696 大韓機械學會論文集 第16卷 第4號, pp. 696~706, 1992.

〈論 文〉

NC 판금작업에서의 자동 공구선정

조경호* · 이건우**

(1991년 12월 9일 접수)

Automatic Tool Selection in Numerically Controlled Sheet Metal Fabrication

Kyung H. Cho and Kunwoo Lee

Key Words: Automatic Tool Selection(자동 공구 선정), Sheet Metal Part(판재 부품), NCT Operation(수치제어 Turret작업), Pattern Matching(형상 비교), Punch(펀치), Nibbling(근사 펀칭)

Abstract

In sheet metal fabrication using NCT(numerically controlled turret), the automatic tool selection for the NCT operation is the major problem to be solved first to improve its production performance. However, the punching tool selection has been done by human experts either manually or semi-automatically. In this paper, we have introduced the *shape-index-set* to handle the shape of sheet metal parts and developed an algorithm through which one can find easily the successive matching curves between two curve lists, one from the punching tool and the other from the boundaries of the sheet metal part. Based on this algorithm, we have also devised the method that can select automatically the tools to punch out the boundaries of sheet metal parts. The result of several computational experiments shows the successful tool selection without any fail.

1. 서 론

최근 다품종 소량생산 및 신제품 수명의 단축 추세에 부응하여, 판재부품 가공분야에선 NCT (numerically controlled turret punch press) 도입이 각광을 받고 있다. 이는 형상이 바뀔 때 마다 고가의 펀치공구을 새로 만들어 사용하는 대신 이미 확보하고 있는 여러가지 공구를 조합하여 복잡한 형상을 펀칭 할 수 있기 때문이다. 이러한 NCT 기계를 효율적으로 사용하기 위해서는 이를 구동하기 위한 펀칭정보를 제공해 주어야 한다. 여기서 펀칭정보라함은 판재부품 경계를 펀칭하기 위해 필요한 공구와, 공구의 정확한 위치 및 방향에 대한 정보를 뜻한다. 이러한 펀칭정보를 전개도(펼쳐진 판재 모델)로부터 추출하는 작업을 공구선정 작업(tool processing 혹은 tool selection)이라 하는데, 현재 대부분의 NCT 사용업체에서는 공구선정 작업을 숙련된 작업자의 수작업에 의존하고 있는 실정이다. 이로 인해 작업자의 숙련도에 따라 NCT기계의 실제 펀칭작업의 생산성이 좌우되고 있으며, 공구선정 작업에 많은 시간이 소요된다.

본 연구에서는 이러한 공구선정 작업의 자동화를

*정회원, 서울대학교 기계설계학과 대학원
**정회원, 서울대학교 기계설계학과

목표로 하였다. 공구선정을 완전 자동으로 행하기 위해 해결해야 할 문제는, 현재 공구대(turret)에 장착되어 있는 공구중에서 어떤 공구를 얼마만큼 직선, 회전 이동시키면 판재 경계와 정확하게 일치하는가 하는 문제와, 복잡한 형상의 펀칭을 위해선 효과적인 공구조합은 어떻게 해야하나 하는 문제로 귀결되므로 이 두가지 문제의 해결방안을 기술한다.

2. 관련 연구

형상 인식(patterm recognition) 혹은 곡선간 비교(curve matching)에 대한 연구는 컴퓨터 비젼(computer vision), 로봇을 이용한 조립(robot assembly), 항공기 사진인식, 문자 인식(letter recognition) 등의 여러분야에서 활발히 연구 발전되어 왔다.

이들 연구의 대부분은 형상의 특징을 기술할 수 있는 여러가지의 척도, 예컨대 임계점(critical point), 분기점(break point) 혹은 경계표식(land mark)[1~5] 등과 같은 것을 이용하여 형상인식 문제를 해결하고자 했다. 또한, 모양자(template)나 필터를 이용한 형상 비교 방법[6,7]도 있다. 이런 방법들은 주로 물체 경계곡선의 곡률 변화가 급격한 경우를 추적 대상으로 삼고있으나, 특징이 뚜렷하지 않은 경우 혹은 다른 형상에 가려지거나 중복되어 이런 특징을 추출할 수 없는 경우(obscured object)에는 취약점을 보인다[8]. 경계 곡선 위의 점들을 모멘트 불변량(moment invariant)[9,10] 또는 푸리에 기술자(fourier descriptor)[11~14]로 기술하여 형상 인식 문제를 연구하는 방법도 많은 발전을 보였다. 모멘트 불변량에 의한 형상인식 방법은 계산시간을 많이 요구하는 것이 단점으로 나타나고, 프리에 기술자에 의한 물체 표현은 물체의 전체경계를 알고 있을 때는 매우 유용하나, 푸리에 전개(fourier expansion)의 주기성 때문에 경계의 일부분만을 기술하는 데는 문제점을 보이고 있다[13]. 형상인식 혹은 물체 분류(object classification)에 확률통계론적 방법을 도입시킨 경우도 있다[15]. Sharir는 리스트 스퀘어 피트(least-square-fit) 개념을 도입하여 이차원 혹은 삼차원 곡선의 일치 여부를 판별하는 방법을 발표[16]하였고, Wolfson[8]은 형상인식 문제에 이 방법을 이용하여 좋은 결과를 얻었다.

반면에, NCT 분야에서의 펀칭 공구의 자동선정에 대한 연구개발 현황은 의외로 낙후되어 있다. Kimura[17]가 판재부품의 굽힘 시뮬레이터(bending simulator) 개발을 발표한 바 있으나, 이는 펀치 공구의 자동선정과는 무관한 일이었다. 현재 상용화된 소프트웨어[18,19]들에서도 공구선정 작업을 전자동으로 처리하지 못하고 있어, 현장에선 대부분 경험이 풍부한 숙련자가 수동 혹은 반자동으로 이를 수행하고 있는 실정이다.

3. 공구의 자동선택 알고리즘

본 논문에서는 다음의 조건을 만족시키는 공구의 자동선택 알고리즘을 개발하였다. 즉, 만일 펀칭되어야할 판재형상의 전부 또는 그 일부분과 일치하는 공구가 있으면 그공구의 자동탐색이 반드시 성공 해야하고 해당 부분의 정확한 펀칭을 위한 공구의 회전, 직선 운동량이 자동으로 결정되어야 한다. 이의 구현을 위해 본 연구에서 개발된 알고리즘을 다음에 단계별로 설명하겠다.

3.1 자료구조 및 경계곡선의 방향성

이차원 편집기(2D editor)를 통하여 사용자에 의해 입력된 판재형상 및 공구형상의 모든 경계 곡선들은 NURB(Non Uniform Rational B-Spline)[20]로 표현되며 시스템에 의해 특정 자료구조에 저장, 관리된다. Fig. 1은 공구의 자동선택 알고리즘의 효과적인 수행을 위해 본 논문에서 사용된 자료구조이다. 모든 공구와 판재의 경계는 최소 갯수의 곡선으로 나타내어지는 것으로 가정한다. 따라서, 동일 직선 위에 존재하면서 서로 연결되어 있는 직선들은 한데 묶여 하나의 직선으로 대체된다. 같은 방법으로, 동일 원호 상에 존재하면서 서로 연결되어 있는 원호들도 한데 묶여 하나의 원호로 대체된다. 이러한 대체 작업은 시스템에 의해 자동으로 선행된다.

후에 기술될 3.5절의 펀칭작업시 판재경계곡선의 빈번한 갱신작업을 용이하게 하기 위하여 판재 경계곡선들의 저장을 위한 자료구조로 이중 열결리스트(doubly-linked-list)를 채택하였다. 또한 이 자료구조는 펀칭현황을 효율적으로 관리하기 위해 각 곡선의 펀칭유무를 나타내는 펀칭플래그(punch-flag)를 갖는다. 즉 이 플래그 값이 'on'이면 이미 펀칭된 판재경계 곡선임을 뜻하고, 반대로 'off'이

```
typedef   struct  curve              CURVE ;
typedef   struct  control            CONTROL ;
typedef   struct  point              POINT ;
typedef   struct  cvs                CVS ;
typedef   struct  sheet              SHEET;
typedef   struct  tool               TOOL;
typedef   struct  shape_index_set    SIS;
struct    control
{  double       hx,hy,hz,h ;  };
struct    point
{  double       x,y,z ;  };
struct    curve
{  int          Order ;      /* order of NURB */
   int          NumCon ;     /* number of control points */
   double       *Knot ;      /* knot vector */
   CONTROL      *Control ;   /* control point list */
   int          *Type ;      /* type of curve */
};
struct    cvs
{  CURVE        *cv ;        /* curve equation */
   CVS          *pre, *next ; /* doubly linked list */
   float        length ;     /* curve length */
   short        punch_flag ; /* punching flag */
};
struct    sheet
{  short        id ;         /* id no. of sheet */
   CVS          *cvs ;       /* pointer to doubly-linked-list */
   SHEET        *next ;      /* pointer to next sheet */
};
struct    tool
{  short        id ;         /* id no. of tool */
   short        index ;      /* indexing flag */
   float        area ;       /* area of tool */
   POINT        centroid ;   /* centroid of tool */
   CURVE        **cv ;       /* array of tool curves */
   SIS          *template    /* SIS list of tool */
};
struct    shape_index_set
{  short        theta1,theta2 ; /* angle index */
   short        length ;     /* curve-length index */
   short        rho ;        /* curvature index */
};
struct    match_inform
{  short        match_gain ; /* match_count or length */
   CURVE        **scv,**tcv ; /* matching curve list of sheet & tool */
   TOOL         *tool ;      /* candidate tool */
};
```

Fig. 1 Data structure for storing boundaries of tool and sheet

면 아직 펀칭되지 않았음을 의미한다. 이런 자료구조의 생성 및 관리는 시스템에 의해 자동으로 수행된다.

Fig. 2에서 보는 바와 같이, 모든 판재경계 및 공구경계 곡선의 방향은 곡선의 진행방향에 대해 펀치에 의해 제거될 부분이 항상 경계 곡선의 오른쪽에 위치 하도록 그 방향이 시스템에 의해 자동으로 조정된다. 이런 방향성은 동일 펀칭 알고리즘으로 판재의 내부경계(punching)뿐만 아니라 외부경계(blanking)를 모두 펀칭할 수 있게 한다.

3.2 형상지표 집합(Shape Index Set)

판재 형상과 공구 형상의 비슷한 부분을 쉽고 빠르게 추적하기 위하여, 다음과 같은 형상지표들의 묶음을 경계곡선 C_i에서의 형상지표 집합(shape index set), $SIS(i)$라 정의한다.

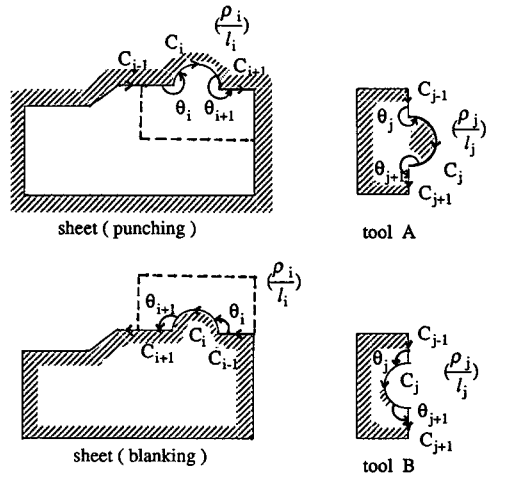


Fig. 2 Curve direction and shape indices in sheet and tool

$$SIS(i) : \{\theta_i,\ \rho_i,\ l_i,\ \theta_{i+1}\}$$

여기서 θ_i는 각도지표로서, 경계 곡선 C_{i+1}와 C_i가 만나는 점에서 두 곡선의 접선이 이루는 각도중 펀칭되어야할 각도에 100을 곱한 값의 정수부분이다. 길이지표 l_i는 곡선 C_i의 길이에 100을 곱한 값의 정수부분이다. ρ_i는 곡선 C_i의 곡률 반경에 관한 지표로 직선일땐 무한대 이고, 반경 r인 원호이면 $\pm r$에 100을 곱한 값의 정수부분이며 펀칭 되어 제거되는 부분을 시계방향으로 돌면 -부호, 반시계 방향으로 돌면 +부호를 갖는다. 자유곡선에선 ρ_i가 일정치 않음으로, 컴퓨터가 제공하는 최소정수를 부여하여 자유곡선임을 나타낸다. 따라서, n개의 경계곡선으로 이루어진 형상은 연속된 n개의 형상지표 집합(SIS list)으로 표현되고, 후에 기술될 3.3절의 일치곡선군의 탐색 및 3.4절의 후보공구의 선정 등에서 필요할 때 마다 시스템은 이들 형상지표 집합들을 자동으로 생성하여 사용한다. 판재 및 공구의 형상 지표들에 대한 예시가 Fig. 2에 나타나 있다.

이렇게 정의된 형상 지표 집합의 특징은 (1) 회전, 병진효과를 배제시키면서 물체의 형상을 간결하게 표현하여 (2) 이차원에서의 형상인식 문제를 형상 지표 집합 리스트의 일차원적 비교 문제로 단순화 시키며, (3) 형상에 대한 다각형 근사(polygon approximation)를 할 필요가 없어 (4) 형상 정보 저장을 위한 기억 장소가 매우 작아도 된다.

3.3 일치 곡선군의 인식

판재 경계가 n개의 곡선으로 이루어져 있고 공구의 경계가 m개의 곡선으로 이루어져 있다면, 판재 경계에 대한 SIS 리스트 $S_i(i=1,...,n)$와 공구 경계의 SIS리스트 $T_j(j=1,...,m)$ 사이에 서로 일치하는 SIS들을 찾기위해 T_j를 S_i의 첫번째 위치부터 계속 오른쪽으로 이동시켜 가면서 가장 많은 SIS가 일치하는 경우를 찾는다. 이 과정을 Fig. 3을 예로 설명하겠다.

이 예에서는 판재경계는 n개의 곡선으로 구성되고 각 곡선의 SIS는 S_1, S_2, ..., S_n이다. 마찬가지로 공구경계는 4개의 곡선으로 구성되어 있고 각 곡선의 SIS는 각각 T_1, T_2, T_3, T_4이다. 또한 각각의 T_j는 그림과 같이 θ_j, l_j, ρ_j, θ_{j+1}의 형상지표를 갖는다. T_1이 S_i의 s번째 위에 놓이도록 공구를 위치시켰을 때를 shift s라 하고, s를 1에서 n까지 shift량을 1씩 증가시키면서 공구를 이동(sliding)시킨다. 만약 S_i와 T_j가 일치하는 형상일때 T_j에 불이 들어오도록 한다면(일치 ; on, 불일치 ; off), 임의의 shift 위치에서 T_j의 on-off 상태로부터 연속적으로 일치하는 SIS리스트와 그 길이를 알 수 있다. 이를 Fig. 4와 같은 구체적인 경우에 적용시켜 보겠다.

Fig. 4에서 모든 숫자는 편의상 경계곡선의 번호를 나타내는 동시에 해당 곡선에서의 형상지표 집합을 의미한다고 가정한다. 탐색결과 Fig. 4(a)에선 판재 경계곡선 2,3,4번과 공구 경계곡선 4,5,6이 연속적으로 일치하고 있음을 보인다. Fig. 4(b)에선 판재 경계곡선 3,4,...,8번과 공구 경계곡선 3,4,...,8이 연속적으로 일치하고, 공구의 위치를 바꾸면 판재 경계곡선 10,11,12번과 공구 경계곡선 10,11,12가 일치됨을 아울러 나타낸다.

이상의 예에서도 나타나듯이 SIS 리스트 비교에 의한 일치곡선군의 인식 방법의 장점은 직선 및 원호로 이루어진 곡선군 사이에서 서로 일치하는 가장 길게 연속한 곡선군을 찾는데 있어 다른 어떤 방법보다도 간단하며 효율적이라는 점이다. 후에 3.5.1 절에서 기술될 Sharir[16]의 방법은 두 곡선 사이의 서로 일치되는 부분을 찾는 알고리즘으로서, 비교하고자 하는 두 곡선위에 등간격의 점들을 생성하여 이들 점들 간의 리스트 스퀘어 피트(least-square-fit) 여부를 계산한다. 그러나, 이 알고리즘은 직선 및 원호 혹은 이들의 조합으로 이루어진 곡선군에 대해선 계산량이 많아 비효율적이다[8]. 따라서, 본 연구에서는 자유곡선의 일치검사에만 Sharir의 알고리즘을 이용한다.

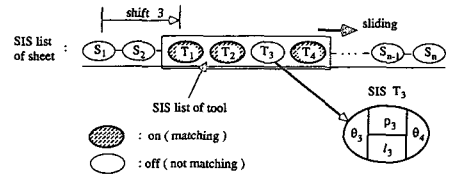


Fig. 3 Pattern matching using shape-index-set lists of sheet and tool

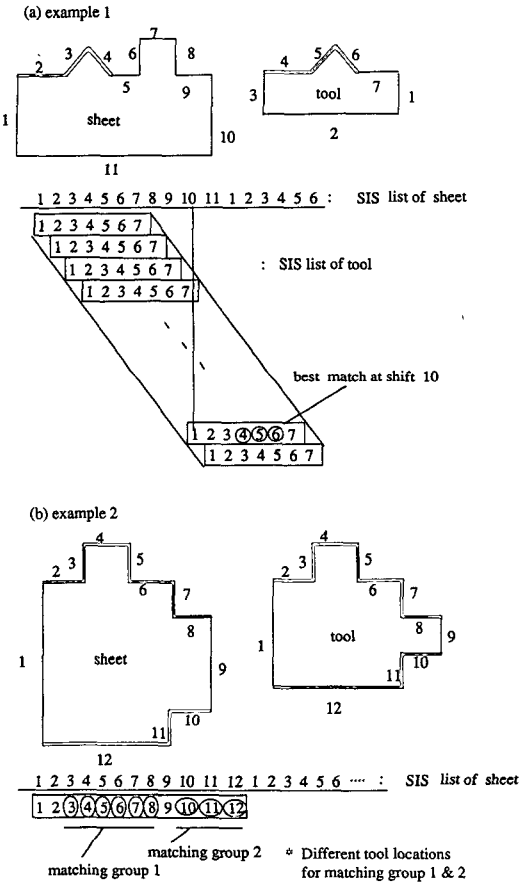


Fig. 4 Examples of pattern matching using SIS list

3.4 후보 공구의 선정

펀칭되어야할 판재형상의 전부 또는 그 일부분과 일치하는 공구를 후보공구라 한다. 만약 후보공구가 존재하면 그 공구의 자동탐색이 반드시 성공해야하고 해당 부분의 정확한 펀칭을 위한 후보공구의 직선, 회전 운동량이 자동으로 결정되어야 한

다. 본 연구에선 이를 위해 앞에서 소개된 형상지표 집합 및 일치 곡선군의 인식기법을 이용하여 다음과 같은 계층적 후보 공구 탐색에 의한 자동 공구 선정 알고리즘을 개발하였다. 각 단계의 구체적인 설명은 다음과 같다.

1단계) 아직 펀칭되지 않은 판재의 경계곡선(punch-flag='off')들로 부터 연속한 *SIS* 리스트 $S_i(i=1,...,n)$을 구성한다. 즉, 각각의 S_i는 연속 곡선들의 형상지표 집합들이다.

2단계) 앞의 3.3절에 의해 모든 공구의 *SIS*리스트 $T_j^k(j=1, ..., m(k),\ k=1, ..., ntool)$ 중에서 S_i의 일부와 허용치 내에서 일치하는 T_j^k의 부분들로 아래와 같은 집합,

$$MPL=\left\{\left(T_{j1}^{k1},\ T_{j1+1}^{k1}, ...\right),\ \left(T_{j2}^{k2},\ T_{j2+1}^{k2}, ...\right),\ ...\right\}$$

을 구성한다. 여기서 $m(k)$는 k번째공구의 경계를 구성하는 곡선의 갯수이고, $ntool$은 펀치공구의 총 갯수이다. 후보 공구의 우선순위는 연속적으로 일치하는 곡선의 수(match-count)가 큰 순서로 하여, 필요한 만큼을 후보 공구 집합으로 선택한다. 이때 하나의 공구와 판재 경계곡선의 여러 곡선군이 일치하는 경우도 있을 수 있다. 즉, 한개의 공구로 판재의 여러 부분을 펀칭할 수 있는 경우가 있기 때문이다. 위의 비교과정을 수행하면서 펀치공구의 정확한 이동 및 회전량을 계산하기 위해 서로 일치하는 곡선 혹은 곡선군의 쌍을 저장한다. 선택된 후보 공구들은 간섭검사 및 펀칭 모듀율로 옮겨져 간섭검사를 통과한 공구로 펀칭한다. 만약, 본 단계에서 후보 공구을 하나도 얻을 수 없거나, 어느 후보 공구도 간섭검사를 통과하지 못하는 경우 다음 단계를 차례로 수행한다.

3단계) *SIS*리스트 S_i와 T_j^k사이에 $\{\rho_{i-1},\ \theta_i,\ \rho_i\}$와 $\{\rho_{j-1}^k,\ \theta_j^k,\ \rho_j^k\}$가 일치하여 판재 모서리의 특정 각도를 펀칭 할 수 있는 후보공구 집합을 구성하여 앞에서 처럼 간섭검사 및 펀칭을 수행한다. 이 경우는 Fig. 5에 보인 펀칭 작업에 해당하는데, 펀칭 횟수를 최소화하기 위해 후보공구의 우선순위는 그림에서 처럼 공구경계와 일치하며 펀칭 가능한 형상경계의 길이(overlap-length)가 큰 순서로 한다. 따라서, Fig. 5와 같은 경우 $t1$을 첫번째 후보공구로 하여 간섭검사 및 펀칭을 수행한다. 만약 $t1$이 간섭검사를 통과하지 못하여 펀칭에 실패하면 두번째 공구 $t2$로 같은 과정을 반복한다. 이때 후보공구가 하나도 없거나, 어느 후보공구도 간섭검사를 통과하지 못하면 다음 단계로 간다.

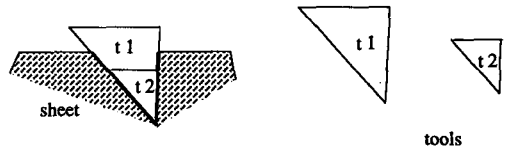


Fig. 5 Example of punching operation

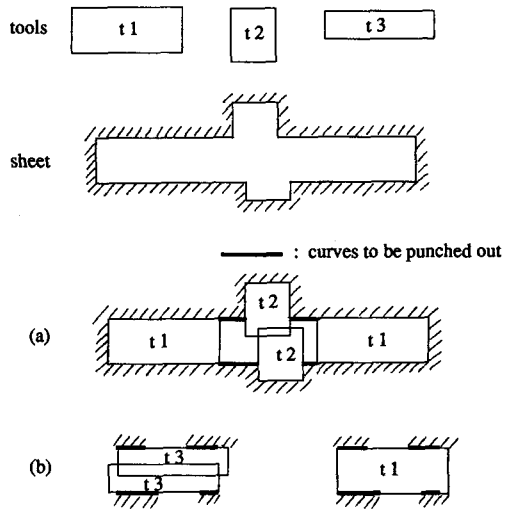


Fig. 6 Example of punching operation

4단계) *SIS*리스트 S_i와 T_j^k사이에 ρ_i와 ρ_j^k가 같아 판재경계 곡선과 곡률반경이 일치하는 곡선을 갖는 후보공구들을 찾아 간섭검사 및 펀칭을 수행한다. 앞에서와 마찬가지로 후보공구의 우선 순위는 공구경계와 일치하며 펀칭 가능한 형상경계의 길이(overlap-length)가 큰 순서로한다. 이 경우는 Fig. 6(b)에 보인 펀칭 작업에 해당된다. 즉, Fig. 6(a)는 공구 $t1$, $t2$의 조합으로 네번의 펀칭작업을 수행한 직후의 상태로서, 아직 펀칭되지 않은 부분이 굵은선으로 표시되어 있다. 이들을 펀칭하기 위한 여러가지의 공구조합 중에서 Fig. 6(b)는 그 일부를 나타내고, $t1$이 가장 좋은 후보공구임을 보인다. 이때에도 후보공구가 하나도 없거나, 어느 후보공구도 간섭검사를 통과하지 못하면 다음 단계로 간다.

5단계) 이 단계까지 오면 시스템은 현 공구대(current turret)에 장착되어 있는 공구로는 현재의 판재경계를 정확하게 펀칭할 수 없음과, 펀칭작업

을 계속하기 위해선 허용오차 내에서의 근사펀칭(nibbling)이 요구됨을 사용자에게 알린다. 이때 사용자가 근사펀칭하기를 원할 경우, 시스템은 원형 혹은 직사각형 등의 근사펀칭용 공구의 형상 및 크기를 사용자가 대화식으로 선택하게 하거나, 이미 지정된 것을 이용할 수 있도록 한다. 선택된 공구가 원형이면 시스템은 허용오차 내에서 근사펀칭하기 위한 공구의 위치 계산 및 간섭검사를 자동으로 수행한다. 만약 곡선경계를 직선경계로 근사펀칭 하고자 한다면, 시스템은 해당 경계를 허용오차 내에서 근사시키는 직선군을 자동으로 생성시켜 해당 경계곡선을 이것으로 대체한 후, 앞의 1~4단계를 반복적으로 수행함으로써 근사펀칭을 완료한다.

위의 어느 단계에서라도 펀칭이 성공하면 판재형상의 경계곡선 리스트는 갱신되고 이들이 모두 펀칭될 때까지 위의 과정을 반복한다.

3.5 간섭검사 및 펀칭

앞에서 선정된 후보 공구로 펀칭하기 위해선 공구의 정확한 위치및 방향을 계산하여야 하고, 공구가 판재의 다른 부분을 잘못 펀칭 하지는 않는지를 검사하여야 한다. 이런 검사과정을 간섭검사라 하는데, 펀칭은 이 검사를 통과한 공구만으로 수행된다. 여기서 펀칭과정은 프로그램 상에서 공구경계와 일치하는 모든 판재경계곡선들을 찾아 그들의 펀칭 플래그(punch_flag)를 'on'으로 바꾸어 주는 작업이다.

이때 후보 공구에 의해 펀칭되는 판재 경계곡선들은 후보공구 탐색시 일치하는 것으로 발견된 곡선들의 갯수보다 더 많아 질 수 있으므로 공구경계와 일치하는 모든 판재경계 곡선을 찾아야 한다. Fig. 7는 이에 대한 예시로, Fig. 7(a)는 판재 경계와 부분적으로 일치하는 공구라 할지라도 간섭이 발생하여 펀칭에 실패하는 경우이고, Fig. 7(b)는 후보공구 탐색결과(굵게 표시된 곡선) 보다 더 많은 판재경계 곡선(점선 표시곡선)들이 펀칭되어 질 수 있음을 보인다. 따라서, 선정된 후보공구를 이용하여 펀칭을 완료하기까지는 다음 과정을 거쳐야 한다.

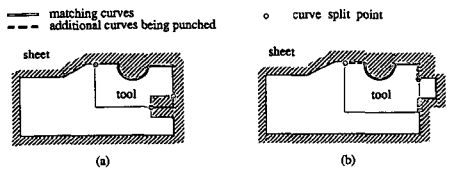
- 공구의 회전, 직선 운동량 계산
- 간섭검사 및 공구경계와 일치하는 판재 경계곡선의 탐색 및 갱신
- 펀칭 정보 저장

(1) 공구의 직선, 회전 운동량 계산

Sharir[16]는 두 곡선의 일치 여부를 판단하기 위해 다음과 같은 알고리즘을 개발하였다. Fig. 8(a)의 곡선 C, C′ 위에서 각각의 점 $\vec{u}_j$, $\vec{v}_j(j=1,2,\ldots,n)$를 등간격으로 구했을 때, 곡선 C를 적절히 이동시켜 곡선 C'에 가장 근접되도록 하는 유클리디안 변환, E,를 다음과 같이 정의된 최소거리자승(least square distance), Δ를 최소화 하도록 결정한다.

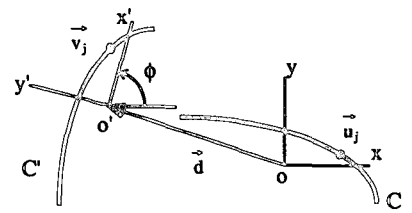
$$\Delta=\min_{E}\sum_{j=1}^{n}|E\vec{u}_j-\vec{v}_j|^2,\ E(\vec{d},\phi)\ :\ \text{Euclidean Transformation}$$

여기서, $\vec{d}$는 곡선 C의 직선이동 벡터이고, ϕ는

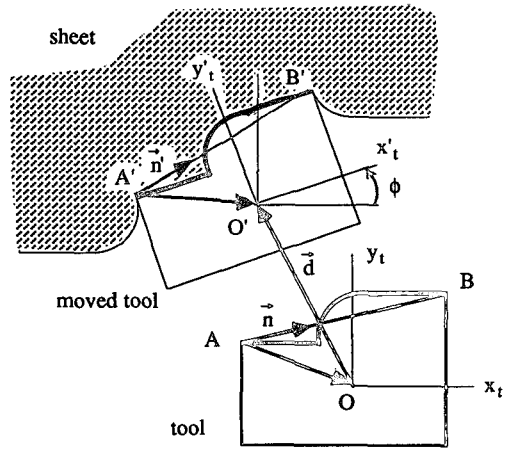


(a) Free-curve matching by Sharir's algorithm
(b) Translation and rotation of the tool

Fig. 7 Interference test and punching



(a) Free-curve matching by Sharir's algorithm



(b) translation and rotation of the tool

Fig. 8 Calculation of the translation and rotation for tool

회전각도를 나타낸다.

위의 Δ를 최소화 하는 $E(\vec{d},\ \phi)$는

$$\vec{d}=\frac{1}{n}\sum_{j=1}^{n}\vec{v}_j,\quad \phi=-\theta$$

로 얻어진다. 이때 θ는 $r^{i\theta}=\sum_{j=1}^{n}u_j\bar{v}_j$에 의해 구해지는데 u_j, v_j는 이차원 벡터 $\vec{u}_j, \vec{v}_j$를 복소수로 나타낸 것이다. 또한, Δ는 다음과 같이 계산된다.

$$\Delta=\sum_{j=1}^{n}|\vec{v}_j|^2-\frac{1}{n}\left|\sum_{j=1}^{n}\vec{v}_j\right|^2+\sum_{j=1}^{n}|\vec{u}_j|^2-2\left|\sum_{j=1}^{n}u_j\cdot\bar{v}_j\right|$$

점 $\vec{u}_j$, $\vec{v}_j$가 정확히 대응되는 경우에는 위의 Δ는 0에 수렴한다. 그러나, 이 알고리즘은 직선 및 원호 혹은 이들의 조합으로 이루어진 곡선군에 대해선 계산량이 많아 비효율적이다[8]. 따라서, 본 연구에서는 자유곡선인 경우에 대해서만 곡선간의 일치여부 검사 및 후보공구의 운동량 계산에만 위의 알고리즘을 이용한다.

반면에 곡선 형태가 직선이나 원호일 경우에는 아래와 같은 간단한 벡터 연산으로 후보공구의 회전, 직선 운동량을 결정한다. 즉 Fig. 8(b)에서와 같이 O, O′를 각각 공구중심 및 그의 이동된 위치라 하고, 서로 일치하는 곡선위의 임의의 대응되는 두점을 A,B 및 A′,B′라 하면, 공구의 회전량 ϕ와 직선운동량 $\vec{d}$는 다음과 같이 구해진다. 벡터 $\overrightarrow{AB}$와 $\overrightarrow{A'B'}$가 이루는 각이 ϕ이고, 벡터 $\overrightarrow{AO}$를 ϕ 만큼 회전시킨 것이 $\overrightarrow{A'O'}$ 이므로 직선운동량 $\vec{d}$는

$$\vec{d}=\overrightarrow{OA'}+\overrightarrow{A'O'}$$

로 계산된다. 여기서 회전각 ϕ의 계산을 위해, 직각좌표(x,y)를 극좌표$(r,\ \theta)$로 표현할 때 동경 θ를 $-\pi\le\theta\le\pi$ 범위에서 계산해 주는 C 언어 제공함수 atan2()를 이용하면 계산이 간편하다. 즉, 벡터 $\overrightarrow{AB}$ 및 $\overrightarrow{A'B'}$ 방향의 단위 벡터를 각각 $\vec{n}$, $\vec{n}'$라 하면 ϕ는 다음과 같이 구해진다.

$$\phi=\text{atan2}(\vec{k}\cdot(\vec{n}\times\vec{n}'),\ \vec{n}\cdot\vec{n}')$$

여기서 $\vec{k}$는 경계곡선이 놓인 평면에 수직한 단위 벡터이다.

(2) 간섭검사 및 판재 경계곡선의 탐색 및 갱신

알고리즘이 간단하고 수행시간이 빠른 다음과 같은 간섭검사 방법을 개발, 사용하였다. 이는 판재 경계 곡선과 공구경계곡선 간의 교점에서 곡선간의 사이각 비교방법으로 다음과 같다.

판재와 공구의 경계곡선 간의 교점 p가 존재하면, 그 교점을 기준으로 곡선들을 분할하여, Fig. 9와 같이 각 곡선에 대응되는 4개의 접선을 t_1, t_2, g_1, g_2로 취한다.

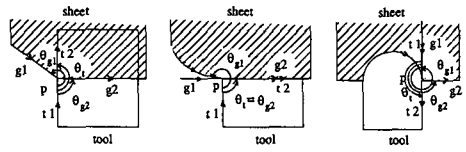


Fig. 9 Interference test between sheet and tool

교점 p에서 접선 t_1을 기준으로 각각의 접선까지 반시계 방향으로 계산된 각도를 θ_t, θ_{g1}, θ_{g2}라 할 때, 간섭이 일어나지 않기 위한 조건은 다음과 같다.

$$0<\theta_t\le\theta_{g2}<\theta_{g1}\le2\pi$$

간섭 검사를 통과한 공구에 의해 제거될 판재경계곡선들의 탐색 및 펀칭은 다음과 같이 수행된다.

공구의 간섭검사를 수행할 때 나타나는 모든 교점에 대해, 그 교점이 경계곡선의 양 끝점이 아닌 곡선 내부에 생길경우, 그곡선을 해당 교점(Fig. 7에서 ●표시)에서 임시로 분할한다.

판재의 모든 곡선과 후보공구의 모든 곡선간의 일치여부 검사를 하여 일치하는 것으로 확인되는 판재의 경계곡선(Fig. 7에서 굵은 선 및 점선 표시 곡선)들을 탐색하여 후보공구 번호, 펀칭되어 제거될 곡선의 수(match-count) 및 그들의 전체 길이(overlap-length) 등의 펀칭정보를 임시 펀칭버퍼(temp-punch-buffer)에 저장한다. 임시 펀칭버퍼에 저장된 모든 후보공구에 대한 정보를 이용하여 가장 바람직한 후보공구(최대 match-count 혹은 최대 overlap-length의 경우)를 선택하여, 이에따라 판재 경계곡선에 대한 이중 연결리스트를 갱신시키고 최종적으로 판재 경계곡선의 펀칭 플래그를 'off'에서 'on'으로 바꾼다. 이때 곡선간의 일치여부 검사는 곡선형태에 따라서, 직선 및 원호 일때는 곡선의 양 끝이 일치하고 길이가 같고 부호를 포함한 곡률 반경이 같으면 일치로 판정하고, 자유곡선인 경우 앞의 3.5.1절에서 소개된 Sharir의 알고리즘을 이용한다.

(3) 펀칭 정보 저장

펀칭정보는 공구 번호, 직선 운동량, 회전 운동량으로 구성되며, 매 펀칭시 스택에 저장하여 추후 판재부품 배치(nesting), 공구경로의 최적화 및

NCT 구동을 위한 천공 테이프 작성 등에 사용되도록 한다. 또한, 모든 펀칭결과는 곧바로 화면에 그래픽으로 나타내도록 하였다.

4. 전산실험 결과 및 검토

알고리즘 구현은 C언어로 하였고, 공구는 모두 회전가능하다는 조건하에서 전산실험을 수행하였다. Fig. 10은 본 알고리즘 테스트를 위해 사용한 펀치공구의 예이고, 이들을 이용한 자동 공구선정 결과를 Fig. 11~16에 보였다. 또한, Table 1에 각 실험 예에서 선정된 공구의 종류, 전체 펀칭 횟수 및 공구선정에 소요된 시간 등을 나타냈다.

예제에서 보는 바와 같이 판재 형상과 일치하는 공구가 있을 때는 한번도 실패하지 않고 해당공구의 탐색이 성공하고 있음을 알 수 있다. 또한, 최적의 공구조합에 대한 평가기준은 아직 마련되어 있지 않으나, 공구조합의 최적화를 위한 노력의 일환으로 펀칭횟수를 최소화 하도록 하였다. 이러한 관점에서 볼 때, 전체적인 펀칭결과도 불필요한 펀칭이 거의 없는 매우 만족스러운 것으로 나타났다.

공구와 공구가 판재경계 곡선 위의 한 점에서 만

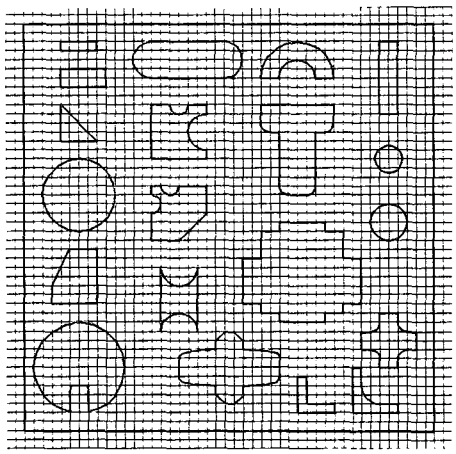


Fig. 10 Examples of tools

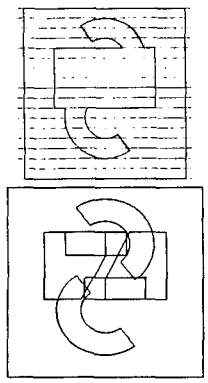


Fig. 11 Tool processing result-example 1

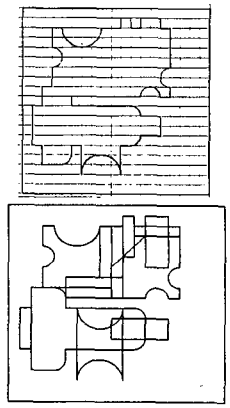


Fig. 12 Tool processing result-example 2

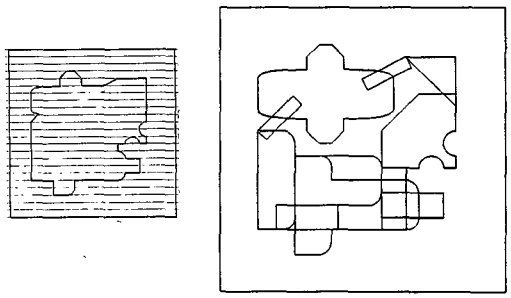
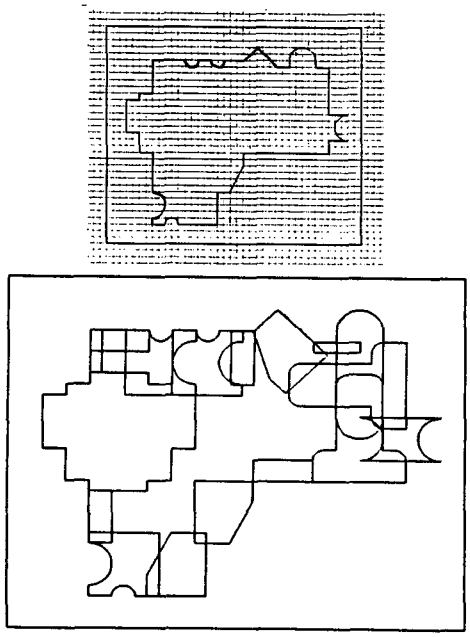


Fig. 13 Tool processing result-example 3

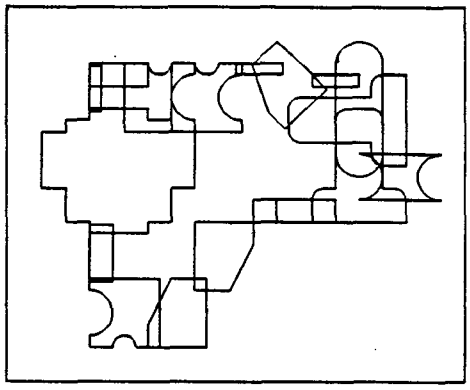
Table 1 Results of automatic tool processing

Example No.	No.of Tools	No.of Hits	Time* (sec)
1	3	6	12
2	6	11	27
3	6	10	29
4(a)	8	17	65
4(b)	8	18	67
5(a)	5	34	76
5(b)	6	41	62
6	10	55	351

*total CPU time on 15 MIPS Unix workstation



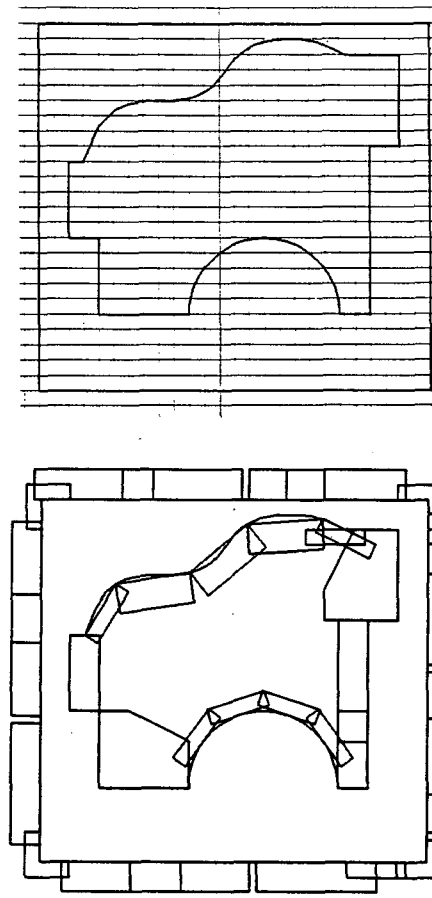
(a)



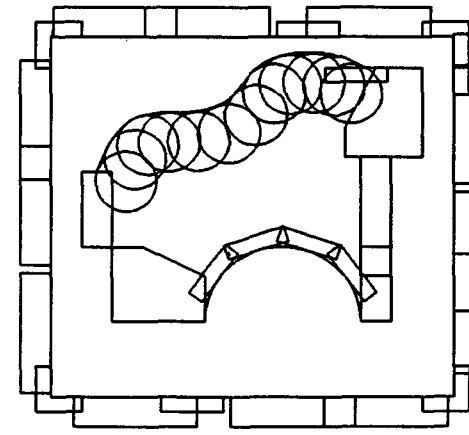
(b)

Fig. 14 Tool processing result-example 4

나는 경우, 판재경계의 펀칭 단면엔 크건 작건 간에 거스러미(burr)가 발생하고, 이는 현재의 NCT 기술로는 피할 수 없는 단점으로 지적되고 있다. 다만 현장에선 적절한 공구 중첩(overlap-hit)을 통하여 거스러미 크기를 최소화시키고자 노력하고 있다. Fig. 14(a)는 공구 중첩에 대한 고려가 없이 공구를 선정한 결과임에 반하여, Fig. 14(b), 15, 16은 공구 중첩이 시스템에 의해 자동으로 고려된 공구선정 결과를 보인다.

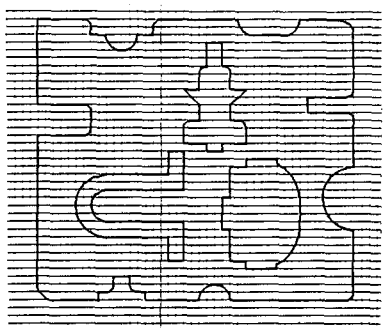


(a)



(b)

Fig. 15 Tool processing result-example 5



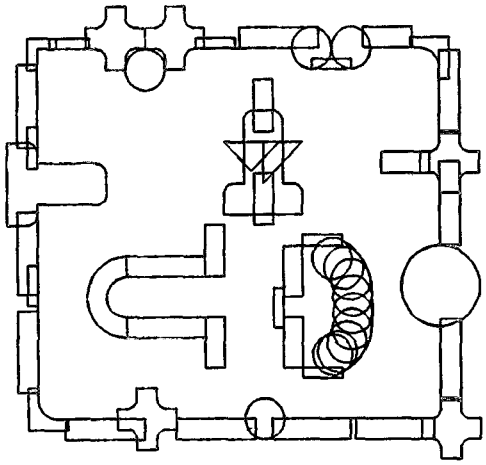


Fig. 16 Tool processing result example 6

Fig. 15~16은 본 연구에서 구현된 여러 종류의 근사펀칭 결과를 보인다. 일반적으로 실무에선 근사펀칭을 위한 수동 공구선정 작업시 공구의 정확한 위치 및 방향 계산의 번거로움 때문에 특정 각도로, 예컨대 45도의 배수 각도 등, 공구 방향이 고정된 근사펀칭만 허용하고, 사용공구 종류도 원형 및 직사각형으로 한정되어 있어 판재부품 설계의 유연성(flexibility)을 크게 제한하고 있다. 이러한 문제점은 현재의 NCT 하드웨어(hardware)의 한계성에도 기인한다. 즉, 현재 보급되어 사용중인 NCT 기계의 공구대(turret)에 장착되는 공구수는 기종 및 용량에 따라 차이는 있지만 대략 40~65개임에 비해 근사펀칭에 사용 가능한 자동회전가능 공구(auto-indexing-tool)의 장착기능은 일반적으로 2개 이내이다. 그러나, 향후 NCT 기계가 개선되어 자동 회전가능 공구수를 늘릴 경우 본 연구결과는 판재부품 설계의 유연성을 향상시킴은 물론 궁극적으로 판재부품 제작시의 생산성 제고에 크게 기여할 것으로 기대된다.

전체적인 작업효율을 높이기 위해 판재 부품의 외곽 펀칭시 완전히 따내지 않고 연결부분(bridge)을 몇 군데에 조금씩 남겨서 펀칭작업 도중엔 판재부품이 떨어지지 않고 철판원단에 붙어 있도록 해야한다. 이를 위해 시스템은 연결부분의 폭과 개략적인 위치를 그래픽 터미널 상에서 사용자로부터 대화식으로 간단히 입력받아 연결부분을 자동생성시킨 후 외곽 펀칭을 시작한다. Fig. 15~16는 이를 고려한 펀칭결과들을 보인다.

결 론

여러번의 전산실험을 통해, 본 연구에서 도입한 형상지표집합(shape-index-set)과 이를 이용한 형상 인식 방법의 강건성(robustness)을 확인할 수 있었다.

또한, 이러한 형상인식 방법에 기초한 계층적 공구탐색 알고리즘을 개발하여, NCT 구동을 위한 펀치공구 선정의 완전자동화를 위한 기틀을 마련하였다. 특히 본 연구결과는 판재경계와 정확히 일치하는 공구가 없을 경우에는 허용오차 내에서 근사펀칭(nibbling)을 할 수 있는 모듀율(module)을 포함하고 있어, 앞으로 NCT 기계의 자동 회전가능 공구(auto-indexing-tool) 장착능력이 개선될 경우 판재부품 분야에서의 설계의 유연성 및 제작시의 생산성 제고에 크게 기여할 것으로 기대된다.

이에따라, 종래의 수동 혹은 반자동에 의한 공구선정으로 야기되었던 제반 문제점, 예컨대, 공구선정의 번거로움 때문에 판재 소모를 최소화하기 위한 최적배치에 제한이 따른다거나, 수동 공구선정을 할 수 있는 소수의 숙련자에게 생산성이 종속하게 되는 등의 문제점을 해결할 수 있게 되었다.

현재 본 연구결과가 현장의 생산성에 실질적으로 기여할 수 있도록 다음 사항들에 대한 추가 연구가 계속되고 있다.

· 구속조건하의 공구경로의 최적화 문제(constrained tool path optimization problem) : 공구의 자동선정 알고리즘과는 무관한 일이나, 저장된 펀칭정보를 이용하여 전체 펀칭작업 시간을 최소화하는 공구경로의 최적화가 이루어져야 한다. 이때 작은 공구의 펀칭을 우선적으로 고려하여 공구의 옆밀림이나 공구손상을 방지토록 하고 정밀도가 요구되는 펀칭부위는 가능한 나중에 펀칭되도록 펀칭순서가 결정되어야 한다.

· 구속조건을 갖는 최적배치 문제(constrained optimal nesting problem) : 다양한 형상의 판재부품들의 배치에 있어 펀칭 여유를 고려한 판재소모(scrap) 및 펀칭횟수의 최소화 문제가 해결되어야 한다.

후 기

본 연구는 91학년도 교육부 학술연구 조성비 지원에 의한 것이며, 이에 관계자 여러분께 심심한 감사를 드립니다.

참고문헌

(1) Freeman, H., 1978, "Shape Description Via the use of Critical Points," Pattern Recognition, Vol. 10, pp. 159~166.

(2) Ayache, N. and Faugeras, O.D., 1986, "HYPER : A New Approach for Recognition & Positioning of 2D Objects," IEEE Trans. on Pattern Analysis & Machine Intelligence, Vol. PAMI-8, Jan., pp. 44~54.

(3) Kalvin, A., Sharir, M. et al., 1986, "2D model Based Boundary Matching Using Footprints," Int. J.Robotics Res., Vol. 5, No. 4, pp. 38~55.

(4) Hong, J. and Wolfson, H.J., 1988, "An Improved Model-Based Matching Method Using Footprints," Proc. Int. Conf. Pattern Recognition, Rome, Italy, Nov., pp. 72~78.

(5) Ansari, N. and Delpt, E. J., 1990, "Partial Shape Recognition : A Landmark Based Approach," IEEE Trans. on Pattern Analysis & Machine Intelligence, Vol. 12, No. 5, May, pp. 470~483.

(6) Niemann, H., 1981, "Pattern Analysis," Springer-Verlag, Berlin, Germany

(7) Schmidt, W., 1990, "Modified Matched Filter for Cloud Clutter Suppression," IEEE Trans. on Pattern Analysis & Machine Intelligence, Vol. 12, No. 6, June, pp. 594~600.

(8) Wolfson, H. J., 1990, "On Curve Matching," IEEE Trans. on Pattern Analysis & Machine Intelligence Vol. 12, No. 5, May, pp. 483~489.

(9) Reeves, A.P. et al., 1988, "3D-Shape Analysis Using Moments & Fourier Descriptors," IEEE Trans. on Pattern Analysis & Machine Intelligence Vol. 10, No. 6, Nov., pp. 937~943.

(10) Dudani, S.A. et al., 1977, "Aircraft Identification by Moment Invariants," IEEE Trans. Comput. Vol. C-26, Jan., pp. 39~46.

(11) Lin, C.C. and Chellapa, R., 1987, "Classification of Partial 2-D Shapes Using Fourier Descriptors," IEEE Trans. on Pattem Analysis & Machine Intelligence, Vol. PAM1-9, No. 5, Sep, pp. 686~690.

(12) Granlund, G.H., 1972, "Fourier Preprocessing for Hand Printed Character Recognition," IEEE Trans. Comput. Vol. C-21, Feb., pp. 195~201.

(13) Wallace, T.P. and Wintz, P.A., 1980, "An Effcient 3D Aircraft Recognition Algorithm Using Fourier Descriptors," Comput. Graphics Image Processing, Vol. 13, pp. 99~126.

(14) Gorman, J. W. et al., 1988, "Partial Shape Recognition Using Dynamic Programming," IEEE Trans. on Pattern Analysis & Machine Intelligence, Vol. 10, No. 2, March, pp. 257~266.

(15) Dubois, S.R. and Glanz, F.H., 1986, "An Autoregressive Model Approach to 2D Shape Classification," IEEE Trans. on Pattern Analysis & Machine Intelligence Vol. PAM1-8, No. 1, Jan., pp. 55~66.

(16) Schwartz, J.T. and Sharir, M., 1987, "Identification of Partially Obscured Objects in two & Three Dimensions by Matching of Noisy Characteristic Curves," Int. J.Robotics Res. Vol. 6, No. 2, pp. 29~44.

(17) Kimura, F. et al., 1987, "Automatic Process Planning for Sheet Metal Parts with Bending Simulation," Intelligent & Integrated Manufacturing Analysis & Synthesis, ASME PED-Vol. 25, pp. 245~258.

(18) BRAVO 3, 1988, "Sheet Metal Design/Fabrication User's Guide" U.S.A.

(19) EUCLID-IS, 1987, "Sheet Metal Module-Reference Manual," Matra Datavision, France

(20) Tiller, W., 1983, "Rational B-Spline Curve and Surface Representation," IEEE, CG A, sep. pp. 61~69.