다면체간의 강건한 민코스키합 경계면 계산

경민호° Elisha Sacks†

아주대학교°, Purdue University†

kyung@ajou.ac.kr, eps@cs.purdue.edu

Robust Computation of Polyhedral Minkowski Sum Boundary

Min-Ho Kyung° Elisha Sacks†

Ajou University°, Purdue University†

요 약

기하학에서 민코스키합은 두 집합에 들어 있는 모든 점들간의 합으로 이루어지는 집합을 구하는 연산으로 정의되는데, 로보틱스, NC 가공, 솔리드 모델링 등의 다양한 분야의 기하학적 문제를 다루는 매우 유용한 이론적 도구로 사용되고 있다. 하지만, 단순한 정의에도 불구하고 수치 연산의 반올림 오차로 인하여 다면체간의 민코스키합을 정밀하고 강건하게 계산하는 것은 매우 어렵다. 본 논문에서는 컨볼루션 계산 방법을 이용하여 다면체 간의 민코스키합 경계를 계산하는 알고리즘을 제안한다. 알고리즘의 강건성을 보장하기 위한 방법으로 CLP(controlled linear perturbation) 기법을 처음으로 적용하였다. CLP는 인위적 교란 방법의 하나로 알고리즘의 강건성을 해치는 반올림 오차에 의한 논리적 오류 발생을 막는다. 본 논문의 알고리즘은 실험 예제들에서 민코스키합의 경계면을 구성하는 완전한 2차원 다양체 구조 메시를 10^{-14}의 정밀도로 출력하고, 이 때 입력 다면체의 꼭지점 좌표는 10^{-10}까지 교란되는 결과를 얻었다.

Abstract

Minkowski sum of two polyedra is an operation to compute the sum of all pairs of points contained in the polyhedra. It has been a very useful tool to solve many geometric problems arising in the areas of robotics, NC machining, solid modeling, and so on. However, very few algorithms have been proposed to compute Minkowski sum of polyhedra, because computing Minkowski sum boundaries is susceptible to roundoff errors. We propose an algorithm to robustly compute the Minkowski sum boundaries by employing the *controlled linear perturbation* scheme to prevent numerically ambiguous and degenerate cases from occurring. According to our experiments, our algorithm computes the Minkowski sum boundaries with the precision of 10^{-14} by perturbing the vertices of the input polyhedra up to 10^{-10}.

키워드: 민코스키합, 삼각형교차, 컨볼루션

Keywords: Minkowski sum, triangle intersection, convolution

1. 서론

두 다면체 A와 B가 주어져 있을 때, 이들 간의 민코스키합은 다음과 같이 정의된다:

$$M = A \oplus B = \{a + b | a \in A \text{ and } b \in B\}. \quad (1)$$

이렇게 정의된 민코스키합은 기하 계산을 포함하는 많은 분야에서 매우 유용하게 사용되고 있다. 예를 들어 로보틱스 분야의 전형적인 문제 중 하나인 충돌 회피 경로 계산 문제는 민코스키합을 이용하여 쉽게 해결할 수가 있다([1] 참조). 로봇 A와 장애물 B가 주어졌을 때, $M = (-A) \oplus B$은 A와 B가 충돌하게 되는 A의 이동 위치를 정의하게 된다. 따라서 $\bar{M}$에 포함되는 위치만을 따라서 이동하게 되면 A와 B는 절대로 충돌하지 않게 된다. 이 외에도 형상의 오프셋 모델링, 두 물체 간의 투과 깊

*투고일: 2010.04.13 / 심사일: 2010.04.18 / 게재확정일: 2010.05.31

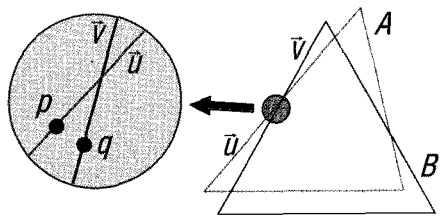


그림 1: 수치 오차로 인한 삼각형 A와 B간의 교차 계산 오류. p와 q는 동시에 교차점이 될 수 없지만 수치 오차로 인하여 모두 교차점으로 계산됨.

이(penetration depth) 계산, 3차원 몰핑 연산, 컴퓨터 애니메이션, 부품 레이아웃, 제품 조립 등의 다양한 문제들을 해결하는데 유용하게 사용될 수 있다.

이러한 유용성에도 불구하고 민코스키합은 실제 문제 해결에 활발히 사용되지 못하고 있다. 그 이유는 민코스키합의 계산에는 여러 가지 예외 경우들(degenerate cases)들이 발생하는데, 이러한 경우들을 모두 분류하여 논리적으로 모순되지 않게 처리하는 것이 매우 어렵기 때문이다. 더군다다 이러한 경우들은 수치적으로 매우 예민하기 때문에, 약간의 수치 오차에 의해서도 논리적으로 모순되는 잘못된 결과를 낳기도 한다. 계산 과정에 발생하는 논리적인 모순은 잘못된 경계면을 찾거나 또는 프로그램 수행 자체를 중단시키는 심각한 상황을 초래하기도 한다. 그림 1은 두 삼각형 A와 B간의 잘못된 교차 계산 예를 보여주고 있다. 두 삼각형의 모서리 $\vec{u}$와 $\vec{v}$가 3차원 공간에서 반올림 오차 이내로 매우 근접하게 지나갈 경우 두 모서리 상의 점 p와 q는 각각 다른 삼각형과의 교차점으로 판단될 수 있다. 하지만 유클리드 기하에서 p와 q가 동시에 교차점이 되는 것은 논리적으로 맞지 않기 때문에 둘 중에 한 교차점만을 선택해야 한다. 여기서 어려운 점은 수치적 계산만으로는 올바른 판단을 할 수 없기 때문에 인접 삼각형들과의 교차 상황과 모순되지 않게 판단해야 하는 것으로 다양한 경우들을 고려해야 한다.

본 논문에서는 수치 오차로 인해 발생하는 논리적 모순과 동시에 예외적인 경우들의 발생을 피할 수 있는 *제어된 선형 교란(controlled linear perturbation, 또는 CLP)* [1] 방법을 도입하였다. CLP는 예외적인 경우들을 피하기 위해 계산기하학에서 일반적으로 사용하는 임의 교란(random perturbation)의 일종으로 선형 근사를 이용하여 최적의 교란 파라메터를 찾아 신뢰할 수 있는 결과를 보장하는 방법이다. 민코스키합 계산에 CLP를 적용함으로써 알고리즘의 강건성을 확보하였고, 예외 경우들이 발생할 수 있는 가능성을 제거하여 알고리즘의 구현을 단순화시켰다.

본 논문에서 제안하는 민코스키합 알고리즘은 기존의 컨볼루션(convolution) 기반 알고리즘과 유사하다. 먼저, 두 다면체에서 법선 방향이 일치하는 기하 요소들(정점, 면, 모서리) 간의 합을 구한다. 여기서 얻어진 결과는 다각형들의 집합이 되고, 이 집합은 민코스키합의 경계를 포함하게 된다. 다음은 이 다각형들 간의 모든 교차선을 계산하고 이 교차선들로 다각형들을 여러 개의 영역으로 분할한다. 마지막으로 다각형 영역들을 탐색하여 민코스키합 경계에 들어가는 영역들을 찾고, 이 영역들을 연결하여 민코스키합 경계를 나타내는 다각형 메시 구조를 구성한다.

본 논문은 다음과 같이 구성된다. 2절에서는 민코스키합 계산과 관련된 기존 연구들을 알아 보고, 3절에서는 CLP에 관한 기본 이론을 설명한다. 다음 4절에서는 알고리즘의 세부적인 내용을 설명하고, 5절에서 이를 이용한 실험 결과를 보여주고, 마지막으로 6절에서 결론을 맺는다.

2. 기존 연구

민코스키합 계산에 관한 연구는 그동안 기하 모델링, 로보틱스, 계산기하학 등의 여러 분야에서 활발히 되어 왔고, 여러 가지 방법들이 제안되었다. 평면 다각형의 민코스키합의 계산 방법은 비교적 많이 연구되었고 효율적이고 강건한 알고리즘들이 많이 알려져 있다([3, 4] 참조). 최근에 Wein [5]은 컨볼루션을 기반으로 비볼록 다각형(nonconvex polygon)을 정확하고 강건하게 계산하는 방법을 제안하였다.

3차원 다면체의 민코스키합 계산은 평면 다각형에 비하여 복잡도가 훨씬 높고 수치 오차에 영향을 많이 받기 때문에 알고리즘의 개발이 매우 어려운 편이다. 볼록 다면체의 경우는 민코스키합이 항상 볼록 다면체로 만들어지고 교차가 발생하지 않기 때문에 정확하고 효율적인 알고리즘들이 여러 가지 개발되어 있다([6, 7, 8, 9, 10] 참조). 반면에 비볼록 다면체의 경우는 복잡도가 최대 $O(n^6)$까지 높아질 수 있기 때문에 효율적인 알고리즘 개발이 어려운데, Kaul [11]은 3차원 다면체 경계의 기하 요소들 간의 합에서 경계면을 추출하는 방법을 제안하였다. Basch등[12]은 다면체의 컨볼루션을 처음 정의하고, 이 콘볼루션을 이용하여 민코스키합을 표현할 수 있음을 보였다. Peternell 등[13]은 연속적인 곡면으로 이루어진 3차원 모델의 민코스키합을 엔벨롭(envelope)에서 샘플링한 점집합을 이용하여 근사적으로 구하는 방법을 제안하였다. Varadhan[14]은 주어진 다면체를 볼록 다면체들로 분할하고 볼록 다면체들 간의 민코스키합을 구한 후에 그 결과를 근사적으로 합하는 방법을 제안하였다. Hachenberg[15]는 3D Nef 다면체를 이용하여 정확하고 강건하게 볼록 다면체 분할하고 민코스키합을 구하였다. 하지만 복잡한 다면체에는 실제로 적용하기 어려운 한계가 있다. Lien은 점집합 근사를 이용하여 다면체의 민코스키합을 근사적으로 구하는 방법([16])과 컨볼루션 및 충돌 테스트를 이용하여 민코스키합 경계를 구하는 방법([17])을 제안하였다.

본 논문에서 제안하는 알고리즘은 Basch등이 제안한 알고리즘에 기반을 두고 있다. 원래 알고리즘은 민코스키합의 특성으

[1] CLP는 Victor Milenkovic와 Elisha Sacks가 제안한 방법으로 논문 발표를 준비 중임. 자세한 내용은 Trac의 박사학위 논문 [2]에 소개되어 있음.

로 인하여 수치 오류에 매우 취약한 문제가 있다. 본 논문에서 기여한 점은 이 취약점을 수치적인 교란 방법을 통해 해결함으로서 실제 구동되는 알고리즘으로 완성한 것이다.

3. CLP(Controlled Linear Perturbation)

민코스키합을 포함한 대부분의 기하 알고리즘들은 문제의 조합 성질을 결정하는 유한 개의 판단 함수(f_i)들로 구성되고 그 함수들의 부호에 따라 결과를 구하는 조합 문제들이다. 알고리즘의 입력을 벡터 $\mathbf{x}$로 나타낼 때, 조합 성질은 $f_i(\mathbf{x})$의 값이 음수인 경우와 양수인 경우로 나누어 결정되고, $f_i(\mathbf{x}) = 0$인 경우는 예외적인 경우(degenerate cases)로 처리한다.

판단 함수 f_i의 계산은 유한 정밀도를 가지는 실수 연산으로 이루어지기 때문에 필연적으로 반올림 오차가 발생한다. 만일 함수 값이 반올림 오차에 비해 충분히 크다면 반올림 오차는 함수값의 부호에 영향을 주지 않는다. 하지만 만일 함수 값이 반올림 오차에 비해 작다면 반올림 오차에 의해 부호가 바뀔 수 있기 때문에 그 결과는 신뢰할 수 없는 것이 된다. CLP는 이러한 불안정한 함수값이 발생하는 경우 입력 벡터 $\mathbf{x}$를 인위적으로 교란하여 함수값을 반올림 오차 범위 밖으로 이동시키는 역할을 한다. 이때 교란값은 다음과 같이 결정한다.

먼저 입력 벡터의 원래 값을 $\mathbf{x}_0$라고 하고, 교란시킬 방향을 $\mathbf{v}$라고 하자. $\mathbf{v}$는 알고리즘 시작 전에 랜덤하게 선택된 임의의 단위 벡터이다. 벡터 $\mathbf{x}$가 $\mathbf{v}$방향으로 길이 δ만큼 교란된다면,

$$\mathbf{x} = \mathbf{x}_0 + \delta \cdot \mathbf{v} \tag{2}$$

와 같이 정의된다. 여기서 δ는 초기에 0으로 설정된다. ϵ을 반올림 오차의 예상 최대치라고 한다면, $|f_i(\mathbf{x})| > \epsilon$인 경우 $f_i(\mathbf{x})$의 부호는 반올림 오차에 의해 영향을 받지 않기 때문에 신뢰할 수 있다. $|f_i(\mathbf{x})| \le \epsilon$ 경우는 다음 조건을 만족하도록 δ를 증가시켜 $|f_i(\mathbf{x})|$가 ϵ보다 커지도록 한다:

$$|f_i(\mathbf{x})| \simeq |f_i(\mathbf{x}_0) + \nabla f_i(\mathbf{x}_0) \cdot (\delta \cdot \mathbf{v})| \ge \epsilon \tag{3}$$

$$\delta \ge (\epsilon - s \cdot f_i(\mathbf{x}_0))/(s \cdot \nabla f_i(\mathbf{x}_0) \cdot \mathbf{v}). \tag{4}$$

여기서 s는 $f_i(\mathbf{x}) < 0$이면 -1, 아니면 1이다. 교란정도는 작을수록 좋기 때문에 δ는 $(\epsilon - s \cdot f_i(\mathbf{x}_0))/(s \cdot \nabla f_i(\mathbf{x}_0) \cdot \mathbf{v})$ 로 설정하는 것이 좋지만 선형 근사에 의한 오차를 고려하여 이 값에 2를 곱하여 δ값으로 설정한다.

민코스키합에서는 여러 가지 예외 경우들이 많이 발생하는데, 이러한 예외 경우들에서는 0 또는 반올림 오차 범위의 작은 값을 가지는 판단함수 $f_i(\mathbf{x})$가 발생하게 된다. 예를 들어 한 삼각형이 다른 다면체의 두 꼭지점과 거의 동시에 접하는 경우 이 들간에 만들어지는 컨볼루션 삼각형 두개는 한 평면 위에서 가깝게 겹치게 된다. 이때 두 콘볼루션 삼각형들의 교차를 구하기 위해 한 삼각형의 꼭지점들을 다른 삼각형의 평면식을 나타내는 함수 f_i에 대입하면 반올림 오차 범위 안에서 매우 작은 값이 구해진다. 이 값은 신뢰할 수 없는 값으므로, 기존 알고리즘들에서는 두 삼각형이 한 평면에 있는 공면(coplanar) 경우로 간주하여 처리하였다. CLP는 f_i를 안전한 범위의 값으로 보내기 위해 입력 삼각형들의 꼭지점 위치를 교란하므로, 콘볼루션 삼각형의 공면성이 자동적으로 깨지게 된다. 이와 같이 CLP를 이용하면 기하 알고리즘의 구현을 어렵게 만드는 예외 경우들을 고려할 필요가 없어지기 때문에 강건한 알고리즘 구현이 비교적 쉬워진다.

4. 민코스키합 알고리즘

민코스키합 알고리즘은 세 단계로 구성되어 있다:

1. 두 다면체의 컨볼루션 계산
2. 합 삼각형의 교차 계산
3. 민코스키합 경계 추출

앞 절에서 설명한 CLP는 알고리즘의 모든 단계에서 발생하는 수치 계산 함수에 관여하여 불안정한 함수값이 발생할 경우 실시간에 입력 벡터를 교란시켜 준다. CLP의 입력 벡터는 민코스키합이 계산되는 다면체 A와 B의 정점 좌표들을 모두 나열한 벡터가 된다. 입력되는 다면체 A와 B는 편의상 모두 삼각형 메시로 되어 있다고 가정한다.

4.1 컨볼루션 계산

두 다면체 A와 B의 컨볼루션은 다음과 같이 정의된다:

$$C = A * B = \{(p+q) | p \in A, q \in B, N(p) \cap N(q) \neq \emptyset\}. \tag{5}$$

$N(p)$는 점 p에서의 접면(tangent plane)들의 법선 방향을 포함하는 집합으로 가우스 맵 S^2의 부분 집합이 된다. 만일 p가 면 f에 속하는 점이면 $N(p)$는 f의 법선 벡터가 된다. p가 모서리 e 위의 한 점이면 $N(p)$는 e를 공유하는 두 면의 법선 벡터을 연결하는 S^2 위의 호(arc)가 된다. p가 다면체의 정점 v이면, $N(p)$는 v를 공유하는 면들의 법선 벡터들을 호로 연결하여 정의되는 S^2 위의 영역이 된다.

컨볼루션 $A * B$는 민코스키합의 경계 $\partial(A \oplus B)$를 포함하는 집합이라고 증명되어 있다. 따라서 $A * B$에서 경계에 포함되는 다각형들만을 결정하면 효율적으로 민코스키합의 경계를 찾을 수 있다. 이 방법은 A와 B의 모든 기하 요소 쌍들을 더하여 여기서 경계면들을 찾는 방법과 비교할 때 시간 복잡도는 같지만 실험적으로는 훨씬 빠른 결과를 보여준다. 그 이유는 $|A| = m$과 $|B| = n$인 경우에 두 방법 모두 $O(mn)$개의 다각형들이 생성되지만, 실제 $A * B$의 복잡도는 대부분의 경우에 $m + n$에 비례하여 증가하므로 후자에 비해 처리할 다각형의 수가 적기 때문이다.

$C = A * B$를 이루는 다각형들은 공통된 법선 벡터를 가지는 정점-면 쌍 또는 모서리-모서리 쌍에 대하여 두 기하 요소의 합으로 구해진다. 정점 v와 면 $f = \triangle(p_0, p_1, p_2)$에 대하여 $N(v) \cap N(f) \neq \emptyset$의 조건은 f의 법선 N_f와 v에 연결된 모든 모서리 $e_i = (v, u_i)$에 대하여 $N_f \cdot (u_i - v) < 0$ (또는 > 0)이 된다. 이 조건을 만족하는 v와 f에 대한 합은

$$C_{vf} = \triangle(v + p_0, v + p_1, v + p_2) \tag{6}$$

이 된다. 모서리 $e_0 = (a, b)$와 $e_1 = (c, d)$가 $N(e_0) \cap N(e_1) \neq \emptyset$이기 위한 조건은 e_0와 e_1에 수직인 $(b - a) \times (d - c)$와 e_0와 e_1에 연결된 모든 모서리 방향과의 내적이 0보다 크면 된다. 이 조건을 만족하는 e_0와 e_1의 합은

$$C_{e_0e_1} = \square(a + c, a + d, b + c, b + d) \tag{7}$$

가 된다. $C_{e_0e_1}$은 교차 계산의 편의를 위하여 두 개의 삼각형 $\triangle(a + c, a + d, b + c)$와 $\triangle(a + c, b + c, b + d)$로 나누어져 저장된다.

4.2 삼각형 교차 계산

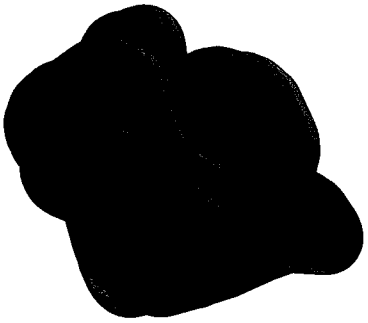
콘볼루션에서 얻어진 삼각형들간에는 많은 교차가 발생한다. 교차되어 나누어진 삼각형의 일부는 $A \oplus B$의 경계에 포함되고 일부는 내부에 들어가게 된다(그림2a 참조). 따라서 경계 추출 전에 모든 삼각형의 교차를 계산하고, 교차에 의해 분할되는 삼각형 영역을 개별 다각형으로 나누어 줄 필요가 있다.

n개의 삼각형 교차 계산은 기하 데이터를 다루는 분야에서 자주 발생하는 문제로서 효율적으로 교차 계산 시간을 하기 위한 여러가지 방법들이 연구되어 왔다. 그 중에서 일반적으로 많이 사용하는 방식은 공간을 분할하여 교차 테스트할 대상의 범위를 한정하는 것으로 k-d tree, octree, OBB 등의 분할 방법들이 있다. 본 논문에서는 k-d tree를 사용하여 공간을 분할하였다.

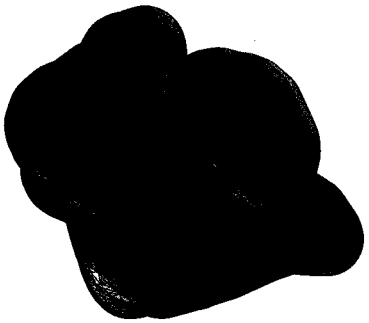
교차 테스트는 k-d tree에서 같은 리프(leaf) 노드에 저장된 삼각형들 간에 수행된다. 삼각형 f_1과 f_2간의 교차 테스트는 먼저 f_1와 f_2의 각 모서리 e_{1i}와의 교차점을 계산하고, 다음 f_2와 f_1의 각 모서리 e_{2j}와의 교차점을 계산한다. 두 삼각형이 교차하면 정확히 두 개의 교차점이 구해지고 이 두 점을 연결하면 교차선이 나오게 된다.

교차선이 구해지면 두 삼각형의 내부 영역 분할을 나타내는 배열(arrangement)를 갱신한다. 삼각형 내부의 배열(arrangement)를 나타내는 구조로는 trapezoidal map을 사용하였다. Trapezoidal map을 갱신하는데 걸리는 시간은 맵 크기를 n이라고 할 때 $O(\log n)$이 된다.

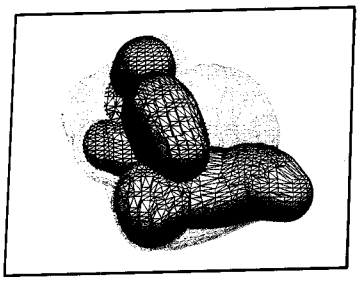
교차 계산이 모두 끝나면 삼각형의 분할된 영역들을 모두 다각형으로 변환하고 인접 관계에 따라 서로 연결시켜 준다.



(a) Sphere * Dragon



(b) 경계면 추출



(c) 절단면

그림 2: Sphere $\oplus$ Dragon의 컨볼루션과 경계면 추출 결과 비교.

그림 3: 경계면에 속하는 인접 다각형 탐색.

4.3 민코스키합 경계 추출

두 다면체의 민코스키합 경계면은 2차원 다양체이기 때문에 한 경계점에서 출발하여 인접한 다각형을 탐색해 가면서 경계면를 결정해 나갈 수 있다. 만일 경계가 여러 개의 연결 요소(connected component)로 구성되어 있을 경우에는 각각의 연결 요소마다 출발점을 찾아주어야 한다. 가장 외곽 경계면의 출발점은 비교적 쉽게 선택할 수 있다. 콘볼루션 C에서 가장 작은(또는 큰) x 좌표를 가지는 정점 v_{xmin}을 살펴보면, 주변 공간이 이 점을 중심으로 민코스키합의 내부와 외부로 나누어지는 것을 알 수 있다. 따라서 v_{xmin}은 항상 경계면에 포함되는 점이 된다. 반면에 그 외의 경계면에서는 이러한 정점을 찾기가 쉽지 않다. 한 가지 방법은 모든 컨볼루션 정점에 대하여 입력 다면체 A와 B 간에 충돌 여부를 테스트하는 것이다. 하지만 이 방법은 시간이 매우 많이 걸리는 단점이 있다. 다른 방법으로는 컨볼루션에 의하여 구분되는 공간 영역을 모두 찾고, 이 영역들 내부의 한 점을 선택하여 민코스키합의 외부인지를 테스트하는 것이다. 이 점에서 두 다면체가 충돌하지 않는다면 해당 영역은 외부 영역이 되고 이 영역에 접하는 다각형을 모두 찾아 경계면을 완성할 수 있다. 본 논문에서는 현재 가장 외곽 경계면만을 출력하도록 구현하였지만, 후자의 방법을 이용하여 쉽게 모든 경계면을 찾도록 확장할 수 있다.

출발점 v_s가 주어지면 너비우선탐색(breadth first search) 방식으로 인접 다각형을 탐색하며 경계면을 찾아간다. 탐색 과정은 다음과 같다:

1. v_s를 포함하고 있는 다각형 f_0을 큐(queue) Q에 추가한다.
2. Q에서 다각형 f를 꺼내고, f를 집합 M_B에 추가한다. M_B는 경계면으로 이미 결정된 다각형들을 포함하는 집합이다. 만일 Q가 비어 있다면 탐색을 끝낸다.
3. f에 인접한 다각형들 중에 경계면으로 판단되는 다각형들을 모두 Q에 추가한다.
4. 단계 2로 간다.

깊이우선탐색을 하지 않은 이유는 스택의 길이가 그래프의 크기에 따라 비례해서 길어지기 때문이다. 민코스키합 경계면의 갯수는 이론적으로 최악의 경우 $O(m^3n^3)$까지 될 수 있는데, 이만한 크기의 스택을 유지하는 것은 효율적이지 못하다.

경계 다각형 f의 모서리 e에 인접한 다각형들 중에 경계면은 e를 중심으로 반시계 방향으로 f에서 가장 가까운 다각형 f'이 된다(그림 3). 민코스키합은 2차원 다양체이기 때문에 경계 다각형에 인접한 다각형 f'은 항상 찾을 수 있다. 모서리 e를 공유하는 다각형은 보통 두 개이지만 e가 교차로 인해 만들어진 경우에는 네 개가 된다.

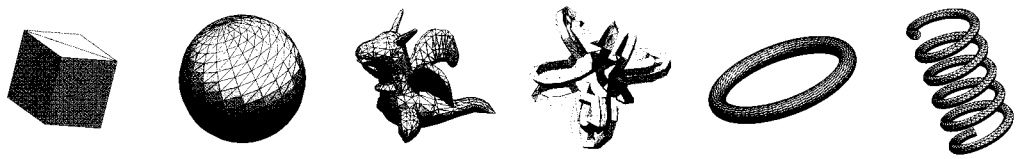
5. 실험 결과 및 분석

본 논문에서 제안한 알고리즘을 다양한 다면체에 적용하여 민코스키합을 구해 보았다. 그림 5은 민코스키합을 구한 예제의 결과를 보여주고 있다. 콘볼루션 결과는 이론적으로 $O(mn)$개의 면이 나올 수 있지만, 본 논문에서 실험한 결과에 의하면 생성되는 삼각형의 갯수는 $O(m+n)$에 비례함을 알 수 있다(표1 참조). 실험 결과를 보면 민코스키합 경계면의 갯수가 컨볼루션 삼각형의 갯수보다 적음을 알 수 있는데, 그 이유는 교차로 인해 나누어지는 늘어나는 다각형의 수보다 민코스키합의 내부에 포함되는 수가 많기 때문이다. 이 예제들은 반올림 오차 한계를 $\epsilon = 10^{-14}$로 설정하고 계산한 것으로 이 때 입력다면체의 교란값 δ는 10^{-12}에서 10^{-8} 사이에 오는 것을 알 수 있다. 이 정도의 교란은 입력 다면체의 크기를 고려했을 때 매우 작은 오차로 기존의 공간 분할, 또는 점 기반 접근 방법들에서는 얻기 힘든 정밀도다.

표2는 실험 예제들의 실제 계산시간을 보여주고 있다. 사용한 하드웨어는 4GB 메모리를 장착한 인텔 듀얼코어 2.4Ghz이고, 소스 코드 컴파일에는 g++ 컴파일러를 사용하였다. 실행 시간은 전반적으로 컨볼루션 삼각형의 갯수에 비례해서 증가하지만 입력 다면체 형상의 복잡도에도 많은 영향을 받는 걸 알 수 있다. Torus $\oplus$ Knot와 Torus $\oplus$ Dragon을 비교하면 컨볼루션 삼각형의 갯수는 같지만 전자가 후자보다 두 배정도 시간이 걸림을 알 수 있다. 이 것은 Knot의 모양이 Dragon의 모양보다 복잡하여 더욱 많은 삼각형 교차가 발생하기 때문이다.

6. 결론

본 논문에서는 3차원 다면체의 민코스키합 경계를 계산하는 강건한 알고리즘을 제안하였다. 알고리즘의 강건성을 보장하기 위하여 입력 다면체를 교란시키는 CLP 방법을 사용하였다. 이 방법은 수치계산에 의한 반올림 오차를 극복하여 알고리즘에서 발생하는 기하학적 판단을 일관되고 안정되게 만들어 준다. 알고리즘은 컨볼루션 계산, 삼각형의 교차 계산, 경계면 추출의 세 단계로 구성되어 있고, 알고리즘 전 단계에 걸쳐 모든 수치 계산에 CLP가 적용되어 실시간으로 적절한 교란값을 찾아간다. 알고리즘은 다섯 가지 다면체를 다양하게 조합하여 테스트하였다. 테스트 결과 기존 민코스키합 연구에 비하여 더 빠른 시간 안에 높은 정밀도의 결과를 얻을 수 있었다.



(a) Cube (b) Sphere (c) Dragon (d) Knot (e) Torus (f) Helix

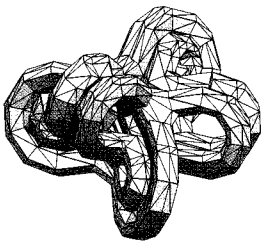
그림 4: 입력 다면체들.

	Object 1	Object 2	Convolution	Boundary	δ
Cube $\oplus$ Knot	12	992	2,564	2,492	2.84366×10^{-12}
Cube $\oplus$ Dragon	12	2,328	3,810	2,726	6.8462×10^{-12}
Cube $\oplus$ Helix	12	4,012	7,335	3,232	7.76923×10^{-12}
Sphere $\oplus$ Knot	760	992	18,589	18,420	3.68975×10^{-08}
Sphere $\oplus$ Dragon	760	2,328	19,074	11,247	1.54632×10^{-9}
Sphere $\oplus$ Helix	760	4,012	40,212	21,270	5.66214×10^{-11}
Torus $\oplus$ Knot	2,068	992	45,922	18,829	8.62922×10^{-10}
Torus $\oplus$ Dragon	2,068	2,328	44,126	18,061	8.4065×10^{-10}
Torus $\oplus$ Helix	2,068	4,012	78,582	10,522	4.8703×10^{-10}

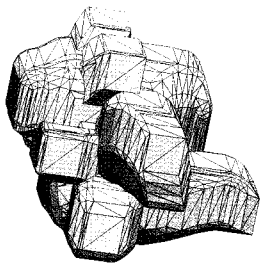
표 1: 민코스키합 계산 결과. (왼쪽부터 입력 다면체와 컨볼루션결과의 삼각형 갯수, 경계면의 갯수, 교란값 δ).

	Convolution	Intersection	Boundary Extraction	Total
Cube $\oplus$ Knot	0.042688	0.526307	0.066381	0.635376
Cube $\oplus$ Dragon	0.095957	0.689891	0.07743	0.863278
Cube $\oplus$ Helix	0.144755	2.767734	0.235784	3.147683
Sphere $\oplus$ Knot	1.40753	8.411562	0.652708	10.4718
Sphere $\oplus$ Dragon	3.49336	3.276912	0.387802	7.158074
Sphere $\oplus$ Helix	4.83775	5.687893	1.0867	11.612343
Torus $\oplus$ Knot	3.2438	32.14462	1.45804	36.84646
Torus $\oplus$ Dragon	7.94231	9.48127	1.05104	18.47462
Torus $\oplus$ Helix	10.921	16.03317	0.992894	27.947064

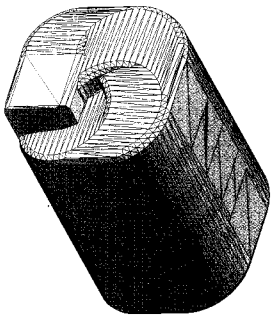
표 2: 예제별 수행 시간 비교(단위 초).



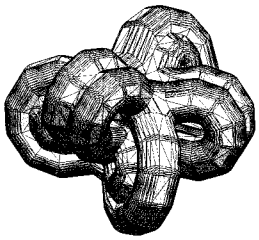
(a) Cube ⊕ Knot



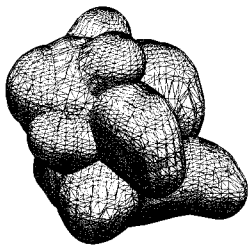
(b) Cube ⊕ Dragon



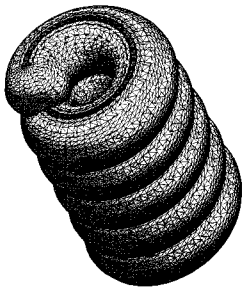
(c) Cube ⊕ Helix



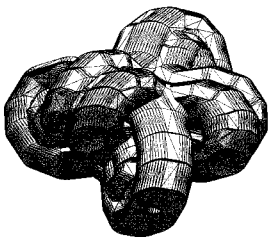
(d) Sphere ⊕ Knot



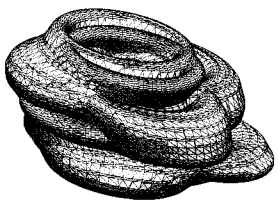
(e) Sphere ⊕ Dragon



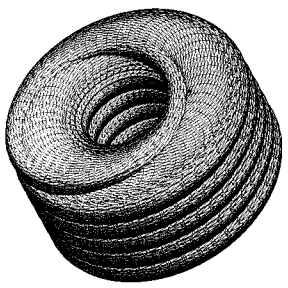
(f) Sphere ⊕ Helix



(g) Torus ⊕ Knot



(h) Torus ⊕ Dragon



(i) Torus ⊕ Helix

그림 5: 민코스키합 계산 결과.

앞으로의 연구 계획은 컨볼루션 계산과 교차 계산을 GPU를 이용하여 가속하는 방법을 고려하고 있다. 이 두 계산은 병렬성이 높기 때문에 GPU 아키텍쳐에 적합한 장점을 가지고 있다. 하지만 아직 대부분의 GPU가 부동소숫점 연산을 단일 정밀도(single precision)만 지원하거나 하기 때문에 반올림 오차 범위가 크고, 따라서 결과의 정확도가 떨어지는 문제가 생길 수 있다.

감사의 글

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2010-0012021).

참고 문헌

[1] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.

[2] S. Trac, "Robust explicit construction of configuration spaces using automated reasoning," Ph.D. dissertation, University of Miami, Dec. 2008.

[3] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," in *Europoean Symposium on Algorithms*, 2000, pp. 20–31.

[4] D. Halperin, "Robust geometric computing in motion," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 219–232, 2002.

[5] R. Wein, "Exact and efficient construction of planar minkowski sums using the convolution method," in *Proc. 14th Annual European Symposium on Algorithms (ESA)*, 2006, pp. 829–840.

[6] L. J. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry," in *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, 1983, pp. 100–111.

[7] L. J. Guibas and R. Seidel, "Computing convolutions by reciprocal search," *Disc. Comp. Geom.*, vol. 2, pp. 175–193, 1987.

[8] P. K. Ghosh, "A unified computational framework for minkowski operations," *Computers and Graphics*, vol. 17, no. 4, pp. 357–378, 1993.

[9] K. Fukuda, "From the zonotope construction to the minkowski addition of convex polytopes," *Journal of Symbolic Computation*, vol. 38, no. 4, pp. 1261–1272, 2004.

[10] D. Halperin and E. Fogel, "Exact and efficient construction of minkowski sums of convex polyhedra with applications," *Cmputer Aided Design*, vol. 39, no. 11, pp. 929–940, 2007.

[11] A. Kaul and M. A. O'Connor, "Computing minkowski sums of regular polyhedra," IBM T. J. Watson Research Center, Yorktown Heights, N. Y., Tech. Rep. RC 18891, 1993.

[12] J. Basch, L. Guibas, G. Ramkumar, and L. Ramshaw, "Polyhedral tracings and their convolutions," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 1996.

[13] M. Peternell and T. Steiner, "Minkowski sum boundary surfaces of 3d-objects," *Graphical Models*, vol. 69, no. 3-4, pp. 180–190, 2007.

[14] G. Varadhan and D. Manocha, "Accurate minkowski sum approximation of polyhedral models," *Graphical Models*, vol. 68, no. 4, pp. 343–355, 2006.

[15] P. Hachenberger, "Exact minkowski sums of polyhedra and exact and efficient decomposition of polyhedra in convex pieces," in *Proc. 15th Annual European Symposium on Algorithms(ESA)*, 2007, pp. 669–680.

[16] J. M. Lien, "Point-based minkowski sum boundary," in *PG '07: Proceedings of the 15th Pacific Conferenc on Computer Grapics and Applications*. IEEE Computer Society, 2007, pp. 261–270.

[17] J.-M. Lien, "A simple method for computing minkowski sum boundary in 3d using collision detection," in *The Eighth International Workshop on the Algorithmic Foundations of Robotics*, 2008.

〈저자소개〉

경민호

· 1993년 포항공과대학교 전자계산학 학사
· 1995년 포항공과대학교 전자계산학 석사
· 2001년 Purdue University, Computer Science, PhD.

Elisha P. Sacks

· 1982년 Carnegie-Mellon University, Computer Science, BS
· 1985년 MIT, Computer Science, MS
· 1988년 MIT, Computer Science, PhD