한국정밀공학회지 제 26권 1호 pp. 63-70 January 2009 / 63
Journal of the Korean Society for Precision Engineering Vol. 26, No. 1, pp. 63-70

스윕라인 알고리즘을 이용한 공구경로의 생성

Tool-Path Generation using Sweep Line Algorithm

✉성길영[1], 장민호[2], 박상철[3]
✉Kil Young Seong[1], Min Ho Jang[2] and Sang Chul Park[3]

1 아주대학교 산업공학과 (Department of Industrial Engineering, Ajou Univ.)
2 단국대학교 기계공학과 (Department of Mechanical Engineering, Dankook Univ.)
3 아주대학교 산업공학과 (Department of Industrial Engineering, Ajou Univ.)
✉ Corresponding author: skyblue@ajou.ac.kr, Tel: 031-219-2429

Manuscript received: 2007.12.5 / Revised: 2008.3.27,10.1 / Accepted: 2008.10.29

Proposed in the paper is an algorithm to generate tool-path for sculptured surface machining. The proposed algorithm computes tool path by slicing offset triangular mesh, which is the CL-surface (Cutter Location surface). Since the offset triangular mesh includes invalid triangles and self-intersections, it is necessary to remove invalid portions. For the efficient removal of the invalid portions, we extended the sweep line algorithm. The extended sweep line algorithm removes invalid portions very efficiently, and it also considers various degeneracy cases including multiple intersections and overlaps. The proposed algorithm has been implemented and tested with various examples.

Key Words: Sculptured Surface Machining (자유곡면 형상 가공), Sweep Line Algorithm (스윕라인 알고리즘), Triangular Mesh (삼각망), Projection (투영)

1. 서론

자유곡면형상의 가공은 제조산업에서 다양한 제품을 가공하는데 사용되고 있다.[1] 주로, 대량으로 생산하는 제품, 예를 들어 가전제품의 외형이나 장난감 등의 대량 생산에 필요한 다이 또는 금형을 가공하는데 있어서 자유곡면 형상의 가공은 필수적이다. 가공의 품질을 결정하는 주요 요소 중 하나가 공구 경로를 생성하는 방법이며, 일반적으로 Cutter-Location(CL)-Points 를 계산하는 것이 공구 경로를 생성하는데 가장 주요한 방법으로 사용되고 있다.

공구 경로를 생성하는 방법에는 공구 경로가 생성되는 표면에 따라 두 가지로 나누어진다. 그 두 가지는 1) Cutter-Contact (CC) Surface 접근법과 2) Cutter-Location (CL) Surface 접근법이다.

CC-Surface 접근법은 먼저, 파트 표면으로부터 CC-points 들의 순서를 얻어내어 공구 경로를 생성한다. 이후 각각의 CC-points 를 CL-points 로 바꾼다. 반면에 CL-Surface 접근법은 CL-Surface 위에 공구 경로를 생성하는 방법이다. 일반적으로는 CL-Surface 접근법이 CC-Surface 접근법보다 cutting-load smoothing, tool path smoothness, 그리고 chip-load leveling 등의 고속가공에 더 적합하다고 알려져 있다.[1]

CL-Surface 방법은 먼저 가공물의 표면과 inverse cutter 의 Minkowski sum 에 의해 정의된 CL-Surface 를 계산한다. 즉, 가공물의 표면을 inverse cutter 로 sweeping 하여 CL-Surface 를 계산한다. 여기에서 inverse cutter 란 Fig. 1 에서 볼 수 있듯이

공구 경로를 생성하는 공구의 중심점은 공구의 끝 부분이 아닌 다소 안쪽에 위치하기 때문에 끝부분과 중심점 사이의 거리를 보상할 필요가 있다. 보상의 필요성을 배제하고 공구경로를 생성하기 위해 공구를 거꾸로 한 것을 inverse cutter 라 하며, 이것을 통해 CL-Surface 를 계산하게 된다. Minkowski sum 은 "유클리드 공간에 있는 집합 A, B 의 합은 A 와 B 의 모든 원소를 더한 것과 같다." 라는 것을 의미 하며 CL-Surface 는 Minkowski sum 에 의해 정의된다.

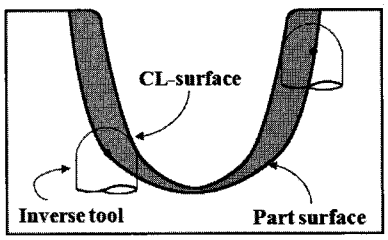
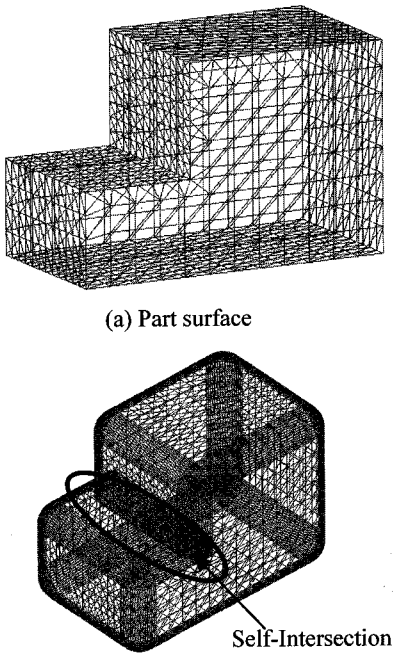


Fig. 1 Cutter location surface (CL-surface)

일반적으로 가공물의 표면은 매개변수 곡면의 집합(parametric surface set)으로 표현하지만, 매개변수 곡면(parametric surfaces)은 Minkowski sum 에서 변형되기 때문에 매개변수 곡면의 형태로 CL-Surface 를 구하는 것은 비효율적이다. 이러한 단점을 보완하고 CL-Surface 산출을 위해, Minkowski sum 에서 변형이 없는 Z-Map 이나 삼각망 등의 표현방법을 이용할 수 있다. 한편, Z-Map model 은 3 축 NC-가공 응용프로그램에서 널리 쓰이지만, 높은 수준의 정밀도를 위해서는 많은 계산 시간과 메모리를 요구하기 때문에 고속정밀가공에는 적합하지 않다.

위와 같은 특성을 고려하여 본 논문에서는 자유형상표면의 가공을 위해 삼각망을 이용하기로 한다. Ball-End Mill 의 경우에 CL-Surface 는 삼각망을 오프셋 하여 얻을 수 있다. 하지만, Fig. 2 와 같이 삼각망을 오프셋 하는 것은 유효한 CL-Surface 를 구하는데 불필요한 삼각형을 포함하고 있으므로 이것에 대한 고려가 필요하다.

Fig. 2(a)는 자동적으로 생성되는 형상과 Fig. 2(b)는 그것의 CL-Surface 를 보여주고 있다. 공구경로를 효율적으로 생성하기 위해서는 CL-Surface 에 포함된 불필요한 삼각형을 제거하는 것이 매우 중요하다. 일반적으로, 삼각망을 절단하여 얻어진 직선이나 호 등은 self-intersection 이나 gouging 을 유발하는 부분을 포함하고 있기 때문에 가공에 바로 사용할 수 없다. 이 문제를 해결하기 위해 두 가지 방법을 생각해 볼 수 있다. 첫 번째로 CL-Surface 를 절단하기 전에 불필요한 삼각형들을 제거하는 방법과, 두 번째로 CL-Surface 를 절단한 후, 그 면에서 불필요한 부분을 제거하는 방법이 있다. 본 논문에서는 두 번째 방법을 사용하기로 한다.



(a) Part surface

(b) CL-Surface

Fig. 2 Triangular mesh offsetting

자유곡면형상의 가공에서 가장 일반적인 가공 유형 중 하나는 regional milling 이다. 이것은 경계선에 의해 제한된 특정 영역에서의 가공을 말한다.[1,6,9] Regional milling 에서의 가공 작업 절차는 Fig. 3 과 같다. Regional milling 에서 공구 경로의 생성 절차는 크게 두 단계로 나누어진다. 먼저 2D 경로 요소(element)로 가공 범위를 채운 후, 2D 경로를 CL-Surface 에 투사한다. 2D 공구 경로 element 들은 선분 또는 경계곡선(boundary curve)의 연속적인 오프셋 커브, 또는 임의 형상의 곡선(shaped-curve) 같은 2D 기하형상을 가진 유형일 것이다.

본 논문의 목적은 자유곡면 형상에서 regional milling 을 위한 공구 경로의 생성 방법을 제안하는

것이다. 여기서 제안하는 절차는 자유곡면형상의 CL-Surface 를 표현하기 위해 삼각망을 이용하며 삼각망에 2D 기하 형상을 이용하여 투사, 절단하여 공구 경로를 계산한다. 다음에는 계산된 공구 경로의 계산을 위해 사용한 line projection algorithm 을 설명한 뒤 고려해 볼 수 있는 degeneracy 문제들을 해결하기 위한 방법을 제시한다. 그리고 이 알고리즘에 대한 수행 테스트를 기술하고 마지막으로 결론을 다루기로 한다.

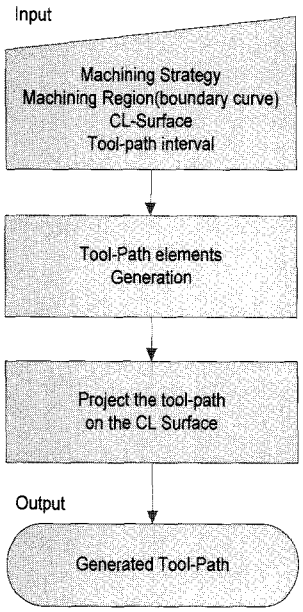


Fig. 3 Tool path generation of regional milling

2. CL-Surface 에서의 Line Projection

Projection 은 크게 두 가지 절차로 나눌 수 있다. 1) 라인을 포함 하는 평면으로 CL-Surface 를 수직으로 절단하여 선분들의 집합을 얻는다. 2) 절단하여 얻어진 선분의 집합에서 경로 생성의 계산에 불필요한 부분을 제거하여 공구 경로를 얻는다. 평면으로 삼각망을 절단하는 첫 번째 방법은 이미 최적화 되어 있으며, 본 논문에서는 Tomas 와 Prosolvia[4] 에 의해 개발 된 방법인 triangle-triangle intersection or triangle-plane intersection 을 사용하기로 한다.

절단하여 얻어진 선분의 집합에서 계산에 불필요한 부분을 제거하는 가장 직관적인 방법은 집합의 모든 선분의 교점을 찾고, 거기에서 불필요한 부분을 제거하는 것이다. 이 방법은 간단하며 타당성에도 문제가 없지만 효율적인 측면에서 볼 때, 바람직하다고 할 수 없다. 그 이유는 첫 번째로, 공구 경로 생성을 위해 알고리즘의 계산을 수 천 만번씩 반복 해야 하는데 여기서 모든 교점을 찾아 계산하는 것은 불필요 하기 때문이다. 예를 들어, Fig. 4 는 선분의 집합과 계산이 필요한 12 개의 교점을 나타내고 있다. 하지만, 공구 경로를 생성하기 위해서는 단지 5 개의 교점만 계산하면 되므로 나머지 7 개의 교점을 계산할 필요가 없다. 두 번째로, 많은 교점들은 단지 선분과 선분을 연결하는 점이므로 다른 일반적인 교점들보다는 비교적 쉽게 계산 할 수 있다. 마지막으로, 이 문제는 Fig. 5 에서 볼 수 있듯이, 다양한 종류의 degeneracy 문제를 포함하고 있다. 그러므로 제안하는 알고리즘은 이러한 문제를 해결할 수 있어야 한다.

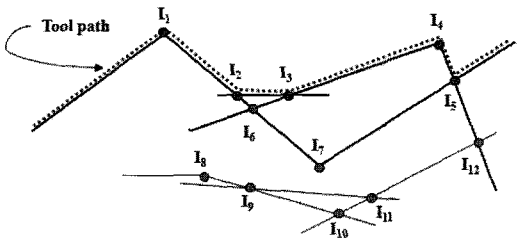
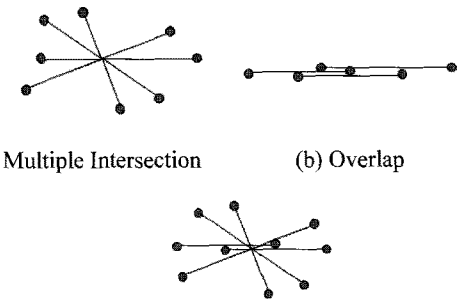


Fig. 4 An example of projection and Intersection Point



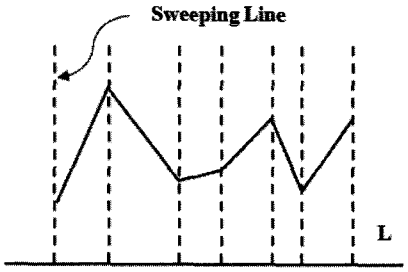
(a) Multiple Intersection (b) Overlap

(c) Multiple intersection and Overlap

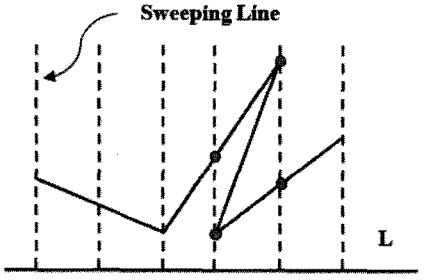
Fig. 5 Degeneracy cases of a polygonal chain intersection problem

본 논문에서 제시하는 알고리즘은 polygonal-chain-intersection algorithm (PCI-algorithm)[7]을 수정하여 개발하였다. PCI-algorithm 은 plane sweeping paradigm[2,3,13] 을 바탕으로 monotone chain[14] 의 특성을 이용하여 효율적으로 공구경로를 계산하는 알

고리즘이다. 여기서 새롭게 개발 된 알고리즘과 PCI algorithm 의 가장 큰 차이는 공구경로를 생성하기 위해 모든 교점을 계산할 필요가 없다는 것이다. 위에서 언급한 monotone chain 의 정의는 다음과 같다. 직선 L 축에 수직인 Ls 와 Chain C 가 최대 1 개의 교점을 가질 때 Chain C 를 L 에 대하여 monotone 이라고 한다. 여기서 직선 L 은 sweeping direction, 직선 Ls 를 sweeping line 이라고 한다. (Fig. 6 참고)



(a) monotone chain with respect to line L



(b) non-monotone chain with respect to line L

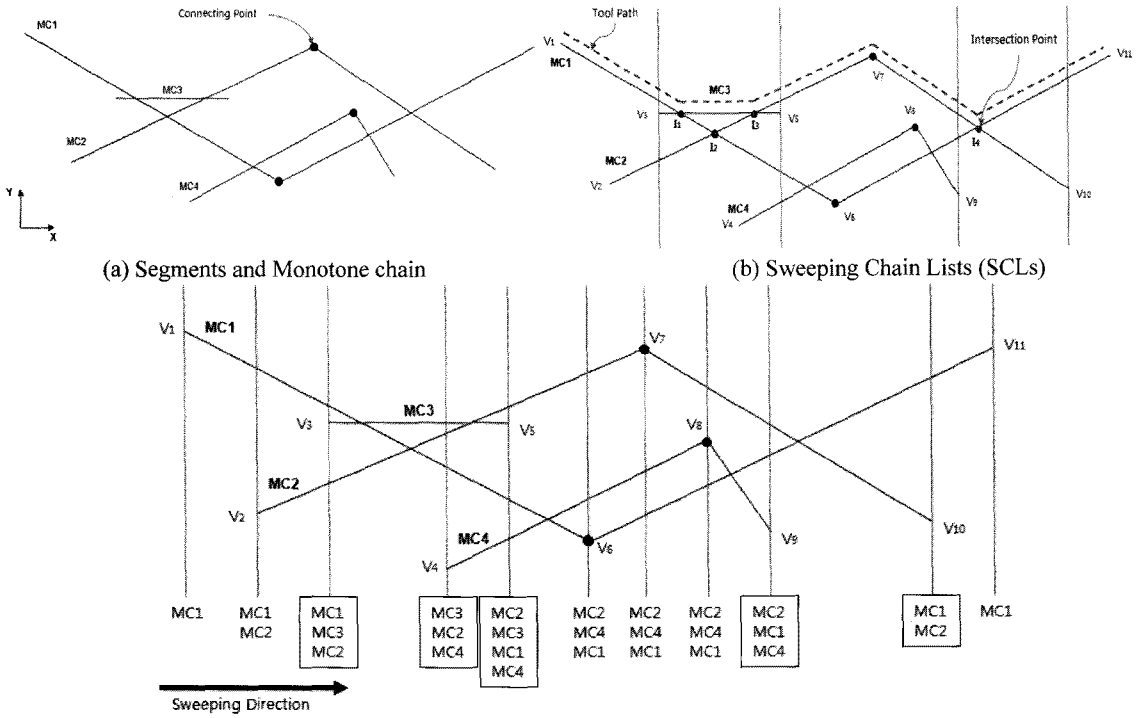
Fig. 6 Monotone chain and non-monotone chain

제안하려는 알고리즘을 Fig. 7 의 예를 통해 설명한다. 먼저, 입력 데이터는 Fig. 7(a) 와 같이 보편적으로 XY 평면상의 선분들의 집합으로 가정할 수 있다. monotone chain 을 파악하기 위해서는 각 선분들의 끝 단을 연결하는 점을 찾아내는 것이 필요한데, 이것은 각 선분들을 정렬하는 과정에서 Sweeping line 과 직교하는 선분들의 끝 단의 (x, y)좌표가 동일한 점을 찾음으로써 쉽게 구할 수 있다. Fig. 7(a) 는 X 축을 기준으로 하는 monotone chain 을 보여주고 있다. monotone chain 은 self-intersection 이 없으며 x 값에 의해 순차로 정렬된다. 이러한 특징을 통해 Sweep Line Method 를 효과적으로 사용할 수 있다. Fig. 7(b) 에서 각각의 꼭지점마다 sweep line 을 정의할 수가 있다. Sweeping Chain list(SCL)란 sweep line 에 의해 교차하는 monotone chain 들의 순서이다. 즉, 하나의 SCL 에서 monotone chain 의 순서는 monotone chain 들과 sweeping line 의 교점의 y 값에 의해 결정된다. 예를 들면, V_2의 SCL 에서 MC1 은 MC2 보다 더 큰 Y 값을 가지기 때문에 MC1 이 MC2 보다 우선순위를 갖는다.

제안하는 알고리즘의 요점은 공구 경로의 생성에 관계없는 교점들의 계산을 피하고자 SCL 을 응용하는 것이다. 만약 연속하는 두 개의 SCL 의 첫 번째 monotone chain 이 같다면, 두 sweep line 사이의 어떠한 교점도 계산할 필요가 없다. 예를 들어 V_1, V_2, V_3 구간과 V_5, V_6, V_7, V_8, V_9 구간은 첫 번째 monotone chain 이 같기 때문에 V_1~ V_3과 V_5~ V_9 구간 범위에서는 어떠한 교점도 계산할 필요가 없으며, 여기서 각각 MC1 과 MC2 가 유효한 공구 경로가 된다.

교점들의 계산이 필요할 때는 연속적인 SCL 들의 첫 번째 monotone chain 이 같지 않을 때이다. 먼저, Fig. 7(b) 에서 V_9 ~ V_{11}에서는 monotone chain 이 바뀌는 것을 볼 수 있다. 이러한 경우에는 Fig. 7(c) 에서 볼 수 있듯이, 교점 I_4 를 계산해야 한다. 더 복잡한 경우로, Fig. 7(c) 의 V_3~ V_5의 sweep line 을 볼 때, 교점 I_1, I_2, I_3은 반드시 계산이 필요하다. V_3에서 첫 번째 monotone chain 은 V_5에서 세 번째 monotone chain 이 된다. 이런 경우에는 범위내의 모든 교점을 계산해야 한다. 이 경우, V_3과 V_5내의 모든 교점(I_1, I_2, I_3)의 계산을 통해서, 유효한 공구 경로를 생성할 수 있다. 먼저 I_1~I_2 에서 첫 번째 monotone chain 은 MC3 가 되며, I_2~I_3 에서도 MC3 가 되므로 I_1~I_3 까지 유효한 공구 경로는 MC3 임을 알 수 있다. I_3 이후에는 MC2 가 유효한 공구 경로가 된다.

제안하는 알고리즘의 장점을 정리하면 다음과 같다. 첫 번째로, 공구경로를 생성하기 위해 불필요한 교점들은 계산하지 않으므로 계산양의 측면에서 효율적이다. 두 번째로 이 알고리즘은 Fig. 5 에서 볼 수 있는 degeneracy 문제들을 해결 할 수 있다. 이 알고리즘은 교점을 찾는 것이 목적이 아니라 유효한 공구 경로를 찾는 것이므로 위의 해결 방법이 타당하다고 할 수 있다. 한편, 이 알고리즘은 공구경로 생성을 위해 line projection algorithm 이 수천 번 반복 되며, 위에서 언급한 overlap 또는 다중교점을 가진 CL-Surface 를 처리해야 하므로 위와 같은 degeneracy 문제를 3 장에



(a) Segments and Monotone chain

(b) Sweeping Chain Lists (SCLs)

(c) Intersection point and Obtained tool-path

Fig. 7 Monotone chain and Sweep chain list and Computing intersection points

서 다룬다.

3. Degeneracy Program

본 논문에서 제안하는 알고리즘은 CL-surface 를 절단하여 공구경로를 생성하는 것이다. 그 절차는 CL-surface 를 절단하여 얻은 선분들의 집합을 통해 공구 경로를 생성한다. 그리고 전체 형상에 대한 공구 경로를 생성하기 위해 이 알고리즘은 수 없이 반복된다. 하지만 CL-surface 를 절단하여 선분들의 집합을 얻는 과정에서 Fig. 5 와 같이 선분들의 다중 교차(multiple intersection), overlap 또는 두 가지가 결합된 형태의 문제가 발생할 수 있다. 이 같은 문제들은 최종적으로 생성되는 공구경로에 더 큰 문제를 야기할 수 있으므로 중요하게 해결해야 할 문제이다.

Degeneracy 문제의 해결하기에 앞서, Fig. 8 과 같이 선분이 sweep line 과 평행한 경우는 선분들을 얻는 과정에서 발생한 에러 요소로서 이러한 요소는 선분의 양단의 y 좌표가 같으므로 쉽게 발견할 수 있으므로 사전에 제거한다.

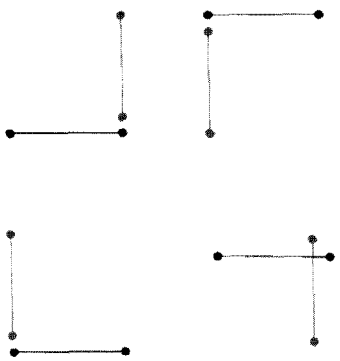
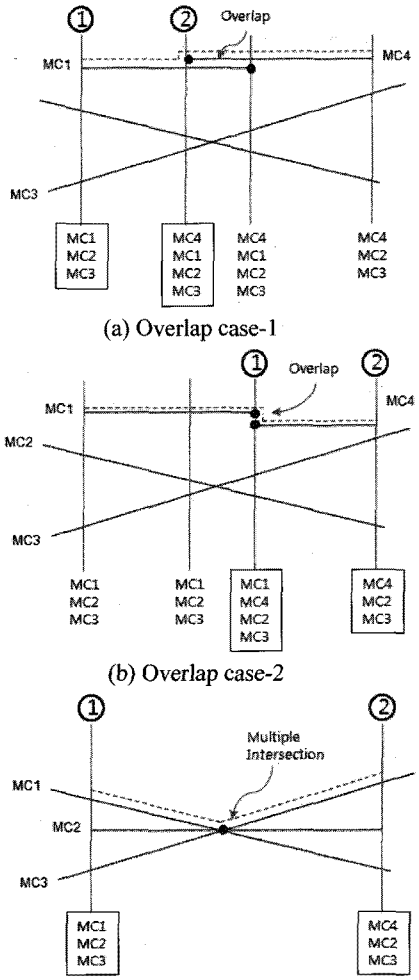


Fig. 8 Error cases of a polygonal chain intersection problem

Fig. 5 에서 볼 수 있는 degeneracy 문제를 해결하기 위한 방법을 Fig. 9 를 통해 설명한다. 선분들 간 overlap 이 있는 경우는 monotone chain 에서 우선순위를 갖는 선분들 사이에 선분으로 연결되지 않는 틈이 존재한다. Fig. 9(a)의 ②번 sweeping line 을 보면 MC1 과 MC4 사이에 폐쇄되지 않고 틈으로 인해 개방 된 것을 확인할 수 있다. 공구 경로는 monotone chain 에 의해 생성되므로 이러한 틈을

line-projection 알고리즘에서 고려해 주어야 한다. 이 문제의 해결 방법은 다음과 같다. projection 알고리즘에서 각 교점에서 monotone chain 의 우선순위를 갖는 선분이 바뀔 때 유효한 경로를 생성하기 위해 교점을 모두 계산하게 된다. 하지만 선분의 중복 문제를 해결하기 위해서는 monotone chain 에서 첫 번째 선분이 바뀌었을 때 그 구간에서의 교점들의 계산에 앞서 overlap 을 먼저 확인 하여야 한다. Fig. 9(a)에서 monotone chain 의 우선순위가 MC1 에서 MC2 로, Fig. 9(b) 에서는 MC1 에서 MC4 로 바뀐 것을 확인할 수 있다. 이 구간 내(①~②)에서는 monotone chain 에서 우선순위를 갖는 두 선분 사이에 교점이 없으므로 교점의 유무를 확인하여 overlap 을 확인하면 된다. overlap 이 존재하는 것이 확인되면 Fig. 9(a)에서는 그 구간에서의 overlap 을 가진 두 선분과 직교하고 있는 sweeping line ②, Fig. 9(b)에서는 sweeping line ①을 선택하여 오버랩을 가진 두 선분과의 교점을 계산하여 sweeping line 과 평행한 두 점 사이의 선분 즉, sweeping line 과 평행인 선분을 연결하여 유효한 공구경로를 생성할 수 있다. 그리고 선분들의 다중 교차에 대한 문제는 Fig. 9(c)에서 볼 수 있듯이 monotone chain 을 이용하여 공구 경로를 생성하는 알고리즘에서 자동적으로 해결이 된다. 예를 들면, Fig. 9(c)에서 ①번과 ②번 sweep line 에서 monotone chain 의 순서는 다르므로 다중교점을 계산하여 공구 경로를 생성한다. 여기서 교점을 이루는 선분인 MC2 는 공구경로의 생성에 영향을 미치지 않으므로 고려할 필요가 없기 때문에 제안한 알고리즘에서 자연스럽게 해결된다. 마지막으로 선분들의 overlap 과 다중 교점 문제의 해결에 대한 것은 다중 교점 문제는 monotone chain 을 이용하여 공구경로를 생성하는 알고리즘에서 자동적으로 생성이 되므로 이것은 overlap 의 경우에서 문제를 해결하는 방법과 동일하다.

이러한 degeneracy 처리를 통해서 공구경로를 생성하기 위한 더욱 강건한 알고리즘을 만들 수 있다. 여기서 제안하는 알고리즘은 직관적이므로 구현에 문제가 없기 때문에 본 논문에서는 생략하기로 한다. 단, 컴퓨터 알고리즘인 이상 floating point error 를 피하기 어렵다. 실제 알고리즘의 구현은 0.0001 이상 가까운 점이면 동일한 점으로 취급을 하므로 degeneracy 가 실제로 발생한 것이든, floating point error 로 발생한 것이든 구별하지 않는다. 물론 이러한 가정으로 인하여 형상왜곡이 발생할 수 있으나 기계가공 정밀도 0.01 을 고려하면 허용 가능하다고 판단하였다.



(a) Overlap case-1

(b) Overlap case-2

(c) Multiple Intersection

Fig. 9 Degeneracy Problem

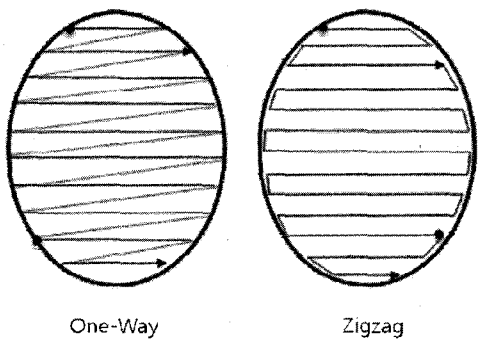
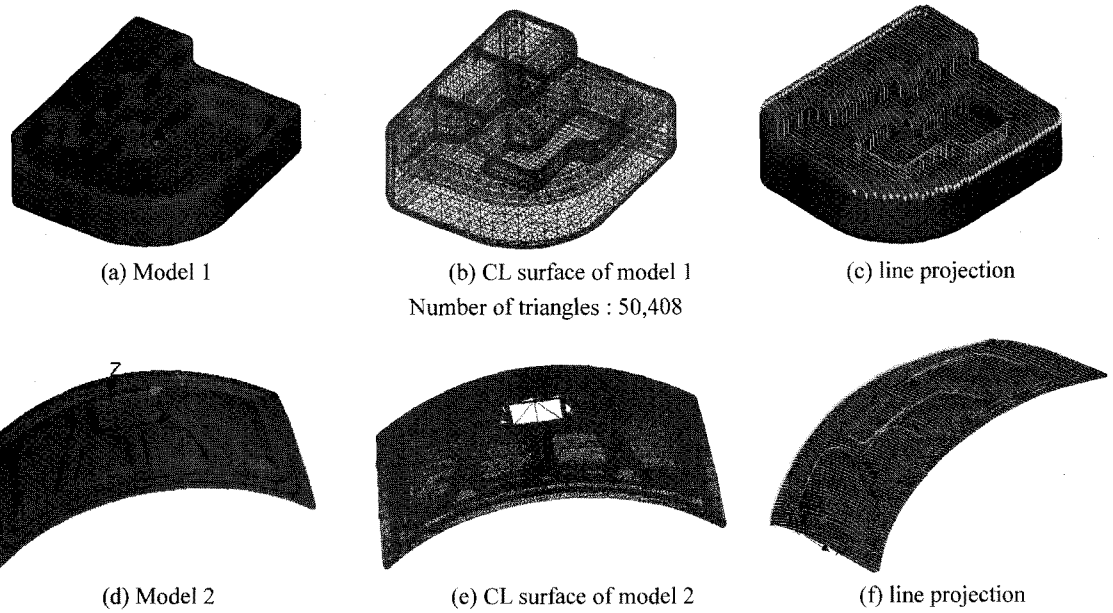


Fig. 10 Direction parallel type tool-path



(a) Model 1 (b) CL surface of model 1 (c) line projection

Number of triangles : 50,408

(d) Model 2 (e) CL surface of model 2 (f) line projection

Number of triangles : 142,487

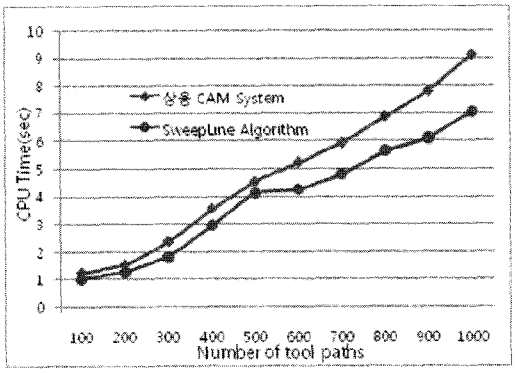
Fig. 11 Two examples for performance test and tool path

4. 알고리즘의 응용

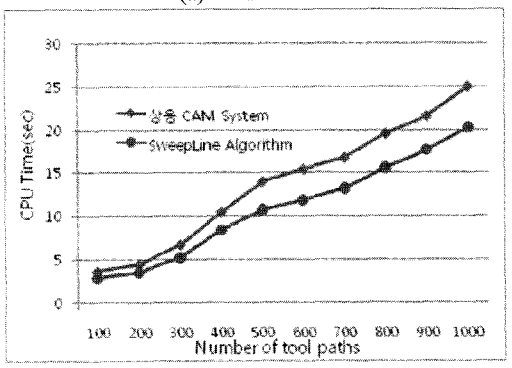
우리가 제안하는 알고리즘은 regional milling 에서 공구 경로를 생성하는데 유용하다. Regional milling 에는 3 가지 종류의 공구 경로에 대한 연구들이 있는데 각각 Contour-Parallel type, Direction-Parallel type (DP type),[8] Space Filling Curve type 이며 이 알고리즘은 DP 유형의 공구경로에 이용될 수 있다.

DP 유형의 기본적인 요점은 특정 가공 방향으로 평행한 선분을 따라 밀링하는 것이며, DP 유형의 밀링에는 단 방향과 지그재그로 밀링하는 방법의 두 가지가 있다.(Fig. 10 참고) 여기서 제안하는 line projection 알고리즘은 위에서 언급한 DP type 의 공구 경로 생성에 유용하다. 또한 공구 경로를 생성하는 과정에서 발생할 수 있는 degeneracy 문제를 모두 고려하여 즉, line projection 알고리즘에서 발생하는 선분들간의 overlap 이나 다중교차의 문제를 해결할 수 있다.

우리가 제안한 알고리즘은 Fig. 11(a), (d)의 예제를 가지고 수행과 테스트를 반복 하였으며, 모델의 삼각형 수는 각각 (50,408), (142,487)개이다. Fig. 11(c), (f)는 CL-Surface 위에 line 을 투사하여 공구 경로를 생성한 것을 보여주고 있다.



(a) Model 1



(b) Model 2

Fig. 12 Plotting of executing times

여기서 제안하는 알고리즘은 pentium4, 1GB 메모리, XP 운영체제를 가진 PC 에서 C++언어로 작성되었다. 그리고 수행능력에 대한 테스트를 Fig. 12 의 그래프로 표시하였다. 그래프를 통해 기존의 상용 CAM 시스템에 비해 계산시간이 약 80% 정도 단축되었으며 공구경로의 개수와 계산시간이 선형관계에 있는 것을 확인할 수 있다. 마지막으로 degeneracy 문제를 체계적으로 분류하여 처리함으로써 더욱 강건한 알고리즘을 제안하였다.

5. 결론

본 논문에서 regional milling 에서 자유 형상 표면 가공을 위한 공구 경로 생성 알고리즘를 설명하였다. 이 알고리즘은 자유형상 표면의 CL-surface 를 표현하기 위해 삼각망을 도입하였으며 2D 기하 요소를 삼각망에 투사함으로써 공구경로를 계산한다. 이 알고리즘은 연구 되고 있는 여러 공구경로 중에서도 DP type 의 공구경로 생성에 유용하다. 즉, regional milling 에서 공구 경로 생성을 위해 line-projection 알고리즘과 degeneracy 처리를 위한 알고리즘을 제안하였다. 구현 상의 부딪히는 가장 어려운 문제는 degeneracy 처리에 대한 문제이며, 본 논문에서는 이러한 문제를 일으키는 경우를 체계적으로 분류하여 처리할 수 있음을 보여주었다. 단, degeneracy 라는 것은 기본적으로 open boundary problem 으로 본 논문에서 밝힌 degeneracy 를 포함했다고 할 수는 없으며, 경험적으로 degeneracy 를 찾아내고 처리할 수 있도록 수정, 보완하였다.

line projection 알고리즘은 평면의 sweeping 방법[13]을 토대로 monotone chain 의 특성을 이용하여 효율적으로 동작한다. 이 알고리즘은 특징은 교점의 최소의 교점만을 계산하므로 매우 효과적이다. 또한 다양한 유형의 degeneracy 문제를 해결 할 수 있다. 이러한 절차는 regional milling 에서 공구경로를 생성하는데 적용하고 수행하였다.

본 논문은 DP type 의 공구경로에 대한 생성방법을 제안하였으며, 이 외에도 자유곡선에 대한 형상가공 방법이나 절차에는 Z-level machining, pencil machining 또는 CP type 등 다양한 유형이 있으며 이러한 가공 방법에 대해서도 효율적인 알고리즘을 개발하는 것이 필요할 것이다.

참고문헌

1. Choi, B. K. and Jerard, R. B., "Sculptured Surface Machining," Kluwer, 1998.
2. Shamos, M. I. and Hoey, D. J., "Geometric intersection problems," Seventeeth Annual IEEE Symposium on Foundations of Computer Science, pp. 208-215, 1976.
3. Preparata, F. P. and Shamos, M. I., "Computational geometry: An introduction," Springer, 1985.
4. Moller, T., "A fast triangle-triangle intersection test," Journal of Graphics Tools, Vol. 2, No. 2, pp. 25-30, 1997.
5. Jiménez, P., Thomas, F. and Torras, C., "3D collision detection: a survey," Computers & Graphics, Vol. 25, Issue 2, pp. 269-285, 2001.
6. Held, M., "On the Computational Geometry of Pocket Machining," Springer-Verlag, 1991.
7. Park, S. C. and Shin, H., "Polygonal Chain intersection," Computers & Graphics, Vol. 25, Issue 2, pp. 341-350, 2002.
8. Park, S. C. and Choi, B. K., "Tool-path planning for direction-parallel area milling," Computer Aided Design, Vol. 32, No. 1, pp. 17-25, 2000.
9. Park, S. C. and Choi, B. K., "Boundary extraction algorithm for cutting area detection," Computer Aided Design, Vol. 33, Issue 8, pp. 571-579, 2001.
10. Choi, B. K. and Park, S. C., "A pair-wise offset algorithm for 2D point-sequence curve," Computer Aided Design, Vol. 31, No. 12, pp. 735-745, 1999.
11. Park, S. C. and Choi, B. K., "Uncut-free pocketing tool-paths generation using pair-wise offset algorithm," Computer Aided Design, Vol. 33, Issue 10, pp. 739-746, 2001.
12. Park, S. C. and Chung, Y. C., "Offset tool-path linking for pocket machining," Computer Aided Design, Vol. 34, No. 4, pp. 299-308, 2002.
13. Bentley, J. L. and Ottmann, T. A., "Algorithms for reporting and counting geometric intersections," Computers IEEE Transactions on, Vol. 28, No. 9, pp. 643-647, 1979.
14. Chandru, V., Rajan, V. T. and Swaminathan, R., "Monotone pieces of chains," ORSA Journal On Computing, Vol. 4, No. 4, pp. 439-446, 1992.