

## 그래픽 하드웨어를 이용한 NC 가공 검증의 고속화

김경범(국민대 자동차공학전문대학원), 이상현(국민대 자동차공학전문대학원),  
우윤환(국민대 자동차공학전문대학원)

### Fast NC Cutting Verification Using Graphic Hardware

G.B.Kim(Kookmin University), S.H.Lee(Kookmin University),  
Y.H.Woo(Kookmin University)

#### ABSTRACT

The z-map structure is widely used for NC tool path verification as it is very simple and fast in calculation of Boolean operations. However, if the number of the x-y grid points in a z-map is increased to enhance its accuracy, the computation time for NC verification increases rapidly. To reduce this computation time, we proposed a NC verification method using 3-D graphic acceleration hardwares. In this method, the z-map of the resultant workpiece machined by a NC program is obtained by rendering tool swept volumes along tool pathes and reading the depth buffer of the graphic card. The experimental results show that this hardware-based method is faster than the conventional software-based method.

**Key Words :** NC simulation(NC 모의 가공), Verification(검증), Z-map(Z-맵), OpenGL, HardWare(하드웨어), Ball-end mill(볼 엔드밀),

#### 1. 서론

NC 가공에서 허용 오차 범위를 넘는 과정삭이나 공구 간섭과 같은 NC 코드의 오류여부를 컴퓨터상에서 검증하고 그 속도를 향상시키기 위한 연구들은 널리 수행되어 왔으며<sup>(2)(3)(6)</sup> 실제 대부분의 상용 CAM 시스템들은 공작물이 실제 가공되는 상황을 실시간으로 화면상에 디스플레이하고 과정삭이나 미절삭등을 검증하여 주는 기능들을 구비하고 있다.<sup>(7)</sup> 하지만 실제 산업 현장에서 필요한 기능들은 실시간으로 NC 가공 시뮬레이션 상황을 확인하는 것 보다는 CAD 모델의 형상과 가공후의 형상과 비교를 하여 과정삭이나 미절삭을 판단하는 것이 일반적이다.

현재 가공중인 공작물의 표현방법중 하나인 Z-map 표현방법<sup>(1)</sup>은 일정한 간격의 (x, y)에 대한 z 값을 2 차원 배열 형태로 저장한 형태를 의미한다. 이 방법은 샘플링 간격이나 방식에 따라 표현의 정확도는 떨어지거나 불리안 작업을 신속하고 안정되게 수행할 수 있다는 장점으로 현재 상용 시뮬레이션 시스템의 대부분이 이방식을 택하고 있다. 하지만

이방식은 사용자가 정의한 공구가 일정한 격자상의 점을 지나는지를 판단하여 그때의 z 값을 공구위치의 값으로 치환하기 때문에 정밀도를 높이기 위해 격자의 간격을 조밀하게 하거나 NC data 가 방대할 경우 많은 계산을 필요로 한다.

이러한 문제점을 해결하기 위하여 본 연구에서는 3 차원 그래픽 하드웨어의 Z-버퍼(buffer)를 이용한 NC 프로그램 검증방법을 개발하였다.

#### 2. 시스템 개요

본 연구는 그래픽 하드웨어의 성능으로 화면상의 픽셀 정보 중 깊이 정보를 알아내어 현재 픽셀의 Z 값을 알아내는 방법으로 OpenGL 라이브러리로 구현하였다.<sup>(4)</sup> Fig. 1(a)는 Z-map 의 계산을 공구 궤적과 Z-map 과의 높이 비교로 계산하는 방법을, Fig.1(b)는 같이 주어진 공구궤적을 그대로 화면에 디스플레이하고 -Z 방향에서의 각 픽셀의 정보를 읽어 들임으로써 일정한 간격의 Z-map 정보를 얻는 방법을 나타낸다.

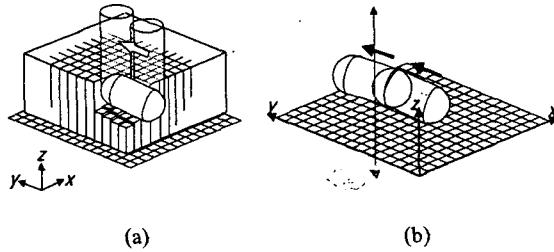


Fig. 1 Z-map 계산 방법 비교 (a)불리안 작업을 이용한 방법 (b)그래픽 하드웨어를 이용한 방법

### 3. 시스템 환경 설정

본 시스템을 사용하기 위해서는 윈도우와 실제 공작물이 그려질 drawing 영역, 그리고 drawing 영역에 매핑될 좌표축을 설정하는 작업이 필요하다.

Fig. 2에서 보이는 것과 같이 X 축 방향으로 NUM\_X\_PIXEL 만큼 Y 축 방향으로 NUM\_Y\_PIXEL 만큼의 크기로 window 를 생성하고 생성된 윈도우 안에 그려질 drawing 영역을 윈도우 크기 픽셀 개수로 일치 시킨다. 그리고 일정한 픽셀의 개수 만큼 좌표를 할당시켜 준다. 만약 (800, 800)픽셀만큼의 윈도우 크기에 왼쪽 하단을 (0, 0), 오른쪽 상단을 (800, 800) 좌표로 할당시켜주면 왼쪽 하단의 구석부분의 픽셀의 좌표는 Fig. 2 와 같이 (0.5, 0.5) 가 된다. 여기서 Z-MAP 의 간격의 좌표 값이 실제 픽셀의 중심의 좌표 값으로 일치시키기 위하여 한 개의 픽셀의 좌표상의 길이의 절반 값(여기서는 0.5)만큼 이동시켜서 픽셀의 위치정보 값을 Z-MAP 의 좌표간격과 일치시킨다. Fig. 3은 0.5 만큼 좌표의 원점을 이동시켜서 픽셀의 중앙위치정보가 실제 Z-MAP 의 간격좌표가 되도록 한 모습이다.

실제로 전체 윈도우의 픽셀 개수와 좌표범위와 비를 조절함으로써 Z-MAP 의 격자의 간격을 조절 할 수 있다. 그 방법으로 윈도우의 좌표범위를 (origin\_x, origin\_y)라 놓고 전체 픽셀의 개수와 좌표 범위의 비를 ortho\_n 이라 놓자.

$$\text{ortho\_n} = \text{NUM\_X\_PIXEL} / \text{origin\_x}$$

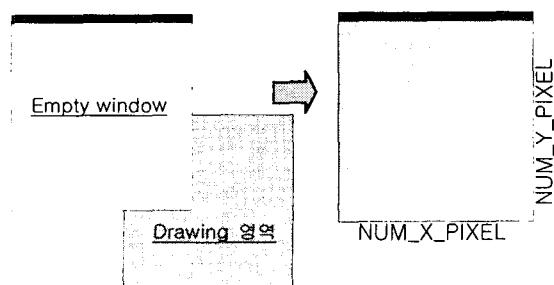


Fig. 2 Window의 크기와 drawing 영역 일치

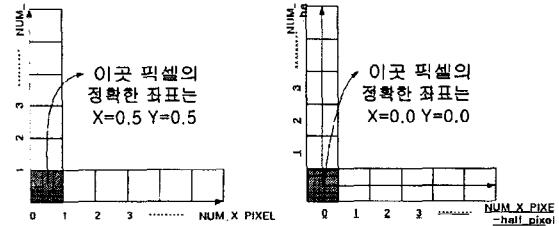


Fig. 3 픽셀중앙의 좌표 설정

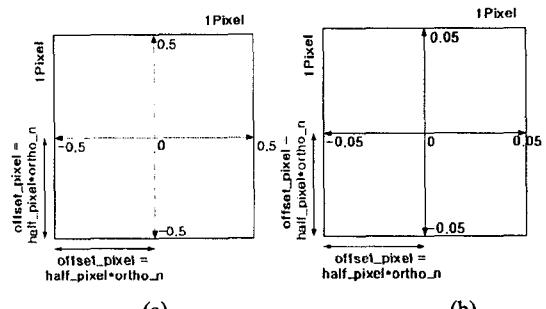


Fig. 4 픽셀당 좌표 설정

여기서  $\text{ortho\_n}$  은 결국 한 픽셀이 차지하는 좌표상의 픽셀 길이를 말한다. 그러므로  $\text{ortho\_n}$  의 값이 줄어들면 한 픽셀의 좌표상 길이가 적어지는 것이며 실제의 간격이 조밀해지는 것이다. 따라서 픽셀 중앙에 Z-MAP 의 간격 좌표를 맞추기 위한 이동 값도 좌표상의 픽셀 길이의 절반 값이 되어야 되므로  $0.5 * \text{ortho\_n}$  이 된다.

Fig. 4 는  $\text{ortho\_n}$  의 변화에 따른 이동 값의 변화를 보여주는 예로서 (a)는  $\text{origin\_x}$  의 길이가 100이고 픽셀의 x 방향 개수가 100 일 때  $\text{ortho\_n}$  이 1, 한

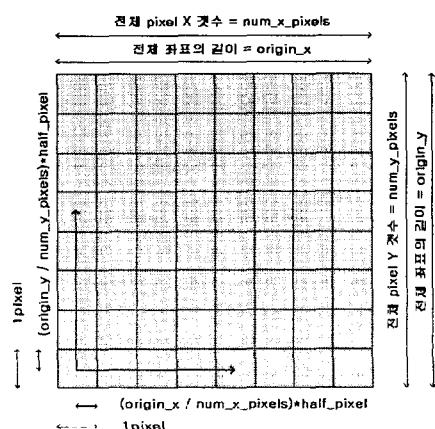


Fig. 5 픽셀과 좌표축의 설정

픽셀의 절반길이는 0.5, 이동 값은 0.5, origin\_x의 범위는 (-0.5, 99.5)이고 (b)는 origin\_x의 길이가 10이고 픽셀의 x 방향 개수가 100 일 때 ortho\_n이 0.1, 한 픽셀의 절반길이는 0.05, 이동 값은 0.05, origin\_x의 범위는 (-0.05, 9.95)이다.

Fig. 5에서 픽셀 개수와 좌표길이와의 관계를 정리하였다..

#### 4. 가공 검증

##### 4.1 공구 이동궤적 그리기

3 축 볼엔드밀 가공 시 NC 공구경로는 일정한 간격의 점들의 집합으로 되어있다.

이러한 NC 경로에 따라서 한 점에서 다음 점으로 가공되어질 때 시작점과 끝점에서 공구의 크기 만큼의 구를 생성하고 두 점의 사이의 각도를 구하여 그 각도에 따라 두 점 좌표상에 그려진 구와 접촉하는 실루엣곡선을 구한다. 그리고 얻어낸 두 실루엣 곡선을 일정한 간격으로 나누어서 두 개의 실루엣 라인을 삼각메쉬로 연결하는 공구 궤적을 만든다. 이러한 작업을 반복하여 공작물의 최종형상을 얻어낸다.

##### 4.2 픽셀의 Z-버퍼의 깊이 읽기

OpenGL의 라이브러리인 glReadPixel()함수로 픽셀의 depth 값을 얻으면 Fig. 6과 같이 눈의 위치에서 가까운 쪽(가장 앞쪽)의 depth 값이 0, 먼 쪽(가장 뒤쪽)의 depth 값이 1이 된다. 이때 각 픽셀의 depth 값은 실제 클리핑 볼륨의 z 방향 범위 view\_volume\_depth에 따라 실제 Z 좌표로 변환시켜 준다. View\_volume\_depth 값은 공작물의 크기에 따라 적절하게 설정하여준다.

예를 들어 depth 값이 0.5이면 좌표상으로는 0.0의 값을 가지고 있고 depth 값이 1이면 background의 depth 값을 가지고 있는 것이다.

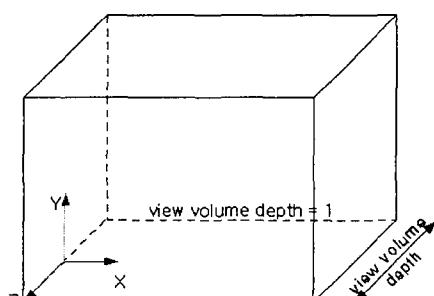


Fig. 6 클리핑 볼륨

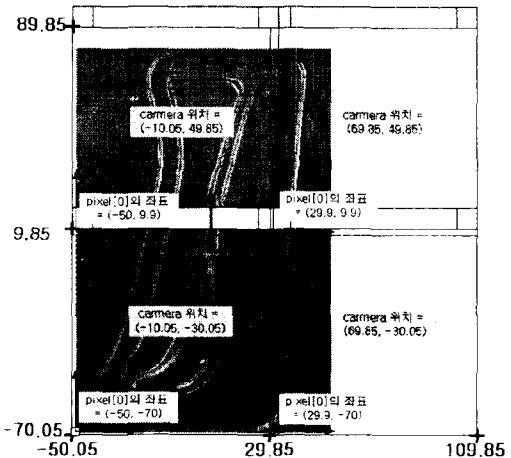


Fig. 7 공작물의 분할 처리

##### 4.3 공작물의 분할 처리법

모니터상의 픽셀의 수는 한정적이기 때문에 공작물의 크기가 크거나 정밀도를 높이기 위해 Z-MAP의 간격을 조밀하게 설정하였을 때는 현재의 윈도우 창 밖으로 공작물이 주어진 윈도우 안에 모두 표현할 수 없게 된다. 이와 같은 경우에는 Fig. 7과 같이 일정한 윈도우 크기로 공작물의 범위를 나눈 다음 glLookAt 함수를 이용하여 카메라의 eye position을 그 윈도우의 크기만큼 바꾸어 가면서 공작물을 나누어 읽는 방법으로 전체 공작물의 depth 값을 읽어 들인다. 이때 각 윈도우의 마지막 픽셀의 read depth 값은 정확성이 떨어지므로 다음 윈도우의 처음 픽셀과 겹치게 하여 읽는다. 그렇게 해서 읽은 각 픽셀의 z 좌표의 값을 순서대로 다음과 같은 구조체에 차례로 담는다.

```
typedef struct{
    int nx, ny;
    double start_x, end_x, xstep;
    double start_y, end_y, ystep;
    double **map;
} TOOL_MAP;
extern TOOL_MAP zmap;
```

여기서

$nx, ny = \text{read pixel}$  영역의 전체 픽셀수  
 $\text{start}_x, \text{start}_y = \text{read pixel}$  영역의 시작점 좌표  
 $\text{end}_x, \text{end}_y = \text{read pixel}$  영역의 끝점 좌표  
 $x\_step, y\_step = \text{픽셀 한 개의 좌표길이}$   
 $\text{map}[ny][nx] = \text{각 픽셀의 } z \text{ 값으로 좌측 하단}$   
 $\text{부터 시작해서 우측 상단 픽셀까지 차례로 저장}$

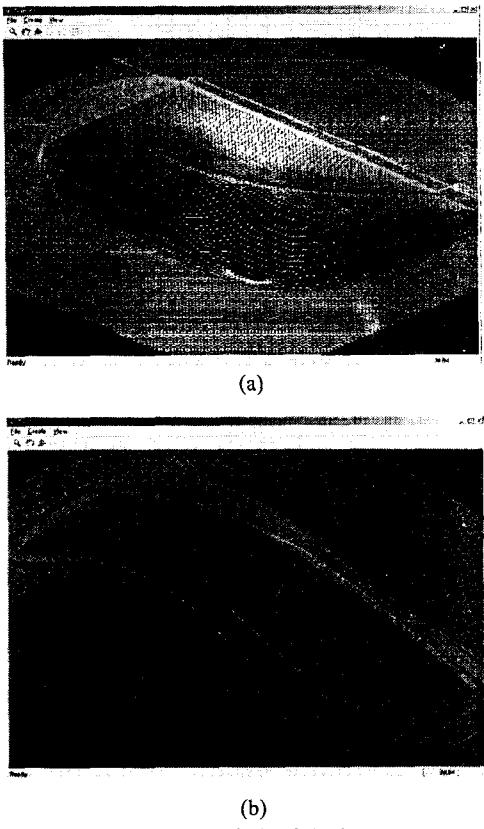


Fig. 8 실제 적용예

### 5. 적용예

Fig. 8(a), (b)는 상용 NC 경로 생성 프로그램에서 얻어낸 NC code 를 이용하여 가공형상을 Z-map 으로 표현한 실제 적용 예이다.

Table 1 은 Fig. 8(a)모델을 본 연구에서 제안된 방법으로 Z-map 을 계산한 시간과 불리안 작업을 통해 Z-map 을 계산한<sup>(3)</sup> 시간을 비교한 표이다.

NC code step	본시스템	직접계산방식
721830	64.687sec	254.281sec
209919	5.219sec	67.813sec
114905	2.969sec	43.156sec
43033	1.219sec	15.656sec

Table 1 Z-map 계산 소요시간 비교

본 프로그램은 CPU P4-1G Hz PC 에서 각각 측정을 하였다.

### 6. 토의

검증 처리 시간의 대부분은 깊이 값을 얻기 위

해 화면에 display 하는 랜더링 과정이 대부분을 차지한다. Z-map 간격을 줄게 하거나 공작물의 크기가 커서 여러 번 나누어서 화면에 display 를 하게 되면 검증 시간은 더욱 많이 소요된다. 이 문제는 OpenGL 의 랜더링 속도를 빠르게 처리하는 알고리즘을 이용하거나 공작물의 범위를 지정함으로서 그 시간을 줄이면 더욱 효과적인 검증 처리를 할 수 있다.

### 7. 결론

그래픽 하드웨어를 이용한 알고리즘은 소프트웨어적인 방법에 비해 빠른 시간 안에 가공 형상을 표현함으로써 빠른 시간에 공작물의 과정이나 미결삭의 신속한 확인에 이용될 수 있으며 나아가 금형의 pattern 가공이나 공구경로 생성에도 응용될 수 있다.

### 참고문헌

1. Choi, B.K., Robert., B and Jerard, R.B., Sculptured Surface Machining, Kluwer Academic Publishers, 1998.
2. 주성욱, 이상현, 박기현, “실시간 3축 NC 밀링 시뮬레이션을 위한 메쉬 간략화 방법,” 한국 CAD/CAM 학회 논문집, 제 5 권 제 4 호 pp347-358
3. 최병규, 전차수, 유우식, 편영식, “CAD/CAM 시스템과 CNC 절삭가공,” 회중당
4. Richard S. Wright Jr Michael Sweet, “OpenGL Super Bible”, fone
5. Hugues Hoppe, “Progressive meshes”, SIGGRAPH 96 Conference Proceeding, pp.99-108, 1996
6. Kevin J. Renze and James H. Oliver, “Generalized unstructured decimation”, IEEE Computer Graphics & Applications, Vol.16 No.6, pp. 24-32, November 1996
7. “Z-master 2000 Reference Manual”, CUBICTEK Co., Ltd