한국 CAD/CAM 학회 논문집
제 10권 제 6호 2005년 12월 pp. 455-464

기술논문

오프셋 삼각형의 절단과 교선 추적에 의한 공구 경로 계산

정 연 찬*

Tool-path Computing by Slicing Offset Triangles and Tracing Intersections

Chung, Y. C.*

ABSTRACT

This paper discusses the methods of computing tool-paths for machining free-form surfaces on 3-axis CNC machines in die and mould making. In computational view this paper describes the characteristics and issues of the geometric information and the shape, which make computing tool-paths difficult. Important points that should be considered in devising a computing method are also discussed. A newly implemented method is explained and compared with an old method for a commercial CAM system. The implemented method is used in a commercial CAM system and the computing time for an example is presented.

Key words : Tool-path generation, Slicing offset triangles, NC milling

1. 서 론

CAM 시스템은 CAD 시스템에서 설계된 형상 정보를 수치제어(NC) 밀링 가공기의 공구 경로 정보(NC 프로그램)로 변환해 주는 역할을 한다. 컴퓨터 및 수치제어 기술의 비약적인 발전과 CAD 시스템의 폭 넓은 사용과 함께 CAM 시스템도 많은 발전이 있었다. 한국에서도 80년대 후반에 KAIST에서 개발한 KAPT를 시작으로 다양한 시스템이 개발되었으며 지금도 꾸준히 개발되고 있다. 2003년 CIMdata의 보고서에 의하면 세 개의 업체가 시장의 주도권을 장악하고 있는 CAD 시스템과 달리 CAM 시스템은 전체 시장의 75% 정도를 상위 20개 이상의 업체가 분할하고 있다[1]. CAM 시스템은 CAD 시스템에 비해 소프트웨어의 규모가 작아서 진입 장벽이 낮고, 사용 분야와 생산 기술의 수준과 방법에 따른 기능의 다양성이 요구되기 때문으로 분석된다.

최초의 공구 경로 자동 프로그래밍 도구로 여겨지는 APT[2]의 개발 이후 CAM 시스템의 지속적이고 비약적인 발전에도 불구하고 사용자의 입장에서는 CAM 시스템이 여전히 만족스럽지 못하다. 가전 사출 금형 혹은 자동차 차체 패널 프레스 금형 가공의 경우 가공 공수의 절반가량은 CAM 시스템에서 생성된 NC 프로그램을 사용하지 않고 있다. 기계 작업자가 파트 프로그램을 수동으로 작성하거나, 수작업 NC 가공으로 금형을 가공하고 있다. 제품의 형상을 결정하는 자유 곡면은 CAM 시스템에서 생성된 NC 프로그램에 의해 가공되지만 형상면 이외의 평면 작업, 구멍 작업, 윤곽 작업 등과 일부 황삭 작업은 기계 작업자가 직접 NC 명령어를 입력하거나 직접 기계를 제어해서 가공하고 있다. 또 CAM 시스템을 활용하는 경우에도 기계 가공 시간보다 더 많은 시간을 공구 경로 준비 혹은 계산에 투입하는 경우가 있다. 결국 CAM 시스템이 보다 더 다양한 용도의 공구 경로를 더 편리하고 빠르게 생성해야 할 필요가 있다.

Table 1은 서로 다른 두 시스템에서 유사한(엄밀히 동일한 공구 경로를 생성할 수는 없다) 공구 경로를 생성하는데 걸리는 계산 시간을 보여주고 있다. 약 6배의 차이를 보이고 있는데 Fig. 1과 같은 자동차 패널 금형의 가공을 위한 공구 경로 계산 시간이다. 그러나 Table 1의 A 시스템도 나름대로의 다른 경쟁력(공구 경로의 품질, 사용 편의성, 시장 선점 등)을 갖고 있으므로 공구 경로 계산 알고리즘을 개선해서 계

*교신저자, 종신회원, 서울산업대학교 금형설계학과
- 논문투고일: 2005. 02. 16
- 심사완료일: 2005. 08. 23

Table 1. Calculation time of two CAM systems (XEON 2.4GHz/Windows2000/2GB RAM)

	A System	B System
setup for calculation	23m	7m
Φ50 contour	4h 04m	3m
Φ50 contour(angle)	4h 04m	3m
Φ50 pencil	3h 12m	6m
Φ40 pencil	3h 05m	10m
Φ50 scanning(X)	20m	5m
Φ50 scanning(Y)	20m	5m
Φ25 clean-up	2h 50m	21m
Φ30 pencil	2h 42m	9m
Φ30 scanning(X)	46m	11m
Φ30 scanning(Y)	43m	13m
Φ16 pencil	1h 48m	10m
Φ16 clean-up	2h 00m	20m
Φ10 pencil	2h 58m	12m
Φ10 clean-up	1h 08m	18m
Φ8 pencil	2h 20m	15m
Φ6 pencil	1h 52m	17m
Φ4 pencil	1h 29m	25m
Total	19h 02m	3h 30m

산 시간의 경쟁력을 확보하려는 시도를 하고 있다(두 제품의 구체적인 명칭을 밝힐 수는 없지만 모두 해외에서 개발된 제품이다).

본 연구에서는 최근의 공구 경로 계산 방법을 소개하고 공구 경로 계산 알고리즘을 고안할 때 고려할 점과 어려운 점을 정리하고자 한다. 그리고 구현 사례 분석을 통해 보다 효과적인 계산 방법에 관한 연구 방향을 모색하고자 한다.

2. 기존 연구

2.1 공구 경로 생성 기술의 변화

70년대 APT 개발 이후 CAM 시스템 개발 초기에는 3차원 자유곡면을 가공할 수 있는 공구 경로를 생성한다는 것 자체가 중요한 주제로 인식되었다. 주로 개별 곡면 단위로 곡면의 매개 변수를 따라 공구 경로를 생성하는 것이 일반적이었다. 자유곡면의 공구 경로 계산은 오목 부위와 볼록 부위의 간섭에 따른 과삭을 제거하는 것이 가장 기본적인 문제다. 과삭 없는 공구 경로를 계산하는 효율적인 방법과 생성된 공구 경로에서 과삭을 검출하는 방법[1]들이 이 시기에 제기되었으며 개념적인 측면에서는 최근의 방법들도 초기

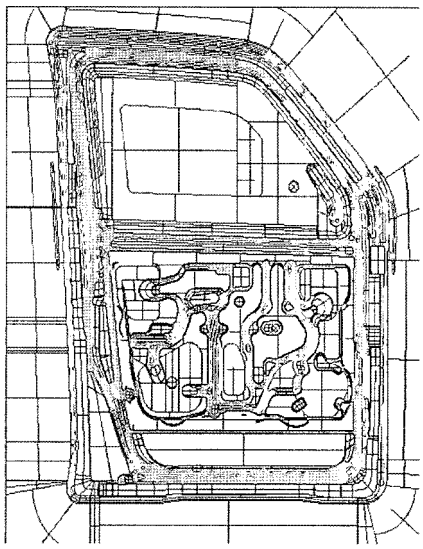

Fig. 1. Example part for bench marking.

에 제안된 방법과 크게 다르지 않다.

컴퓨터 및 컴퓨터 그래픽 성능이 일정한 수준에 도달한 1980년대 후반과 1990년대 초에는 여러 개의 곡면을 동시에 고려한 공구 경로 생성이 주요 과제로 인식되었다[5,6]. 여러 개의 트림 곡면(trimmed surface)으로 구성된 형상을 한꺼번에 가공할 수 있는 주사선 경로(공구 경로가 XY 평면에서 서로 평행)가 가장 기본적인 공구 경로였다.

고속가공(이송속도 5,000 mm/min 이상, 주축회전수 10,000 RPM 이상) 기술이 등장하면서 잔삭(혹은 미삭) 제어와 고속 가공 지원을 통한 가공 생산성 및 가공 품질 향상이 공구 경로 생성의 최대 과제로 부상하였다.

컴퓨터 계산 능력이 더 좋아지고 사용 가능한 메모리 용량이 커지면서 공구 경로 계산 방법도 변화를 가져오게 되었다. Flutter와 Todd의 연구[3]에서는 공구 경로 생성 기술의 최근 변화로 다면체 모델(polyhedral model)의 사용을 지적하고 있다. 다면체 모델에 근거한 공구 경로 생성은 이미 1980년대에 시도 되었다[7]. 일반적인 금형 형상의 경우 원하는 정밀도와 다면체 모델로 근사하면 많게는 수백만 혹은 수천만 개의 다면체로 표현되는데, 그것을 컴퓨터에서 저장 처리하는 것이 최근에 가능해지면서 다면체 이용 방법이 새롭게 부각되고 있다. 그리고 컴퓨터 그래픽스, 3차원 측정, 쾌속 조형 등의 분야에서도 삼각형에 기반을 둔 다면체 모델이 다양하게 쓰이고 있다.

다면체 모델은 매개변수형 곡면을 다면화(tessella-

tion)해서 얻을 수 있으며, 다양한 방법론이 연구 되었다[8-11].

2.2 삼각망에 기초한 공구 경로 생성의 방법

매개 변수 곡면을 공구 경로 계산에 직접 사용하는 방법은 일반적인 복잡한 3차원 복합(곡면이 여러 개) 곡면을 가공하기에는 어려움이 있고, 최근의 추세가 다면체를 이용하는 것이므로 논의에서 제외하자[3]. 그리고 다면체의 경우 대부분 한 면(face)이 삼각형인 삼각망(triangular net)을 많이 쓰고 있으며, 삼각형이 아닌 다각형(polygon)으로 일반화하는 경우에도 개념적인 방법은 크게 다르지 않으므로 본 연구에서는 삼각망으로 형상 모델을 한정하고자 한다.

삼각망에 기초한 공구 경로 계산은 크게 두 가지로 나뉠 수 있다. 첫 번째는 공구 접촉점(CL-point)을 탐색하는 방법이고 다른 하나는 삼각망을 오프셋해서 오프셋 모델에 공구 경로를 투영하는 방법이다.

공구 접촉점 탐색 방식은 Fig. 2와 같이 무한 거리에 있는 공구를 공구 축 방향으로 삼각망 모델에 접근시킬 때 최초로 닿는 점을 찾는 방식이다. 이 방법은 공구 형상에 비교적 자유로우며 알고리즘이 단순하지만 계산량이 많다는 단점이 있다. 이 방법은 공구 경로를 연속적인 곡선으로 생성하지 못하고 공구 경로에 놓이는 점들을 짧은 간격으로 샘플링 해야 한다. 중요한 가공 품질로 여겨지는 모서리의 경우 정밀도를 높이기 위해서는 더욱 많은 샘플링이 필요하며, 그 결과 계산량이 증가하게 된다. Hwang 등의 연구[12]는 효율적으로 접촉점을 찾는 방법을 제시하고 있다.

삼각망을 오프셋 한 후 그 면에 가공 경로를 투영하는 방법은 계산량이 적고 정밀한 계산이 가능하다. 그러나 오프셋 알고리즘이 다소 복잡하고 다양한 형상의 공구를 지원하기 위해서는 그에 맞는 삼각망 오프셋 알고리즘이 필요하다. 삼각망 오프셋 방법은 모서리와 꼭지점의 오프셋 면 표현 방식에 따라 명시적(explicit) 오프셋과 암시적(implicit) 오프셋 두 가지로 나눌 수 있다(명칭은 저자가 임의로 붙인 것임). 명시적 오프셋 방식은 삼각형뿐만 아니라 삼각형의 모서리와 꼭지점의 오프셋 면을 모두 삼각형으로 표현한다. 따라서 오프셋 된 결과가 새로운 삼각망이 된다[13]. 암시적 오프셋 방식은 모서리와 꼭지점의 오프셋 면을 수식으로 표현한다. 이때 그 오프셋 면의 경계는 별도의 식으로 표현된다[14]. 그런데 두 방법 모두 오프셋 면의 꼬임을 완전히 해결하는 것이 용이하지 않기 때문에 공구 경로 계산에서 곡선의 꼬임을 해결하는 방법을 사용하고 있다[14,15]. 솔리드(water tight solid)로 표현 가능한 삼각망의 경우에는 오프셋 모델의 꼬임을 해결하는 방법이 최근에 연구되었다[16].

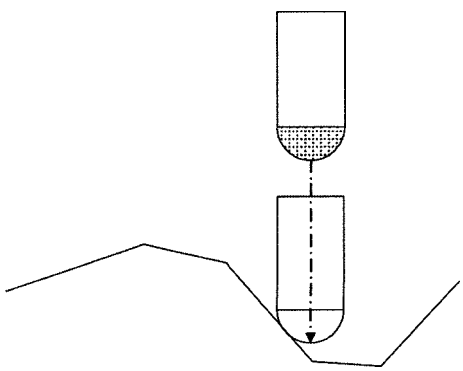

Fig. 2. Searching CL-point in 2D view.

3. 공구경로 생성 알고리즘의 고려 점

3.1 모델의 문제

공구 경로를 생성하기 위한 입력 형상 모델(삼각망)은 결코 완벽한(혹은 깨끗한) 모델이 아니다. 통상 공구 경로 생성을 위해 입력되는 형상 모델의 상태 혹은 문제점은 다음과 같다.

1) 겹침 혹은 틈새가 있다.
2) 역구배가 있다.
3) 단차가 있다.
4) 비정상적 위상정보가 있다.
5) 비정상적으로 작거나, 긴 삼각형이 있다.

결국 형상 모델의 문제점을 미리 해결 하거나 나중에 그런 문제를 만나도 문제가 없는 알고리즘을 고안해야 한다. 공구 접촉점 탐색 방식은 형상 모델을 변형하지 않기 때문에 이러한 문제점에 비교적 강건하다. 그러나 형상 모델을 오프셋 하는 방식은 삼각형의 법선 정보와 위상정보에 의지하기 때문에 모델의 상태에 매우 민감하다.

3.2 생성할 경로의 종류

최근에 많이 쓰이는 공구 경로의 종류는 주사선, 등고선, 투영(2D를 3D로), 펜슬, 잔삭, 3차원 오프셋 가공 등이다. 공구 경로의 종류와 이름은 CAM 시스템별로 상이하지만 전체적으로는 유사성이 있으며, 구체적인 형태는 관련 인터넷 사이트[17-19]에서 확인할 수 있다. 최근의 추세로는 경로의 간격을 조절해서 절삭 부하를 일정하게 할 수 있어야 하며 잔삭 경로를 안정적으로 계산할 수 있어야 한다. 주사선 경로를 이

용해서 여타의 다른 공구 경로를 계산하는 방법을 많이 쓰지만 계산 시간과 정밀도를 고려한다면 직접 계산하는 것이 좋다.

3.3 사용할 공구

일반적인 금형 가공에서는 라운드엔드밀(round-end mill)을 지원할 수 있으면 된다. 라운드엔드밀을 지원할 수 있으면 그 특수한 경우인 볼엔드밀(ball-end mill)과 평엔드밀(flat-end mill)이 가능하기 때문이다. 특수하게는 APT의 일반 공구를 지원해야 할 경우도 있겠지만 통상의 금형 가공에는 아직 쓰이지 않고 있다.

4. 구현 사례

4.1 구현 동기 및 개요

Table 1에서 계산 시간을 비교한 A 회사의 CAM 시스템은 공구의 접촉점을 탐색하는 방식으로 모든 공구 경로를 계산하고 있었다(회사와 제품명을 밝힐 수 없으나 해외 개발 회사로 10년 이상 수백 업체에 판매). 그러나 최근 B사와의 경쟁에서 계산 시간이 가장 치명적인 문제점으로 지적되어 공구 경로 계산 방법을 전반적으로 수정하게 되었다. 검토 끝에 새롭게 채택된 방법은 삼각형으로 구성된 다면체를 암시적 오프셋 하고, 오프셋 다면체를 잘라서 생성된 곡선을 정규화 하는 방식으로 기본적인 개념은 Jun 등의 연구[14] 방법과 같다. 그리고 첫 단계 개발로 볼엔드밀 공구만 계산할 수 있도록 구현되었다.

공구 경로 생성의 전체적인 알고리즘은 다음의 다섯 단계로 구성된다.

1) 형상 모델의 삼각화(triangulation)
2) 삼각망 모델의 오프셋(offset)
3) 오프셋 모델을 공구 경로 평면으로 절단
4) 교선의 추적(tracing) 및 연결
5) 교선의 정렬 및 정리(regularize)

입력 형상 모델은 일반적인 CAD 시스템에서 모델링된 데이터이며 삼각화를 거치면 다면체 모델인 삼각망 모델이 얻어진다. 얻어진 삼각망 모델을 오프셋 한 후 오프셋 모델을 공구 경로 평면으로 절단하면 초기 공구 경로인 교선(intersection curves)이 얻어진다. 그런데 오프셋 모델이 꼬임을 갖고 있는 비정규 모델(non-regular model)이므로 얻어진 교선도 꼬이거나 끊김이 있다. 교선을 공구 경로 방향으로 정렬하고 꼬임을 제거해서 연결하면 완전한 공구 경로가 생성된다. 기타 자세한 내용은 Jun 등의 연구[14]에서 찾아볼 수 있다.

새로운 CAM 시스템을 개발하는 것이 아니라 공구 경로 계산 모듈의 일부를 변경하는 것이었기 때문에 A사 CAM 시스템의 개발 환경을 그대로 따랐다. 사용된 컴파일러는 마이크로소프트사의 Visual C++ 6.0이며, OS는 Windows XP이다. 개발 검토에서 알고리즘 구상, 설계 구현, 최종 테스트까지 3명이 참가하여 약 20개월 정도 소요 되었다.

4.2 형상 모델의 삼각화 및 삼각형 데이터 구조

A사에서 기존에 사용하던 방법이 삼각형 모델에서 공구의 접촉점을 찾는 방식이었기 때문에 곡면 모델을 삼각화해서 삼각형 모델을 만드는 부분은 안정적이었다. 그리고 삼각화를 하는데 걸리는 시간은 예상보다 길었지만 특정 곡면 모델에 대해서 한번만 수행하는 연산이고, 새롭게 개발하기에는 그 부담이 너무 크기 때문에 기존의 프로그램을 그대로 쓰기로 하고 저장 구조만 달리 하였다. 기본적으로 삼각형을 저장하는 배열과 삼각형의 꼭지점을 저장하는 두 개의 배열을 갖는 간단한 구조를 취하였다. 그러나 대규모 데이터의 경우 연결된 하나의 배열로 메모리를 할당하기 어렵고, 모든 데이터를 메모리에 상주 시키는 것이 어렵기 때문에 배열 관리는 대용량을 처리할 수 있는 별도의 프로그램을 사용하였다.

하나의 곡면에서 삼각화된 삼각형은 공통 꼭지점(vertex)과 공통 모서리(edge)를 가지고 있다. 즉, 하나의 곡면에서 생성된 삼각형들은 위상정보를 갖고 있다. 그러나 서로 다른 곡면에서 생성된 삼각형들은 비록 기하학적으로 인접해 있더라도 공통의 꼭지점, 모서리 정보를 갖지 않기 때문에 곡면 사이에 틈새가 벌어져 있기도 하고 겹쳐져 있기도 하다.

사용된 삼각형의 데이터 구조는 Fig. 3과 같다. 이때, TriId는 인접한 삼각형의 ID(삼각형 배열에서의 인덱스)를 나타낸다. 인접한 삼각형이 없는 경우에는 (-1)을 저장한다. VtxId는 세 꼭지점(vertex)의 ID(꼭지점 배열에서의 인덱스)를 나타낸다. 그리고 Convexity는 삼각형 각 변에 해당하는 모서리를 '볼록', '평평', '오목', '없음'의 네 가지로 분류한다. 모서리를 공유하는 두 삼각형의 법선 벡터로 '볼록', '오목' 혹은 '평평'으로 판정되며, 인접 삼각형이 없는 최외곽 모서리의 경우에는 '없음'으로 판정된다. 삼각형의 법선 벡터는 언제든지 계산가능 하지만 계산시간의 절약을 위해 별도로 'Normal' 변수에 저장하였다. 이때 Vector3D 형식은 3개의 단정도 실수(float, 4바이트) 좌표값을 표현한다. 실제 구현에서는 36바이트에 하

```
class Triangle {
    int             TriId[3];
    int             VtxId[3];
    unsigned char   Convexity[3];
    Vector3D        Normal
}
```

Fig. 3. Data structure of Triangleclass Vertex.

```
class Vertex {
    Position3D      Pos
    int             TriId;
    unsigned char   VtxN;
}
```

Fig. 4. Data structure of Vertex.

나의 삼각형을 저장하기 위해 TriId와 Convexity를 각각 30바트와 2비트로 표현 하였다.

꼭지점의 데이터 구조는 Fig. 4와 같다. 꼭지점의 좌표값 (x, y, z)는 배정도 실수(double, 8바이트)값으로 Pos에 저장되며, 이 꼭지점을 가지는 삼각형의 ID와 그 삼각형에서 이 꼭지점이 삼각형에서 몇 번째(0, 1, 2) 꼭지점인지를 저장한다.

4.3 삼각망의 오프셋

삼각망으로 표현된 다면체의 오프셋 모델은 삼각형과 모서리, 꼭지점의 오프셋으로 얻어진다. 먼저 삼각형의 오프셋은 Fig. 5와 같이 법선벡터 방향으로 삼각형의 세 꼭지점을 공구반경 R만큼 이동하면 삼각형을 오프셋할 수 있다. 그러나 오프셋 삼각형을 실제로 표현, 저장하기 위해서는 많은 추가 메모리(삼각형의 개수×3 Vtx/삼각형×3 double/Vtx×8 Byte/double)가 필요하다. 본 연구에서는 식 (1)와 방법으로 매번 계산하는 방법을 사용하였다.

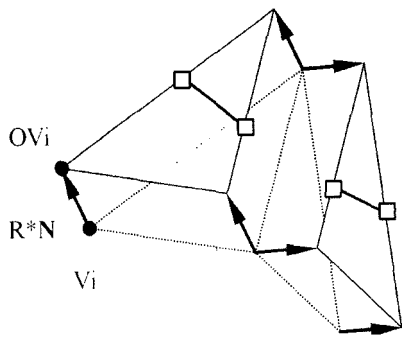

Fig. 5. Offsetting triangles and slicing.

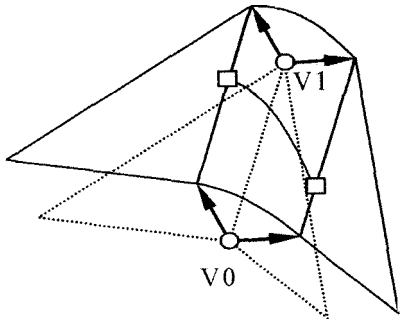

Fig. 6. Offsetting edges and slicing.

$$OVi = Vi + R * N \quad (1)$$

where OVi = ith vertex of the offset triangle
Vi = ith vertex of a triangle
N = unit normal vector of a triangle
R = offset radius

볼엔드밀 공구 경로 계산의 경우 모서리의 오프셋은 원통 면(cylindrical surface)이다. 이때 만들어지는 원통 면은 Fig. 6에서 보는 것처럼 꼭지점 V0, V1에서 만들어지는 원호(circular arc)와 두 오프셋 삼각형의 변이 모서리 오프셋 면의 네 경계 곡선을 이루게 된다. 원호는 모서리 벡터(edge vector)에 수직하고 각각의 꼭지점이 놓이는 평면에 정의된다. 원호는 2차 유리 베지어 곡선(quadratic rational bezier curve)으로 표현 하였다. 모서리 오프셋 면은 모서리가 '볼록'으로 표시되어 있는 경우에만 필요하다. '오목' 혹은 '평평'인 경우에는 그 모서리가 공구 경로에 영향을 미치지 않기 때문이다.

꼭지점의 오프셋 면은 구면(spherical surface)이며, 그 경계 곡선은 해당 꼭지점과 주변 모서리에 의해 정의되는 원호로 구성된다. 따라서 모서리 오프셋 면 정의에 사용된 원호를 경계 곡선으로 사용하면 된다. Fig. 7은 꼭지점의 오프셋 면을 보여 주고 있다. 해당 꼭지점의 인접 모서리가 모두 오목이면 오프셋 면이 생성되지 않는다. 볼록과 오목이 혼재한 경우에는 Jun 등의 연구[4]에서 기술한 방법과 같이 단순화할 수 있다. 그러나 본 연구에서는 볼록 모서리의 원호 곡선만으로 경계를 삼고 경계의 꼬임을 그대로 두었다(4.6절 교선의 정렬, 절단, 연결에서 해결).

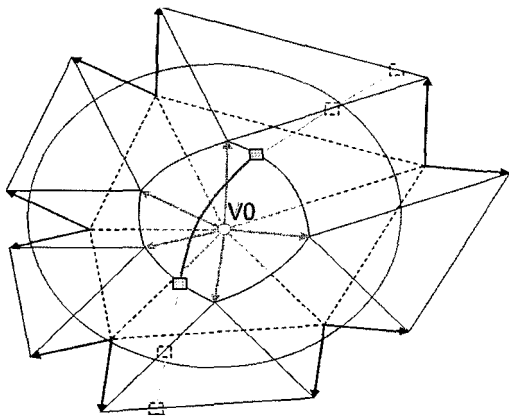

Fig. 7. Offsetting vertices and slicing.

4.4 오프셋 모델의 절단

오프셋 모델은 공구의 중심 궤적면(CL면: cutter location면)이므로 오프셋 모델을 공구 경로가 놓이는 평면으로 절단하면 최종 공구 경로를 얻을 수 있다. 4.3절에서 얻어진 오프셋 모델은 세 종류(삼각형, 모서리, 꼭지점)의 오프셋 면으로 구성된다. 따라서 세 오프셋 면을 공구 경로가 놓이는 평면과 절단하는 방법을 본 절에서는 설명하려고 한다.

4.4.1 삼각형 오프셋 면의 절단

4.3절에서 설명한 것처럼 삼각형의 오프셋 면은 삼각형이므로 삼각형과 평면의 교선 문제가 된다. 삼각형과 평면의 교선 문제는 다양한 해법이 존재하지만 본 연구에서는 삼각형의 세 변과 평면과의 교점을 구하는 문제로 변형하여 해결 하였다. Fig. 5에서 사각형 점이 평면과의 교점에 해당한다. 하나의 삼각형에서 생기는 교점이 두개인 경우에 그 교점을 이은 직선이 삼각형과 평면의 교선이 된다. 실제 구현에서 공차 때문에 교점이 2개 이상인 경우가 있는데 같은 점을 버린다. 그리고 교점이 1개인 경우에는 교선을 형성하지 않는다.

4.4.2 모서리 오프셋 면의 절단

4.3절에서 설명한 것처럼 볼엔드밀의 경우 모서리의 오프셋 면은 원통 면이다. 원통 면을 평면으로 절단해서 생기는 교선은 타원호(elliptic arc)이며, 원통면을 이루는 네 경계 곡선(원호와 직선 각각 2개)과 평면의 교점을 구하면 쉽게 얻을 수 있다[14]. 오프셋 삼각형의 변과 생기는 교점은 4.4.1절에서 이미 구해졌으므로 원호와 평면과의 교점만 계산하면 된다. 3차원 공간에 놓인 원호와 평면의 교점은 해석적인 방법으로도 구할 수 있지만 Newton-Raphson 방법을 사용하였다. 해석적 방법으로 구현한 결과 수치 오류가 누적되어 원하는 정밀도를 맞출 수 없었다.

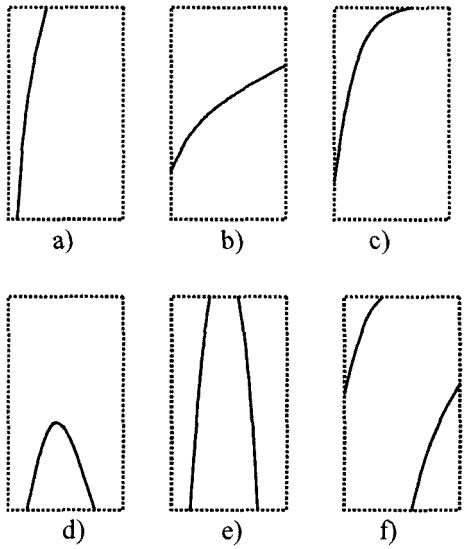

Fig. 8. Slicing cylindrical surface.

Fig. 8는 모서리의 오프셋 면인 원통 면을 평면으로 자를 때 생기는 단면 곡선의 경우를 보여 주고 있다. 그림의 사각형에서 아래와 위쪽의 변이 원호가 이루는 경계곡선이다. 원호는 평면과 두 점에서 교차할 수 있지만 선분은 평면과 한번만 교차하기 때문에 Fig. 8와 같은 경우의 수만 존재 한다. 그림에서 e)와 f)는 네 개의 교점으로 두개의 교선이 생기는 경우이다. Fig. 6의 예는 Fig. 8의 b)에 해당한다. 모서리 오프셋 면의 절단으로 생성되는 교선은 모두 타원호(elliptic arc)이므로 2차 유리 베지어 곡선으로 저장 하였다.

4.4.3 꼭지점 오프셋 면의 절단

꼭지점 오프셋 면은 4.3에서 설명한 것과 같이 구면이다. 구면을 평면으로 절단해서 얻어지는 교선은 원호(circular arc)가 된다. 3축 가공의 경우 오프셋 면을 이루는 경계 곡선과 평면의 교점이 존재하는 경우에만 꼭지점의 오프셋 면이 절단되므로 경계 곡선의 교점을 연결하는 방법으로 오프셋 면의 절단을 수행한다. Fig. 7과 같이 해당 꼭지점을 공유하는 모서리가 모두 볼록하다면 교점은 두개만 얻어지므로 쉽게 원호를 구성할 수 있다. 그러나 교점이 두개를 초과하는 경우에는 확실한 교선을 골라내기 어렵다. 특히 해당 꼭지점을 공유하는 모서리 중에 오목 모서리가 있다면 문제는 더욱 복잡해진다. 따라서 교점이 두개를 초

과하는 경우에는 어느 한 점을 포함하는 반원(half circle)을 교선으로 채택하였다.

4.5 교선의 추적(tracing)과 연결

공구 경로를 형성하기 위해서는 오프셋 모델을 절단해서 얻어진 각각의 교선(intersection curve)을 하나로 연결을 해야 한다. 이것을 위해 삼각망의 위상정보를 활용한 추적(tracing) 방법을 사용하였다.

4.4절에서 설명한 방법으로 삼각형, 모서리, 꼭지점의 오프셋 면을 평면으로 자른 교선을 계산하고 모두 저장 해 두었다. 저장된 임의의 교선 하나를 선택한다. 그 교선이 삼각형을 잘라서 생긴 교선(line)이라면 그 끝이 항상 모서리에서 멈추게 된다. 그 끝과 일치하는 모서리 오프셋 면의 교선(elliptic arc)을 연결한다. 모서리 오프셋 면의 교선 끝은 이웃한 삼각형의 교선으로 연결되거나 꼭지점 오프셋 면의 교선(circular arc)으로 연결되게 된다. 계속 추적, 연결을 반복하다가 연결할 교선이 없으면 멈추게 된다. 오목한 모서리를 만나거나 복잡한 꼭지점을 만나는 경우 혹은 모델의 경계에 도달하면 더 이상 연결한 교선을 찾지 못하게 된다. 이 경우에는 새로운 초기 교선을 꺼내서 추적과 연결을 반복한다.

Fig. 9에서 보는 것과 같이 처음 (a-b)로 표현되는 교선으로 시작하는 경우에 b점에서 모서리 오프셋 교선인 (b-c)를 연결하고, c점에서 다시 (c-d)로 연결한 후 (d-e)로 연결하게 된다. e점에서는 더 이상 연결할 교선이 없으므로 추적을 중지한다.

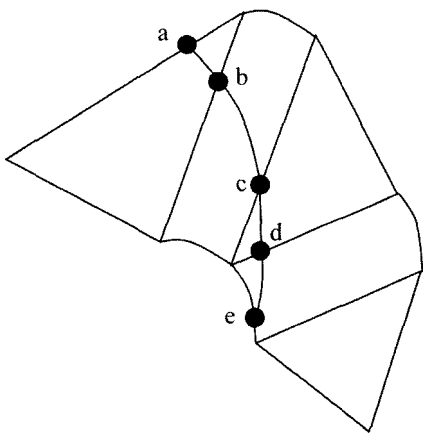

Fig. 9. Tracing intersection curves.

4.6 교선의 정렬, 절단, 연결

삼각망 모델이 완전히 볼록인 경우가 아니라면 4.5절의 교선 추적 연결로는 최종 가공 경로를 얻을 수 없다. 일반적인 모델의 경우 오프셋 모델을 평면으로 자르면 Fig. 10의 a)와 같이 여러 개의 교선으로 구성되며 불필요한 곡선이 생성된다. 3축 가공의 경우 공구 축 방향에서 보이는 곡선만 올바른 가공 경로에 해당하므로 쉽게 불필요한 부분을 제거할 수 있다. 불필요한 부분의 제거는 기존 연구[20,21]에서 설명하고 있는 방법을 사용하였으며 개략적인 절차는 다음과 같다.

먼저 모든 교선의 방향을 공구 진행 방향과 일치시킨다. Fig. 10의 a)와 같이 공구 축이 Z축과 평행하고 가공 방향이 X축과 평행한 경우 교선을 이루는 점의 x 좌표값이 증가하는 방향으로 교선의 방향을 맞춘다. x 좌표값이 증가하다가 감소하는 경우에는 그 점에서 교선을 분할하고, 방향이 반대인 교선을 뒤집는다(기존 연구[20,21]의 monotone chain 분할). 모든 교선의 방향이 공구 진행 방향으로 맞춰진 후에는 각 교선의 교점을 찾고 교점에서 교선을 분할한다. 이때 교선은 선분(삼각형 오프셋 면과의 절단선)과 2차 유리 베지어 곡선(모서리와 꼭지점 오프셋 면과의 절단선)으로 구성되어 있으므로 '선분과 선분', '선분과 2차 유리 베지어 곡선', '2차 유리 베지어 곡선과 2차 유리 베지어 곡선' 등의 세 가지 종류의 교점을 구하는 함수가 필요하다. 2차 유리 베지어 곡선의 교점 계산은 곡선 분할(subdivision)을 통해 곡선을 선분화해서 '선분과 선분'의 교점 문제로 변환하였다.

이제 교선의 방향은 공구 진행 방향으로 맞추어져 있고, 모든 교점에서 교선을 분할했기 때문에 각각의 교선은 서로 꼬임(혹은 교점)이 없다. z 좌표값이 가장 큰 교선을 선택하고 연결된 교선을 추적하면 최종적인 공구 경로를 얻게 된다. Fig. 10의 b)는 공구 경로로 사용할 수 있는 최종적인 곡선을 보여주고 있다.

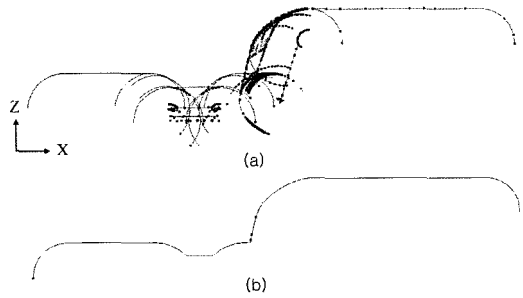

Fig. 10. Removing invalid segments.

4.7 구현 결과

Table 2는 A사와 B사 시스템 및 본 연구에서 구현된 방법으로 같은 공구 경로를 계산하는데 걸리는 시

Table 2. Comparison of computing time (XEON 2.4GHz/ Windows 2000/2GB RAM)

	A system	B system	Proposed method
Φ50 scan(X)	20m	5m	7m
Φ50 scan(Y)	20m	5m	7m
Φ25 clean-up	2h 50m	21m	35m
Φ30 scan(X)	46m	11m	10m
Φ30 scan(Y)	43m	13m	10m
Φ16 clean-up	2h 00m	20m	28m
Φ10 clean-up	1h 08m	18m	23m
Total	8h 07m	1h 33m	2h 00m

Table 3. Analysis of computing time

Computing step	Ratio
Offsetting model	10.5%
Slicing offset model	23.2%
Tracing intersections	20.1%
Trimming curves	46.2%
Total	100%

간을 보여준다. 이때 사용된 예제 모델은 Fig. 1이며 사용된 컴퓨터 사양도 동일하다. 모든 종류의 공구 경로 계산 모듈을 새로 구현한 것이 아니기 때문에 주사선 공구 경로와 잔삭 공구 경로만을 비교 하였다. 본 연구에서 구현된 방법이 B 시스템에 비해 30% 정도 느리며 주로 잔삭 공구 경로 계산에 더 많은 시간이 걸림을 알 수 있다. 그러나 A 시스템에 비해서는 현격히 빠른 계산 시간을 보이고 있다. 21개의 실제 자동차 프레스 금형 형상 데이터로 다양한 조건의 테스트를 해 본 결과 주사선 공구 경로의 경우 공구의 직경이 30 mm 이상으로 큰 경우에는 B사가 빠르고, 공구가 작아질수록 본 연구에서 구현한 방법이 빠른 것으로 나타났다.

Table 3은 Fig. 1의 예제를 공구 직경 30 mm, 경로 간격 0.7 mm로 X축과 평행한 주사선 공구 경로 생성할 때 걸리는 계산 시간을 각 계산 단계별 비율로 보여 주고 있다. 교선을 정렬하고 절단 연결하는 부분(4.6절)에서 많은 시간을 소비하고 있었다. 추후에 집중적인 개선이 필요한 부분이다. 잔삭 공구 경로 계산의 경우에도 계산 시간 분석을 통해 알고리즘을 개선한다면 전체적인 계산 시간이 더욱 빨라질 것으로 기대 된다.

Fig. 11은 새롭게 구현된 방법으로 계산한 잔삭 공구 경로를 보여주고 있다(직경 25 mm 볼엔드밀로 가공된 면을 직경 12 mm 볼엔드밀로 잔삭). 이 공구 경로는 XY-평면에 수직한 평면이 아닌 임의의 평면으로 오프셋 모델을 절단하는 방법을 사용했다.

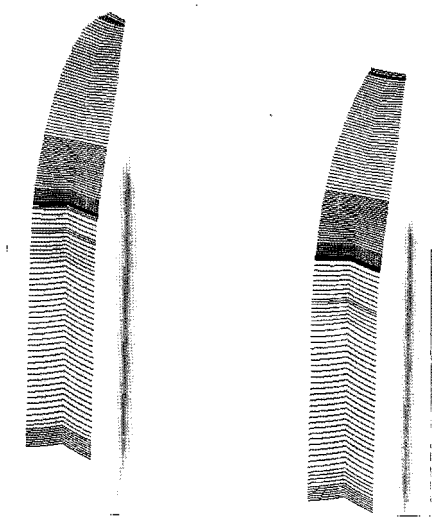

Fig. 11. Example tool-path: clean-up.

5. 결 론

본 연구에서 구현된 오프셋 모델의 절단과 교선 추적 방법은 기존의 공구 접촉점 계산 방법에 비해 70%이상 빠른 방법으로 평가된다. 그러나 삼각망의 오프셋으로 공구 경로를 생성하는 다른 방법들[13-15,22]과 객관적인 계산 시간 비교를 할 수 없었다. 향후 관련 연구의 촉진과 객관적인 평가를 위해 사용된 예제의 공유 등이 필요하겠다.

예상한대로 본 연구에서 구현된 방법은 공구 접촉점을 탐색하는 방법에 비해 형상 모델의 품질에 보다 민감한 특성을 보였다. 이로 인해서 프로그램의 테스트와 안정화에 많은 노력이 들었다. 형상 곡면의 법선을 오프셋 방향으로 사용하고 있기 때문에 역구배가 있는 형상 혹은 법선을 판별하기 곤란한 수직 측벽의 경우에는 계산의 어려움이 있었다.

Fig. 12와 같이 하나의 꼭지점을 공유하는 삼각형이 지나치게 많은 경우 삼각형의 폭이 아주 좁아져서 삼각형을 자른 선분의 하나의 길이가 아주 짧아지게 된다(10E-4 이하). 그러나 개개의 선분은 무의미할 정도로 짧지만 그것을 연결한 교선은 충분히 길기 때문에 삭제할 수는 없었다. 결국 소수점 계산 오류를 막을 대책이 필요하다. 그리고 이 경우에 삼각형 법선 벡터도 그 결과가 불안정하여 '불록'과 '오목', '평평'이

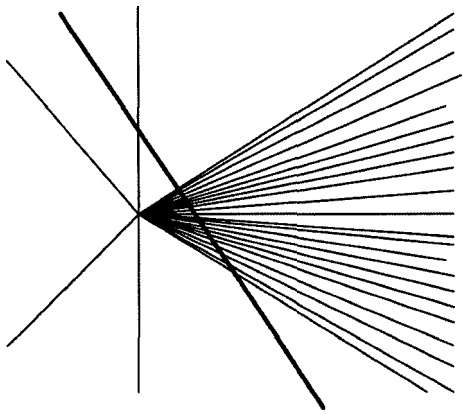

Fig. 12. Round-off error.

혼재하는 것으로 나타났다.

새롭게 구현된 방법은 기존의 공구 접촉점을 탐색하는 방법에 비해 많은 메모리를 사용하게 되었다. 그 결과 삼각망을 이루는 삼각형의 개수가 대략 1,000만 개 이상이면 윈도 OS의 메모리 한계인 2GB 메모리를 모두 소진하는 것으로 나타났다. 따라서 1,000만개 이상의 삼각형 모델은 별도의 모델 분리 작업을 통해 계산을 수행할 수 있도록 구현하였다.

A사에서는 Fig. 13와 같이 실제 가공을 통해 의도하는 가공 정밀도와 거칠기를 만족하는지 테스트하였으며, 21개의 테스트 모델에 대해 25가지의 계산 조건으로 새로운 공구 경로 생성 방법을 테스트 하고 실제 형상 가공을 통해 가공 품질을 비교 측정하였다. 가공 품질은 전반적으로 예전의 방법보다 우수한 것으로 판명되었다. 그리고 테스트에서 발생한 다양한 문제점들을 발견 제거하였으며 최근에는 안정되고 빨라진 CAM 시스템을 사용자들에게 최종 발표 하게 되었다.

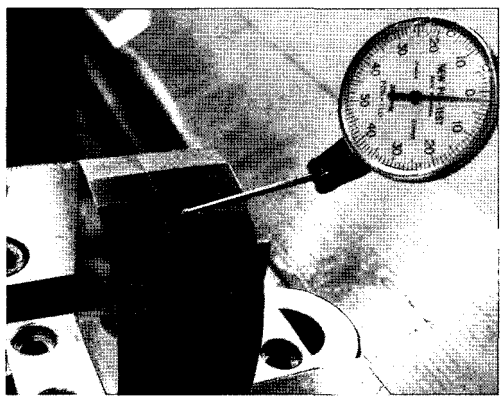

Fig. 13. Machining and accuracy test.

참고문헌

1. The State of the NC Software Market, CAD/CAM Publishing, Inc., 2003.
2. Encyclopedia of the APT Programming Language, CAM-I Inc., 1973.
3. Flutter, A. and Todd, J., "A Machining Strategy for Tool Making", *Computer-Aided Design*, Vol. 33, pp. 1009-1022, 2001.
4. Anderson, R. O., "Detecting and Eliminating Collisions in NC Machining", *Computer-Aided Design*, Vol. 10, pp. 231-237, 1978.
5. Elber, G. and Cohen, E., "Toolpath Generation for Freeform Surface Models", *Computer-Aided Design*, Vol. 26, No. 6, pp. 490-496, 1994.
6. Choi, B. K., Lee, C. S., Hwang, J. S. and Jun, C. S., "Compound Surface Modeling and Mmachining", *Computer-Aided Design*, Vol. 20, No. 3, pp. 127-136, 1988.
7. Hansen, A. and Arbab, F., "Fixed-axis Tool Positioning with Built-in Global Interference Checking for NC Path Generation", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 6, pp. 610-621, 1988.
8. Austin, S. P., Jerard, R. B. and Drysdale, R. L., "Comparison of Discretization Algorithms for NURBS Surfaces with Application to Numerically Controlled Machining", *Computer-Aided Design*, Vol. 29, No. 1, pp. 71-84, 1997.
9. Piegl, L. A. and Richard, A. M., "Tessellating Trimmed NURBS Surfaces", *Computer-Aided Design*, Vol. 27, No. 1, pp. 6-26, 1995.
10. Piegl, L. A. and Tiller, W., "Geometry-based Triangulation of Trimmed NURBS Surfaces", *Computer-Aided Design*, Vol. 30, No. 1, pp. 11-18, 1998.
11. 정재호, 박준영, 트림된 "NURBS곡면의 효율적인 삼각화 알고리즘", 한국CAD/CAM학회 논문집, 제5권, 제2호, pp. 144-154, 2000.
12. Hwang, J. S. and Chang, T. C., "Three-axis Machining of Compound Surfaces Using Flat and Filleted Endmills", *Computer-Aided Design*, Vol. 30, No. 8, pp. 641-647, 1998.
13. Kim, S. J. and Yang, M. Y., "Triangular Mesh Offset for Generalized Cutter", *Computer-Aided Design*, in press
14. Jun, C. S., Kim, D. S. and Park, S., "A New Curve-based Approach to Polyhedral Machining", *Computer-Aided Design*, Vol. 34, pp. 379-389, 2002.
15. Park, S. C., "Sculptured Surface Machining Using Triangular Mesh Slicing", *Computer-Aided Design*, Vol. 36, pp. 279-288, 2004.
16. 정원형, 정춘석, 신하용, 최병규, "옵셋팅을 위한 정규 삼각망 추출", 한국CAD/CAM학회 논문집, 제9권, 제3호, pp. 203-211, 2004.
17. PowerMILL, Delcam, www.powermill.com
18. WorkNC, Sescoi, www.sescoi.com

19. SPEED Plus, CSCAM, www.cscam.co.kr
20. Park, S. C., Shin, H. and Choi, B. K., "A Sweep Line Algorithm for Polygonal Chain Intersection and Its Applications", *Proceedings of IFIP WG5.2 GEO-6 Conference*, Tokyo University, pp. 187-95, 1998
21. 박상철, 신하용, 최병규, "점열곡선의 꼬임을 효율적으로 찾는 알고리즘", 한국CAD/CAM학회 논문집, 제4권, 제3호, pp. 190-199, 1999.
22. 추원식, 안성훈, 김동수, 전차수, "MIMS: 웹기반 마이크로 머시닝 서비스", 한국CAD/CAM학회 논문집, 제9권, 제3호, pp. 246-252, 2004.

정 연 찬

1989년 한양대학교 산업공학과 학사
1991년 KAIST 산업공학과 석사
1996년 KAIST 산업공학과 박사
1998년~1999년 다이믈러크라이슬러 연구원
1991년~2002년 큐빅테크 수석 연구원
2004년~현재 서울산업대학교 금형설계학과 교수
관심분야: 디지털 제조 공학, 측정 데이터 처리, 가공 경로 생성 및 검증