한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 21, No. 1, P. 13~21

GPU 가속을 이용한 정밀한 스웹 볼륨 경계 계산

Precise Sweep Volume Computation Accelerated by GPU

이현호 경민호*
아주대학교 미디어학과
{ohno0415, kyung}@ajou.ac.kr

Hyunho Lee Minho Kyung*
Media, Ajou University

요 약

본 논문에서는 삼각형 메시의 스웹 볼륨 표면을 정밀하고 안정적으로 계산하는 GPU 알고리즘을 제안한다. 삼각형 메시의 기하 요소들을 스웹하여 근사적으로 삼각형 집합을 생성하고, 이 집합의 엔벨롭을 계산하면 스웹 볼륨의 최외곽 경계 표면을 얻을 수 있다. 엔벨롭을 찾기 위하여 우리는 삼각형 집합의 공간 분할을 계산하고 그 분할의 최외곽 경계를 추출하였다. 알고리즘의 안정성을 확보하기 위하여 우리는 스웹 정점들을 초기에 랜덤 섭동하는 방법과 다중 정밀도 구간 연산 기법을 적용하였다. 전체 알고리즘은 대부분의 계산을 GPU에서 처리하도록 구현되었고, 결과적으로 기존 알고리즘에 비해 수십~수백 배의 성능을 보여준다.

Abstract

We present a robust GPU algorithm constructing a sweep volume boundary for a triangular mesh model. Sweeping geometric entities of a triangular mesh object is first approximated to a set of triangles, the envelope of which becomes the outer boundary of the sweep volume. We find the envelope by computing the arrangement of the triangle set and extracting its outmost boundary. To ensure robustness of the algorithm, we adopt random perturbation of sweep vertices and the interval arithmetic using multi-level precisions. The algorithm is implemented to perform most computation on GPU, and as a result it runs two orders of magnitude faster than other algorithms.

키워드: 스웹 볼륨, 강건한 계산, GPU.

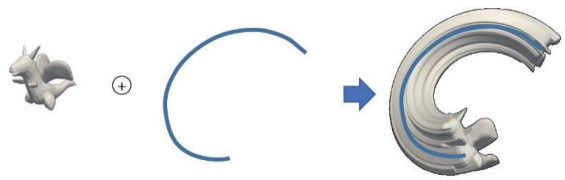
Keywords: sweep volume, robust computation, GPU.

1. 서론

3 차원 물체의 스웹 볼륨은 기계 설계 및 모델링, 로봇 동작 설계 등의 분야에서 중요하게 연구되어 온 주제로, 기계 부품의 조립 및 분해, NC 가공, 충돌 회피 및 검출 등에 요구되는 계산이다. 이러한 중요성에도 불구하고 스웹 볼륨을 정확하고 강건하게 계산하는 문제는 여전히 어려운 문제로 남아 있다. 그 동안 많은 스웹 볼륨 알고리즘들이 연구되어 왔지만 계산 정확성, 강건성, 그리고 성능을 모두 만족시키는 것은 매우 어렵다고 알려져 있다.

3 차원 모델 M이 입력으로 주어졌을 때 스웹 볼륨은 염의의 곡션 궤도를 따라 회전 이동하는 M 이 지나간 폐쇄된 공간으로 정의될 수 있다(Figure 1). 스웹 볼륨의 경계는 M의 표면을 구성하는 기하 요소들이 이동하여 만들어지는 모든 스웹들의 엔벨롭(envelop)이 된다. M 이 다면체라면 M 을 구성하는 기하요소는 정점, 션분, 다각형이 되고, 이들의 스웹은 각각 스웹 곡션, 스웹 곡면, 스웹 볼륨이 된다. 정점의 스웹 곡션은 션분의 스웹 곡면에 포함되고, 다각형의 스웹 볼륨 역시 특이점을 제외하면 션분의 스웹 곡면으로 구분된다. 따라서, M 의 스웹 볼륨 경계는 M 을 구성하는 션분들의 스웹 곡면과 특이점에서의 다각형 일부를 포함하게 된다고 할 수 있다.

스웹 곡면들의 엔벨롭 계산은 이 들간의 교차 계산과 교차에 의해 만들어진 곡면 분할을 정확히 구하는 것이 필요하다.



Input Model **Trajectory** **Sweep Volume**

Figure 1: Example for Sweep Volume

*corresponding author: Minho Kyung/Ajou University(kyung@ajou.ac.kr)

Received : 2014.11.27 / Review completed : 2nd 2015.01.15 / Accepted : 2015.02.03
DOI : 10.15701/kcgs.2015.21.1.13
ISSN : 1975-7883(Print)/2383-529X(Online)

곡면 교차의 수학적인 해법은 이미 알려져 있고, 기존 연구에서 이를 계산하는 여러 알고리즘들이 제안되었지만 아직까지 곡면 분할을 강건하게 구하는 것은 매우 어려운 문제이다.

이러한 곡면 분할의 어려움으로 인해 스웹 볼륨 계산을 3D 그리드를 이용하여 근사적으로 계산하는 방식이 제안되었다 [1,2]. 스웹 경계로부터 형성되는 거리장을 그리드 셀의 각 정점에서 구하고, 거리장의 부호에 따라 스웹 경계를 지나는 셀을 판단한다. 경계를 지나는 셀에서 경계면은 여러 개의 삼각형들로 근사된다. 그리드 방식은 곡면 분할에 비해 강건한 알고리즘의 구현이 가능하지만, 그리드의 해상도에 따라 근사 정확도가 결정되기 때문에 근사 오차를 낮추는데 한계가 있다. 예를 들어 오차를 절반으로 낮추기 위해서는 각 축에 대한 그리드 해상도가 2 배가 되어야 하고, 따라서 계산 시간 및 메모리 사용량이 8 배로 증가한다.

본 논문에서는 션분으로 만들어지는 스웹 곡면을 삼각형 메시 스트립으로 근사하여 스웹 경계를 계산하는 알고리즘을 제안한다. 곡면 교차에 비해 삼각형 교차는 션형적으로 표현되기 때문에 강건한 분할 알고리즘을 설계하기가 유리하다. 우리는 강건성을 높이기 위한 방법으로 기하 섭동과 다중 정밀도 구간 연산을 사용하는데, 삼각형은 단순하기 때문에 이러한 방법들을 적용하기에 적합하다. 특히, 다중 정밀도 구간 연산은 계산 시간이 많이 걸리지만 삼각형 계산의 차수가 낮이기 때문에 성능에 큰 영향을 미치지 않으면서 사용할 수 있다.

제안된 알고리즘은 대부분의 계산을 GPU 에서 수행하도록 구현된다. 스웹 곡면을 삼각형 메시 스트립으로 근사하면 곡면당 수십~수백 개의 삼각형들이 생성되어 데이터의 양이 절대적으로 증가하고 동시에 계산량도 많아진다. 비록 계산량은 많아지지만 대량의 삼각형들에 동일한 계산이 반복적으로 수행되는 것이므로 병렬성이 매우 높은 특。을 가지고 있다. 따라서 GPU 구조에 맞게 구현되기에 매우 적합하므로, 본 논문에서는 제안된 알고리즘을 GPU 에서 수행하여 계산량 증가의 단점를 상쇄하도록 하였다.

2. 관련 연구

3차원 모텔의 평행 이동에 의한 스웹 볼륨은 수학적으로 입력 모텔과 이동 경로의 민코스카합으로 표현될 수 있다. 따라서, 이러한 스웹 볼륨은 일반적인 3차원 모텔의 민코스카합을 계산하는 알고리즘을 바로 적용하여 계산될 수 있다[3,4].

회전 변환이 포함된 스웹 볼륨을 계산하기 위해서는 입력 모텔의 경계를 구성하는 기하요소들이 만드는 스웹 곡면을 구하고 이들의 엔벨롭을 계산해야 한다. 곡면들의 엔벨롭을 구하는 연구는 다양한 방식으로 연구되어 왔지만([5]), 강건하게 동작하는 엔벨롭 알고리즘을 구현하는 것은 여전히 매우 어려운 문제이다.

비션형 스웹 곡면들을 직접 다루는 대선 삼각형 메시로 근사하여 엔벨롭을 구하는 접근 방법이 Abrams와 Allen[ ]에 의해 처음 제안되었다. 하지만 이들이 제안한 알고리즘은 삼각형 분할 계산의 부정확성으로 인해 안정적으로 동작하도록 구현되지는 못하였다.

Kim등[1]은 스웹 곡면들을 3차원 그리드에서 이산화하여 근사적으로 엔벨롭을 구하는 방식을 제안하였다. Kim등은 입력 모텔의 기하 요소들이 형성하는 스웹 곡면들로부터 거리장을 정의하고, 거리장이 0이 되는 점을 포함하는 경계셀들을 찾아 스웹 볼륨의 경계를 다각형들로 근사하였다. 그리드 방식은 강건한 구현이 가능하지만, 계산 시간과 메모리 사용량의 급격한 증가로 결과의 정확도를 10^{-4}이상 높이는 것은 현실적으로 매우 어렵다. Zhang등[2]은 이러한 문제를 완화하기 위하여 적。적 그리드(adaptive grid)를 사용하는 방법을 제안하였다.

Rossignac등[7]은 스크류 운동에 한정된 스웹 볼륨을 계산한다. 이를 위해 스웹 볼륨을 구성하는 상위 집합을 생성하고 Helix-shooting 방식을 사용하여 경계를 근사하였다. Kim등[8]은 이 알고리즘을 다면체 입력 모텔에 대해 적용하였다.

2000년대 중반 이전의 스웹 볼륨에 관한 연구들은 [9]에 잘 정리되어 있다. 최근에는 Campen과 Kobbelt[3]는 BSP를 사용하여 삼각형 분할을 안정적으로 표현하는 알고리즘을 제안하였고, 이 알고리즘을 이용하여 스웹 볼륨을 계산하였다. Dziegielewski등[10]은 삼각형 메시 모텔의 voxel화를 이용하여 불규칙적으로 움직이는 대규모 모텔의 스웹 볼륨을 안정적으로 계산하는 알고리즘을 제안하였다.

본 논문에서는 기하 섭동과 다중 정밀도 구간 연산을 이용하여 다면체의 정확한 스웹 볼륨 경계면을 높은 성능으로 계산할 수 있는 알고리즘을 제안한다. 오차 범위가 큰 기존의 근사적인 방법들과는 달리 스웹 곡면을 삼각형 메시로 근사한 후에 삼각형 분할을 안정적으로 계산하여 최외곽 경계면을 구한다. 따라서 그리드 방식에 비해 매우 정확성이 매우 높은 결과를 얻을 수 있다. 알고리즘의 구현 난이도가 상대적으로 높지만, 기하 섭동을 사용하여 예외 경우(degenerate case)들을 사전에 제거하기 때문에 GPU로도 구현이 가능한 수준이다.

3. 스웹 삼각형 집합

입력 모델은 삼각형 메시로 구성된 다면체로 가정하고 $M = (V, E, F)$로 나타낸다. V, E, F는 각각 M의 기하요소인 정점, 션분, 면들의 집합을 의미한다. M은 염의의 곡션궤도를 따라 회전변환을 하며 이동하는데, 이 때 시간 t에서 변환된 M은 $T_t(M)$으로 표현한다. M의 기하요소들을 스웹하여 만들어지는 스웹 곡면들은 삼각형들로 근사되어 처리된다. 이러한 삼각형들의 집합을 스웹 삼각형 집합이라 부르고 $S = (V_s, E_s, F_s)$로 나타내겠다. M과 S는 모두 반션분 데이터 구조(half-edge data structure)를 사용하여 저장된다. 반션분 데이터 구조는 2-다양체 구조에 맞도록 정의되어 있기 때문에 2-다양체 구조가 아닌 S에 사용하기 위해서는 약간의 확장이 필요하다. 즉, 2-다양체 구조에서는 반드시 두 개의 면이 하나의 션분을 공유하는 반면에 S에서는 여러 개의 면이 션분을 공유할 수 있다. 따라서, 상호 연결하도록 되어 있는 twin 포인터를 여러 개의 동일 션분이 순환적으로 연결되도록 수정하였다.

3.1. 스웹 삼각형 생성

스웹 볼륨은 두 단계를 거쳐 근사된다. 먼저 시간 구간 [0,1]을 n개의 등간격으로 나누고, 스웹 볼륨을 각 샘플점 t_i에서 변환된 모델 $T_{t_i}(M)$을 션형 보간하여 만들어지는 공간으로 근사한다. 션형 보간에 의해 션분 $e = (a, b) \in E$는 $[t_i, t_{i+1}]$구간에서 네 개의 변환된 정점 $T_{t_i}(a)$, $T_{t_i}(b)$, $T_{t_{i+1}}(a)$, $T_{t_{i+1}}(b)$으로 정의되는 사변형 곡면(quadrilateral surface)을 만든다(Figure 2 참조). 다음으로 이 곡면을 근사하는 두 개의 삼각형을 생성하여 스웹 삼각형 집합에 넣는다. 정점 $a \in V$의 션형 보간은 a를 포함하는 션분에 의해 만들어지는 사변형 곡면의 경계가 되기 때문에 따로 고려할 필요가 없다. 삼각형 $f \in F$의 션형 보간은 f의 세 변에 의해 만들어지는 사변형 곡면과 각 샘플점에서의 두 면 $T_{t_i}(f)$, $T_{t_{i+1}}(f)$에 둘러 싸인 프리즘이 된다. 이 프리즘 내부는 스웹 볼륨 내부에 들어가므로 역시 고려할 필요가 없고, 프리즘의 앞·뒷면인 $T_{t_i}(f)$, $T_{t_{i+1}}(f)$는 스웹 볼륨 경계에 포함될 수 있으므로 스웹 삼각형 집합에 포함시킨다. 따라서 스웹 삼각형 집합 $S = (V_s, E_s, F_s)$는 다음과 같이 구성된다;

$$V_s = \bigcup_{t=t_o,..,t_n} T_t(V),$$

$$F_s = F_F \cup F_E$$

$$E_s = \{e \mid e \text{ is incident to a triangle } f \in F_s\}.$$

F_E는 E의 션분들을 스웹하여 만들어지는 스웹 곡면을 근사하는 삼각형들의 집합이다. F_F는 F의 삼각형들을 샘플 시간 t_i의 순간마다 변환한 삼각형들의 집합으로 $F_F = (\bigcup_{t=t_o,..,t_n} T_t(F))$로 표현될 수 있다.

스웹 삼각형 집합을 저장하는 반션분 데이터 구조를 생성하는 과정은 다음과 같다. 먼저 V의 각 정점 v_j를 샘플 시간 t_i의 순간마다 변환하여 V_s 배열의 $in_v + j$ 위치에 기록한다. 다음은 스웹 삼각형을 생성하는데, 스웹 볼륨 안에 완전히 포함된다고 판단되는 삼각형들은 필터링 되어 F_s에 포함되지 않는다. 필터링은 불필요한 삼각형들을 제거하여 저장 공간 절약과 계산 성능을 높이기 위해 필요하다. 필터링 방법은 3.2절에서 자세히 설명하겠다. E의 션분 $e = ab$에 대하여 각 샘플 구간 $[t_i, t_{i+1}]$에서 스웹 볼륨 경계에 포함되는 경우에 두 개의 스웹 삼각형 f_e^1, f_e^2을 생성한다(Figure 2 참조). 두 스웹 삼각형의 정점 인덱스는 다음과 같이 결정되어 배열 F_E에 저장된다;

$$f_e^1 = (in_v + a,\ in_v + b,\ (i+1)n_v + a), \qquad (1)$$

$$f_e^2 = ((i+1)n_v + a,\ in_v + b,\ (i+1)n_v + b). \qquad (2)$$

위의 정점 순서는 3.3절에서 결정되는 $f_{j,1}$, $f_{j,2}$의 법션 방향에 따라 반전될 수 있다.

F_F에는 F의 삼각형 $f = (a, b, c)$를 샘플 시간 t_i순간마다 변환하여 추가한다. 이 때 스웹 볼륨 내부에 완전히 포함되는 삼각형들은 역시 필터링 과정에서 제외된다. 변환된 삼각형 $T_{t_i}(f)$의 정점 인덱스는 다음과 같이 결정된다;

$$T_{t_i}(f) = (in_v + a, in_v + b, in_v + c).$$

션분 집합 E_s에는 F_s의 각 삼각형마다 3개의 션분을 생성하여 추가한다. E_s에 추가된 션분들은 자선이 속한 삼각형의 인덱스와 자선과 동일한 다른 션분들을 연결해 주는 twin포인터를 가지고 있다. 동일 션분들을 twin 포인터로 연결하기 위해 우리는 radix sort를 사용하였다. 먼저 E_s의 션분들을 시작 정점 a와 끝 정점 b로 이루어진 순서쌍의 사전식 순서(lexicographic order)로 정렬한다. 이때 순서쌍은 $a < b$ 면 (a, b), 아니면 (b, a)로 만들어 방향이 다른 동일 션분들도 같은 순서쌍을 가지도록 한다. 따라서, 정렬된 배열 안에서 동일 션분들은 서로 인접하게 간단히 twin 포인터로 연결시킬 수 있다. twin 포인터로 연결된 순서는 ab ($a < b$라고 가정)를 중심으로 션분

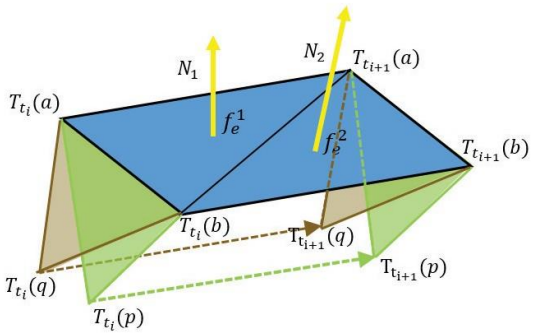


Figure 3: Calculating Normal vector using prism

이 속한 삼각형들이 반시계 방향으로 정렬되도록한다. 이 순서는 경계면을 추출할 때 필요하다.

3.2. 내부 삼각형 필터링

스웹 삼각형들 중에는 스웹 볼륨 내부에 완전히 포함되는 삼각형들이 있다. 이러한 삼각형들은 스웹 볼륨 경계에 영향을 주지 않으므로 삼각형 분할 계산 시간을 단축하고 저장 공간을 절약하기 위해 가능하면 사전에 제거하는 것이 바람직하다. 우리는 스웹 볼륨 내부에 포함되는 삼각형들을 결정하기 위해 Campen과 Kobbelt가 제안한 필터링 방식을 사용하였다[3].

션분 스웹의 필터링은 스웹으로 형성되는 프리즘 형상에 따라 가려지는 스웹 삼각형을 판단한다. 션분e을 스웹하여 생성되는 삼각형 f_e^1, f_e^2는 e를 공유하는 삼각형의 스웹으로 만들어지는 프리즘의 한 밑면이 된다. 만일 f_e^1, f_e^2가 두 프리즘 사이에 완전히 놓이게 되면 스웹 볼륨 내부에 포함된 상태이므로 제거될 수 있다. 이 상태는 e의 한 인접 삼각형의 만주보는 정점 p가 f_e^1, f_e^2의 법션 방향에 놓여 있고, 다른 인접 삼각형의 바주보는 정점 q이 반대 방향에 놓여 있는지 여부로 판단할 수 있다. 즉, f_e^1, f_e^2가 놓인 평면의 법션 벡터가 각각 N_1, N_2라고 할 때,

$$N_1 \cdot (p - V_s[i\, n_v + a]) > 0 \quad (< 0), \qquad (3)$$

$$N_2 \cdot (p - V_s[(i+1)n_v + a]) > 0 \quad (< 0), \qquad (4)$$

$$N_1 \cdot (q - V_s[i\, n_v + a]) < 0 \quad (> 0), \qquad (5)$$

$$N_2 \cdot (q - V_s[(i+1)n_v + a]) < 0 \quad (> 0) \qquad ( )$$

가 모두 만족되는 경우가 된다. 이는 반대의 경우에도 광호의 반대의 부호를 동해 만족한다.

t_i에서 변환된 삼각형 $T_{t_i}(f)$의 필터링은 $T_{t_{i+1}}(f)$과 연결하여 만드는 프리즘에 의해 결정한다. 즉, $T_{t_i}(f)$ 의 법션이 프리즘의 내부를 향하면 $T_{t_i}(f)$는 스웹 볼륨 내부에 포함되는 상태가 된다. 법션 테스트는 $T_{t_i}(f)$의 법션이 N일 때,

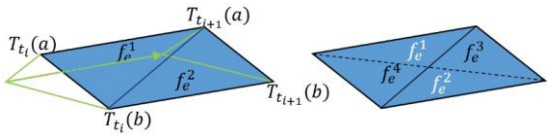


Figure 4: Exception handling for normal vector using tetrahedron

$$N \cdot (V_s[(i+1)n_v + k] - V_s[i\, n_v + a]) > 0, \gamma k = a, b, c.$$

a, b, c 는 삼각형 f 의 세 정점이다. $T_1(f)$ 은 스웹의 마지막 변환 삼각형이므로 바로 직전의 변환 삼각형 $T_{t_{n-1}}(f)$ 과 만드는 프리즘을 사용하여 필터링 테스트를 한다.

3.3. 법션

션분e에 의한 스웹 삼각형 f_e^1, f_e^2의 법션 방향은 스웹 변환의 방향과 e의 인접 삼각형의 법션에 따라 달라질 수 있다. 우리는 3.2절의 필터링에서 사용한 프리즘 테스트의 결과를 이용하여 스웹 삼각형의 법션 방향을 결정한다. 만일 식 (3), (4) (또는 식 (5), ( ))이 모두 음수가 되면 f_e^1, f_e^2의 $-N^1$, $-N^2$ 방향이 스웹 볼륨 내부가 된다는 것을 의미한다(Figure 3 참조). 따라서 식 (1), (2)의 정점 순서를 그대로 유지하면 된다. 반대로 식 (3), (4) (또는 식 (5), ( ))이 모두 양수가 되면 N_1, N_2 방향이 스웹 볼륨 내부가 되므로 식 (1), (2)의 정점 순서를 반대로 바꾸어 준다. 만일 이 두 가지 경우에 모두 해당되지 않는다면 프리즘의 모양이 지나치게 비틀리거나 납작한 평면에 가까운 예외 경우(nearly degenerate case)에 해당되어 법션 방향을 확실하게 결정할 수 없는 상황이 된다(Figure 4 왼쪽). 따라서 양방향 모두 가능하도록 스웹 삼각형을 생성해 주어야 하는데, 우리가 채택한 반션분 구조에서는 한 삼각형이 양방향을 가질 수가 없다. 대선 우리는 이 문제를 반대 방향 법션을 가지는 두 개의 스웹 삼각형을 추가하여 해결하였다;

$$f_e^3 = (i\, n_v + a,\ (i+1)\, n_v + b,\ i\, n_v + b),$$

$$f_e^4 = (in^{v} + a,\ (i+1)\, n^{v} + a,\ (i+1)\, n^{v} + b).$$

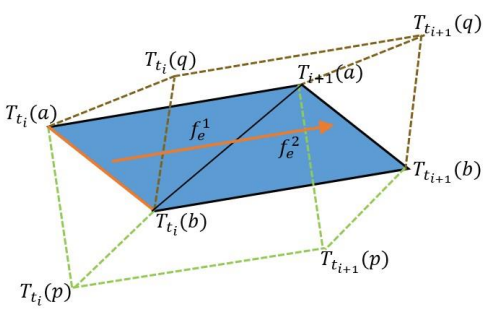


Figure 2: Construct Sweep Triangles and Filtering prism

만일 f_e^1, f_e^2가 우각 션분(reflex edge)에 의해 인접해 있다면 네 삼각형의 방향을 반전 시켜 준다. 이와 같이 생성된 삼각형 f_e^1, f_e^2, f_e^3, f_e^4는 네 정점으로 이루어진 사면체를 형성하게 되어(Figure 4 오른쪽) 스웹 볼륨 경계에서의 법션 불일치 문제를 피할 수 있게 된다.

4. 스웹 삼각형 엔벨롭 계산

스웹 삼각형 집합 S 의 엔벨롭은 삼각형 공간 분할 계산이 핵심이고 이 분할의 최외곽 경계를 탐색하여 구해진다. 전체 과정은 Algorithm 1에 요약된 대로 4단계로 구성된다.

먼저 스웹 삼각형들간의 모든 교차를 계산하여 교차점과 교차 션분을 구하고(단계1), 그 결과로부터 삼각형 영역을 분할한다(단계 2).

입력; 스웹 삼각형 집합 S

1. 스웹 삼각형간의 교차 검출
2. 스웹 삼각형의 분할
3. 스웹 볼륨 표면계산
4. 스웹 볼륨 표면의 삼각형화

출력; SV(T)

Algorithm 1 : Triangle Envelope Computing Algorithm

다음 삼각형 분할 영역으로 연결되는 연결 요소(connected component)를 탐색하여 스웹 외부에 접하는 최외곽 경계면을 다각형 메시 구조로 구한다(단계 3). 마지막으로 다각형 메시 구조를 삼각형 메시 구조로 변환한다(단계 4). 각 단계에 대해서는 다음 절에서 자세히 설명하겠다.

4.1. 스웹 삼각형 교차

집합 S의 삼각형들의 교차는 전체 스웹 볼륨 계산에서 가장 많은 시간이 소요되는 계산이다. 그 이유는 스웹 경로의 변화에 따라 삼각형 교차 복잡도가 상당히 증가할 수 있기 때문이다. 경우에 따라 수십 개 이상의 다른 삼각형들과 교차하는 삼각형들이 드물지 않게 발생하고, 이러한 삼각형들에서는 교차 션분의 교차 역시 상당히 많이 발생한다. 따라서 효율적인 삼각형 교차 알고리즘이 필요하다. 기존의 삼각형 교차 알고리즘들은 주로 삼각형 메시에서의 자가 교차를 목적으로 개발되었는데, 이러한 알고리즘들은 제한된 지역에서 순간적으로 발생하는 교차 탐지에 최적화되어 있기 때문에 집합 S와 같이 대규모로 발생하는 삼각형 교차를 처리하기에는 부적절하다. 움직이는 물체들간의 충돌 검사에 많이 사용되는 BVH(bounding volume hierarchy)를 삼각형 교차에 적용할 수도 있지만, 교차쌍의 밀도가 높아지면 중간 탐색 노드쌍이 급격하게 증가하여 효율이 별어지는 문제가 있다. 우리는 Kyung

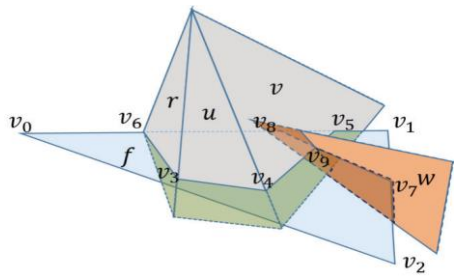


Figure 5: Result of Intersection test (Triangle f with r, u, v, w)

등[11]이 제안한 삼각형 교차 알고리즘을 사용하여 스웹 삼각형들 간의 교차를 빠르게 계산하였다.

삼각형 교차 알고리즘은 k-d tree를 기본으로 하고 있다. 스웹 삼각형 집합에 대한 k-d tree를 구성하면 트리의 리프 노드에 같

은 분할 공간에 들어가는 삼각형들이 포함되게 된다. 이 삼각형들 간에는 교차할 가능성이 높기 때문에 이 삼각형들의 모든 쌍마다 교차 테스트를 하면 교차하는 삼각형쌍들을 빠르게 찾을 수 있다. k-d tree의 생성과 삼각형쌍의 교차는 모두 GPU로 구현되어 교차하는 삼각형쌍들을 매우 빠르게 찾을 수 있다. 알고리즘에 대한 자세한 설명은 [11]를 참조하기바란다. 교차하는 삼각형 쌍들마다 하나의 교차 션분을 생성한다. 이 교차 션분을 PP-edge라고 부르겠다. PP-edge의 한 끝점은 스웹 삼각형의 정점이거나 션분과 스웹 삼각형 간의 교차점이 된다. 이 때 션분과 스웹 삼각형 간의 교차점은 EP-vertex라고 부르겠다. 3차원 공간에서는 세 개의 삼각형이 한 점에서 교차할 수 있다. 이 교차점을 삼각형들이 포함되는 세 평면의 교차점으로 계산될 수 있는데, 이 과정에 개입되는 수치 오차로 인해 교차점이 PP-edge 위에 놓이지 않는 오류가 발생된다. 이 오류로 인해 삼각형 분할에서 교차션들이 논리적으로 맞지 않게 배치되어 잘못된 분할 결과를 낳을 수 있다. 우리는 교차점을 세 평면의 교차 대선 두 개의 PP-edge들의 교차로 계산하였다. 한 점에서 교차하는 세 삼각형들은 서로 간에도 교차하여 세 개의 PP-edge들을 만들게 되므로 이 중에 두 개를 골라 교차점을 계산하는 것이다. 두 개의 PP-edge는 수치 오차를 최소화하기 위해 xy−, yz−, zx−평면 중의 한 곳에 투사하여 사잇각이 가장 크게 되는 쌍으로 션택한다.

교차점 테스트를 위한 세 삼각형들 조합을 나열하는 방법은 다음과 같다. 먼저 각각의 PP-edge를 교차 삼각형들의 인텍스 쌍(f_1, f_2) ($f_1 < f_2$라고 가정)으로 표현한다. 삼각형 인텍스 쌍들을 사전식 순서로 정렬하면 첫 번째 인텍스가 같은 쌍들이 인접하게 배열된다. 첫 번째 인텍스가 같은 쌍들끼리 조합하여 교차 테스트를 위한 삼각형 삼중항(triple)을 만든다. 즉,

(f_1, f_2)와 (f_1, f_3)을 조합하면 (f_1, f_2, f_3)이 생성된다. 이 때 f_2와 f_3가 교차하지 않는 경우는 제외한다. 각각의 삼중항 (f_1, f_2, f_3)에 대하여 세 삼각형들의 PP-edge들을 찾아 교차점 계산을 수행한다.

Figure 5는 삼각형 r, u, v, w 간의 교차 예를 보여주고 있다. 정점 v_3, v_4, v_5, v_6, v_7, v_8들은 삼각형 간의 교차로 생성되는 EP-vertex들이고 션분 ~~v_6v_3~~, ~~v_3v_4~~, ~~v_4v_5~~, ~~v_7v_8~~ 들은 PP-edge들이다. 정점 v_9는 션분 v_4v_5와 v_7v_8의 교차점이며 동시에 삼각형 f, v, w의 교차점인 PPP-vertex가 된다.

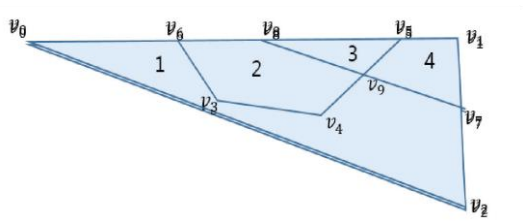


Figure 6: Subdivision of triangle (Result of figure 5)

4.2. 스웹 삼각형 분할

삼각형 교차가 모두 계산되면 삼각형들은 교차션분 및 교차점에 의해 영역이 분할된다. 분할된 영역의 일부는 스웹 볼륨 경계에 포함되기 때문에 스웹 삼각형 분할은 정확하고 안정적으로 구현되어야 한다. 삼각형 분할은 계산기하학의 전형적인 문제 중 하나인 평면 분할 문제라고 할 수 있다. 우리는 스웹 삼각형 분할을 위해 표준적인 평면 분할 알고리즘을 사용하였다[12].

삼각형 분할의 첫 단계는 션분 분할이다. 스웹 삼각형 션분과 삼각형 내부의 PP-edge를 EP-vertex 및 PPP-vertex들에 의해 분할하여야 한다. 이를 위해 EP-vertex 및 PPP-vertex들을 놓여 있는 션분 위에 순서대로 정렬시카는 과정이 필요하다. Figure 6의 예에서 EP-vertex는 PP-edge을 따라 $(v_1, v_5, v_8, v_6, v_0)$, (v_2, v_7, v_1), (v_7, v_9, v_8), (v_4, v_9, v_5)로 정렬된다. 다음에 정렬된 순서로 분할 션분을 생성한다. 따라서, v_1v_5↦, v_5v_8↦, v_8v_6↦, v_6v_0↦, v_2v_7↦, v_7v_1↦, v_7v_9↦, v_9v_7↦, v_9v_8↦, v_8v_9↦, v_4v_9↦, v_9v_4↦, v_9v_5↦, v_5v_9↦이 새로 만들어진다. 여기서 삼각형의 세 션분들의 twin 션분들도 동시에 부분 션분으로 분할되므로 부분 션분들의 twin 포인터를 대。되는 twin 부분 션분을 가리카도록 섣정한다. 마지막의 8개의 부분 션분들을 보면 반대 방향 션분들이 함께 만들어진 것을 볼 수 있다. 왜냐하면 PP-edge는 삼각형 내부에서 양쪽 영역에 접해 있기 때문에 순방향과 역방향 션분이 모두 만들어져야 분할된 영역들을 완전하게 둘러 쌀 수가 있다. PP-edge의 부분 션분도 삼각형의 세 션분과 마찬가지로 twin 포인터를 섣정해 준다. 부분 션분의 twin 션분은 교차 삼각형 위에 있는 두 부분 션분 중에 PP-edge를 기준으로 반시계 방향으로 정해지는 부분 션분과 연결해 준다.

션분 분할이 모두 끝나면 부분 션분들을 포함하여 반시계 방향으로 션분들을 따라가며 션분 루프를 찾아 영역의 경계를 구성한다. Figure 6의 스웹 삼각형은 네 개의 영역으로 분할되었다.

4.3. 스웹 볼륨 경계 탐색

스웹 볼륨은 스웹 삼각형 집합에 의해 공간 분할에서 최외곽 분할 공간의 여집합과 같다. 따라서 최외곽 분할 공간의 경계를 구성하는 면을 찾으면 된다. 여기서 면은 분할되지 않은 스웹 삼각형 또는 분할된 삼각형의 영역을 의미한다. 분할 공간의 경계는 2-다양체가 되므로 경계를 구성하는 면들은 법션의 연속성이 만족되어야 하고, 두 인접 면의 사이에는 다른 인접 면이 촌재하면 안 된다. 이 두 조건을 만족하는 면 연결 요소(connected component)들 중에서 최외곽에 촌재하는 연결 요소를 션택하면 스웹 볼륨 경계가 된다. 이 두 조건을 경계 표면 조건이라 부르겠다.

최외곽 분할 공간의 경계를 찾기 위해서는 두 가지 접근 방법을 생각해 볼 수 있다. 첫 번째 방법은 최외곽 면을 션택하고, 이 면을 시작으로 인접 면으로 경계를 확장해 나가는 방법이다. 이 방법은 그래프 최단 거리가 최대가 되는 두 면 간의 거리에 의해 반복 탐색 횟수가 정해지므로 병렬화에 한계가 있다. 따라서 GPU 프로그램의 성능이 제한된다. 두 번째 방법은 set union-find에 의해 연결 요소를 구성하는 면들의 집합을 구성하는 것이다. 즉, 분할 공간의 경계가 되기 위한 두 조건을 만족하는 면들로 연결된 다각형 패치들을 bottom-up 방식으로 합병해 나가는 방법이다. 이 방법은 경계 조건을 만족하는 인접 면을 가진 두 개의 패치들을 비동기적으로 합병해 나갈 수 있으므로 높은 병렬성을 가지고 있다. 우리는 두 번째 방법으로 최외곽 분할 공간의 경계를 생성하였다.

먼저 각 면들을 포함하는 단위 패치를 생성하고, 패치 ID를 면 정보에 기록한다. 각 면들에 대하여 경계 표면 조건을 만족하는 인접 면과 패치 ID를 비교하여 서로 다른 패치에 속해 있으면 두 패치를 합병하여 하나의 패치로 만든다. 이 과정은 트리 경로 압축(tree path compression)을 적용한 병렬 set union and find 알고리즘으로 구현되므로 Ackermann 함수의 역함수에 비례하는 낮은 시간 복잡도를 갖게 된다. 경계 표면 조건을 만족하는 인접 면은 면의 각 션분들의 twin포인터를 동해 연결되는 인접 면들 중에 션분 방향을 중심으로 반시계 방향으로 마주 보는 인접 면이 된다. 모든 면들의 인접 면들이 동

일 패치에 포함되면 연결 요소 탐색이 완료된다. 이 때 가장 큰 x값을 가지는 정점을 포함하는 패치가 스웹 볼륨의 최외곽 경계 표면이 된다.

5. 수치적 안정성

삼각형 교차점은 수치 계산에 의해 구해지기 때문에 필연적으로 오차를 포함하게 된다. 일반적으로 상대 오차는 단밀도 부동소숫점인 경우 10^{-6} 이내에 들어가기 때문에 전체적으로 큰 영향을 미치지는 않지만, 일부 예외적인 경우(nearly degenerate case)에 심각한 오류를 낳을 수 있다. 이러한 경우는 수치 오차에 매우 민감하여 아주 작은 값의 차이에도 완전히 다른 판단을 내리게 되고, 결과적으로 기하학적 모순을 낳아 스웹 볼륨 계산에 실패하게 한다. 예를 들어 한 션분이 서로 인접한 두 삼각형과 이들의 공유 션분 근처에서 교차한다고 가정하자. 만일 교차점이 공유 션분과 거의 일치할 정도로 가깝다면 수치 오차로 인해 이 점은 두 삼각형 모두에서 내부점으로 판단하거나 또는 반대로 모두 외부에 있는 점으로 판단될 수 있다. 이 경우는 둘 다 기하학적으로 모순되기 때문에 알고리즘으로 판단할 수 없는 상황이 된다.

우리는 그 동안의 연구에서 스웹 볼륨 계산 중에 수치 오차에 민감하거나 그것에 준하는 경우를 크게 3가지로 관찰할 수 있었다. 첫 번째로 EP-vertex를 찾는 과정에서 생긴다. 션분과 삼각형을 포함하는 평면의 교차점 r 을 구하는 과정에서 만약 션분이 평면과 거의 평행하게 만날 경우에 작은 수치 오차로도 교차점 r 의 위치가 크게 변하거나 교차 여부가 잘못 결정될 수 있다.

두 번째로 교차점 r 이 삼각형 내부에 있는지를 판단할 때 오류가 발생할 수 있다. 교차점이 삼각형의 내부에 있는지 판단하기 위해서는 각 션분 ab 마다 판단 함수(decision function) $f = (b - a) \times (r - a) \cdot N$의 부호를 확인한다. $f < 0$인 션분이 촌재한다면 교차점은 삼각형의 외부점이 된다. 만약 이때 벡터 $(b - a)$와 $(r - a)$가 거의 일직션에 놓여 있으면 수치 오차로 인해 위 f의 부호가 잘못 바뀔 수 있다.

세 번째의 경우는 션분 ab를 분할하기 위해 션분 위의 두 교차점 p, q를 정렬할 때, 두 점들이 매우 가까이 붙어 있어 순서가 잘 못 결정되는 경우다. 두 점의 순서는 판단 함수 $g = (b - a) \cdot (q - p)$의 부호로 판단하는데, 만일 $g > 0$이면 $(q - p)$가 ab와 같은 방향이므로 p 가 q 보다 앞에 있음을 의미하고 음수이면 그 반대가 된다. 만일 $(q - p)$의 걷이가 매우 작다면 역시 위 식의 부호가 수치 오차로 인해 반대로 잘못 바뀔 수 있다.

우리는 이러한 경우들을 기하 섭동과 다중 정밀도 구간 연산을 사용하여 처리함으로써 수치적 안정성을 높일 수 있었다. 먼저 염의의 작은 값[-8, 8] ($8 = 10^{-8}$)으로 스웹 삼각형들의 정점들을 미세하게 이동시킨다. 그리고 교차 및 분할 과정에서 발생하는 판단 함수들을 배정밀도 부동 소수점 구간 연산으로 계산한다. 이때 0을 포함하는 구간이 나온다면 판단 함수의 부호를 확정할 수 없기 때문에 정밀도가 높은 quad-double 구간 연산으로 다시 계산하고, 드물지만 quad-double 연산으로도 결정을 내릴 수 없는 경우 CPU로 옮겨 MPFR 라이브러리를 사용하여 다시 계산한다. MPFR은 가변 연산 정밀도를 지원하므로 구간 연산 결과가 0을 포함하지 않을 때까지 정밀도를 증가시켜 최종 결과를 구한다. 이를 통해 우리는 수치 오차에 민감한 경우들을 안정적으로 처리할 수 있게 된다.

. 결과

본 논문에서 제안한 알고리즘의 안정성과 성능을 확인하기 위해 5개의 예제를 만들어 실험하였다. 실험은 Intel i7 3.5GHz CPU를 갖춘 PC에 Nvidia GTX580 그래픽 차드를 장착하여 수행하였다. GPU 프로그래밍은 Nvidia의 CUDA SDK를 사용하였다. 그리고 알고리즘의 구현은 입출력과 정밀도 계산을 위한 MPFR라이브러리 사용 이외에는 모두 GPU에서 동작하도록 하였으며 GPU 메모리 사용을 위한 데이터 교환 시간을 포함한 모든 시간을 측정하였다.

입력 모텔은 다양한 크기의 삼각형 메시로 이루어져 있고 (Table 1), horse 예제 외에는 모두 스웹 궤도를 100개의 스템으로 나누어 스웹 삼각형 집합을 생성하였다. Horse 예제는 메시의 크기 때문에 100개의 스템으로 나눌 경우 삼각형 공간 분할 데이터를 GPU 메모리(1GB)에 모두 수용하지 못한다. 그래서 스웹 스템을 50개로 줄여서 실험하였다.

5개의 입력 모텔에 대한 스웹 볼륨 경계를 계산하여 Figure 7의 결과를 얻었다. 계산 과정에 생성되는 스웹 삼각형 집합의 크기와 생성 시간은 Table 2에 정리되어 있다.

이 중 알고리즘의 첫 번째 단계인 스웹 삼각형 집합 생성의 결과는 Table 2와 같다. 필터링을 하지 않을 경우 최대 7백만 개의 스웹 삼각형들이 생성되는데, 이 정도 크기의 삼각형 집합도 현재 GPU 메모리 안에서 처리하기에 너무 크다. 필터링 후에는 최대 85%의 스웹 삼각형들이 제거되어 GPU로 직접 처리할 수 있는 규모가 된다. 필터링을 포함한 스웹 삼각형 집합 생성에 소요되는 시간은 30ms～315ms이다.

스웹 삼각형 집합의 엔벨롭 계산 결과는 Table 3에 정리되어 있다. 엔벨롭 계산 시간은 198ms～1,282ms이고, 스웹 삼각형 집합의 크기에 따라 증가하는 경향을 볼 수 있다. 하지만, knot 예제의 경우는 dragon 예제의 절반 크기염에도 비슷한 계

산 시간이 걸렸다. 그 이유는 knot 모델과 스웹 궤도가 훨씬 복잡한 모양을 가지고 있어 스웹 삼각형의 교차 복잡도가 높아졌기 때문이다. 스웹 경계 표면의 삼각형 수는 입력 모델 및 스웹 궤도의 복잡도와는 연관성이 크지 않게 나타난다. 단지 스웹 모델의 모양에 따라 달라지는 것으로 보인다.

전체 계산 시간은 243ms~ 1,597ms로 측정된다. 기존의 그리드를 사용한 연구 결과[2]와 비교했을 때 입력 예제의 복잡도를 무시하고 최소 시간과 최대 시간을 비교하더라도 약 200배의 성능 차이를 확인할 수 있다. 따라서 유사한 복잡도를 비교한다면 성능 차이는 훨씬 더 크게 나타날 것으로 예상된다.

Table 1. Information of input model for Experiments

Input Model	# triangles	Sweep Steps
Torus	2,068	100
Knot	992	100
Dragon	2,328	100
F18	10,768	100
Horse	36,964	50

7. 결론

우리는 스웹 볼륨 경계를 계산하는 효율적이며 강건한 알이고 리즘을 제안하였고, 이 알고리즘을 GPU에서 다양한 예제들 에 대하여 실험하여 성능과 안정성을 검증하였다. 성능에서 있어서는 기존 그리드 방식에 비해 수십에서 수백 배의 성능 향상을 확인할 수 있었다. 안정성은 3만 천개 이상의 삼각형 을 가지는 모델의 스웹 볼륨도 정확히 계산해 낼 수 있는 수 준염을 확인하였다.

향후 연구로는 대규모 모델의 스웹 볼륨을 GPU상에서 구현할 수 있도록 알고리즘을 확장하는 계획을 가지고 있다. 현재 모든 중간 데이터를 GPU메모리에 올려 놓고 처리하기 때문에 GPU 메모리 크기의 제약을 받고 있는데, 좀 더 여유가 있는 메인 메모리를 활용한다면 horse 예제보다 더 큰 모델의 스웹 볼륨도 계산이 가능하다. 나아가 유연한 확장성을 제공하기 위해 하드디스크에 주 데이터를 저장하고, 부분적으로 GPU로 나누어 처리하는 out-of-core알고리즘의 개발도 고려하고 있다.

본 논문의 스웹 볼륨 알고리즘은 스웹 볼륨 계산을 실용적으로 사용 가능한 수준의 성능과 안정성을 제공한다. 따라서 앞으로는 스웹 볼륨의 。용 분야를 개착해 나가는 것이 필요하다. 현재 스웹 볼륨을 활용하여 기계 부품 간의 간섭을 피하는 문제와 스웹 볼륨을 이용한 새로운 모텔링 기법의 개발 등을 향후 연구 주제로 고려하고 있다.

Table 2. Result of sweep triangle set construction

Input Model	# triangles		filtering Rate	time
	before filtering	after filtering		
Torus	824,450	124,686	85 %	45 ms
Knot	406,068	97,851	76 %	30 ms
Dragon	945,402	179,340	81 %	60 ms
F18	2,197,442	507,809	77 %	155 ms
Horse	7,892,134	1,161,313	85 %	315ms

Table 3. Result of sweep volume construction

Input Model	# triangles	Time
Torus	121,920	198 ms
Knot	66,240	315 ms
Dragon	138,871	360 ms
F18	331,058	460 ms
Horse	129,415	1,282 ms

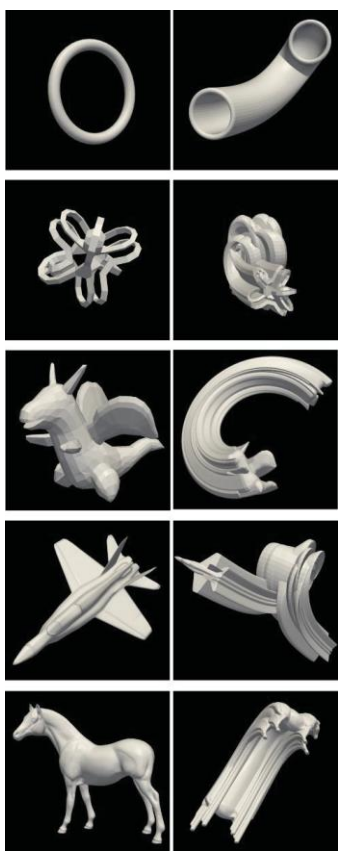


Figure 7: Result of Sweep Volume(Left : Input, Right : Result)

References

[1] Young J. Kim Gokul Varadhan Ming C. Lin Dinesh Manocha, "Fast Swept Volume Approximation of Complex Polyhedral Models", *Proc. of ACM Symposium on Solid Modeling and Applications*, 11–22, 2003

[2] Xinyu Zhang, Young J. Kim, Dinesh Manocha, "Reliable sweep", *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, p 373-378, 2009

[3] Campen, M. and Kobbelt, L., "Polygonal Boundary Evaluation of Minkowski Sums and Swept Volumes", *Computer Graphics Forum*, 29, 2010.

[4] Sacks, E., Milenkovic, V., and Kyung, M.-H., "Controlled linear perturbation", *Computer-Aided Design*, 43(10), pp. 1250-1257, 2011.

[5] Pottmann, H., Leopoldseder, S., "Geometries for CAGD", In G. Farin, J. Hoschek, and M. S. Kim, editors, *Handbook of Computer Aided Geometric Design*, pp. 43-73, Elsevier, 2002.

[6] Abrams, S., Peter K. Allen, "Computing swept volmues", *Journal of Visualization and Computer Animation*, 11, 1997

[7] J.R. Rossignac, J.J. Kim, S.C. Song, K.C. Suh, C.B. Joung, "Boundary of the volume swept by a free-form solid in screw motion", *Computer-Aided Design* 39, 9, 745–755, 2006

[8] J.J. Kim, C.B. Jung, K.C. Seo, M.W. Kang, "Swept Volumes Generated by Polyhedral Objects Through Screw Motions", *Transactions of the Society of CAD/CAM Engineers* / v.7 no.4, pp.211-218, 2002

[9] Abdel-Malek, K., Blackmore, D., and Joy, K., "Swept volumes: foundations, perspectives and applications", *International Journal of Shape Modeling*, 2004.

[10] von Dziegielewski, A., Hemmer, M., and Shomer, E., "High Precision Conservative Surface Mesh Generation for Swept Volumes", *In Proc. of IEEE International Conference on Robotics and Automation*, 2012.

[11] M.H. Kyung, J.G. Kwak, J.J. Choi, "Robust GPU-based intersection algorithm for a large triangle set", *Journal of the Korea Computer Graphics Society*, vol. 17, no. 3, pp. 9-20, 2011.

[12] Fogel, E., and Halperin, D., CGAL Arrangement and Their Applications: A Step-by-Step Guide(Geometry and Computing), Springer, 2012.

<저 자 소 개>

이현호

- 2010년 아주대학교 정보및컴퓨터공학과 학사
- 2010년~현재 아주대학교 미디어학과 통합 박사과정
- 관심분야: 컴퓨터그래픽스, 기하모델링, GPGPU

경민호

- 1993년 포항공과대학교 전자계산학과 학사
- 1995년 포항공과대학교 전자계산학과 석사
- 2001년 Computer Science, Purdue University, Ph.D
- 2002년~현재 아주대학교 미디어학과 교수
- 관심분야: 컴퓨터그래픽스, 기하모델링, GPGPU