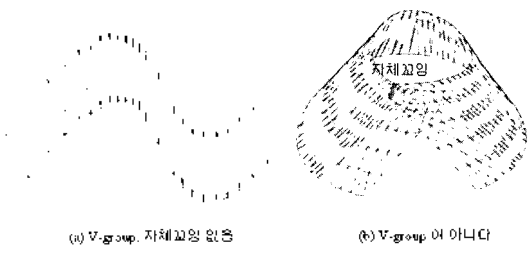


Fig. 2. The concept of a V-group.

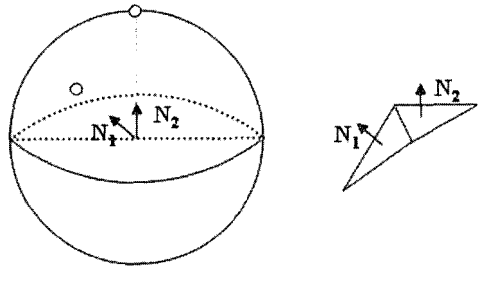


Fig. 3. Gauss Map of two triangles.

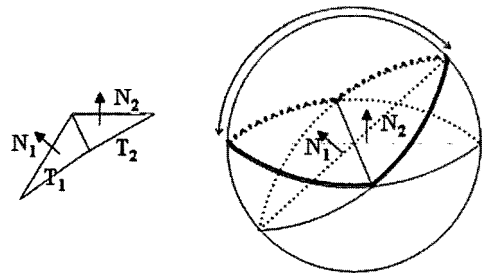


Fig. 4. Visibility Map of two triangles.

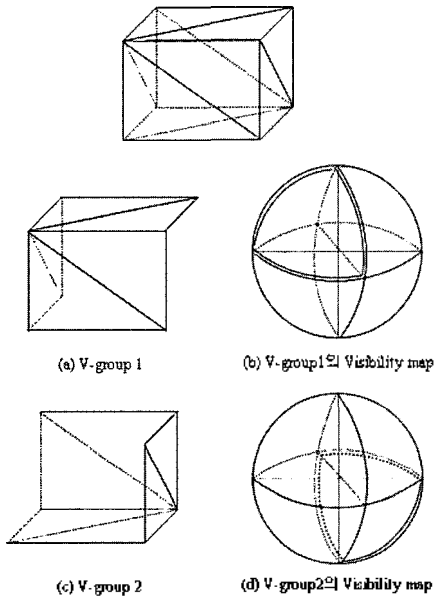


Fig. 5. V-group extraction.

을 위해서 가시성 맵을 활용한다. Fig. 5에서는 박스모양 삼각망에 대하여 V-group을 추출하는 과정을 보여주고 있다. 직관적으로 생각해보면 박스모양 삼각망의 가시성 맵은 공집합이다. 왜냐하면 어떠한 한 방향에서도 박스모양 삼각망을 구성하는 모든 삼각형들을 볼 수는 없기 때문이다. V-group을 추출하는 과정은 다음과 같이 기술 된다. 1) 삼각형들을 담을 수 있는 삼각형 집합 Tri-S를 정의한다. 2) Seed 삼각형을 하나 선택해서 삼각형 집합 Tri-S에 넣는다. 3) Tri-S에 속하는 삼각형들에 이웃한 각 삼각형에 대해서 그 삼각형이 Tri-S에 추가 되었을 때 Tri-S의 가시성 맵이 공집합이 되는지를 본다. 만약 인접 삼각형이 Tri-S에 추가되어도 Tri-S의 가시성 맵이 공집합이 아니면 추가 시킨다. 4) Tri-S를 더 이상 확장 시킬 수 없을 때 까지 단계 3을 반복한다. 5) Tri-S가 더 이상 확장 될 수 없으면 Tri-S의 가시성 맵으로부터 Tri-S에 속하는 모든 삼각형들을 볼 수 있는 Vector V를 하나 선택한다. 6) Tri-S에 속하는 삼각형들을 둘러싸는 경계곡선 C를 구한다. 7) 구해진 경계곡선 C를 V에 수직한 평면에 투영하고 투영된 이차원 곡선이 자체 꼬임이 있는지를 살핀다. 8) 만약 투영된 곡선에 자체 꼬임이 없다면 Tri-S를 V-group으로 등록한다. Fig. 5에서처럼 박스모양 삼각망은 두 개의 V-group으로 나누어 질 수 있다.

이런 방식으로 주어진 삼각망으로부터 V-group들을 추출하는 것이 가능하며, 추출된 V-group들을 이용함으로써 TTI 횟수를 줄일 수 있다. 왜냐하면 V-group에 속하는 삼각형들 사이에는 꼬임이 없다는 것이 보장되므로 만약 임의의 두 삼각형이 같은 V-group에 속하는 경우는 TTI를 수행할 필요가 없기 때문이다. 본 논문에서는 이러한 V-group 방식과 더불어서 전통적인 공간 분할 방식을 함께 사용한다. 공간 분할 방식에는 다양한 종류가 있지만, 본 논문에서는 그 중에서 가장 간단한 종류인 Octree 방식을 이용한다. Octree 방식은 삼각망이 존재하는 공간을 각 격자가 일정 개수 이하의 삼각형을 포함하도록 분할 한다. 이때 만약 어떤 두 삼각형이 서로 다른 격자에 포함된다면 그 두 삼각형은 절대 교차하지 않을 것임을 알 수 있다. 결과적으로 본 논문에서는 다음의 두 조건을 모두 만족하는 삼각형 쌍들에 대해서만 TTI를 수행한다. 1) 두 삼각형이 서로 다른 V-group에 속한다. 2) 두 삼각형이 동일한 격자에 속한다. 만약 어떤 두 삼각형이 이러한 두 조건을 모두 만족하면 비로소 TTI를 수행하는데, 본 논문에서는 TTI를 위해서 Tomas와 Prosolvia가 개발한 TTI 알고리즘을 채택하였다[9]. 본 논문에서 채택한 TTI 알고리즘은 매우 효율적이며, 소스코드가 오픈 되어 있고 인터넷 상에서 접근 가능하다(http://www.acm.org/jgt/papers/Moller97/).

3. V-group의 추출

본 장에서는 주어진 삼각망으로부터 V-group을 추출하는 알고리즘을 자세히 기술하도록 한다. 제안된 알고리즘은 다음과 같이 기술될 수 있다.

V-group 추출 알고리즘
//입력: 하나의 삼각망
//출력: 추출된 V-group들
//Tri-S: 삼각형 집합
//V-map: Tri-S의 가시성 맵
1. 초기화 단계
  1) Tri-S를 비운다:

2. V-group 추출 단계

Repeat {

1) Seed=삼각망에서 방문하지 않은 임의의 삼각형을 선택한다;

2) **Recursive-Expansion** (Tri-S, V-map, Seed) ;

3) V=V-map에 속하는 임의의 Vector를 선택한다;

4) Prj-curve=V에 수직한 평면에 Tri-S의 경계곡선을 투영한다;

5) 만약 Prj-curve가 자체꼬임을 가지지 않으면 Tri-S를 V-group으로 등록한다;

6) Tri-S를 비운다;

} **Until** (삼각망에서 방문되지 않은 삼각형이 없을 때까지);

3. 출력 단계: 등록된 V-group들을 출력한다.

제안된 V-group 추출 알고리즘을 구성하는 여러 단계들 중에서 Step 2-2와 Step 2-5가 가장 복잡한 계산을 요구하므로 전체 알고리즘의 효율성을 위해서는 최적화 시킬 필요가 있다. Step 2-5는 투영된 이차원 곡선이 자체 꼬임을 가지는지를 테스트하는 것인데, 사실 이 문제는 다각형의 단순성(Simplicity)을 테스트하는 것과 동일한 문제이다. 다각형의 단순성을 테스트 하는 문제는 이미 잘 알려져 있고 가용한 알고리즘들도[3,7] 여럿 있다. 본 논문에서는 박상철, 최병규의[3] 점열곡선의 꼬임을 효율적으로 찾는 알고리즘을 채택하였다. 그 알고리즘은 원래 임의의 점열곡선의 자체 꼬임을 찾기 위해서 개발 되었지만, 다각형의 단순성 테스트를 위해서도 적용 될 수 있다. 특히 투영된 곡선을 이루는 직선 세그먼트들의 개수가 많은 경우에 더욱 효율적으로 작동한다.

Step 2-2는 Tri-S의 가시성 맵인 V-map이 공집합이 아닌 한 계속해서 인접한 삼각형들을 Tri-S에 집어 넣음으로써 Tri-S를 확장해 나가는 과정이다. 재귀 함수인 Recursive-Expansion 함수는 다음과 같이 기술 될 수 있다.

Recursive-Expansion (Tri-S, V-map, tri)

//입력: 삼각형 집합 (Tri-S), 가시성 맵 (V-map), 새롭게 추가될 삼각형 (tri)

1. tri를 추가할지 결정.

If (tri를 추가해도 V-map이 공집합이 아니면)

{

1) Tri-S에 tri를 추가한다;

2) V-map을 업데이트한다;

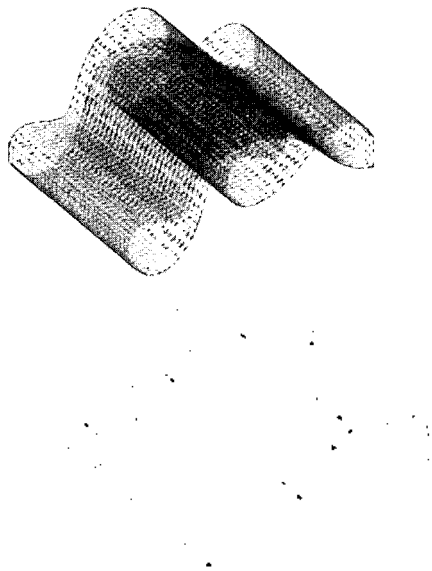


Fig. 6. V-group examples.

3) tri를 방문한 것으로 표시한다;

} **Else**

Return ;

2. 재귀적으로 확장.

For (i = 0 ; i < 3 ; i++)

{

1) adj-tri=tri의 i번째 변을 공유하는 인접 삼각형;

2) if (adj-tri가 방문되지 않았으면)

Recursive-Expansion (Tri-S, V-map, adj-tri);

}

Fig. 6는 하나의 삼각망에서부터 추출된 V-group들의 예를 보여준다.

4. 실험 및 평가

제안된 알고리즘은 구현되었으며, 다양한 예제에 대해서 테스트 되었다. Fig. 7은 세 개의 꼬인 삼각 망을 보여주며, 각각의 삼각 망들은 약 8천, 1만, 2만개의 삼각형들로 이루어진다.

위의 세 가지 삼각 망에 대해서 제안된 알고리즘을 적용하여 실험을 수행하였으며, 그 결과는 Table 1에 정리되어 있다. 예를 들어 Fig. 7(a)는 8,546개의 삼각형들로 이루어진 삼각 망이며 실제 꼬임은 180개를 가지고 있다. 만약 이 예제에 대해 brute force 방식으로 TTI를 수행한다면 28,474,831번의 TTI를 수행해야 하며, 기존 Octree 방식을 사용하면 65,306번의

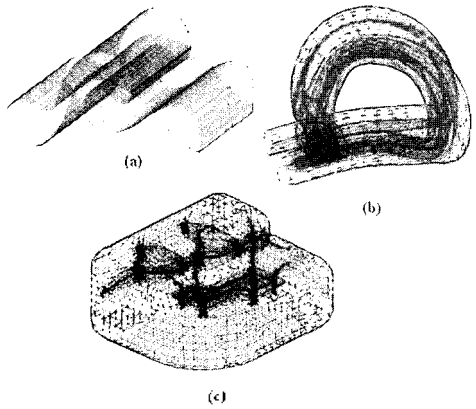


Fig. 7. Triangular meshes for the test.

Table 1. Test results

	Fig. 7(a)	Fig. 7(b)	Fig. 7(c)
삼각형 개수	8,546	10,327	19,972
꼬임 개수	180	674	2,710
Brute force TTI pairs	28,474,831	43,501,128	187,915,834
Octree TTI pairs	65,306	249,250	369,862
제안된 방식 TTI pairs	4,482	110,222	155,510

TTI가 필요하다. 하지만 본 논문에서 제안된 알고리즘을 적용한 결과 4,482번의 TTI만을 수행함으로써 자체 꼬임을 모두 찾아 내었다.

5. 결 론

본 논문에서는 삼각망의 자체꼬임을 효율적으로 찾기 위한 알고리즘을 제안하였다. 제안된 알고리즘의 특징은 TTI 횟수를 줄이기 위해서 삼각형들의 가시성 정보를 활용하는 V-group 방식에 있다. V-group 방식은 Topology에 기반하여 TTI 횟수를 줄이는 방식이라 볼 수 있으며, 이 방식의 장점은 가까이 인접한 삼각형들이라도 같은 V-group에 속하면 꼬임이 없음이 보장되므로 TTI 횟수를 줄인다는 데 있다. 이러한 특징인 전통적인 공간분할 방식의 약점을 보완해준다. 공간분할 방식은 기하정보(Geometry)에 기반하여 TTI 횟수를 줄인다고 볼 수 있으며, 주로는 멀리 떨어진 삼각형들이 서로 다른 격자에 속하게 되면 TTI를 수행할 필요가 없다는 것을 알게 됨으로써 TTI 횟수를 줄이는 방식이다. 이렇게 V-group 방식과 전통적인 공간분할 방식이 상호 보완적인 특성을 가지고 있음으로써, 본 논문에서는 이 두 방식을 결합하여 TTI 횟수를 줄였다.

현재 개발된 알고리즘에서는 V-group을 추출하기 위해서 최초 삼각형을 임의로 선택하고 있다. 하지만 만약 최초 Seed 삼각형을 임의로 선택하기 보다는 어떤 선택법을 개발해서 Seed 삼각형을 선택함으로써 더 많은 삼각형을 포함하는V-group을 추출할 수 있다면 제안된 알고리즘을 더욱 효율적으로 할 수 있을 것이다. 이 부분은 추후 연구과제로 남기고자 한다.

감사의 글

본 연구는 아주대학교 장학연구비로 수행되었습니다.

참고문헌

1. Choi, B. K. and Jerard, R. B., Sculptured Surface Machining, Kluwer, 1998.
2. Woo, T. C., "Visibility maps and spherical algorithms," *Computer-Aided Design*, Vol. 26, No. 1, pp. 6-16, 1994.
3. 박상철, 최병규, "점열곡선의 꼬임을 효율적으로 찾는 알고리즘," 한국CAD/CAM학회 논문집, 제4권, 제3호, pp. 190-199, 1999.
4. Volino, P. and Thalmann, N. M., "Collision and self-intersection detection: efficient and robust solutions for highly deformable surfaces," Proceedings of Eurographics Workshop on Animation and Simulation, pp. 55-65, 1995.
5. Tomas, M. and Prosolvia, C. A. B., "A fast triangle-triangle intersection test," *Journal of Graphics Tools*, Vol. 2, No. 2, pp. 25-30, 1997.
6. Jimenez, P., Thomas, F. and Torras, C., "3D collision detection: a survey," *Computers and Graphics*, Vol. 25, No. 2, pp. 269-285, 2001.
7. Shamos, M. I. and Hoey, D. J., "Geometric intersection problems," *in Proc. 17th Annu. Conf. Foundation of Computer Science*, pp. 208-215, Oct 1976.

박 상 철

1994년 KAIST 산업공학과 학사
1996년 KAIST 산업공학과 석사
2000년 KAIST 산업공학과 박사
2001년~2004년 미국 DaimlerChrysler ITM research engineer
2004년~현재 아주대학교 산업정보시스템공학부 교수
관심분야: CAD/CAM, Virtual Manufacturing System, Discrete Event System Modeling & Simulation