한국 CAD/CAM 학회 논문집
제 11권 제 1호 2006년 2월 pp. 20-26

학술논문

잔삭 가공을 위한 펜슬커브 생성

박상철*, 박태종**

Pencil Curve Computation for Clean-up Machining

Park, S. C.* and Park, T. J.**

ABSTRACT

This paper presents a procedure to compute pencil curves from a triangular mesh which is offset with the radius of a given ball-end mill. An offset triangular mesh has numerous self-intersections caused by an abundance of invalid triangles, which do not contribute to the valid CL-surface. Conceptually, we can obtain valid pencil curves by combining all intersections lying on the outer skin of the offset triangular mesh, i.e., the valid CL-surface. The underlying concept of the proposed algorithm is that visible intersections are always valid for pencil curves, because visible intersections lie on the outer skin of the offset model. To obtain the visibility of intersections efficiently, the proposed algorithm uses a graphics board, which performs hidden surface removal on up to a million polygons per second.

Key words : Pencil curve, Clean-up machining, Visibility

1. 서 론

일반적으로 기계가공은 황삭, 중삭, 정삭, 그리고 잔삭의 네 단계로 구분된다. 잔삭 가공은 날카로운 코너 부위 등에서 정삭 가공 후에 남은 살을 더 작은 공구로 제거하는 가공을 말한다. 잔삭 가공이 제거하는 살의 양은 작지만 최종 가공형상의 품질에 큰 영향을 미치고, 가공 후 사상 시간에도 큰 영향을 주므로 매우 중요한 단계이다. 이러한 잔삭 가공에서 발생할 수 있는 문제점은 가공 량의 변동에 따른 공구 떨림 현상이다. Fig. 1(a)와 Fig. 1(b)처럼 정삭 가공 후에 남은 살은 제품의 곡률에 따라 그 깊이의 변화가 매우 클 수 있다. 이러한 경우 잔삭 가공 시 가공 량의 급격한 변화에 따라 공구가 떨릴 수 있고 따라서 가공 품질에도 악 영향을 미칠 수 있는 것이다.

이러한 문제의 해결을 위해서 일반적으로 사용하는 방법이 펜슬 커브를 이용한 펜슬 가공의 수행이다. Fig. 1(c)에서 보듯이 펜슬커브를 계산하여 잔삭 가공 전에 펜슬가공을 해줌으로써 잔삭 단계에서의 전체적인 가공 량이 일정하도록 한다. Fig. 2에서 보여지듯이 펜슬 가공을 위한 첫 번째 단계는 주어진 ball-end mill의 반경을 고려하여 펜슬 커브를 계산해내는 것이다.

본 논문에서는 3축 기계에 대한 펜슬 커브를 추적하기 위해서 삼각망을 사용한다. 메쉬 모델을 표현하

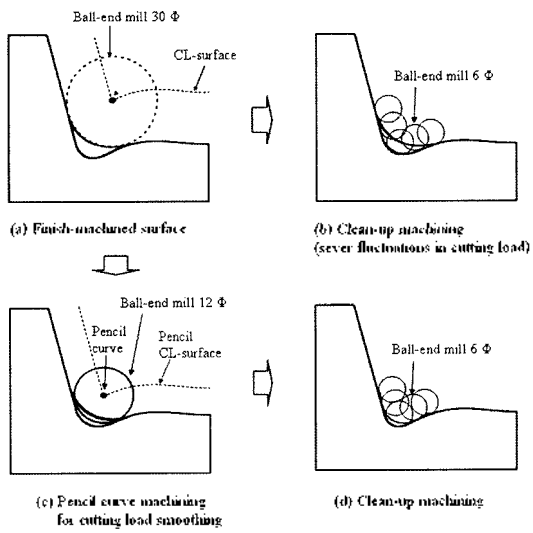


Fig. 1. Pencil curve machining & clean up machining.

*교신저자, 종신회원, 아주대학교 산업정보시스템공학부
**아주대학교 산업정보시스템공학부
- 논문투고일: 2005. 05. 22
- 심사완료일: 2005. 09. 07

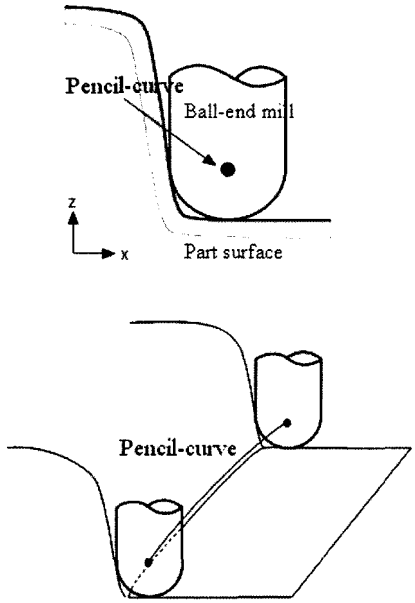


Fig. 2. Pencil curve tracing.

기 위해서 삼각망을 사용하는 것은 이미 널리 사용되고 있다. ball-end mill의 CL 표면은 삼각망을 옵셋함으로 얻을 수 있다. 하지만 삼각망의 옵셋 작업을 수행하게 되면 삼각형의 개수는 폭발적으로 늘어나게 된다. 삼각망의 옵셋은 공구의 형상에 따라 다르지만 볼 엔드밀인 경우, 삼각형은 옵셋 후 삼각형으로 에지는 옵셋 후 실린더로 그리고 꼭지점은 옵셋 후 구의 형상으로 대응된다[12]. 이때 옵셋 삼각망의 위상정보는 원래 삼각망 모델의 위상정보를 이용하여 재구성하게 된다. 일반적으로 옵셋 삼각망은 많은 비유효 삼각형들을 포함하고 있으며 이것이 자체 꼬임에 의해 수많은 교선들을 야기한다. 우리는 이러한 교선을 옵셋 망의 외피에 놓여 있는 펜슬 커브에 대해 유효한 교선과, 옵셋 메쉬로 둘러싸인 비유효한 교선 2가지로 분류할 수 있다. 유효한 교선은 가공에 사용할 ball-end mill로 목수의 접촉을 만드는 부품의 표면 위의 오목한 엣지와 관련되어 있으므로 이러한 교선을 합쳐서 펜슬 커브를 만들 수 있다.

본 논문의 목적은 비유효한 삼각형들을 포함하는 옵셋 삼각망에서 펜슬 커브를 계산하는 알고리즘을 제안하는 것이다. 추출한 펜슬 커브는 잔삭 가공 이전의 럭리프 가공에 사용될 수 있다. 제안된 접근 방식의 특성 중 하나는 교선의 가시성 정보를 얻기 위해서 그래픽스 하드웨어를 사용했다는 점이다. 이것과 관련하여, 삼각망에서 삼각형의 가시성 정보를 얻기 위해 3D 그래픽스 하드웨어를 사용하는 연구 결과가 있다. 하지만 이에 대한 연구 결과들은 삼각형의 가시성에 초점을 맞춘 반면 본 논문에서는 교선들의 가시성에 중점을 두도록 한다.

2. 펜슬 커브 계산에 대한 접근

본 논문에서 부품에 대한 가공 시 3축 기계를 이용한다는 가정을 한다. 이것은 부품 표면의 가공 면적이 Z축 양의 방향으로만 접근 가능함을 의미한다. 입력 값은 유효하지 않은 삼각형들을 포함하는 옵셋 삼각망이고 목표는 삼각망에서 펜슬 커브를 계산하는 것이다. 옵셋 삼각망의 외부 표면을 통해 유효한 CL면을 만들어 내며 펜슬 커브는 삼각망의 외부 표면에 놓인 교선들을 연결함으로 얻을 수 있다. 이를 위해 우리는 먼저 아래 3가지 과정으로 구성된 일반적인 해법을 생각할 수 있다. 1) 옵셋 삼각망의 모든 교선을 찾는다. 2) 옵셋 삼각망으로 각 교선을 비교해서 교선들의 가시적인 영역을 추출한다. 3) 교선들의 추출된 부분들을 결합해서 펜슬 커브를 계산한다. 이 3가지 과정에서 과정 2)는 몇 가지 심각한 계산상 문제점을 내포하고 있다. 옵셋 삼각망은 수 만개의 교선을 포함하고 있을 수 있다. 이렇게 많은 교선의 수에 대해서 수 만개 옵셋 삼각망과 비교를 통해서 모든 교선들의 가시적인 영역을 추출해 내는 것은 바람직하지 않다.

위에서 언급한 어려움을 피하기 위해서 교선의가시성에 대한 정보를 얻는데 3D그래픽스 하드웨어를 사용한다. Balasubramaniam과 Sarma *et al.*[13]은 5축 기계에서 충돌을 피하는 공구의 자세를 결정하기 위해 3D 그래픽스 하드웨어를 사용할 것을 제안했다. 그들은 주어진 장면을 표현하는데 최적화된 그래픽스 모드를 사용했다. 장면에 대한 계산(표현)을 수행한 후 그들은 가시적인 개체를 결정하기 위한 시도를 했다. 가시적 개체를 결정하기 위해 그들은 색을 통해서 각 개체에 대해 부호를 매긴다. 24비트 컬러 보드는 16,777,216색들을 허용한다. 픽셀 맵 상에 나타난 색들을 조회함으로 주어진 원래 장면에서 어떤 가시적 개체를 찾는 것이 가능하다. 이 접근 방식의 장점은 효율성으로 인해 주어진 기하 모델의 철저한 탐색을 가능하게 했다는 점이다. 그러나 접근 방식의 이산적 특성으로 인해 제한된 해상도가 잠재적인 약점이라 할 수 있다.

2^{24} 즉, 약 1600만개의 개수가 넘는 삼각형이 생성되면 3D 그래픽스 하드웨어를 사용하는 방법을 활용할 수 없다. 하지만 대부분의 실제 대형 금형 모델 일

지라도 삼각형의 개수가 천 만개가 넘는 경우는 극히 드물다고 할 수 있으므로 본 논문에서 제안된 방법은 실용적인 측면에서는 큰 문제가 없을 것이다.

본 논문에서는 그들의 개념을 옵셋 삼각망으로부터 펜슬 커브를 계산하는 문제에 적용한다. 그래픽스 모드를 사용할 것이며, 그래픽스 모드는 초당 수백만 다각형까지 숨겨진 면의 제거를 수행한다. 먼저 우리는 유효하지 않은 삼각형들을 포함하는 옵셋 삼각망을 갖는 것을 가정한다. 그리고 각 삼각형에 대해 색으로서 부호화한다. 삼각형의 수가 2^{24}보다 작다면 모든 삼각형들이 각기 다른 색을 가질 수 있다. 다음으로 픽셀 맵위에 옵셋 삼각망을 탑뷰로 그릴 수 있다. 픽셀 맵 상에 나타나는 색들을 조회해서 가시적인 삼각형을 결정하는 것이 가능하다. 또한 가시적인 교선을 만들어내는 삼각형들을 추출할 수 있다. Fig. 3에서 픽셀 맵은 3개의 삼각형으로 구성된다. 픽셀 맵에 대한 조사를 통해서 삼각형 A, B, C이 서로 인접해 있음을 쉽게 알 수 있다. A와 C는 옵셋 삼각망에서 서로 엣지를 공유하고 있고 반면에 A와 B는 엣지를 공유하고 있지 않다고 가정하자. 삼각형 A와 B(A와 C)는 가시적인 교선을 만들 수 있음(없음)을 의미한다. 만약 두 삼각형이 픽셀 맵 상에서 인접해 있고 3차원 공간에서 엣지를 공유하지 않는다면 이것은 가시적 교선을 만들 수 있는 잠재적인 후보가 된다.

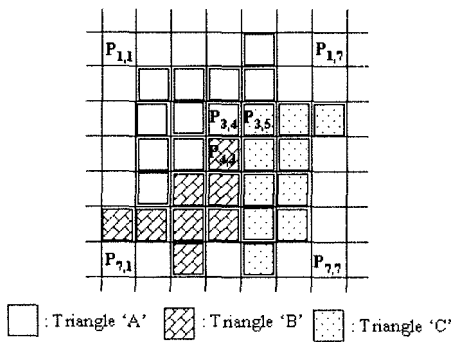
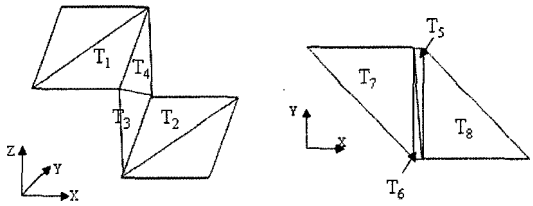


Fig. 3. Fining potential candidateds by scanning the pixel map.

가시적 후보는 두 가지 이유로 인해서 항상 가시적 교선임을 보장할 수 없다. 첫째는 추적하는 삼각형의 존재할 가능성이며 둘째는 픽셀 맵의 제한(1600*1200)된 해상도이다. Fig. 4(a)에서 보는 바와 같이 삼각형 T1과 T2는 서로 만나지 않지만 잠재적 후보를 만든다. T1과 T2는 3차원 공간에서 어떤 엣지를 공유하지 않지만 삼각형의 색은 T1, T2 사이의 수직하는 삼각형 T3, T4로 인해 픽셀 맵 상에서 인접하게 된다. Fig. 4(b)는 이 방식의 이산적 성격으로 인해 야기되는 또 다른 경우를 보이고 있다.

Fig. 4(b)에서 T5와 T6가 픽셀 맵상에서 하나의 픽셀을 차지할 만큼 충분히 크지 않다고 가정하자. 그렇다면 삼각형 T7, T8은 픽셀 맵상에서 서로 인접하기 때문에 잠재적 후보를 만들어 낸다. 하지만 실제로 서로 만나지 않는다. 만약 옵셋 삼각망이 수직의 삼각형들을 가지고 있지 않고 해상도가 충분히 높다면 펜슬 커브 추적은 오직 가시적인 삼각형과 그것들 사이의 교선들에 대한 고려를 통해서 쉽게 수행할 수 있다. 하지만 실질적으로 옵셋 삼각망은 수직의 삼각형들을 포함하고 있을 수 있으며 하나의 픽셀을 차지할 수 없는 작은 삼각형으로 구성되어 있을 수 있다. 그러나 이러한 요소들은 다음 섹션에서 소개할 제안된 알고리즘에서 중요하지 않다. 제안된 알고리즘은 가시적인 교선들을 펜슬 커브를 추적하는 것에 대한 씨드로서 사용하기 때문이다.



(a) Vertical triangles T_3, T_4 (b) Small triangles T_5, T_6

Fig. 4. Potential candidates having no intersections.

3. 펜슬 커브의 계산

본 논문에서 제안한 알고리즘은 아래 4가지 단계로 구성된다. 1) 주워진 옵셋 삼각망의 삼각형 간 모든 교선을 찾는다. 2) 만약 교선이 다른 교선과 교차한다면 그 교선은 쪼갠다. 3) 탑뷰로 옵셋 삼각망을 픽셀 맵 위에 그린 후 픽셀 맵을 사용하여 가시적인 교선들을 추출한다. 4) 가시적인 교선으로부터 시작해서 펜슬 커브를 추적한다.

3.1 주워진 옵셋 삼각망의 삼각형 간 모든 교선을 찾는다.

과정1)은 삼각망 교선을 구하는 문제이며 이것은 TTI(triangle-to-triangle intersection) 문제의 일부로 생각할 수 있다. 삼각망의 교선을 구하는 과정의 효율

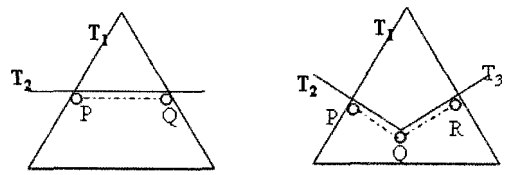
성을 높이기 위해서 기본적인 TTI 시험이 적용 될 수 있도록 시간과 위치를 제한한다. 전통적으로 TTI 시험의 수를 줄이기 위해서 공간 나눔 방법을 사용한다. 이 방법들의 개념은 간단하다. 삼각망이 차지하는 공간을 분할함으로써 분할된 공간상에서의 삼각형들은 동일 공간상의 삼각형 사이에서만 그 교선을 고려할 수 있다. 두 개의 교차하는 삼각형이 교선을 만드는데 우리는 과정1) 이 후에도 삼각형들과 교선들의 간의 관계가 유지됨을 가정한다.

3.2 교선이 다른 교선과 교차한다면 그 교선은 쪼갠다.

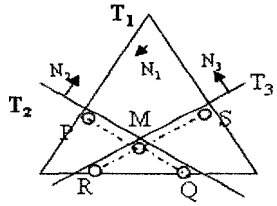
과정 1)에서 찾아낸 교선으로 2)의 과정에서는 교선간의 교차가 있다면 교선을 분할한다. 삼각형과 교선의 관계를 이용하여 이 과정은 쉽게 수행할 수 있다. 다수의 교선을 갖는 각 삼각형에 대해 삼각형 상의 교선의 교차점을 분할할 수 있다. 이 분할 과정의 목적은 펜슬 커브에 대해 부분적으로 유효한 교선들을 제거해 주는 것이다.

3.3 옵셋 삼각망을 픽셀 맵 위에 그린 후 픽셀 맵을 사용하여 가시적인 교선들을 추출한다.

과정 3)에서 우리는 픽셀 맵을 이용하여 가시적인 교선을 추출할 필요가 있다. 가시적인 교선에 대해 가시적인 삼각형과 잠재적인 후보를 찾는 것에 대한 개요는 Fig. 3에서 보인 예에서 충분히 설명되어 있다. 이미 과정 1)에서 모든 교선을 계산하였기 때문에 각 가시적 후보들이 교선을 갖는가에 대해서 즉각적으로 알 수 있다. T1과 T2 사이에 가시적 후보를 가지고 있다고 가정한다. 만약 T1과 T2가 서로 교차하지 않는다면 더 이상 고려할 필요가 없다. 그런데 T1과 T2가 교선 PQ를 형성한다면 2가지 경우로 분류할 수 있다. (1) PQ 전체가 가시적인 경우, Fig. 5(a)에서 보듯 다른 삼각형이 PQ를 분할하지 않는다. (2)부분적으로 PQ가 가시적인 경우, Fig. 5(b)에서 보듯 다른 삼각형이 PQ를 분할한다. 만약 다른 삼각형들이 PQ를 분할하지 않는다면 PQ는 두 가시적인 삼각형을 통해 만들어진 교선이기 때문에 PQ는 가시적이라고 할 수 있다. Fig. 5(b)는 T3가 PQ를 PM과 MQ두 부분으로 분할하는 것을 보이고 있다. 이 경우에는 연관된 삼각형의 법선 벡터에 대해 고려함으로써 가시적인 부분을 추출할 수 있다. Fig. 5(b)의 예에서 PM(MQ)가 분할하는 삼각형 T3의 위에(아래) 있기 때문에 PM(MQ)가 가시적(비가시적)이라고 설정할 수 있다. 과정3) 은 아래 2가지 이유로 인해 펜슬 커브를 정의하는데 유효한 모든 교선들을 추출할 수 없다. (1) 수직한 삼각형이 생성하는 수직의 교선은 펜슬 커브의 유효한 부분이 될 수 있으나 top view에서는 이러한 교선들이 가시적이지 않다. (2) 가시적이지만 작은 교선은 픽셀 맵의 제한 때문에 놓칠 수도 있다. 위와 같은 이유로 인해서 교선을 놓칠 수도 있지만 그것은 그렇게 중요하지 않다. 과정 3)에서 확인한 가시적인 교선들은 다음 과정의 씨드로 사용한다.



(a) Intersection PQ is not split.
➔ Intersection PQ is visible



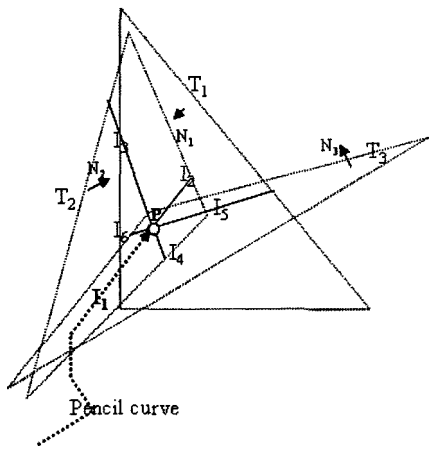
(b) Intersection PQ (between T_1 & T_2) is split by T_3
PM is visible, and MQ is not visible, because of T_3

Fig. 5. Extracting visible intersections.

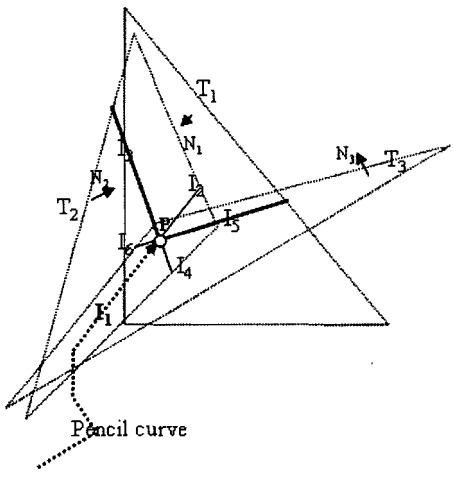
3.4 가시적인 교선으로부터 시작해서 펜슬 커브를 추적한다.

과정 4)는 가시적인 교선에서 시작하는 펜슬 커브를 추적한다. 과정 3)에서 확인한 가시적 교선들 사이에서 씨드 교선을 선택한다고 가정한다. 선택한 씨드 교선으로 펜슬 커브가 확대됨에 따라 증대되는 매개 펜슬 커브를 만들 수 있다. 매개 펜슬 커브를 확대하기 위해서 과정 2)의 모든 교선들 사이에서 중간 매개 펜슬 커브의 양 끝점과 만나는 교선들을 확인할 필요가 있다. 만약 만나는 교선이 없다면 펜슬 커브의 확대를 중단한다. 만약 만난다면 2가지 경우가 있다. (1) 하나의 교선, 그리고 (2) 다수의 교선이 있다. 하나의 교선이 만나는 경우 단순히 그 교선을 펜슬 커브에 더하면 된다. 다수의 교선이 만나는 경우의 예는 Fig. 6에서 보이고 있다. 이 경우 만나는 교선 사이에서 펜슬 커브에 대해 유효한 교선을 정할 필요가 있다. Fig. 6은 3개 삼각형과 6개의 교선으로 구성되며 삼각형은 각 T1, T2, T3이고 교선은 각 I1, I2(T2, T3의

교선), I3, I4(T1, T2의 교선), I5, I6(T1, T3의 교선)이다. Fig. 6에서와 같이 현재 펜슬 커브의 끝 점을 P(I1의 경우 이미 펜슬 커브에 포함되어 있다.)라고 가정한다. 5개의 교선 중에서 유효한 교선을 찾아내기 위해서 관련된 삼각형들과 삼각형들의 법선 벡터를 이용할 필요가 있다. 관련된 삼각형들의 법선 벡터에 기초해서 각 삼각형 T1, T2, T3에 대해 각 교선 I2, I4, I6가 뒤에 있음으로 I2, I4, I6는 유효하지 않다고 할 수 있다. 결과적으로 2개의 유효한 교선 I3, I5가 남으며 이 교선은 펜슬 커브를 추적하는데 새로운 씨드로 사용할 수 있다. 씨드(과정 3)에서 발견한 유효한 교선들)의 초기 집합이 펜슬 커브에 대해 모든 유



(a) Contacting intersections with P = {I2, I3, I4, I5, I6}



(b) Valid intersections = {I3, I5}

Fig. 6. Tracing pencil curves.

효한 교선을 포함하고 있지는 않지만 과정 4)는 매개 펜슬 커브를 확장함으로 모든 유효한 교선들을 발견한다.

과정 4)에 대한 형식적 기술은 아래와 같으며 이 과정은 펜슬 커브에 대한 반복적 확대하는 반복적인 함수를 사용하고 있다.

과정 4: 펜슬 커브의 추적

// 입력

// Intn_S : 과정 2)의 모든 교선의 집합

// Vis_S : 과정 3)의 가시적인 교선들의 집합

// 출력:

// Pcv_S : 펜슬 커브들의 집합

// 주요 변수 :

// Seed_S : 펜슬 커브 추적 에 대한 씨드 교선들의 집합

// TmpPenCV : 확장을 위한 일시적인 펜슬 커브

1. 초기화 단계
   (1) Pcv_S를 비운다
   (2) Seed_S = Vis_S
2. Repeat {
   (1) Seg = Seed_S로부터 교선을 잡고 Seed_S에서 그 교선을 제거한다.
   (2) 만약 (Seg = = NULL)
   Return Pcv_S (과정 4)의 끝)
   (3) Seg를 TmpPenCV에 더한다.
   (4) **Recursive-Expansion** (Intn_S, Seed_S, TmpPenCV, true) // 앞 쪽에 대한 확장
   (5) **Recursive-Expansion** (Intn_S, Seed_S, TmpPenCV, false) // 뒷 쪽에 대한 확장
   (6) Pcv_S에 TmpPenCV를 더한다.
   (7) TmpPenCV를 비운다.
   }

Recursive-Expansion (Intn_S, Seed_S, TmpPenCV, flag)

// Intn_S : 과정 2)의 모든 교선들의 집합

// Seed_S : seed 교선의 집합

// TmpPenCV : 확장될 펜슬 커브

// Flag : true or false, 만약 true(falsc)면 TmpPenCV를 앞쪽(뒤쪽)으로 확장한다.

1. 만약 (Flag = = true)
   Contact_S = TmpPenCV의 끝 점에 대해 만나는 교선을 찾는다
   그렇지 않으면

Contact_S = TmpPenCV의 시작점에 대해 만나는 교선을 찾는다.
2. Contact_S에서 비 유효한 교선들을 제거한다
3. 만약 (Contact_S의 수 = = 1){
(1) Seed_S = Seed_S – Contact_S
(2) TmpPenCV에 Contact_S의 원소를 더한다.
(3) **Recursive-Expansion** (Intn_S, Seed_S, TmpPenCV, flag)
}
4 Else { // Contact_S의 수 = = 0 또는 >1
Seed_S = Seed_S + Contact_S
Return
}

제안한 알고리즘은 많은 예를 통해서 실험했다. Fig. 7는 두 옵셋 삼각망과 각기 다른 펜슬 커브를 보이고 있다. 알고리즘은 C++언어를 사용했으며 Window XP 운영체제 1GB 메모리를 가진 펜티엄 4 프로세서 퍼스널 컴퓨터 환경에서 구현되었다. 수행 능력에 대한 시험을 위해서 Fig. 7에서 보인 예에 대한 실행 시간을 수집했고 결과는 Table 1에서 보이고 있다. Table 1에서 실행 시간은 많은 실질적 요소들에 영향을 받지만 본 논문에서 제안한 접근 방식의 실행이 합리적인 시간 내에 완료될 수 있음을 보이기에는 충분하다.

Table 1. Execution times for examples shown in Fig. 7

	Example A	Example B
삼각형 개수	13,880	21,414
모든 교선의 개수	4274	7655
유효한 교선의 개수	929	2744
실행시간	21s	37s

4. 결 론

펜슬 곡선은 물론 황삭 및 정삭 단계에서도 선택적으로 펜슬 커브를 이용한 펜슬 커브를 사용할 수 있으나 잔삭 가공의 단계에서는 필수적으로 사용되는 것으로 알려져 있다. 하지만 본 논문의 제안된 방법이 잔삭 단계에만 한정되 사용되는 것은 아니며 황삭 및 정삭을 위해서도 활용될 수 있다.

본 논문에서는 주워진 볼엔드밀의 반지름 만큼 옵셋한 삼각망으로부터 펜슬 커브를 계산하는 과정을 제안했다. 옵셋 삼각망은 내부의 수많은 비유효 삼각형들의 교차로 인해 수많은 교선들이 생긴다. 개념적으로 옵셋 삼각망의 외피에 놓인 모든 교선들을 결합해서 우리는 유효한 펜슬 커브를 얻을 수 있다. 제안된 알고리즘의 기본적인 개념은 가시적 교선들은 옵셋 모델의 외피에 놓여 있기 때문에 가시적 교선들은 항상 펜슬 커브에 대해서 유효하다. 교선의 가시성에

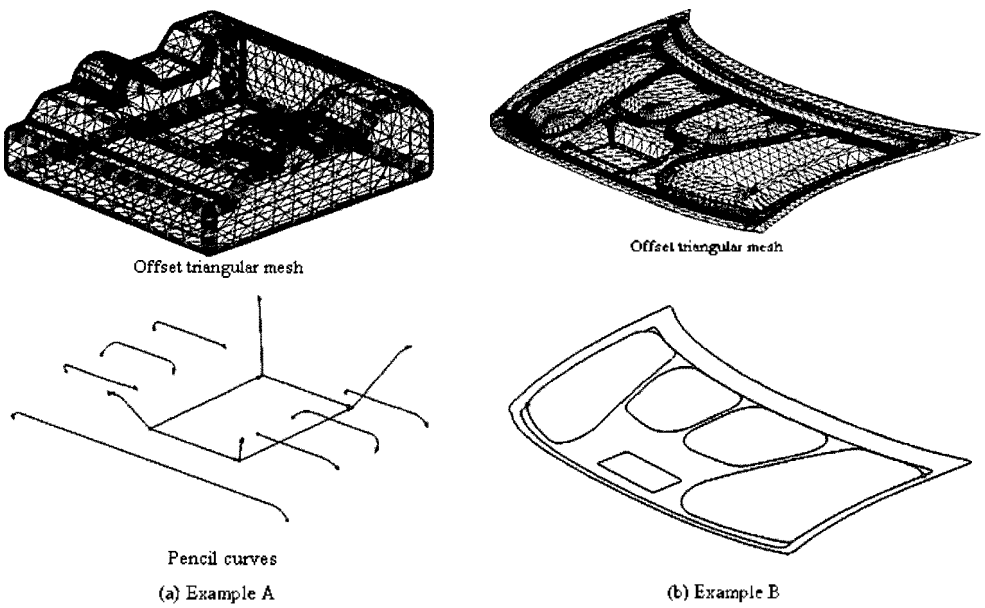
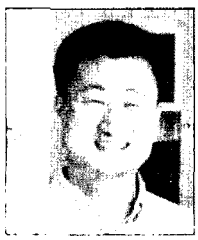


Fig. 7. Examples of pencil curve computation.

대한 정보를 효율적으로 얻기 위해서 제안된 알고리즘은 그래픽스 보드를 사용하였으며, 그래픽스 보드는 초당 수백만의 다각형까지 보이지 않는 표면의 제거 작업을 수행한다. 옵셋 삼각망에서의 삼각형의 개수가 2^{24}보다 작으면 각 삼각형에 서로 다른 색을 할당할 수 있으며 그것들을 그래픽스 보드에 그릴 수 있다. 픽셀 맵상에 나타난 색들을 조회함으로 가시적인 삼각형과 잠재적으로 가시적 교선들을 만들어 낼 수 있는 후보들을 찾아낼 수 있다. 다음으로 잠재적 후보에 의해 만들어진 교선으로부터 가시적 교선들을 얻을 수 있다. 우리는 잠재적인 후보들에서 만들어진 교선으로부터 가시적 교선들을 추출해 낼 수 있다. 가시적 교선들은 팬슬 커브에 대한 모든 교선들을 포함하고 있지 않을 수도 있기 때문에 가시적 교선에서 시작하여 팬슬 커브를 추적할 필요가 있다.

참고문헌

1. Choi, B. K. and Jerard, R. B. "Sculptured Surface Machining - theory and Applications", Kluwer Academic Publishers, 1998.
2. Balasubramaniam, M., Laxmiprasad, P., Sarma, S. and Shaikh, Z., "Generating 5-axis Paths Directly from a Tessellated Representation", *Computer-Aided Design*, Vol. 32, pp. 261-277, 2000.
3. Balasubramaniam, M., Sarma, S. and Marciniak, K., "Collision-free Finishing Toolpaths from Visibility Data", *Computer-Aided Design*, Vol. 35, pp. 359-374, 2003.
4. Ren, Y., Yau, H. T. and Lee, Y. S., "Clean-up Tool Path Generation by Contraction Tool Method for Machining Complex Polyhedral Models", *Comput. Ind.*, Vol. 54, pp. 17-33, 2004.
5. Zhu, W. and Lee, Y. S., "Five-axis Pencil-cut Planning anD Virtual Prototyping with 5-DOF Haptic Interface", *Computer-Aided Design*, Vol. 36, pp. 1295-1307, 2004.
6. Park, S. C., "Triangular Mesh Intersection", *Visual Computer*, Vol. 20, No. 7, pp. 448-456, 2004.
7. Park, S. C., "Sculptured Surface Machining Using Triangular Mesh Slicing", *Computer-Aided Design*, Vol. 36, No. 3, pp. 279-288, 2004.
8. Klass, R. and Kuhn, B., "Fillet and Surface Intersections Defined by Rolling Balls", *Computer Aided Geometric Design*, Vol. 9, pp. 185-193, 1992.
9. Lee, Y. S., Ma, Y. and Jegadesh, G., "Rolling-ball Method and Contour Marching Approach to Identifying Critical Regions for Complex Surface Machining", *Comput. Ind.*, Vol. 41, No. 2, pp. 163-180, 2000.
10. Xiuzhi, Q. and Brent, S., "A 3D Surface Offset Method for STL-format Models", *Rapid Prototyping Journal*, Vol. 9, pp. 133-141, 2003.
11. Maekawa, T., "An Overview of Offset Curves and Surfaces", *Computer-Aided Design*, Vol. 31, pp. 165-173, 1999.
12. Jun, C. S., Kim, D. S. and Park, S., "A New Curve-based Approach to Polyhedral Machining", *Computer-Aided Design*, Vol. 34, No. 5, pp. 379-389, 2002.



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

박 태 종

2005년 아주대학교 산업공학과 학사
관심분야: CAD/CAM, Virtual Manufacturing System