1074 大韓機械學會論文集 第17卷 第5號, pp. 1074~1083, 1993.

〈論 文〉

NC파트프로그램의 검증 및 오류 수정에 관한 연구

김찬봉* · 박세형** · 양민양***

(1992년 9월 9일 접수)

A Study on Verification and Editing of NC Part-program

Chan Bong Kim, Se Hyung Park and Min Yang Yang

Key Words: NC(수치제어), CAD/CAM(컴퓨터원용 설계 및 가공), Verification(검증), Z-map, Part Program(파트프로그램)

Abstract

NC simulation has been used to replace the test cutting of NC machining. Although it can reduce the NC part programming effort, it still has a problem. If any error is found during the simulation, then the part has to be reprogrammed and it is time consumming. This paper presents a method for verifying and editing the NC code after the post-processing without going back to the part programming stage. A data structure and an algorithm to verify and edit the NC code interactively with the aid of graphics is introduced. Z-map method is used for the shaded image display and cross-sectional view display of the macined parts. The method was implemented in a IBM/PC-386 with MS-Windows software, and the multi-window function of the MS-Windows is used for the simultaneous editing and verification.

1. 서 론

최근 생산제품의 수명주기의 단축에 대처하기 위하여 FMS라는 생산시스템이 도입되고 있다. 특히 금형과 같은 다품종 소량생산 품목은 경쟁력을 높이기 위하여 무인가공을 하려고 하는 욕구가 높아지고 있다. 이러한 욕구는 CAD/CAM기술의 발달을 가속시켰으며, 제품의 설계및 생산에 이르기 까지 통합된 체계를 이룰 수 있는 컴퓨터 통합 생산시스템(CIM : computer integrated manufacturing)으로 까지 발전하고 있다.

그러나, 금형과 같은 품목을 가공하는 공정을 무인화하여 완전한 FMS를 구축하는데는 많은 문제점이 있는것이 실정이다. 그중 기계 가공중 예기치 못한 상황 즉, 공구의 마모 또는 파손과 같은 상황에 대처하는 기술의 부족과 NC파트프로그램의 오류에 따른 가공불량이 대표적인 예이다.

NC기계가공을 할 경우에 발생하는 오류는 부적절한 공구궤적, 과대절삭(over cut)과 과소절삭(under cut), 공구와 피삭재와의 충돌등 기하학적인 오류(geometric errors)와 공구의 휨, 떨림, 마모, 파괴등 물리적인 오류(physical errors)로 크게 나눌 수 있다. 이러한 오류들은 프로그래머(programmer)또는 가공자의 실수로 인해서 발생할 수 있으며 그 원인은 다양하다(Fig. 1). CAD/CAM시스템을 이용하거나 APT 계열의 CAM시스템을 이용하여 NC파트프로그램을 얻을 경우, 도면의 판독잘못이나 CAD데이터 혹은 CAM데이타의 부정확한 입력, 부적절한 절삭 조건의 부여, 가공준비상태의 부적절등이 NC가공의 불량을 초래

*한국과학기술원, 정밀공학과
**정회원, 한국과학기술연구원 CAD/CAM실
***정회원, 한국과학기술원 정밀공학과

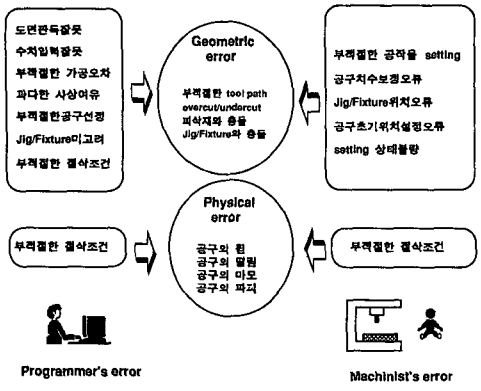


Fig. 1 Errors in NC machining

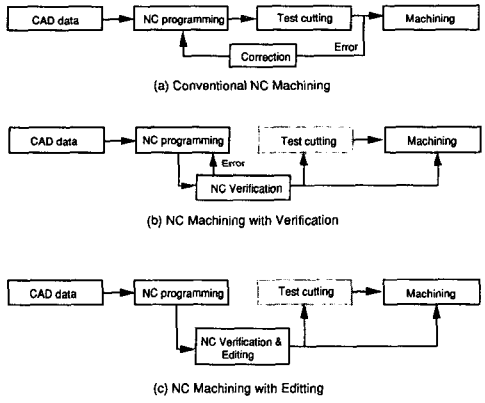


Fig. 2 Comparision of the NC machining methods

한다. 이러한 문제들은 NC파트프로그램이 절삭경로 위주로 작성되며 절삭조건의 결정, 최적경로의 결정 등은 프로그래머에 의존하기 때문에 발생된다.

NC파트프로그램의 오류를 찾아내어 수정하기 위하여 Fig. 2(a)와 같이 실가공전에 플라스틱재료나 왁스와 같은 연한 재료를 이용하여 모의가공을 하는 방법이 이용되고 있으며 현재도 많이 이용되고 있다. 이 방법은 인력 및 기계의 낭비를 초래하며 프로그래머, 가공자, 검사자가 다르기 때문에 정확한 검사, 확인 및 수정이 어려운 단점이 있다. 이의 개선을 위하여 컴퓨터를 이용하여 공구궤적을 검증하는 방법이 개발되어 사용되고 있으나 공구궤적에 오류가 발생했을 경우 그 위치와 정도를 파악하지 못하여 다시 NC프로그램을 해야하는 단점이 있다(Fig. 2(b)). 이러한 두가지 방법을 이용하여 NC공구궤적을 검증하기 위하여 미국의 경우만도 연간 20억달러가 소비되고 있으며 NC파트프로그래머의 작업시간의 13%를 오류수정에 소비하고 있는 것으로 발표된 바 있다.[1]

기존에 발표된 많은 NC시뮬레이션 시스템은 주로 검증에 국한된 것으로 공구이동을 소거체적(swept volume)으로 나타내고 부울리안 연산(boolean operation)을 이용하여 가공상황을 묘사하고, 레이캐스팅(ray casting)방법을 이용하여 형상을 처리하였다.[2~5] 최근 상용화된 CAM시스템에서도 레이캐스팅 방법을 이용한 검증방법이 이용되고 있으며,[6] Z-map을 이용하거나[7] 솔리드 모델링(solid modeling)방법을 이용하여 시뮬레이션을 수행하는 연구도 많이 진행되고 있다.[8]

본 연구에서 제시된 방법은 Fig. 2(c)에 나타낸 방법으로 공구궤적의 검증과 더불어 NC공구궤적의 잘못된 부분과 정도를 찾아내 수정할 수 있도록 하여 기존의 NC검증시스템의 비효율성을 해결하고자 했다.

2. NC공구 궤적의 검증 및 편집

2.1 연구의 개요

CAD/CAM시스템에서 산출되는 CL데이타는 사용하고자하는 공작기계에 따라 후처리(post-processing)에 의하여 NC파트프로그램으로 만들어진다. 본 연구에서는 공작기계에 직접 입력되는 NC파트프로그램을 이용하여 공구이동 명령 뿐만 아니라 공작기계의 상태를 나타내는 정보를 검증할 수 있도록 하였다.

본 연구에서 개발한 NC파트프로그램의 검증 및 편집을 위한 시스템의 전체적인 구조는 Fig. 3에 나타낸 바와 같다. NC파트프로그램의 공구이동정보(motional command)와 공작기계의 상태를 나타내는 정보(auxiliary command)에 대한 검증을 위해 와이어프레임 기법과 솔리드 모델링 기법을 이용하였으며, 공구이동에 의해 창성되는 공작물의 형상과 최종 공작물의 형상을 Z-map으로 변환하여 비교하여 가공오차가 발생한 부위를 찾아 수정 및 편집할 수 있도록 하였다.

공구궤적의 검증을 위한 와이어프레임 도시 방법과 솔리드 도시 방법을 소개하며, 이를 위한 Z-map 변환과정을 설명하고 효과적인 수정 및 편집을 위한 알고리즘과 데이터 구조를 설명하고자

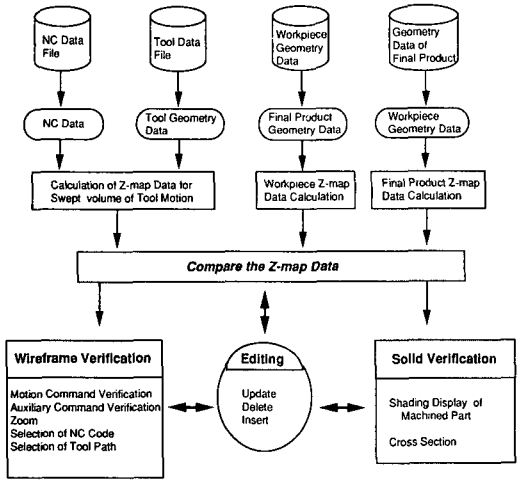


Fig. 3 Structure of NC verification and editing system

한다.

2.2 NC 공구궤적의 검증

NC파트프로그램의 각각의 라인은 공구의 이동정보(motional command)와 공작기계의 상태를 나타내는 정보(auxiliary command)의 연속으로 되어있다. NC파트프로그램내에 있는 각각의 정보에 대한 오류를 찾기위해서는 공구궤적의 와이어 프레임 도시(wire frame display)방법과 가공된 형상을 볼 수 있는 솔리드 도시(solid display) 방법을 모두 이용할 수 있어야 한다.

공구궤적의 와이어 프레임 도시는 처리속도가 빠르고 부분확대(zooming)를 할 수 있기 때문에 공구궤적의 오류를 효과적으로 찾을 수 있고, 파트프로그램 각각의 블럭(block)과 그 블럭에 해당되는 공구궤적의 도시를 연관시킬 수 있어 수정 및 편집을 용이하게 수행 할 수 있다. 그러나 복잡한 형상일 경우 형상을 식별하기 어렵고 가공된 후의 형상을 예측하기 어렵다는 단점이 있다.

솔리드 모델링 기법을 이용하여 가공된 형상을 도시하는 경우는 처리속도는 상대적으로 많이 소요되나 가공된 후의 형상을 볼 수 있고, 단면도시(cross section display)와 가공된 공작물의 상대좌표값을 알 수 있고, CAD데이타와 비교를 통하여 가공불량의 위치와 정도를 찾아낼 수 있다.

본 연구에서는 와이어프레임 도시와 솔리드 도시를 이용하여 공구궤적의 오류를 찾아낼 수 있는 방법에 대하여 연구하였다. 와이어프레임 방법은 일반적인 방법으로 컴퓨터를 이용한 검증에 많이 이용되고 있는 방법으로 설명을 약하고, Z-map을 이용한 솔리드 표현에 의한 검증에 대해 설명하고자 한다. 우선 공구궤적을 Z-map으로 변환하는 방법에 관하여 설명하고 CAD 데이터를 Z-map으로 변환하는 과정을 설명한다.

(1) Z-map을 이용한 가공형상 검증

공작물의 형상정보를 Z-map데이터로 변환하기 위하여 Fig. 4와 같이 공작물이 차지하는 x, y 평면을 일정한 크기로 분할 하고 매 공구의 이동에 대한 소거체적(Swept volume)을 계산하여 Z-map데이터와 비교한다. 이러한 과정을 공구의 이동이 있을 때마다 반복하여 공작물의 형상 변화를 갱신하면 가공된 형상정보에 대한 Z-map정보를 얻을 수 있다.

공구의 이동에 의해 형성되는 소거체적을 구하기 위해서는 공구의 이동시 매 순간마다 공구의 이동체적을 합해야 한다. 즉, 공구의 이동에 의하여 형성되는 체적을 S라 할 때 S는 다음과 같이 나타낼 수 있다.[9]

$$S=\bigcup_{i=0}^{\infty} G(t_i) \tag{1}$$

여기서, $G(t_i)$: 공구의 이동순간 t_i에서 공구의 이동에 의한 체적

만약 공구가 볼엔드밀(ball end mill)일 경우 공구의 이동 체적은 공구의 볼 부분만으로 구성된 체적으로 단순화 시킬 수 있고, 두개의 구면(spherical surface)과 실린더면(cylindrical sur

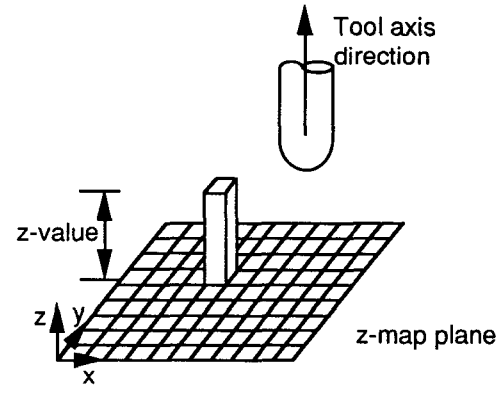


Fig. 4 The concept of z-map

face)으로 이루어진 포락면(envelope)으로 나타낼 수 있다. 이에대한 공구의 이동체적은 다음과 같이 나타낼 수 있다.

$$S=S(C)\cup S(S) \tag{2}$$

여기서, S : 볼엔드밀의 이동에 의해 형성된 체적

$S(C)$: 실린더 부분에 대한 체적

$S(S)$: 구부분에 대한 체적

공구의 이동체적을 xy평면에 투영하여 투영역역내에 들어오는 격자들에 대하여 xy평면으로부터 이동체적면까지의 높이(z값)를 구하면 공구이동체적에 대한 격자정보가 완성된다.

공구의 이동체적에 대한 격자 정보가 결정되면 공작물의 격자 정보와 비교하여 가공시 공구와 공작물이 접촉되는 면적이 계산되고 공작물에 대한 Z-map정보가 갱신된다. 즉, 초기 공작물의 형상을 $W(0)$라하고 i번째 공구의 이동에 의하여 생성되는 공작물의 형상을 $W(i)$라 하면 다음과 같이 나타낼 수 있다.

$$W(i)=W(i-1)-S(i) \tag{3}$$

여기서, $W(i)$: i번째 까지 공구이동에 의하여 생성되는 공작물의 형상

$S(i)$: i번째 공구이동에 대한 체적

공작물에 대한 격자 정보의 갱신은 공구이동 체적에 해당하는 격자중 이동체적의 Z값이 공작물의 Z값보다 작은 경우 Z값을 공구의 이동면의 값으로 대체시켜 주면된다.

이렇게 Z-map을 만들어두면 가공된 후의 모습을 여러각도로 도시해 볼수있고, 단면을 볼 수 있으며 CAD데이터와의 비교를 통하여 과대절삭/과소절삭 부분을 발견할 수 있다.

(2) Error Map 생성을 위한 CAD 데이터의 Z-map변환

CAD데이터와 가공된 공작물의 Z-map 데이터의 비교에 의한 오류 발견을 위해서는 CAD데이터를 Z-map데이터로 변환해야 한다. CAD에서 자유곡면은 대부분 $f(u, v)$로 표현되는 u, v의 매개변수 함수식으로 표현되는데 이를 Z-map데이터로 변환하기 위해서는 자유곡면이 포함되어 있는 격자의 중심좌표 x^0, y^0가 주어졌을 때 그것으로부터 u_0, v_0를 구하여 그곳의 Z값 $Z=f(u_0, v_0)$를 자유곡면이

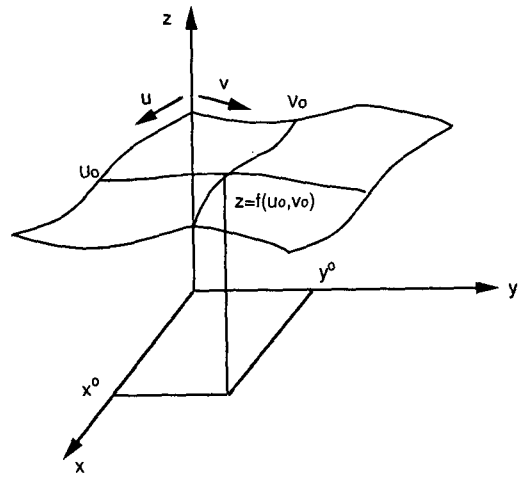


Fig. 5 Intersection between vertical line nd parametric surface

차지하는 모든 격자에 대해 구하면 된다. (Fig. 5)

u_0, v_0를 구하기 위해서는 구하고자하는 x,y평면상의점(x^0, y^0)과 초기점 $p=(u, v)$를 이용하여 다음과 같이 Newton-Raphson방법을 위한 식을 표현한다.[10]

$$m_{1u}\Delta u+m_{1v}\Delta_v=x^0-f_x(u, v) \tag{4}$$

$$m_{2u}\Delta u+m_{2v}\Delta_v=y^0-f_y(u, v) \tag{5}$$

여기서,

$$m_{1u}=\frac{\partial f_x(u_i, v_i)}{\partial u}$$

$$m_{1v}=\frac{\partial f_x(u_i, v_i)}{\partial v}$$

$$m_{2u}=\frac{\partial f_y(u_i, v_i)}{\partial u}$$

$$m_{2u}=\frac{\partial f_y(u_i, v_i)}{\partial v}$$

이것으로부터 매개변수의 증분 Δu와 Δv를 다음과 같이 구할 수 있다.

$$\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}=\begin{bmatrix}m_{1u} & m_{1v}\\ m_{2u} & m_{2v}\end{bmatrix}^{-1}\begin{bmatrix}x^0-f_x(u_i, v_i)\\ y^0-f_y(u_i, v_i)\end{bmatrix} \tag{6}$$

매개변수의 증분에 대한 정보를 이용하여 새로운 매개변수 u, v값을 다시 설정한다.

$$u_{i+1}=u_i+\Delta u \tag{7}$$

$$v_{i+1}=v_i+\Delta v \tag{8}$$

이런과정을 반복하여 오차가 허용범위 이내에 들때까지 반복하여 u_0, v_0를 정하고, 이로부터 CAD

데이터를 Z-map데이터로 변환한다. CAD데이터의 Z-map데이터는 공구궤적 Z-map데이터와 비교되어 Error Map을 생성하는데 이용된다.

2.3 NC파트프로그램의 편집

검증기능을 이용하여 발견된 NC파트프로그램의 오류는 편집기능을 이용하여 수정된다. NC블럭의 한 줄에 해당하는 3차원 공구의 이동은 2차원 변환이 되어 화면에 도시되는데, 이러한 공구궤적중 오류가 발생한 공구궤적을 효과적으로 찾아 수정하기 위해서는 화면에 도시된 각각의 공구궤적이 NC파트프로그램내에 어느 블럭에 의하여 도시된 것인지 쉽고 빠르게 찾을수 있어야 한다.

본 연구에서는 NC파트프로그램의 각 라인과 각각의 라인에 의하여 도시된 공구궤적을 연결하여 불량이 있는 공구궤적을 선택했을 때 그 공구궤적에 해당하는 NC파트프로그램을 찾아 수정하거나, 잘못된 NC파트프로그램의 일부분을 선택했을 경우 그에 대한 공구궤적을 찾아 확인 및 수정할 수 있도록 하였다.

이러한 작업을 효과적으로 수행하기 위해서는 특별한 데이타구조와 알고리즘이 필요하다. Fig. 6은 수정및 편집을 위한 데이터 선언을 보여주고 있다. NC파트프로그램의 각각의 라인은 삽입, 소거, 복사등이 쉽도록 이중 연결구조(double linked list structure)로 연결되어있고, 각각의 라인은 공구궤적과의 연결을 위하여 화면에 2차원 변환되어 도시된 x,y값을 갖는다. (Fig. 7) NC파트프로그램의 한 라인은 가변길이를 갖는 문자열(string)로 저장되어 불필요한 기억영역의 소비를 막았다. 또한, 전체 그래픽 화면을 16등분 하여 각각의 구간을 지나는 공구궤적에 대한 라인 번호를 저장함으로써 그래픽 화면상의공구궤적의 선택과 Zooming등의 그래픽 처리가 빨리 이루어 질 수 있도록 하였다. 즉, 그래픽 화면상의 임의의 공구궤적을 선택했을 경우 선택된 점이 속하는 구간을 선정하고 그 구간을 지나는 공구궤적들을 검사하여 선택한 점과 거리가 최소인 공구궤적을 선택할 수 있도록 하였다. (Fig. 8) 또한 NC파트프로그램을 선택했을 경우도

```
/* NC 블럭의 각 라인의 관계와 화면정보를 위한 Double Linked List 구조 */
typedef DLL huge *LPDLL

struct dll {
    HANDLE prev;  /* 이전라인을 가리키는 handle */
    HANDLE next;  /* 다음라인을 가리키는 handle */
    float  x_scr; /* 화면좌표의 x 값 */
    float  y_scr; /* 화면좌표의 y 값 */
    HANDLE str;   /* NC코드의 한 라인을 저장하기 위한 handle */
} DLL;


/* 원호보간에대한 정보 저장을 위한 메모리에 대한 구조 */
typedef CIRCLE huge *LPCIRCLE;

struct hcircle {
    HANDLE hCircle      /* 원호보간이 있는 NC 라인의 handle */
    int    num_index;   /* 원호를 직선으로 나타낼 때의 직선의 갯수 */
    HANDLE hCirScr;     /* 좌표계 저장을 위한메모리에 대한 handle */
} CIRCLE;

/*원호보간의 화면좌표계 저장을 위한 메모리 구조*/
typedef CIRSCR huge *LPCIRSCR;

struct CirScr {
    int  x_scr;  /* 화면좌표의 x값*/
    int  y_scr;  /* 화면좌표의 y값*/
}CIRSCR;

/* 분할된 각 영역에서 공구궤적의 저장을 위한 메모리 구조 */
HANDLE Sec[16];
```

Fig. 6 Data structure declarations

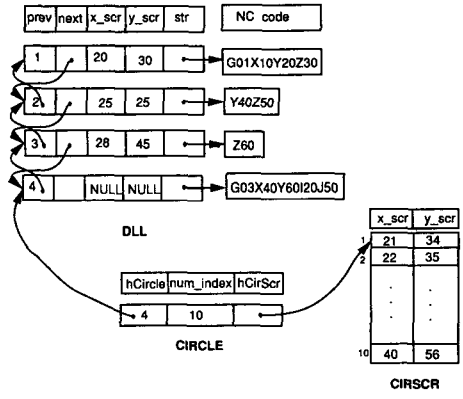
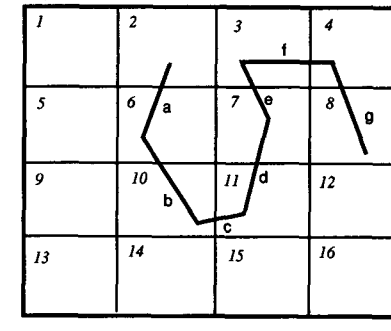


Fig. 7 Data structure for the NC editing



Sec[1] = [], Sec[2] = [a], Sec[3] = [e,f], Sec[4] = [f,g]
Sec[5] = [], Sec[6] = [a,b], Sec[7] = [d,e], Sec[8] = [g],
Sec[9] = [], Sec[10] = [b,c], Sec[11] = [c,d], Sec[12] = [],
Sec[13] = [], Sec[14] = [], Sec[15] = [], Sec[16] = []

Fig. 8 Sectioning of the graphical screen

그 라인이 갖는 화면 좌표 x, y값으로 부터 직접 화면좌표를 얻을 수 있어 그라인에 해당하는 공구궤적을 찾을수 있게된다. 이에대한 알고리즘은 다음과 같다.

```
procedure Select _NC_Code(px, py) ;
var px : x value of picked point on screen;
    py : y value of picked point on screen;
    no_sec : number of section include the picked point;
    x_scr : x_scr value of DLL structure indicated by Sec[no_sec] [i] ;
    y_scr : y_scr value of DLL structure indicated by Sec[no_sec] [i] ;
begin
    no_sec : =Include_Section·px, (py);
    i : =0;
while Sec[no_sec] [i] is not empty do
    x_scr : =x_scr_DLL(Sec[no_sec] [i];
    y_scr : =y_scr_DLL(Sec[no_sec] [i];
    length : =Length(px, py), drawd line by(x_scr, y_scr));
    if length is minimum then
Selected_NC_code_Number : =Sec[no_xec] [i];
    end;
i : =i+1;
    end;
    end;
```

이렇게 선택된 공구궤적이 결정되면 이중 연결구조의 데이터를 이용하여 그전이나 그 후의 공구궤적을 쉽게 선택하여 지우거나 대체하거나 새로운 코드를 추가 할 수 있다.

3. 시스템의 개발 및 적용실례

NC 데이터 검증 및 수정을 위한 시스템은 IBM 386 PC에서 MS-C와 MS-SDK (software development kit)를 이용하여 개발하였으며 MS-Windows와 함께 운용되도록 하였다. MS-Windows와 MS-SDK를 이용하였기 때문에 다중화면분할(multi-windows)이 가능하고 메모리 사용이 편리하며 사용자가 사용하기 편한 그래픽 기능을 부여하여 비 전공자라 할 지라도 쉽게 사용할 수 있도록 하였다.

NC데이터의 검증 및 수정과정은 앞서 설명한 Fig. 3과 같이 먼저 NC데이터와 CAD데이터를 읽어 Z-map으로 변환하고, NC데이터를 와이어프레

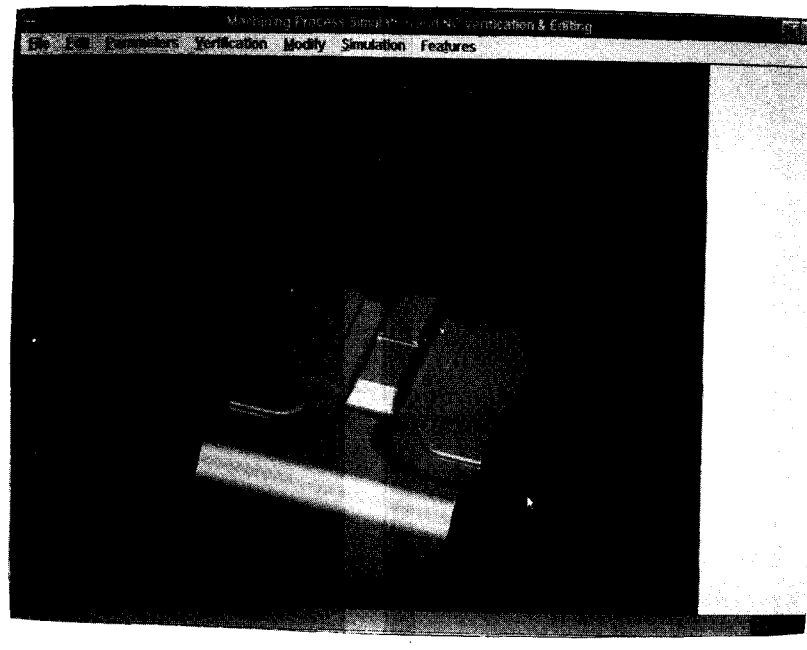


Fig. 9 CAD surface of car body

임과 솔리드를 이용하여 검증하고 Error-Map을 구한 후 공구궤적을 그 위에 그린후 과대절삭 또는 과소절삭부분에 대한 공구궤적을 선택하여 수정한다.

Fig. 9는 자동차 판넬의 CAD데이터를 음영처리한 모습이고, Fig. 10은 이 모델에 대한 공구궤적과 공작기계의 상태를 나타내는 ICON을 도시하여 각각의 공작기계의 상태를 검증할 수 있도록 하였

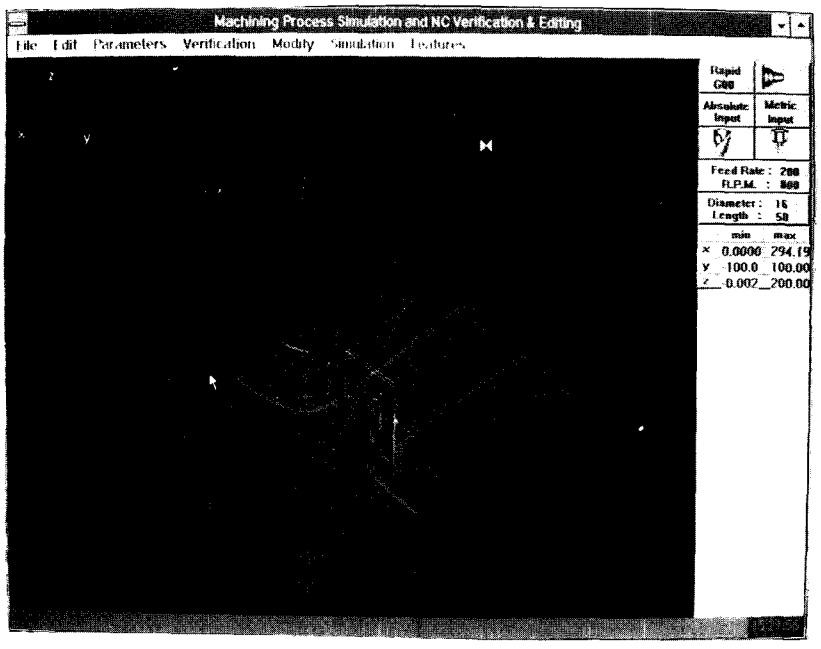


Fig. 10 The tool path for CAD data

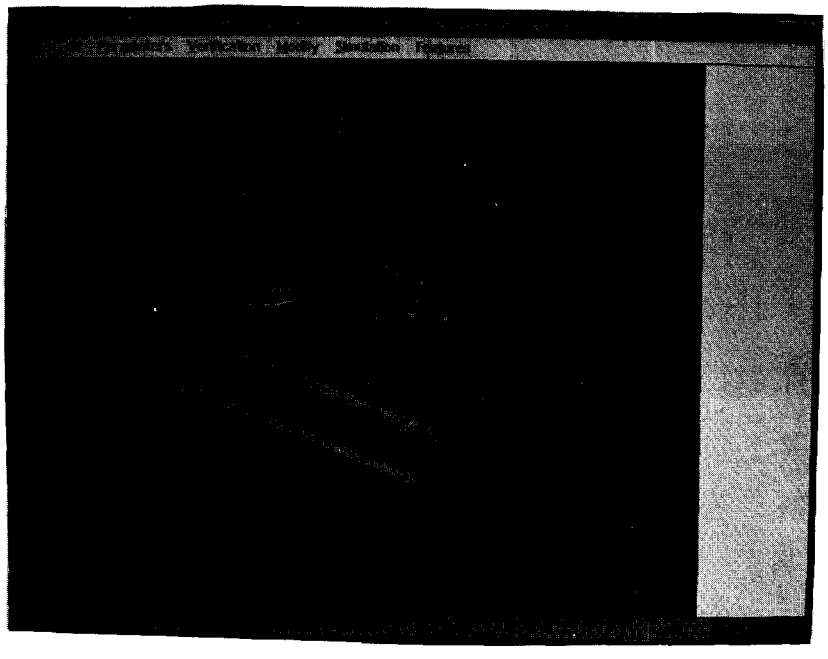


Fig. 11 Machined surface of CAD data

다. 이러한 공구궤적을 이용하여 가공하였을 경우 최종적으로 나타날 공작물의 모습을 Z-map 데이터를 이용하여 그래픽 처리한 모습을 Fig. 11에 나타냈다. Fig. 12는 CAD데이터와 가공된 공작물의 Z-map 데이터를 이용하여 가공오차의 분포를 나타낸 Error Map으로 전체적으로 곡률이 있는 부위에서는 공구의 반지름을 잘못 선택하여 과소절삭(under cut)이 발생했고, 어떤 부분에서는 공작물

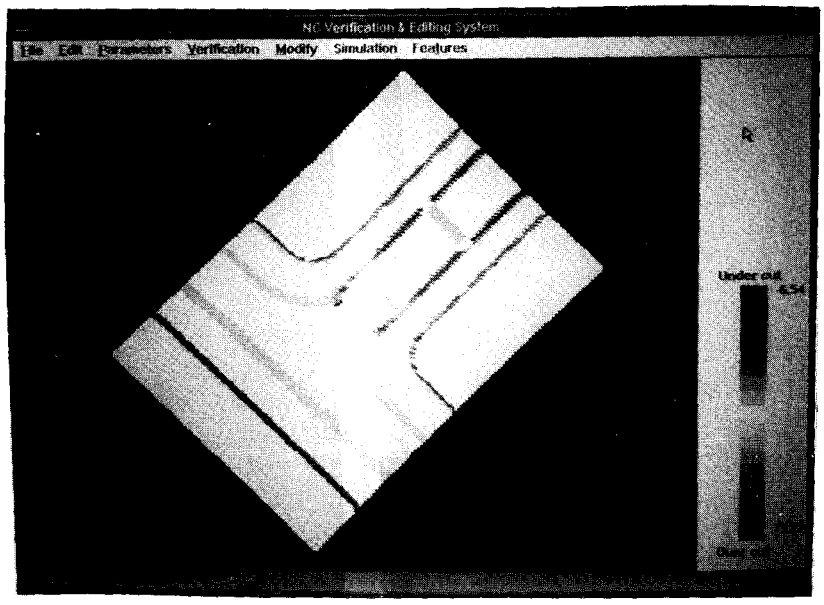


Fig. 12 Error map

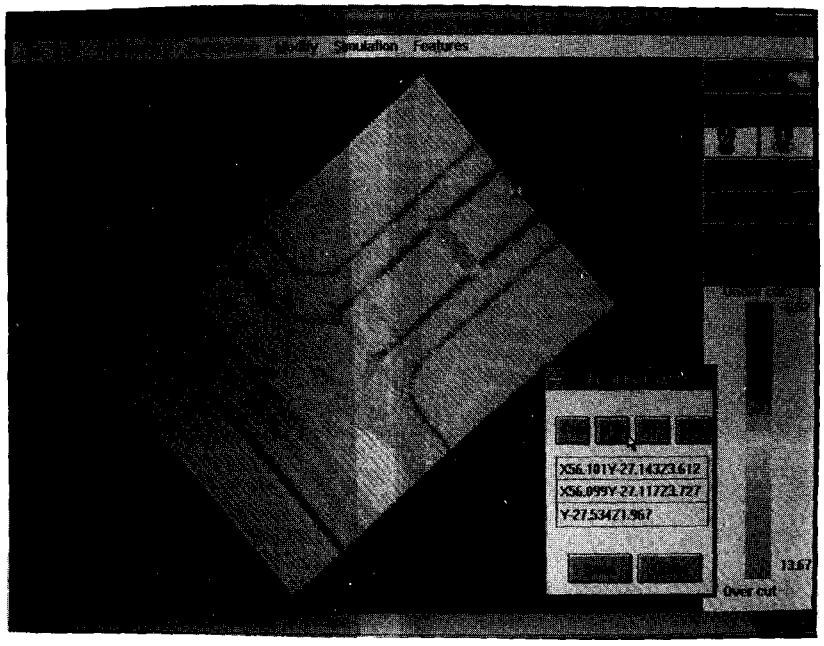


Fig. 13 Error map with tool path

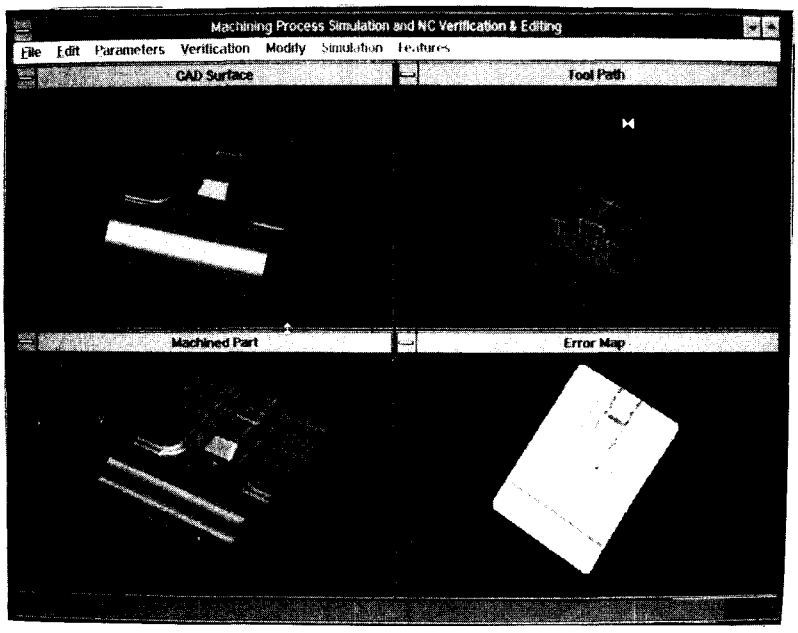


Fig. 14 Co-display of verification results

과 공구와 간섭이 발생하여 과대절삭이 발생한 모습을 보여주고 있다. Fig. 13은 Error Map위에 공구궤적을 겹치도록하여 과대절삭 부위에 대한공구궤적을 찾고 그 공구궤적을 수정하는 과정을 나타내고 있으며 Fig. 14는 수정된 공구궤적, 수정된 공구궤적에 의한 가공형상 그리고 수정된 후의 Error-Map을 한 화면에 동시에 나타낸 모습이다.

4. 결 론

본 연구에서는 NC공구궤적의 검증 및 수정할 수 있는 방법에 관하여 연구하였다. NC파트프로그램의 오류를 찾기 위하여 와이어 프레임 처리와 솔리드 형상처리를 이용하여 공구궤적을 검증하였고 CAD데이터와 가공된 후의 데이터를 비교하여 오류가 발생하는 부위와 정도를 찾아 낼수 있었다. 또한 공구궤적과 NC파트프로그램을 빠르게 연결할 수 있는 데이터 구조와 알고리즘을 이용하여 오류가 발생한 부위에 대한 공구궤적과 NC파트프로그램을 쉽게 찾고 수정할수 있었다.

본 논문에서는 형상오류만을 다루었으나, 앞으로 물리적인 오류를 발견하여 수정할 수 있는 방법을 개발하여 프로그램에 추가할 예정이다. 이를 통하여 오류가 전혀 없는 NC 정보를 생성하여 가공무인화를 추구하고자 한다.

참고문헌

(1) Frederick Mason, 1992, "Verification : Best thing to happen to NC," American machinist, April., pp. 39~43.

(2) Vocker, H.B. and Hunt, W.A., 1981, "The Role of Solid Modeling in Machining Process Modeling and NC Verification," SAE Techincal Series 840195.

(3) Sungurtekin,U.A. and Voelcker,H.B., 1986, "Graphical Simulation and Automatic Verification of NC Machining Program," Proceedings of the IEEE International Conference on Robotics and Automation, April, pp. 156~165.

(4) Leu,M.C., Park,S.H. and Wang,K.K., 1986, "Geometric Representation of Translational Swept Volumes and its Applications," J. of Engineering for Industry, Vol. 208,May, pp. 113~119.

(5) Wang,W.P., 1988, "Solid Modeling for Optimizing Metal Removal of Three-dimensional NC End Milling," Journal of Manufacturing Systems, Vol. 7, No. 1, pp. 57~65.

(6) Appleton,E., 1990, "Tool Path Simulation-Software Enabnles Engineers to Visualize Machining Processes in Advance," Computer Graphics World, September, pp. 85~91.

(7) Takeuchi, Y., Sakamoto,M., Abe, Y. and Orita,R., 1989, "Development of a Personal CAD/CAM System for Mold Manufacturing Based on Solid Modeling Techniques," Annals of the CIRP, Vol. 38, pp. 429~432.

(8) Takeshi Kishinami, Satoshi Kanai, Hiroyuki Shinjyo, 1989, "An Application of Voxel Representation to Machining Simulator" JSPE-55-01, pp. 105~110.

(9) Sehyung Park, 1992, "Simulation of CNC Machining Using a Ball Endmill," Ph.D.Thesis, KAIST.

(10) 최병규, 1990, "Surface Modeling for CAD/CAM," 한국과학기술원 산학협동공개강좌 교재, pp. 297~317.