한국 CAD/CAM 학회 논문집
제 2 권 제 3 호 1997년 9월 pp. 161-174

기준평면과 경계상자를 이용한 NC 절삭과정의 그래픽 시뮬레이션

이철수*, 박광렬**

Graphic Simulation of Material Removal Process Using Bounding Box and Base Plane

Cheol-Soo Lee* and Gwang-Ryeol Park**

ABSTRACT

In this paper, the techniques for graphic simulation of material removal process are described. The concepts of the bounding box and base plane are proposed. With these concepts, a real-time shaded display of a Z-map model being milled by a cutting tool following an NC path can be implemented very efficiently. The base planes make it possible to detect the visible face of Z-map model effectively. And the bounding box of tool sweep volume provides minimum area of screen to be updated. The proposed techniques are suitable for implementation in raster graphic device and need a few memories and a small amount of calculation. Proposed method is written in C and executable on MS-Windows95 and Window-NT.

Key words : Bounding box, Base plane, NC Cut Simulation, Virtual Machining, Visible Check, Z-map

1. 서 론

NC 가공하기 위한 형상은 CAD/CAM 시스템에서 형상을 모델링하고 NC 코드를 생성한다. 복잡한 형상일수록 NC 코드가 길고 문자 그대로 NC 코드를 확인하기는 힘들며 형상에 대한 직관을 가지기도 어렵다. NC 코드를 검증을 실제 절삭 가공으로 하기에는 재료의 낭비가 있고 저가의 대체 피삭재로 사용하더라도 시간적, 기계 장비 공수의 낭비가 있다. 최근 컴퓨터 그래픽 관련 기술에 힘입어 절삭 모델을 모의 가공하는 방법이 확산되고 있다. 절삭 가공이나 기계 동작의 그래픽 시뮬레이션을 통해 가공 오류, 기계 파손을 방지함으로써 생산 계획의 정확한 수행이나 물적 손실의 감소를 실현할 수 있다. 모의 가공은 물리적으로는 힘든 여러가지 처리가 가능하며 비디오 화면이나 사진처럼 시각적인 사실감을 주는 그래픽 시뮬레이션이 가능하다.

*종신회원, 전남대학교 산업공학과
**학생회원, 전남대학교 산업공학과

3차원 모델을 컴퓨터 화면에 디스플레이하려면 형상 데이터를 갱신하고 가시성 판단(visibility check) 및 쉐이딩 처리를 하는 과정에서 많은 메모리와 계산 시간이 소요되는 것이 일반적이다. 따라서 처수에 치중하는 설계나 시각적 사실감에 치중하는 디자인 분야의 모델과는 별도로 NC 절삭 모델에 적합한 형상 데이터의 구조 및 디스플레이 방법이 필요하다. NC 절삭 가공의 그래픽 시뮬레이션은 Fig. 1과 같은 절차로 이루어 지는데 데이터 및 디스플레이 갱신의 계산량은 일반적으로 NC 코드의 길이만큼 늘어나게된다. Z-map 구조는 형상의 갱신에 대한 계산이 빠르고 간단하여 NC 절삭 모델의 구조로 많이 사용된다[1,2].

본 논문에서는 Z-map 등과 같은 $z=f(x, y)$ 방식이 지원되는 형상 데이터 구조와 NC 절삭 모델의 가공 과정이나 결과를 그래픽 시뮬레이션하는 효율적인 방법을 제안한다. 이 방법은 공구와 모델 및 공구의 경계 정보를 이용하여 기준평면을 매개변수 공간과 대응시킨 후 실물 좌표를 구하는 방식을 이용하여 갱신과 가시성 판단에 필요한 계산량을 줄이도록 하였다. 특히 Z-map과 유사한 구조와 잘 공조하는데 가시성

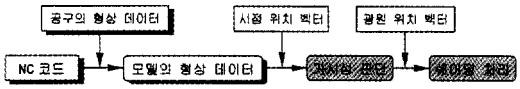


Fig. 1. Graphic simulation process of material removal model.

판단의 계산량이 기존의 방법에 비해 상대적으로 매우 적고 화면의 해상도를 충분히 반영해주며 별도의 메모리를 거의 사용하지 않는 장점을 가지고 있다.

1.1 형상 데이터 관련 연구

3차원 형상을 보관하고 조작하기 위한 많은 연구가 컴퓨터 그래픽 관련 분야 등에서 이루어져 왔다. 솔리드 모델을 위한 CAD/CAM 시스템의 내부 표현은 기본적으로 전통적인 제도 방식과 유사하게 솔리드의 경계를 모서리 및 꼭지점의 형태로 표현하는 B-rep(Boundary Representation)방식과 기본 형상(Primitive)들의 관계를 연산관계에 의해 표현하는 CSG(Constructive Solid Geometry) 방식이 있다. B-rep는 일반적인 형상을 보다 잘 표현하지만 일반적으로 많은 시간을 필요로하며 이와는 대조적으로 CSG는 구축이 더 쉽고 디스플레이가 더 용이하다[3]. NC 절삭 모델의 경우에는 주로 Octree, Voxel, Z-map, M-map 방식과 같은 CSG 지향적인 셀(Cell) 분할 방식의 데이터 구조를 많이 취하는데 공구 스윕 볼륨에 의한 갱신에 대한 계산량과 방법이 빠르고 간단하기 때문이다. Octree와 M-map[4]은 정밀도와 메모리 면에서 효율적이지만 메모리를 할당과 재귀적인 반복이나 연결 리스트의 탐색 등에 별도의 시간이 소요된다. Voxel은 정밀도에 따라 메모리가 급증하기 때문에 정밀도가 낮은 솔리드 모델의 데이터 구조로 주로 사용될 수 있다. Z-map은 측벽이나 동굴 형상의 표현이 불가능하지만 매우 간단하게 구현되고 갱신에 대한 계산량이 적고 빠르기 때문에 5축 NC 가공이 아닌 경우에 NC 모델의 형상 데이터로 많이 사용된다[1,2].

Z-map은 Fig. 1의 (a)와 같이 xy 평면 상의 격자와 해당 z값으로 표현되는 형상 Map 구조인데 고정된 메모리 크기를 유지하면서 주어진 정밀도를 만족해 주며 데이터의 조작이 빠르고 간단하기 때문에 NC 절삭 모델의 표현에 매우 유리하다. M-map은 Fig. 2의 (b)와 같이 몇 개의 도메인이 별도의 map을 가지고 있는 방식이다. 각 도메인 별로 격자의 간격을 달리 줄 수 있고 격자 별로 z값을 연결된 리스트(linked list) 형태로 보관하므로 4,5축 가공에 의한 측벽이나 동굴 형상을 표현할 수 있다[4]. 특히 3축 가공과 같이 동굴 형상이 없는 경우는 M-map에서 한 격자가 여러 개의 z값을 보관하지 않도록 단순화할 수 있다. 이 경우는 각 도메인은 별도의 Z-map을 하나씩 가지는 경우가 된다. 이같이 단순화된 M-map이나 일반적인 Z-map을 NC 절삭 모델의 형상 데이터 구조로 가지는 경우는 공구의 이동에 의한 절삭 볼륨을 반영할 때 공구 바닥면만 고려하면 되므로 구현이 매우 간단하고 빠르다. 직선 이동하는 공구 바닥면의 궤적은 이동 방향에 의한 실루엣 곡선이 만드는 룰드 곡면이다[4,5]. 실루엣 곡선에 의한 공구의 바닥면의 궤적을 구해 Z-map에 반영하는 관련 연구가 이미 자세하게 이루진 바 있다[6].

1.2 디스플레이 관련 연구

NC 가공 과정을 디스플레이하는 경우는 공구의 이동 등을 애니메이션하면서 갱신된 형상을 반영하여 다시 디스플레이해 주는 과정이 반복된다. 가공 과정 또는 가공 완료된 NC 절삭 모델의 형상 데이터나 디스플레이 방법 등을 조작하면 실제 가공으로는 힘든 결과를 얻을 수 있다. 임의의 단면을 설정해 잘라보거나 일차로 가공한 형상에 대한 여러가지 다른 NC 코드에 의한 가공 시도가 가능하다. 원하는 특징 형상을 각기 다른 색으로 출력하여 오류가 있을 만한 위치를 시각적으로 확인해 보거나 공구에 따른

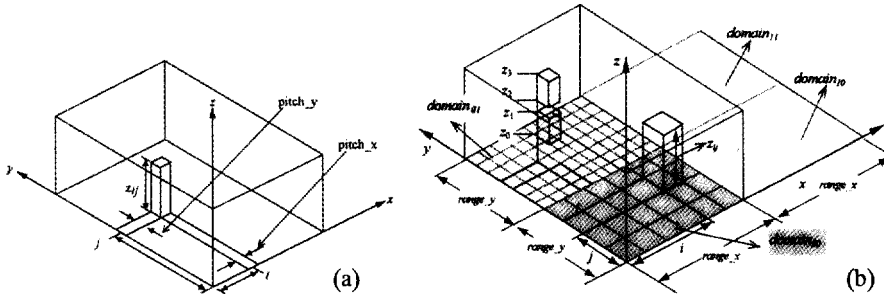


Fig. 2. Representations of Z-map and M-map.

절삭 부위를 색으로 구별해 볼 수 있다. 그리고 화면에 디스플레이된 모델의 특정 부위를 포인팅하여 좌표나 거리 등의 치수 등의 정보를 알아볼 수 있다.

디스플레이 과정에서는 가시성 판단(visible check)이 중요한데 3차원 형상을 2차원 화면에 보일때 시점의 위치에 따라 형상의 가시면을 재 탐색해야 한다. 일반적인 컴퓨터 그래픽스 분야에서 주로 사용되는 가시성 판단 알고리즘은 기본적으로 두가지 범주로 구분할 수 있는데[7,9] 관련된 알고리즘과 함께 Table 1에서 간단히 소개하고 있다.

모의 NC 가공에 많이 이용되는 Z-map 구조는 각 격자를 삼각형 개체로 변환하여 Z-buffering이나 Ray-tracing 등의 방법으로 가시면 판단을 하고 쉐이딩처리하여 디스플레이할 수 있다[1]. Z-buffering을 지원하는 Windows95/NT 환경에서 지원하는 그래픽 라이브러리인 OpenGL을 이용하면[8] PC 환경에서 NC 절삭 모델을 간단하게 디스플레이해 볼 수 있는데 가공 과정을 디스플레이하기 위해서는 공구의 이동에 의해 모델의 Map이 갱신될 때마다 각 개체(삼각형)의 좌표와 법선 벡터를 다시 계산해야 한다. 이 때문에 많은 중복 계산이 발생하고 Z-buffer를 위한 많은 메모리가 요구된다. 일반적인 Z-buffering 방법을 사용한다면 디스플레이 결과는 우수하지만 PC 환경에서 만족할 만한 디스플레이 속도를 얻기위해서는 하드웨어적으로 병렬 처리를 해주는 고가의 그래픽 장비를 별도로 구비해야하는 경우가 많다.

등척 투영(isometric projection)에 의해 화면의 네 개의 화소의 위치와 3차원 상의 좌표를 일대일로 대응시켜 Z-map을 디스플레이하는 방법이 제안된 바 있다[2]. 이 방법은 가시성 판단에 대한 계산은 매우 간단하고 빠르므로 실시간으로 NC 코드에 의한 절삭과정을 디스플레이할 수 있다. 그러나, 등척이 되는 시점이 8방향으로 제한되어 있고 복셀의 세 가시 면을 각 x, y, z 방향의 세가지 색 만으로 표현하는데 별도의 쉐이딩처리를 하더라고 비스듬한 면이 계단 형상으로 거칠게 보이는 것을 피할 수 없다. 이는 Z-map의 한 격자가 작은 정육면체인 복셀(voxel)을 위로 쌓은 형태로 디스플레이되기 때문인데 복셀의 한 면이 화면 상의 네 개의 화소에 해당한다. 즉, 화면의 해상도(resolution)를 충분히 활용하지 못하므로 Z-map이 정확한 z값을 가지더라도 z방향으로 Z-map의 격자 간격만큼 디스플레이의 정밀도가 떨어지게 된다.

가시성 여부가 결정되면 광원의 위치에 따라 음영처리를 해주는 쉐이딩을 하면 된다. 쉐이딩은 광원

Table 1. Visible detection methods

범주	알고리즘	가시 판단 방법	특 징	형상 갱신의 반영

Object Space (연속적)	Area Subdivision[9]	**For**(모든 분할된 View Area) {
		For(모든 개체) do(현재 선택된 View Area 분류*)
		If(분할조건**) do(현재 선택된 View Area 재분할)
		}
		*분류: 둘러쌈/겹침/내부/외부 중 하나로 분류
		**분할조건: (모두 외부 or 둘러쌈/겹침/내부가 하나 or 하나가 모두를 둘러쌈)이 아닌 경우
			보이는 영역에 대한 쉐이딩 계산이 최소화 되지만 계산 시간이 많이 걸림	View Area 전체에 대해 재 분류 및 분할 작업 필요

	Octree Methods[9]	**For**(모든 8분할 개체(cubic): 바깥쪽부터 재귀적탐색) Do(해당 Viewplane*을 4분할함) **Do** Viewplane을 화면으로 대응시킴. ***Viewplane**: 최초 Cubic의 Visible Side 중 하나(4분할면)	형상데이터가 Octree 형태로 되어 있어야 함.	Swept volume에 곡면있는 경우 viewplane의 재분할, 재분류 작업에 재귀적 반복횟수 급증
Image Space (점샘플링)	Z-Buffer[7] Depth-Buffer	**For**(모든 개체) **For**(모든 (x,y)화소) do z값 비교	효과가 우수하나 계산량, 시간 많이 소요	어떤 형상 데이터 구조를 선택하느냐에 따라 알고리즘의 효율이 달라짐. 곡면이 포함된 형상은 삼각형화된 개체로 변환하여 가시면을 탐색하는 방법 주로 사용함. 형상의 갱신 때마다 개체 전체의 탐색 필요.
	Painter's Algorithm[7]	**For**(z순서로 정열된 모든 개체) **For**(모든 (x,y)화소) do 화소 색칠	불필요한 반복 계산 및 디스플레이 시간	
	Ray Tracing[7]	**For**(모든 (x,y)화소) **For**(모든 개체) do z값 비교	개체가 많을수록 많은 비교 시간	
	Scan-Line Method[7]	**For**(y 순서로 정열된 모든 화소) **For**(x 순서로 정열된 모든 개체) **For**(모든 x) do z값 비교	화소 및 개체의 정렬 작업 필요	

과 면과의 거리나 광선 벡터와 면이 이루는 각도에 따라 명암의 정도를 결정하면 된다. NC 절삭 모델의 경우 텍스춰 매핑이나 복잡한 렌더링 알고리즘을 사용하지 않더라도 간단하게 쉐이딩 처리된 디스플레이로도 시각적 사실감을 느낄 수 있다.

1.3 제안된 방법에 의한 NC 절삭 가공의 디스플레이

Z-map이나 M-map 등의 형상 Map은 공구의 이동을 반영하기가 매우 간단하고 빠르다. 이러한 장점을 유지하면서 화면 및 형상 Map의 갱신 후보 영역의 크기를 줄이고 가시성 판단 방법을 점 샘플링 방식을 사용하면 절삭 과정의 디스플레이를 매우 효율적으로 빠르게 구현할 수 있다. 또한 공구 이동 궤적을 실루엣 곡선에 의해 구하고 형상 Map의 격자 간격을 적절히 조정하면 허용 오차 내의 정밀도를 갖는 모의 가공 결과를 얻을 수 있다.

NC 절삭 모델은 가공 전보다 가공 후에 그 크기가 더 작다. 절삭 가공 전의 작업물 형상이 항상 직육면체 형상의 상자라면 가공 후의 절삭 모델은 이 상자에 포함될 수 있다. 화면에 이 상자와 절삭 모델이 동시에 디스플레이 된다면 절삭 모델은 모의 가공 중에 계속 갱신 되지만 상자는 늘 같은 형상을 유지하게 될 것이다. 제안된 방법은 이러한 상자를 정의하고 상자의 각 면이 화면에서 항상 같은 좌표에 대응된다는 점을 이용한다. 2차원 화면 좌표계와 실물이 있는 3차원 실제 좌표계 사이에 매개변수 공간을 두면 화면 좌표계 상에서 구한 매개변수로 실제 좌표계에서 특정 평면 위의 좌표를 구하는 방법으로 Z-map을 사용하는 경우 일반적인 가시성 판단 방법보다 계산 시간을 현저하게 줄일 수 있다. 공구 이동에 의한 Z-map의 갱신은 실루엣 곡선을 이용하면 공구의 이동 궤적을 보다 정확하게 반영할 수 있고 공구가 포함되는 상자를 정의하면 공구의 이동궤적에 의해 갱신해야 할 Z-map의 영역이나 화면의 갱신 영역을 산출할 수 있다.

이와 같이 모델이 포함되는 경계상자(Bounding Box)와 기준평면(Base Plane)의 개념을 이용하면 화면의 점 좌표에 대응하는 3차원 모델의 표면 좌표를 산출할 수 있다. 화면에서 임의로 선택한 NC 절삭 모델 표면 위에 있는 점의 절삭 깊이 또는 그 부분의 법선 벡터를 계산할 수 있으며 두 점 간의 실제 길이도 계산할 수 있다. 제안된 방법으로 NC 절삭을 모의 가공을 하는 과정은 Fig. 3과 같다. Fig. 3에서 (a) 부분은 NC 코드와 무관하게 이미 상수로 정의되는

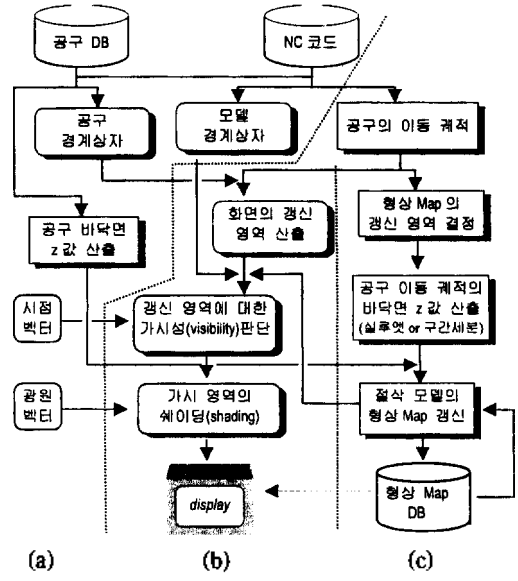


Fig. 3. Flow diagram of proposed graphic simulation method of material removal process (a) Constant, (b) Updating the display, (c) Updating the model map.

부분이다. 여기서 이미 계산된 값을 가지고 (c)부분에서 NC 코드에 의해 모의 가공에 의해 형상 Map을 갱신하고 (b)부분에서 디스플레이를 갱신한다. NC 코드에 의해 모의 가공을 수행하는 과정 중에 항상 일정하게 상수로 정의되는 부분은 주어진 공구에 따른 공구 바닥면과 공구 경계상자, NC 절삭 가공을 할 작업물의 크기에 의한 적절한 모델 경계상자, 광원의 위치 등이 있다. 형상 Map의 갱신은 NC 코드에 의해 공구의 직선 보간 이동에 의한 공구 바닥면 궤적을 이용하며 디스플레이는 갱신된 형상 Map과 관련된 화면 일부를 갱신하는 과정이다.

화면에 어떤 개체를 디스플레이를 한다는 것은 실제로 개체가 화면에서 차지하는 영역의 모든 점(화소; pixel)의 색을 결정한다는 것과 같다. 따라서 변경된 형상 Map 부분을 포함하는 화면의 최소 갱신 영역을 결정하는 것이 중요하다. 본 연구에서는 화면의 갱신영역 결정과 가시면 탐색과정에서 공구 경계상자 및 모델 경계상자 그리고 기준평면의 개념을 이용하여 계산량과 디스플레이 시간을 줄이도록 하였다.

2. 가시평면과 쉐이딩

2.1 모델 경계상자와 기준평면

NC 절삭 모델을 포함하는 최소의 직육면체를 '모

델 경계상자'라고 하자. 가공 할 피삭재(raw stock)의 크기를 미리 설정하거나 NC 코드 전체를 미리 스캔하여 모델 경계상자의 각 면의 경계 좌표는 미리 결정할 수 있다.

절삭 모델과 모델 경계상자가 화면으로 투영되어도 절삭 모델은 모델 경계상자 내부에 그려지게 된다. 예를 들어 절삭 모델과 모델 경계상자를 무한히 먼 거리에서 주어진 시선 벡터 방향으로 바라볼때 절삭 모델과 모델 경계상자는 화면에 Fig. 4의 (a)와 같이 보일 것이다. 만약 모델 경계상자가 투명한 유리이고 모델은 불투명한 재질이라면 Fig. 4의 (a)에 있는 화면 상의 한 점은 (b)처럼 절삭 모델의 표면 위의 점 **M**인지 (c)처럼 모델 경계상자 표면 위의 점 **A**인지 알 수 없다. 이는 3차원 모델을 2차원 화면으로 투영할 때 공간 상에서 시선 벡터의 연장선이 모델 경계상자 또는 절삭 모델의 표면과 부딪히는 교점이 화면의 같은 좌표로 대응되기 때문이다.

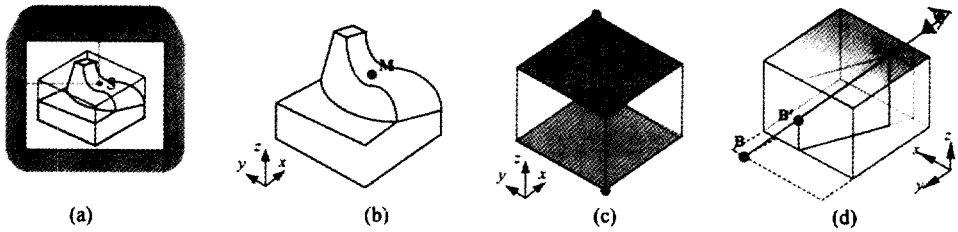
모델 경계상자는 시선 벡터에 따라 1개에서 3개까지의 면을 동시에 보일 수 있는데 어느 면이 보이는지를 시선 벡터와 모델 경계상자의 경계 정보에 의해 알 수 있다. 대부분 Fig. 4와 같이 세 면을 동시에 보는 경우가 많은데 (a)와 같이 화면에 나오는 경우에 모델 경계상자는 (c)와 같은 경우이며 관점을 달리하면 Fig. 4의 (d)와 같이 시점, 시선 벡터 그리고 모델 경계상자가 보이게 된다. Fig. 4에서 색칠된 면을 포함하는 평면을 '기준평면'이라고 하면 화면 상에 투영된 기준평면은 화면 좌표계의 점들(pixels)이 되고 공간 상에 있는 기준평면의 평면의 식이나 법선 벡터를 모델 경계상자의 경계 정보로 쉽게 구할 수 있다. 일반적으로 평면의 식과 한 점을 지나는 직선의 방향을 알고 있으면 평면과 직선의 교점을 구할 수 있으므로 Fig. 4의 (d)와 같은 경우에 시점과 시선 그리고 기준평면을 알면 교점 **A**와 **B** 그리고 **B'** 등을 모두 구할 수 있게 된다.

2.2 기준평면과 매개변수 공간

3차원 상의 기준평면이 2차원 화면으로 투영된 평면을 '화면의 기준평면'이라 하자. 2차원의 화면 좌표계와 3차원의 실제 좌표계에 각각 기준평면이 있을 때 두 기준평면을 매개변수 공간으로 매핑하면 두 기준평면의 대응관계를 유도해 낼 수 있다. 화면의 기준평면은 실제로 화면의 점(화소) 필드이며 색이 요소인 2차원 배열이다. 즉, 화면 상의 모든 점은 3차원 상의 기준평면 위의 한 점이 투영된 것이라고 볼 수 있고 화면 상의 경계 상자의 각 꼭지점의 좌표는 모델 경계상자의 꼭지점의 실제 좌표가 투영 및 윈도우-뷰포트 변환된 것이다[9,10].

예를 들면 공간 상의 기준평면의 식이 z=1일때 화면 상의 임의의 점이 나타내는 실제 3차원 공간의 좌표는 $(x, y, 1)$이고, x, y값은 화면의 기준평면과 매개변수 공간 그리고 매개변수 공간과 실제 3차원 공간 상의 기준평면의 대응관계에 의해 산출할 수 있다. Fig. 5의 (a)에 있는 모델 경계상자의 윗면에 있는 화면 상의 점 **P**는 공간 상의 실제 좌표에서도 모델 경계상자의 윗면 내부에 있지만 화면의 점 **A1**과 **A2**은 공간 상에서 모델 경계상자의 윗면 외부에 존재하게 된다. 이때 화면의 모든 화소를 Fig. 5의 (b)처럼 **P0**를 시작점으로 하는 벡터 **S**와 **T**를 이용하여 매개변수 공간으로 대응시킬 수 있다. 각 방향을 u, v라는 매개변수로 표현하면 u, v값이 0과1 사이인 경우는 공간 상에서도 윗면의 내부가 되고 그 외의 경우는 기준평면에 있지만 모델 경계상자의 윗면 밖에 있는 외부 점이다.

Fig. 5의 (a)와 같이 모델 경계상자의 윗면을 포함하는 기준 평면이 있고 화면 상에서 경계 상자 윗면의 꼭지점이 **P0**, **P1**, **P2**, **P3**일 때 화면에 있는 임의의 점 **P**의 매개변수 u, v의 값을 구해보자. 화면 좌표계에서 시작점과 끝점이 **P0**(x0, y0), **P1**(x1, y1)인 벡터와 **P0**(x0, y0), **P3**(x3, y3)인 벡터 **S**, **T**를 구할 수 있고 Fig. 5의 (c)와 같이 각 벡터의 길이 만큼을 1로



(a) (b) (c) (d)

Fig. 4. Model and bounding box of model (a) Model and Bounding box of model on screen. (b) Cutting model. (c) Bounding box of model.

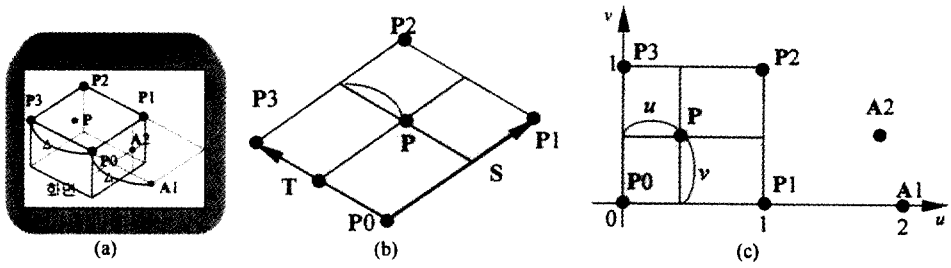


Fig. 5. Parametric space and base plane of screen (a) Viewing screen, (b) Base plane of screen, (c) Parametric space.

하는 매개변수 u, v의 공간을 생각할 수 있다.

두 벡터 S, T를 이용하여 화면의 기준 평면을 매개변수 u, v로 표현하면 식 (1)과 같다.

$$\begin{aligned} \mathbf{P} &= \mathbf{P0} + u\,(\mathbf{P1}-\mathbf{P0}) + v\,(\mathbf{P3}-\mathbf{P0}) \\ &= \mathbf{P0} + u\mathbf{S} + v\mathbf{T} \end{aligned} \tag{1}$$

화면에 있는 상의 임의의 점 P에 대응하는 매개변수 u, v의 값은 식 (1)을 이용하여 연립방정식을 세워 풀면 식 (2)와 같이 구해진다.

$$\begin{aligned} u &= ((x3-x0)(y-y0)-(x-x0)(y3-y0))/ \\ &\quad ((y1-y0)(x3-x0)-(x1-x0)(y3-y0)) \\ v &= ((x1-x0)(y-y0)-(x-x0)(y1-y0))/ \\ &\quad ((y3-y0)(x1-x0)-(x3-x0)(y1-y0)) \end{aligned} \tag{2}$$

식 (2)에 의해 매개변수가 구해지면 화면 상에 있는 점 P에 해당하는 실제 공간에 있는 기준평면의 점의 좌표를 구할 수 있다. 점 P가 경계 상자의 윗면에 있으면 매개변수 u와 v는 모두 0과 1사이의 값이 되는데 점 A1과 A2는 각각 u=0, v=-1과 u=0, $-1 < v < 0$의 값을 가지게 될 것이다. 즉, A1과 A2에 대응하는 공간 상의 점은 v가 0과 1사이가 아니므로 모델 경계상자의 표면 밖의 점이라고 판단할 수 있다. 모델 경계상자의 옆면 표면에 있는 점 A2는 경계 상자의 윗면을 이용하여 기준평면을 정하면 표면 외부의 점이지만 옆면을 이용하여 기준평면을 정하면 표면 내부의 점이다. 이처럼 화면에 있는 기준평면에서 임의의 좌표에 해당하는 매개변수를 구하면 실제 공간 상에서 이 점에 대응하는 좌표를 얻을 수 있다. 화면의 임의의 점은 모델 경계상자의 어느 면을 이용하여 기준평면을 결정하느냐에 따라 모델 경계상자 표면의 내부 또는 외부로 판단될 수 있다.

기준평면을 하나만 설정하고 교점을 구하여 실수 범위의 매개변수 u, v를 구할 수도 있고 기준평면을 모델 경계상자의 각 여섯 면을 모두 고려하여 화면에 보이는 면을 알아낸 다음 그 면과 교점을 구하여 그 교점에 해당하는 0부터 1까지의 값만 가지는 매개변수 u, v를 구할 수도 있다. 화면에 보이는 경계상자의 면은 많아야 세 면으로 국한되는데 실제로 기준평면을 설정할 경우 한 면 또는 두 면만 보이는 경우는 계산의 간편과 속도를 위해 별도로 처리할 필요가 있다. Fig. 6은 기준평면을 여러 개로 하였을 때 시점의 위치에 따른 기준평면의 설정의 각 경우를 보인 것인데 빗금친 부분을 포함하는 평면이 기준평면이다. 이러한 기준평면의 설정 방법의 차이는 다음에 설명할 가시면을 판단할 때 계산 방법과 계산량의 차이를 가져오게 된다.

2.3 매개변수와 3차원 기준평면 상의 좌표

화면의 기준평면에 의해 매개변수가 구해지면 화면 상의 점이 실제로 3차원 상의 기준평면에서 어느 위치에 있는지는 모델 경계상자의 각 모서리에 매개변수를 곱하기만 하면 쉽게 계산 할 수 있다. 예를 들어 모델 경계상자의 이웃하지 않고 마주보는 두 꼭지점 좌표가 (Pmin.x, Pmin.y, Pmin.z)과 (Pmax.x, Pmax.y, Pmax.z)일때 모델 경계상자의 길이, 폭, 높이인 Psiz.x, Psiz.y, Psiz.z는 식 (3)과 같다.

$$\begin{aligned} \mathrm{Psiz.x} &= \mathrm{Pmax.x} - \mathrm{Pmin.x} \\ \mathrm{Psiz.y} &= \mathrm{Pmax.x} - \mathrm{Pmin.y} \\ \mathrm{Psiz.z} &= \mathrm{Pmax.x} - \mathrm{Pmin.z} \end{aligned} \tag{3}$$

Fig. 7과 같이 모델을 시선 벡터 V(Vx, Vy, Vz)로 바라볼 때 공간 상의 점 A와 B를 기준평면을 이용하여 구해볼 수 있다. 점 A와 B의 z좌표는 각각 Pmax.z와 Pmin.z이다. 앞서 설명한대로 화면 상의 기준평면을 이용하여 구한 매개변수 u, v를 이용하면 A의 좌표는 식 (4)와 같다.

$$\begin{aligned} \mathrm{Ax} &= \mathrm{Pmin.x} + u \cdot \mathrm{Psiz.x} \\ \mathrm{Ay} &= \mathrm{Pmin.y} + v \cdot \mathrm{Psiz.y} \\ \mathrm{Az} &= \mathrm{Pmax.z} \end{aligned} \tag{4}$$

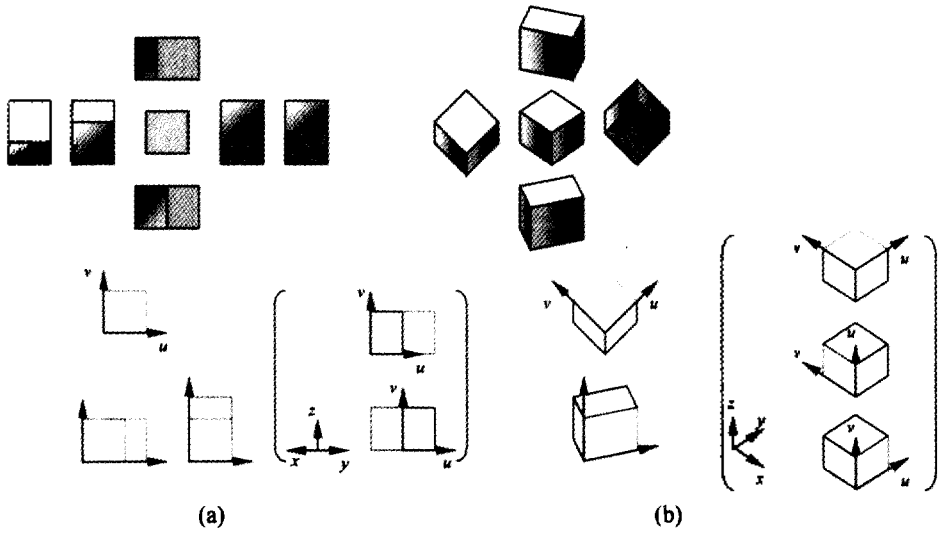


Fig. 6. Choosing the base plane. (a) One or two visible faces on screen, (b) Three visible faces on screen.

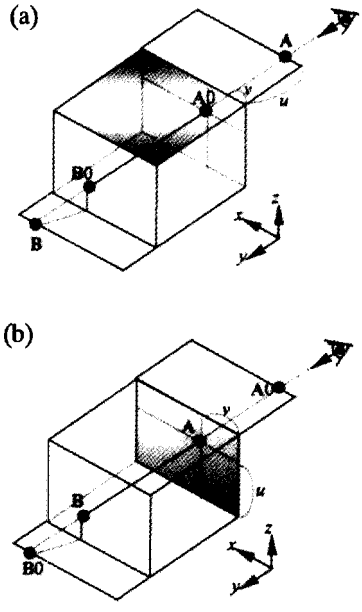


Fig. 7. u, v Parameters depending on base planes (a) Base plane using top face of bounding box, (b) Base plane using side face of bounding box.

방향벡터가 V이고 A를 지나는 직선을 시선벡터의 연장선이라 하자. z=Pmin.z인 평면과 만나는 점 B는 이 연장선 직선의 방정식과 기준평면의 평면의 방정식을 연립하면 식 (5)와 같이 구할 수 있다.

$$Bx = Ax + d \cdot Vx$$
$$By = Ay + d \cdot Vy \qquad (\text{단, } d = Psiz.z / Vz) \qquad (5)$$

$$Bz = Pmin.z$$

2.4 기준평면에 의한 가시면 판단(visible surface check)

일반적인 가시면 판단 방법은 대부분 시점에서 바라보는 시선 벡터의 연장선과 부딪히는 모든 면을 탐색하며 가장 가까운 면을 가시면으로 판단한다. 곡면의 경우는 적당한 삼각형들로 근사화하여 삼각형 평면과 직선의 교점을 찾는데 모든 삼각형을 검색하는 것이 일반적이다. 그러나, 제안된 방법에서는 화면의 화소에 대응하는 기준평면 상의 실제 좌표를 구한 다음 이 좌표의 점이 시선 벡터 방향으로 진행하며 부딪히는 절삭 모델의 표면의 좌표를 구한다. 즉, Z-map을 형상 Map으로 하는 경우 시선 벡터의 연장선을 xy 평면에 투영하여 연장선이 지나는 격자에 대해서만 비교를 수행한다. 이 방법은 매우 빠르고 구현 방법이 간단하다. 또 쉐이딩을 하기위해서는 가시면의 법선 벡터가 필요한데 가시면으로 판단되는 경우에만 그 격자를 삼각형화하여 법선벡터를 구하면 된다. 이러한 가시면 판단 방법을 Table 1에서 서술한 알고리즘과 비교하면 다음과 같다.

제안된 가시면 판단 방법
for (갱신 영역의 모든 (x,y)화소) {
  for (투영된 시선 벡터가 지나는 격자) {
    Z-map 경우: 해당 격자의 z값과 시선의 높이 h를 비교

M-map경우: 해당 격자의 상하 두 z값과 시선의 높이 h를 비교

}

}

화면의 한 점(화소)이 가시면인지 확인하려면 그 화면의 점에 해당하는 두 점 **A**와 **B**를 이용한다(Fig. 8참조). **A**와 **B**는 식 (4)와 식 (5)로 구할 수 있다. Z-map의 z값을 길이로 하고 *xy* 평면에 수직이고 모델 경계상자 바닥을 시작점으로 하는 벡터를 Z-map에서 각 격자의 높이 벡터가 **h**일때 형상 Map이 가지고 있는 그 격자의 높이벡터 **z**와 **h**를 비교하면 가시면을 쉽게 찾을 수 있다. **A**점과 **B**점을 이은 시선벡터의 연장선과 이 연장선을 포함하면서 *xy* 평면에 수직인 평면을 생각하자. 이 평면과 *xy* 평면과의 교선은 시선벡터 연장선의 높이 벡터가 *xy* 평면에 수직으로 투영된 직선과 일치하는데 이 직선이 지나는 격자는 격자의 인덱스가 정수이므로 Raster Graphic에서 직선을 그릴 때 사용하는 Brensenham Line Drawing Algorithm이나 Line DDA 방법[6]을 이용하면 각 격자의 인덱스(gx(*i*), gy(*i*))는 다음과 같은 과정으로 구할 수 있다.

Step 1. Get_Two_Point_By_Base_Plane_of_Model_Box(↓n1, ↓n2, ↑**A**, ↑**B**)

S : 직선의 시작점 **S**=(Ax, Ay)(기준 평면의 점 A가 *xy* 평면에 투영된 점)

E : 직선의 끝점 **E**=(Bx, By)(기준 평면의 점 B가 *xy* 평면에 투영된 점)

i : 격자의 인덱스

n : 격자의 개수

(gx(*i*), gy(*i*)): 격자의 인덱스 리스트(단, *i*는 격자의 인덱스이며 *i*=0, ···, n)

Step 2. Get_Grid_Index_List_By_Line_Drawing_Algorithm(↓**S**, ↓**E**, ↑*n*, ↑gx, ↑gy)

Step 3. Calculate **h***i*

Case Z-map: Get **z***i*

if (**h***i* < **z***i*) then visible = TRUE: goto step 5

Case M-map: Get **z_low** = **z***ij*, **z_high** = **z***i,j*+1 **for** all *j* of each *i*

if (**z_low** < **h***i* < **z_high**) then visible = TRUE: goto step 5

Step 4. Increase (*i*)

If (*i* < =*n*) then goto step 3

else visible = FALSE

Step 5. Return (↓visible, ↓gx(*i*), ↓gy(*i*))

Fig. 8의 (a)와 (b)에서 *i*번째 시선벡터의 연장선의 높이 벡터 **h***i*를 보이고 있는데 색칠된 단면은 시선벡터를 포함하면서 *xy* 평면에 수직인 평면으로 모델 경계상자를 자른 단면을 의미한다. Fig. 8의 (c)는 **h***i*를 구하는 방법인데 투영된 시선 벡터의 연장선의 높이 벡터는 Fig. 9와 같이 단면에 생기는 삼각형은 닮은꼴이므로 격자의 높이 *h*는

$$H = H \cdot l / L \qquad (\because L : H = l : h) \tag{6}$$

이다.

Fig. 9의 (a)와 (b)는 형상 Map이 가지고 있는 높이 벡터 **z***i*를 보이고 있다. 색칠된 단면은 앞서 설명한 모델 경계상자를 잘랐던 평면으로 NC 절삭 모델을 자른 단면이다. Fig. 9의 (c)는 형상 Map이 M-map인 경우인데 이 경우는 높이 벡터의 시작점이 하나의 격자에서 여러개가 있을 수 있는데 한 격자에 대해서 내부가 여러개 있을 수 있기 때문이다.

Fig. 9의 (a)에서 보이듯 시선벡터가 처음으로 절삭 모델의 내부로 들어갈때의 격자를 가시 격자(visible grid)라 하자. 이 가시격자 *i*의 인덱스가 (gx(*i*), gy(*i*))일때 그 격자 부분의 절삭 모델 높이는

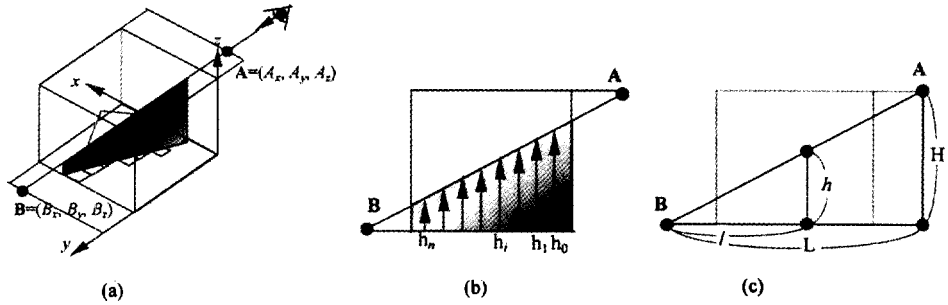


Fig. 8. Heights of extended viewing vector (a) Section plane by viewing vector (Shaded plane), (b) Cutaway view of the section, (c) Geometry involved in intersection points of base planes and extended viewing vector.

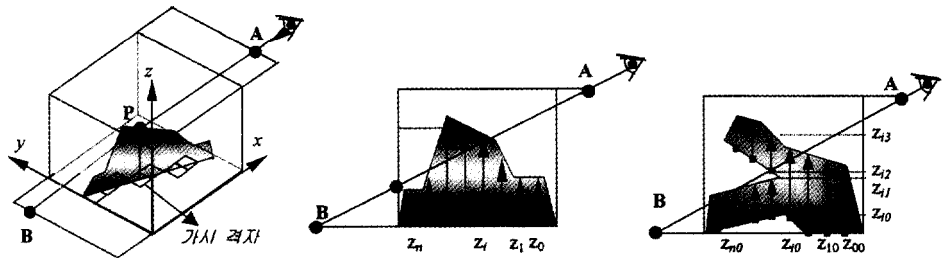


Fig. 9. Cutaway views of model map and heights of extended viewing vector.

Map((gx(i), gy(i))이다. 따라서 Fig. 9에서 보이는 가시면에 해당하는 점 **P**의 좌표는 식 (7)과 같다.

$$\mathbf{P} = (\text{pitch\_x} \cdot gx(i),\ \text{pitch\_y} \cdot gy(i),\ \text{Map}(gx(i), gy(i))) \tag{7}$$

주위의 격자의 형상 Map 상의 높이를 이용하면 삼각형의 세 꼭지점의 좌표 **P1**, **P2**, **P3**를 결정할 수 있고 가시면에 대한 법선벡터 **n**도 구할 수 있다(식 (8)).

$$\mathbf{n} = (\mathbf{P1} - \mathbf{P}) \times (\mathbf{P2} - \mathbf{P}) / |(\mathbf{P1} - \mathbf{P}) \times (\mathbf{P2} - \mathbf{P})| \tag{8}$$

단, $\mathbf{P} = (\text{pitch\_x} \cdot gx(i),\ \text{pitch\_y} \cdot gy(i),\ \text{Map}(gx(i), gy(i)))$

$\mathbf{P1} = (\text{pitch\_x} \cdot (gx(i)+1),\ \text{pitch\_y} \cdot gy(i)),\ \text{Map}(gx(i)+1, gy(i)))$

$\mathbf{P2} = (\text{pitch\_x} \cdot gx(i),\ \text{pitch\_y} \cdot (gy(i)+1),\ \text{Map}(gx(i), gy(i)+1))$

2.5 쉐이딩(shading)

화면에 그려져야할 영역의 모든 화소에 대해 가시면을 찾고 그 가시면(삼각형)의 법선 벡터와 광원 위치 벡터 또는 광선 벡터와의 교각 및 가시면과 광원과의 거리 등을 이용하여 화소의 색을 결정하면[7,9] 실물을 보는 것처럼 쉐이딩된 형상을 볼 수 있다. 본 연구에서는 계산량을 줄이기위해 광원 벡터의 위치를 z축에 놓고 가시면과 광원과의 거리를 무시하면 가시면의 법선 벡터만으로 색을 결정하여 쉐이딩하였다.

이러한 모델 경계상자의 경계정보에 의한 가시성 판단 방법은 NC 절삭 모델처럼 단일 모델이 주어진 영역 내에서만 갱신이 이루어지는 경우에 Z-map이나 M-map과 잘 공조하게 되어있다. 화면의 갱신 영역을 알 수 있으면 이 영역의 색을 구하면 갱신할 이미지(image)가 된다. 이처럼 갱신 전의 이미지는 화면에 있고 갱신할 이미지가 메모리에 구해놓으면 더블 버퍼링(double buffering) 등의 방법으로 절삭 가공의 애니메이션 디스플레이를 쉽게 할 수 있다[7,9].

2.6 공구 경계상자에 의한 화면 갱신 영역의 산출

공구의 형상과 크기는 이미 알고 있으므로 절삭 모델의 모델 경계상자와 마찬가지로 공구가 차지하는 공구가 들어가는 최소 직육면체인 '공구 경계상자'도 정의할 수 있다. 공구 경계상자는 공구가 이동하며 발생되는 형상 Map의 갱신될 격자 영역에 대한 정보와 절삭 과정을 시뮬레이션할 경우 화면의 갱신 영역에 대한 정보를 제공하게 된다. 화면의 갱신은 이 영역에 대해서만 처리하면 된다.

이 영역을 구하기 위해서 Fig. 10의 (a)와 같이 정지해있는 공구의 경계상자를 화면에 투영하면 공간상의 8개의 꼭지점이 투영된 8개의 2차원 좌표를 모두 포함하는 외곽의 점 S_{min}과 S_{max}을 구한다. NC 절삭 가공은 주로 공구가 직선 보간 이동으로 이루어져 있는데 이러한 경우 시작 위치와 끝 위치에 의한 공간 상의 두 개의 공구 경계상자를 생각할 수 있다. 이 두 상자의 16개의 모서리를 화면에 투영한 16개의 2차원 좌표를 포함하는 외곽의 점 S_{min}과 S_{max}를 구하면 이 두 점을 대각선으로하는 직사각형이 공구 이동 궤적에 대한 최소로 갱신해야 할 화면의 사각형 영역이 된다. 이 과정은 시점이 고정된 상태에서 계산이 이루어지기 때문에 좌표값 외에는 모두 상수이며 간단한 사칙연산 만으로 구해질 수 있다. Fig. 10의 (c)는 공구 이동에 따른 화면 영역을 보인 것이다. 이 영역 내 화소에 대해서만 가시성을 판단함으로써 계산량과 그리는 시간을 줄일 수 있다.

3. 형상 Map 데이터의 갱신

공구가 NC 코드에 의해 이동하는 경로는 직선과 원호가 있는데 원호 경로를 직선 보간하면 공구의

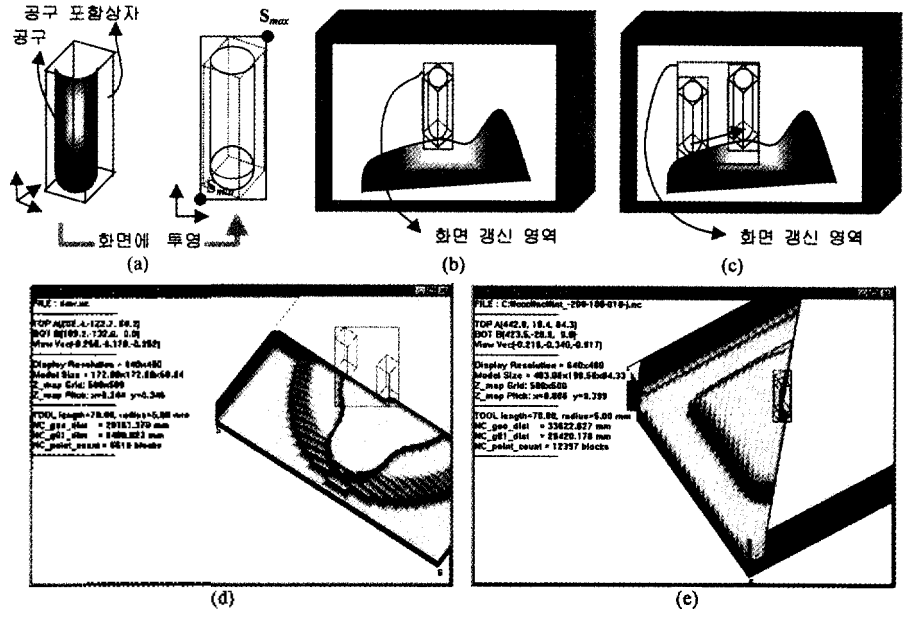


Fig. 10. Determination the updating area of screen (a) Bounding box of a tool, (b)(c) Updating area using bounding box, (d)(e) Graphic demonstrations with displaying the updating area by a tool path.

모든 이동 경로는 직선 보간 이동으로 나타낼 수 있다. 공구 이동 궤적을 구하면 형상 데이터 Map을 갱신하여야 한다. 공구가 공간 상에서 이동할 때 xy 평면에 수직으로 투영될 때 해당되는 격자의 인덱스를 구하거나 격자에 해당하는 공구 바닥의 높이를 구하는 방법은 두가지를 생각해 볼 수 있다. 격자 간격에 비례하여 직선 구간을 세분하고 공구의 바닥 형상을 반영하여 형상 Map을 변경하는 경우와 공구의 이동 방향에 의한 실루엣 곡선을 구하고 실루엣 곡선이 만드는 공구 이동 궤적인 룰드 곡면을 구하여 형상 Map에 반영하는 것이다[4].

3.1 세분된 직선 구간에 의한 공구 이동 궤적

한 격자의 인덱스를 공구의 중심으로 하고 공구의 크기 만큼의 주위 격자의 z값을 갱신하면 정지된 공구가 차지하는 영역을 형상 Map에 반영할 수 있다. 공구 이동의 시작과 끝 점을 xy 평면에 투영하고 그 좌표를 격자의 인덱스로 변환한 후 두 인덱스 좌표가 만드는 직선의 자취가 지나는 격자가 세분된 직선 구간의 공구 중심이 된다.

Z-map의 격자의 간격 때문에 발생하는 오차와는 별도로 공구의 위치 좌표를 정수화된 격자로 맞추기 때문에 발생되는 오차가 있다. 이 오차는 공구의 실제 중심과 격자의 인덱스로 정수화된 중심의 물리적

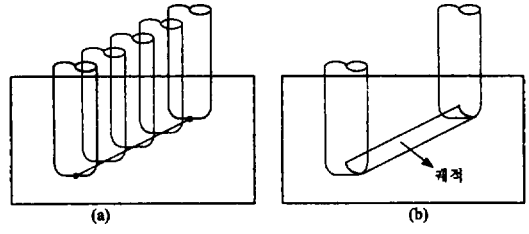


Fig. 11. Swept volume by linear movement of tool (a) Point-to-point method using line segment, (b) Ruled surface using silhouette curve.

인 차를 반영하여 선보간 방식으로 공구의 바닥 높이 값을 보정해 주면 된다. 격자의 간격이 충분히 작은 경우는 별 의미가 없고 모든 격자의 갱신에 적용되어야하므로 계산 시간이 늘어나게 된다. 실루엣 곡선에 의한 방법보다는 구현이 쉽고 계산도 매우 간단하다. 예를 들어 공구의 바닥 모양을 Z-map 형태로 가지고 있다가 형상 Map에서 공구의 이동하면서 차지하는 부분과 각 격자의 값을 비교하는 방법을 사용하면 속도도 빠르고 매우 간단하다.

3.2 실루엣 곡선에 의한 공구 이동 궤적

공구의 반경을 R, 공구의 원주를 따라 생긴 필렛의 반경을 r이라고 하고 공구의 진행이 xz 평면에서 이

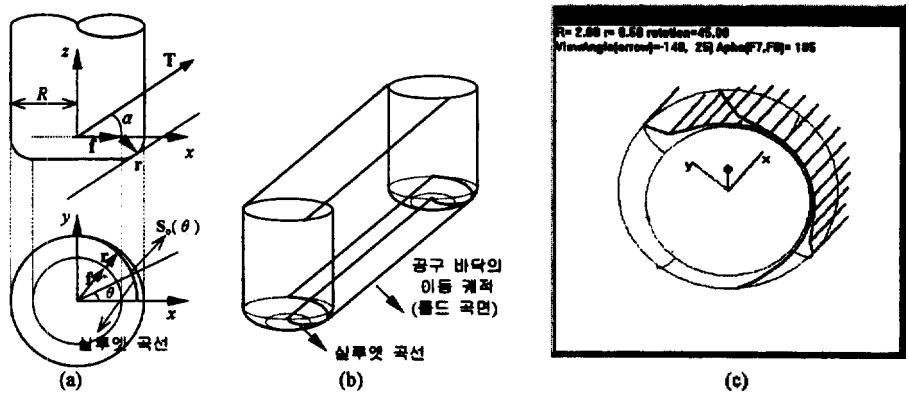


Fig. 12. The swept volume by a tool move. (a) Silhouette curves, (b) Ruled surface by sweeping the silhouette curves, (c) Approximation of silhouette curves.

루어진다고 할 때 x축과 이루는 각도를 α라 하자. xy 평면에서 원주의 방향과 x축과 이루는 각도를 θ라 하면 실루엣 곡선 $S_o(\theta)$는 식 (9)와 같이 주어진다[5].

$$S_o(\theta) = \mathbf{f} + \mathbf{r} = \begin{pmatrix} (R-r)\cos\theta + r_n\cos\theta \dfrac{\tan\alpha}{\sqrt{\cos^2\theta+\tan^2\alpha}} \\ (R-r)\sin\theta + r_n\sin\theta \dfrac{\tan\alpha}{\sqrt{\cos^2\theta+\tan^2 a}} \\ \dfrac{-r\cos\theta}{\sqrt{\cos^2\theta+\tan^2 a}} \end{pmatrix} \quad (9)$$

Fig. 12의 (a)는 필렛티드 엔드밀 공구가 xy 평면에 투영되었을때의 실루엣 곡선을 보인 것이다. 실루엣 곡선이 공구 이동 방향으로 직선 이동하며 만드는 곡면은 룰드곡면이며 공간 상에서 보면 Fig. 12의 (b)와 같다.

식 (9)에서 구한 구한 실루엣 곡선 $S_o(\theta)$는 공구의 이동방향 중에서 각도 α만 고려했기 때문에 xy 평면 상에서의 이동 방향을 고려해 주어야한다. xy 평면 상의 이동 방향 벡터와 x축이 이루는 각도가 ϕ라면 z축을 중심으로 그만큼 회전해야 모든 진행 방향을 감안한 곡선이 된다(식 (10)).

$$S(\theta) = S_o(\theta) Rot(z, \phi) \quad (10)$$

식 (10)에서 구해진 공간 상의 실루엣 곡선 $S(\theta)$를 이용하여 Z-map 구조의 형상 데이터를 갱신하기 위해서는 Z-map의 갱신할 영역의 격자의 인덱스와 그곳의 z값을 알아내야 한다. $S(\theta)$를 $A(A_x, A_y, A_z)$와 $B(B_x, B_y, B_z)$로 옮겨 놓으면 두개의 공간 곡선이 된다. 여기서 구하고자 하는 공구의 궤적은 두 공간 곡선을 직선으로 연결하는 룰드 곡면(ruled surface)이다. 룰드 곡면이 생성되면 이 곡면을 xy 평면 상에 투영했을 때 차지하는 영역과 겹치는 Z-map의 격자

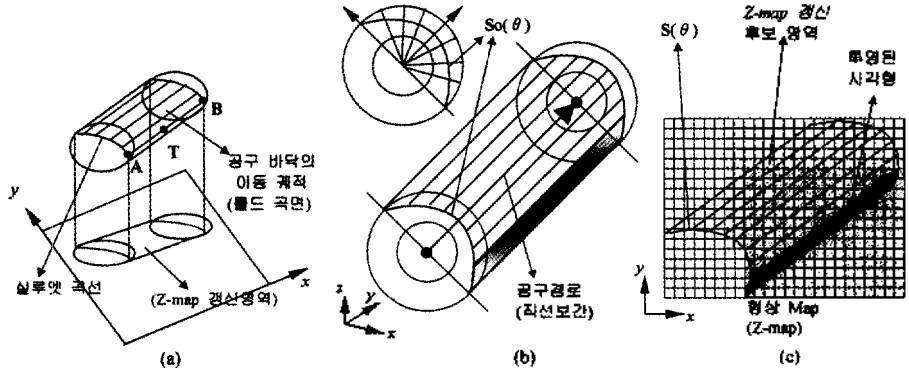


Fig. 13. Swept volume and updating area of Z-map (a) Projecting the ruled surface, (b) Ruled surface by silhouette curve, (c) Updating area of Z-map.

들이 갱신되어야 할 후보 격자이며 공구의 높이 보다 높은 z값은 갱신되게 된다.

예를 들어 Fig. 13의 (a)와 같은 공간 상의 공구 이동은 실루엣 곡선을 분할한 각도에 따라 몇 개의 사각형으로 근사화 할 수 있다. Fig. 13의 (b)는 그 예인데 이러한 공간 상의 사각형을 *xy* 평면 상에 투영하여 Fig. 13의 (c)와 같이 Z-map의 격자 위에 놓으면 사각 영역의 갱신될 후보 격자 영역을 알 수 있다. 이 중에서 투영된 사각형 내부에 포함되는 격자에 대해 z값을 갱신하면 되는데 z값은 공구가 직선 보간 이동이므로 시작점과 끝점의 두 z값을 이용하여 선형 보간하면 된다.

이처럼 공구의 형상에 의한 갱신되어야 할 Z-map의 각 격자의 z값을 계산하는 방법으로 미배개변수식에 의한 방법이 소개된 바 있는데 공구 바닥면의 정의하고 격자의 인덱스만으로 z값을 구할 수 있다[2]. 이 방법은 일반적인 공구 형상의 이동 궤적면의 모델링이 가능하고 삼각함수를 사용하지 않으면서 계산량이 적어 빠르고 매우 정확하게 Z-map을 갱신할 수 있다.

4. 적용 및 평가

제안된 방법을 Pentium 166 Mhz과 32 M의 메모리를 장착한 IBM-PC에서 Window 95/NT 운영체제에서 C언어로 구현하였다. Fig. 14는 화면 윈도우를 640*480의 해상도(resolution)로 하고 전체 블록이 5,610개이며 총 경로 길이가 30,249 mm인 NC 데이터를 가지고 모의 가공한 결과이다. Fig. 14의 (a)는 초기 상태인데 여기서는 모델 경계상자와 모양이 일치하지만 직전에 가공한 모델을 Z-map으로 변환하여 사용할 수 있다. Fig. 14의 (b)와 (c)는 가공 과정과 결과 화면이다. 가공 과정의 디스플레이는 시선벡터가 V=(-0.1111, -0.002, -0.994)일때 262초가 소요되었다. 그러나 화면에 가공 과정을 꼭 볼 필요가 없을 때는 가공 과정의 디스플레이를 갱신하지 않고 내부적으로 Z-map을 모두 갱신한 후 가공 결과 형상만 디스플레이하면 되는데 이 과정은 13+6=19초만 소요되었다(Table 2).

Table 2에서 Z-map을 갱신하며 디스플레이하는 경우 시선 벡터에 따라 그래픽 시뮬레이션하는 시간이 2~4배 정도 차이나는 것을 볼 수 있다. 이 것은 모델 경계상자의 윗면과 바닥면 만을 이용하면 기준 평면을 설정하면 시선 벡터가 *xy* 평면과 이루는 각이 작을수록 시선벡터의 연장선은 길어지 때문이다. 시선벡터의 연장선을 *xy* 평면에 투영하여 정수화된

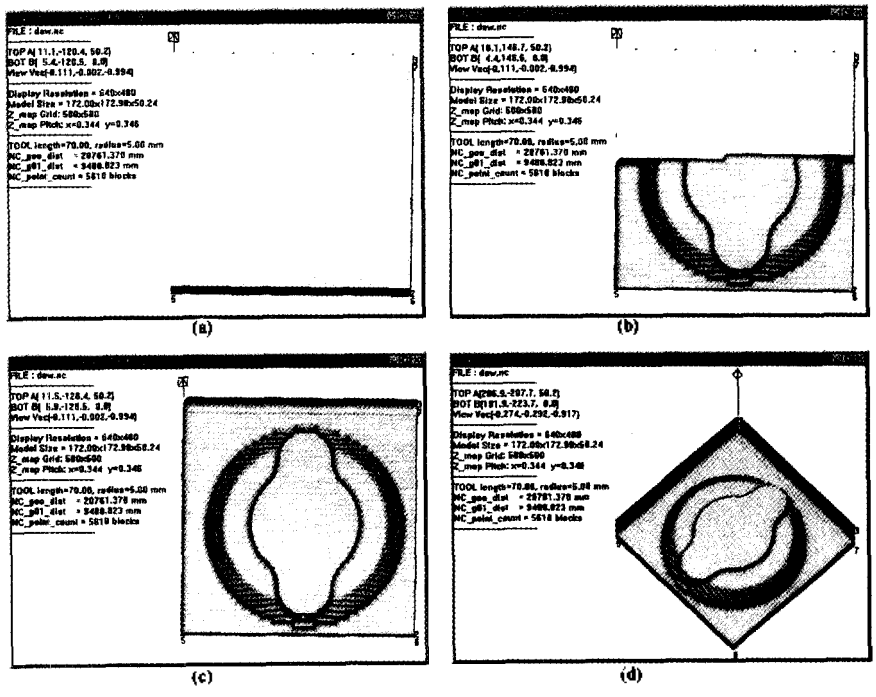


Fig. 14. Graphic simulation of material removal process by proposed method (a) Initial model: Raw-stock, (b) Updating model: Cutting process, (c) Finished model, (d) Another viewpoint.

격자의 인덱스를 구하기 때문에 연장성이 길수록 가시성을 비교하는 횟수가 늘어 계산량은 증가하는 것이다. 모델 경계상자의 여섯 면을 모두 기준평면으로 이용하면 편차는 줄어든다. 그러나 가시면 탐색을 위한 계산횟수는 기준 평면과 가시면과의 거리에 비례하기 때문에 계산시간의 편차는 존재한다.

Table 2. Result of testing the proposed method (Tool Radius=10 mm, Z-map=500×500 grids, Resolution=640×480 pixels)

Model Name	Raw Stock Size (w*/*h)	데이터 Point 수	실제 가공거리	Z-map 갱신	시선 벡터 (View Vector)	Z-map 갱신 후 Display 시간	Z-map 갱신과 동시 Display
Daw	172*172*34 mm	5,610개	9,488 mm	13 sec	(-0.111, -0.002, -0.994)	6 sec	262 sec
					(-0.274, -0.292, -0.917)	7 sec	534 sec
Speaker	300*420*50 mm	8,120개	53,791 mm	18 sec	(-0.006, -0.111, -0.994)	8 sec	125 sec
					(-0.210, -0.340, -0.917)	13 sec	582 sec
Pan	130*130*24 mm	14,609개	30,102 mm	76 sec	(-0.111, -0.013, -0.994)	23 sec	1350 sec

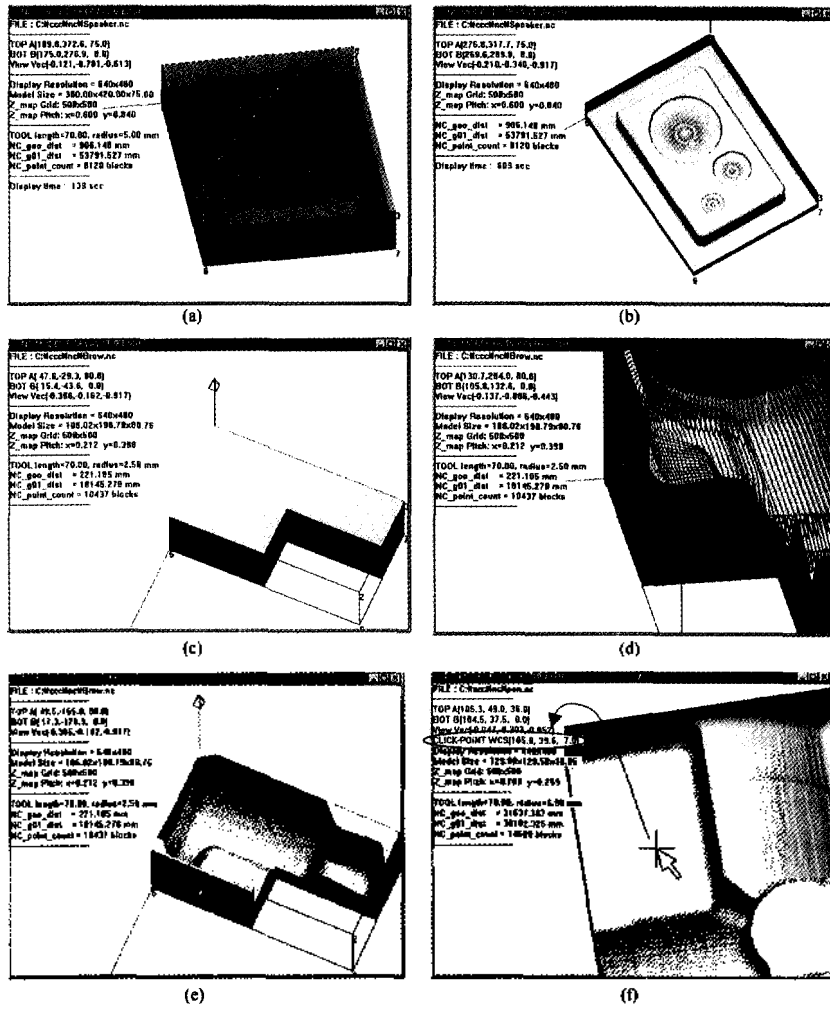


Fig. 15. Additional function of graphic simulation using proposed method (a) Tool path, (b) Result of virtual machining, (c) Cutaway view of raw stock, (d)(e) Finished model: Cutaway view, (f) Check the coordinates of pointing position.

제안된 가시면 판단 방법은 점 샘플링 방식이므로 화면상에서 모델이 차지하는 영역이 클수록 공구가 절삭해서 갱신될 화면 영역이 커져 디스플레이 속도가 떨어지게 된다. 실험 결과 이러한 요인에 비해 Z-map 격자의 개수는 디스플레이 속도에 큰 영향을 미치지 않았다. 따라서 격자의 간격을 줄여 고정도의 모의 가공을 수행하더라도 가공과정의 화면 디스플레이 시간이 큰 영향을 받지 않는다.

Fig 15의 (a)는 가공 경로를 출력한 것이고 (b)는 이 NC 데이터를 가지고 모의 가공한 결과이다. Fig. 15의 (c), (d), (e)처럼 실제 가공에서는 힘든 단면 형상도 모의 가공에서는 쉽게 설정해 볼 수 있다. 또, 제안된 가시면 판단 과정에서 화면의 특정 좌표에 해당하는 실제 3차원 좌표를 구하는 방법을 이용하면 Fig. 15의 (f)와 같이 마우스 등으로 포인팅한 곳의 좌표를 계산할 수 있으므로 절삭 깊이나 두 점 사이의 거리 등을 화면 상에서 알아 볼 수 있다.

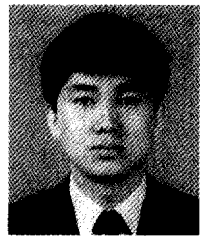
5. 결 론

본 논문에서는 NC 절삭 모델의 표현에 많이 사용되는 데이터 구조인 Z-map이나 M-map을 위한 효율적인 가시성 판단 방법과 형상 데이터의 갱신 방법을 이용하여 NC 가공의 절삭 시뮬레이션 과정을 리얼타임으로 쉐이딩하여 보여주는 방법을 제시하였다. 점 샘플링 방법이면서도 형상 데이터와의 비교 횟수를 최소화하면서 가시성을 판단하기 때문에 적은 계산량과 화면의 갱신영역이 보장된다. 또 모델을 포함하는 상자의 경계정보 외에는 별도의 메모리를 사용하지 않기 때문에 고성능의 컴퓨터시스템이 아니더라도 구현이 가능하다.

제안된 경계상자와 기준평면의 개념을 이용하면 화면에 디스플레이된 점의 좌표에서 실제 절삭 모델의 점의 좌표를 계산할 수 있으므로 절삭 깊이나 두 점간의 거리 등을 확인해 볼 수 있어 직관적인 NC 코드의 검증이 가능하다. 또한 실제로는 보기 힘든 단면 형상이나 여러가지 가공 방법 등을 같은 모델에 적용함으로서 물리적인 부재를 사용하는 실제 가공의 경우에 들어가는 물적, 시간적 공수의 낭비를 줄일 수 있다.

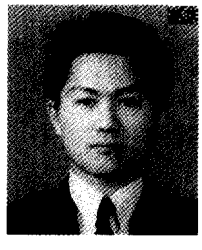
참고문헌

1. Hook, T.V., "Real-time Shaded NC Milling Display", *Proceeding of SIGGRAPH in computer graphics*, Vol. 20, No. 4, pp. 15-20, 1986.
2. Hsu, P.L. and Yang, W.T., "Realtime3D simulation of 3-axis milling using isometric projection", *Computer-Aided Design*, Vol. 25, No. 4, pp. 215-224, 1993.
3. Zeid, I., *CAD/CAM Theory & Practice*, McGraw-Hill, 1992.
4. 이철수, CAD/CAM, (주) 터보테크 출판부, 1997.
5. 박배용, 안정호, "커습 하이트의 기하학적 해석", 한국CAD/CAM학회 학술발표회 논문집, pp. 59-64, 1996.
6. 정연찬, "금형가공용 통합 CAM 시스템을 위한 가공 데이터의 모의 가공과 검증에 관한 연구", 한국과학 기술원 박사학위 논문, 1996.
7. Joy, K.I., Grant, C.W. and Max, N.L., and Lansing Hatfield, *Tutorial:Computer Graphics: IMAGE SYNTHESIS*, IEEE Computer Society Press, 1988.
8. Neider, J., Davis, T. and Woo, M., *OpenGL Programming Guide*, Addison-Wesley Publishing Company, 1993.
9. Hearn, D. and Baker, M.P., *Computer Graphics*, Prentice Hall, 1994.
10. Anand, V.B., *Computer Graphics and Geometric Modeling for Engineers*, John Wiley & Sons, Inc., 1993.



이 철 수

1984년 한양대학교 산업공학과 학사
1986년 한국과학기술원 산업공학과 석사
1990년 한국과학기술원 산업공학과 박사
1990년 ~ 현재 전남대학교 산업공학과 부교수
관심분야 : CAD/CAM과 CNC 콘트롤러



박 광 렬

1993년 전남대학교 산업공학과 학사
1997년 전남대학교 산업공학과 석사
1997년 ~ 현재 전남대학교 산업공학과 박사과정
관심분야 : CAD/CAM, NC