공구 경로 생성을 위한 아일랜드를 포함하는 영역의 오프셋

박 상 철*

(2001년 3월 8일 접수, 2001년 9월 26일 심사완료)

Offsetting a Region Including Islands for Tool-Path Generation

Sang-Chul Park

Key Words : Region Offset (영역 오프셋), Tool-Path Generation (공구경로 생성), PWID Offset (PWID 오프셋), Pocket Machining (포켓 가공)

Abstract

This paper presents a region offsetting algorithm for tool-path generation. The proposed region offsetting algorithm is developed by expanding the 'PWID offset algorithm [Choi and Park, 1999]' designed to offset a simple polygon. The PWID offset algorithm has three important steps; 1) remove 'local invalid ranges' by invoking a PWID test, 2) construct a raw offset curve and 3) remove 'global invalid ranges' by finding self-intersections of the raw offset curve. To develop a region offsetting algorithm, we modified the PWID offset algorithm by expanding the concept of the 'global invalid range' in the third step. The time complexity of the proposed algorithm is approximately $O(n)$, where n is the number of points, and it is free of numerical errors for practical purposes. The proposed algorithm has been implemented and tested with various real regions obtained by intersecting a sculptured surface with a plane.

1. 서 론

다각형들로 정의되는 영역의 오프셋은 아직도 활발히 연구되고 있는 대표적인 이차원 기하문제 중의 하나이다. 이 문제에 대해서 지난 20 여년간 수 많은 알고리듬들이 제안되었음에도 불구하고 여전히 현실적으로 구현하기 어려운 문제일 뿐 아니라 여러 분야에 걸쳐서 이차원 영역의 오프셋이 요구되기 때문이다. 대표적인 응용분야로는 캐드캠 분야의 공구경로 생성,[1~8,14] 가공영역 모델링[6,13] 그리고 컴퓨터그래픽스의 곡선 모델링 및 폰트디자인 등이 있다. 또한 로보틱스에서의 충돌 없이 움직이는 로봇의 경로를 구하는 문제 그리고 반도체 디자인에서의 칩 설계 등에 활용될 수도 있다.

특히 본 논문은 NC 가공을 위한 공구경로 생성에 영역 오프셋을 이용하는 것에 초점을 두고 있다. 영역 오프셋을 이용하여 생성되는 NC 공구경로의 대표적인 예는 포켓, 황삭 가공[3,4,6,14,15]이며 이는 몰드금형의 가공을 위해서는 필수적이라 할 만큼 중요한 공정이다. Fig. 1 은 영역 오프셋을 이용한 포켓가공 경로의 예를 보여준다. 이때 필요한 오프셋이 다각형을 대상으로 한다고는 하지만 실제로는 **'수천(수만)개의 점으로 이루어진 다각형'**일 가능성이 많기 때문에 수치에러가 유발될 수 있고 따라서 오프셋이 쉽지 않은 것이다.

본 연구에서는 아일랜드를 포함하는 이차원 영역을 오프셋 할 수 있는 알고리듬을 제안하는 것을 목표로 한다. 논문의 구성은 다음과 같다.

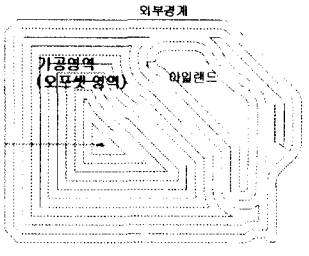

Fig. 1 Tool-path generated by region offsetting

* 회원, 큐빅기술연구소
E-mail : psc@cubictek.com
TEL : (02)3664-4700 FAX : (02) 3664-4701

우선 하나의 다각형(simple polygon)에 대한 기존 오프셋 알고리듬들을 다음 장에서 살펴보고, 제안된 알고리듬의 설명을 위한 용어들을 정의한다. 그리고 본 논문에서 제안한 영역 오프셋 알고리듬의 기반이 되는 '**PWID 오프셋 알고리듬**'[1]을 자세히 설명하도록 한다. 그리고 설명된 PWID 오프셋 알고리듬을 확장하여 개발된 영역 오프셋 알고리듬을 설명하고 구현된 알고리듬을 이용한 공구경로 생성의 예제와 더불어 결론을 맺고자 한다.

2. 다각형 오프셋에 대한 기존 연구

기존의 다각형 오프셋에 관한 연구는 크게 세 가지 방식으로 나눌 수 있는데, Voronoi diagram,[6,8,13] Pair-wise[1,5,11,12,14] 그리고 Pixel based[7,10] 방식이다. 이 세가지 방식들의 특징과 장단점에 대해서 기술하면 다음과 같다.

Voronoi diagram 방식은 다각형을 구성하는 선분들의 Voronoi diagram 을 구하여 오프셋을 수행하며 오프셋 결과의 자체 꼬임이나 유효하지 않은 고리(loop)의 발생과 같은 문제를 피할 수 있는 장점이 있다. 또한 한번 Voronoi diagram 을 구성하게 되면 임의의 오프셋 거리가 주어져도 일정 시간에 결과를 구할 수 있으므로 하나의 다각형에 대해서 매우 많은 횟수의 오프셋 하는 경우에 경제적이라 할 수 있다. 그러나 이 장점은 다르게 이야기 하면 한번만 오프셋 하는 경우에는 Voronoi diagram 이 오히려 비 경제적일 수도 있다는 말이 된다. n 개의 점(혹은 선분)으로 이루어진 다각형의 Voronoi diagram 은 $O(n \cdot logn)$ 시간에 구할 수 있는 것으로 알려져 있으나,[13] Fig. 2 에서와 같은 *near circular singularity* 가 발생하는 부분에서는 수치에러 때문에 Voronoi diagram 을 구성하는 것이 실제적으로 매우 어렵다.[9] 물론 원호 근사를 통하여 어느 정도는 near circular singularity 를 해결할 수 있겠지만, 주어진 공차범위 내에서 촘촘한 점들을 원호들로 잘 근사 하는 문제는 결코 실제적으로 쉬운 문제가 아닐 뿐더러, 때로는 주어진 점들의 특성에 따라 원호 근사가 거의 효과가 없는 경우도 충분히 있을 수 있다.

일반적으로 Pair-wise 오프셋 방식은 다음과 같은 절차를 따른다. 1) 다각형을 구성하는 각 선분들을 각각 오프셋 하고, 불연속이 되는 부분은 원호로 채워서 raw offset curve(초기 오프셋)를 구성한다. 2) raw offset curve 의 꼬인 점들을 모두 구한다. 3) 구해진 꼬인 점들을 기준으로 raw offset curve 를 여러 개의 고리들로 분할한다. 그리고 4) 유효하지 않은 고리들을 제거하고 남은 고리들만

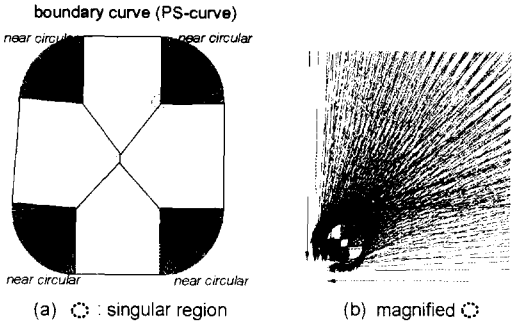

Fig. 2 Near circular singularity in Voronoi diagram

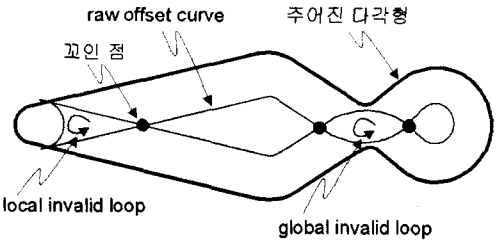

Fig. 3 Local invalid loop and global invalid loop

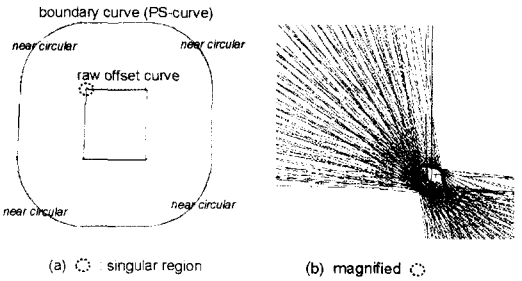

Fig. 4 Near circular singularity in pair-wise offset

을 유효한 오프셋 결과로 출력한다. 이러한 일련의 과정이 개념상으로는 간단하지만 실제적으로 상당히 많은 계산시간을 요구하며, 수치에러에 민감하므로 실제적으로 잘 작동하게 구현하기가 힘들다. Fig. 3 에서 나타나 있듯이 유효하지 않은 고리에는 local invalid loop 와 global invalid loop 가 있다. 일반적으로 raw offset curve 에서 하나의 꼬인 점으로 막힌 것을 local invalid loop 라고 하고 두개 이상의 꼬인 점으로 막힌 것을 global invalid loop 라고 한다. 기존의 연구들은 주로 유효하지 않은 고리들을 효율적으로 제거하는 문제에 집중하였으며, 그 결과 많은 휴리스틱 알고리듬들이 제안되었다. 그러나 이러한 알고리듬들이 반드시 정확한 결과를 보장하는지는 못하는 것이어서, 고리의 유효성 여부를 확실히 검증하기 위해서는 원래의 주어진 다각형과의 거리 비교를 해야만 하는 것으로

알려져 있다. 하지만 Pair-wise 오프셋 방식에서 가장 심각한 문제는 Fig. 4 과 같은 상황에서 발생하는 near circular singularity 라고 할 수 있다. Voronoi diagram 방식과 비슷한 어려움이지만, 실제 계산상의 어려움은 더 심각해서, raw offset curve 의 꼬임점들 모두 구하는 것 자체가 불가능해진다.

Pixel based 방식은 오프셋 결과를 이차원 pixel 로 근사하여 구하는 방식이며, 안정적이라는 점과 꼬인 점 그리고 유효하지 않은 고리와 같은 문제를 피할 수 있다는 장점을 가지고 있다. 그러나 정밀도를 높이려면 필요한 메모리가 폭발적으로 늘어날 수 있다는 단점을 안고 있어 효용에 한계점을 드러낸다.

이러한 어려움들을 극복하기 위해 최근에 제안된 다각형 오프셋 알고리듬이 '**PWID 오프셋**'[1] 알고리듬이다. PWID 오프셋 알고리듬은 Pair-wise 방식에 속한다고 볼 수 있지만 기존의 어려움, 특히 near circular singularity 를 효율적으로 극복한다. PWID 오프셋 알고리듬의 가장 큰 특징은 주어진 다각형에서 인접한 선분들끼리 발생 시키는 local interfering range(local invalid loop)를 **'PWID-test'**를 이용하여 미리 제거함으로써 대부분의 꼬인 점들이 미리 제거된 raw offset curve 를 구성한다. 그리고 구성된 raw offset curve 에서 global interference 를 일으키는 꼬인 점들을 구하여 global interference(global invalid loop)들을 제거한다. 그 후에 남은 부분은 모두 유효한 오프셋 결과임을 증명하였다. 결과적으로 PWID 오프셋 알고리듬은 near circular singularity 를 자연스럽게 해결할 뿐만 아니라, 유효하지 않은 고리 제거를 위해 많은 시간을 소모할 필요도 없다. 하지만 제안된 PWID 오프셋 알고리듬은 다각형의 오프셋만을 대상으로 하였기 때문에 그 응용 범위가 그리 넓지는 못했다. 특히 공구경로 생성을 위해서는 아일랜드를 포함하는 영역에 대해서 오프셋을 수행하는 것이 필수적이지만 기존 PWID 오프셋 알고리듬으로는 한계가 있었다. 본 논문에서는 PWID 오프셋 알고리듬을 확장하여 아일랜드를 포함하는 영역 오프셋을 가능하게 함으로써 가공경로 생성에 활용함을 목적으로 한다.

3. 용어 정의

앞으로 설명할 알고리듬들의 체계적인 설명을 위해서 몇 가지 용어를 정의하도록 한다. 우선 다각형을 구성하는 점을 꼭지점 (***vertex***) 그리고 연속한 두 개의 꼭지점을 잇는 선을 변(***edge***)이라 부르기로 한다.

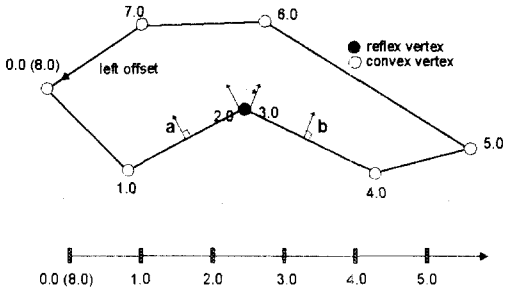

Fig. 5 Parameterization of a PS-curve

정의 1 Vertex type): 하나의 꼭지점에 연결된 두 개의 변이 영역 안쪽으로 이루는 각도가 π 보다 크면 ***reflex vertex*** 라고 부르고 π 보다 작으면 ***convex vertex*** 라 부른다.

정의 2 Segment type): 두 개의 꼭지점을 연결하는 하나의 변을 ***linear segment*** 라 하고, reflex vertex 는 ***reflex segment*** 라 부른다.

이제부터 다각형을 linear/reflex segment 들의 연속으로 이루어진 하나의 매개변수 곡선으로 보고, 각각의 segment 에 매개변수 범위를 1 씩 배정하도록 한다. 예를 들면 Fig. 5 의 다각형은 8 개의 segment (7 개의 linear segment 와 1 개의 reflex segment)로 구성되었으며, 매개변수 영역은 t [0, 8] 이 된다. 매개변수 값 t 에서의 위치는 **P(t)**로 표현하고, unit normal 은 **N(t)**로 나타낸다. 단 normal 의 방향은 오프셋 방향과 동일하다. 여기서 주목해야 할 것은, reflex segment 는 하나의 점에 해당하지만, unit normal 의 값은 직전 linear segment 의 normal 에서 직후 linear segment 까지의 normal 까지 변하게 된다는 것이다. Fig. 5 에서 **a** 와 **b** 는 두 번째와 네 번째 segment 들의 unit normal 이고, 세 번째 reflex segment (t [2,3])의 unit normal 은 **a**(t = 2)에서 **b**(t = 3)까지 변하게 된다.

정의 3 Tangential circle): 다각형에서 하나의 segment 상의 한 점 P(t)에 접하고 반지름 ρ 를 가지는 원을 P(t)의 ***tangential circle*** 이라 한다. 또한 접점 P(t)의 매개변수 t 가 해당 segment 의 끝점에 해당할 때는 ***end tangential circle*** 이라 부른다. 매개변수 t 에서의 tangential circle 의 중심 C(t)는 다음과 같이 정의된다. $\mathbf{C(t) = P(t) + \rho \cdot N(t)}$, 이때 ρ 는 오프셋 거리이며, **N(t)**는 t 에서의 unit normal 이다. 하나의 tangential circle 이 동시에 하나이상의 segment 와 접하게 되면 ***common tangential circle*** 이라 부른다.

정의 4 Interfering point/range): 다각형의 매개변

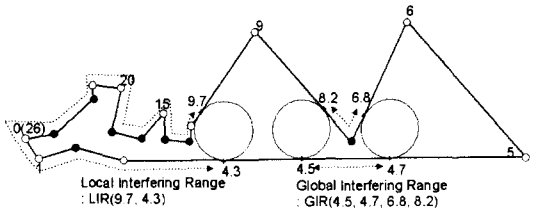

Fig. 6 Local & global interfering ranges

수 영역에 속하는 매개변수 t 에서의 tangential circle 이 다각형과 교차(혹은 포함)를 일으키면, 매개변수 t 는 ***interfering point*** 라고 한다. interfering point 들의 range 는 ***interfering range*** (IR)이라 부른다. 이때 local invalid loop 에 해당하는 IR 은 ***local interfering range*** (LIR)라 하고 global invalid loop 에 해당하는 IR 은 ***global interfering range*** (GIR)이라 한다.

Fig. 6 에 그려진 다각형에는 26 개의 segment 들로 (18 linear, 8 reflex segments) 이루어져 있으며 세 개의 common tangential circle 과 두 개의 interfering range 가 그려져 있다. Fig. 6 의 다각형의 매개변수 영역은 [0, 26]이며, 그려진 두 개의 interfering range 는 다음과 같다. LIR = [9.7, 4.3] ≡ [9.7, 26]∪[0, 4.3]; GIR = [4.5, 4.7]∪[6.8, 8.2]. GIR 은 항상 두 개의 common tangential circle 로 막혀있게 되고, LIR 은 항상 하나의 common tangential circle 로 막혀있게 된다.

Convex vertex 는 항상 interfering point 이므로 모든 convex vertex 들은 local interfering range 를 찾기 위한 local seed point 로 사용될 수 있다. 만약 매개변수 t=j 의 convex vertex 를 seed point 로 삼아서 **'PWID-test'**를 수행하는 방식은 다음과 같다. 우선 t=j 에서 시작하는 segment 를 전진 segment (f_seg)라 하고 t=j 에서 끝나는 segment 를 후진 segment (b_seg)라 한다. 그러면 PWID-test 를 위한 추적방향은 seed point 를 공통 출발점으로 하게 되며, f_seg 는 전진하고 b_seg 는 후진하면서 추적하게 된다. 즉 f_seg 의 end point 는 t=j+1 이 되고, b_seg 의 end point 는 t=j-1 이 되는 것이다. 이제 두 segment 의 end point 중에 interfering point 가 있으면 그 segment 는 추적방향의 다음 segment 에 의해 대치되게 된다. PWID-test 에 대한 구체적인 설명은 다음 장에 주어진다.

정의 5 Interfering segment): 두 개의 segment S1, S2 가 PWID-test 에 의해 test 된다고 하면 1) 만약 S1 의 end tangential circle 이 S2 와 교차하면 S1 은 ***full-interfering segment*** 가 된다. 2) 만약 S1 의 end tangential circle 이 아닌 S1 의 tangential circle 이 S2 와 교차하면 S1 은 ***partial-interfering segment*** 가 된다. 3) 만약 S1 의 모든 tangential circle 이 S2 와 교차하지 않으면 S1 은 ***reverse-interfering segment*** 가 된다.

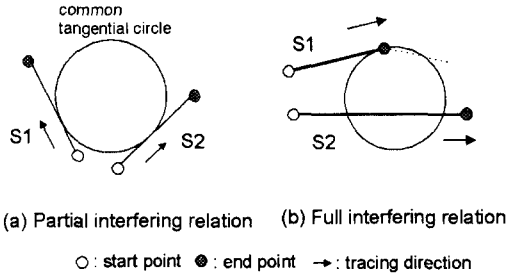

Fig. 7 Partial and full-interfering relations

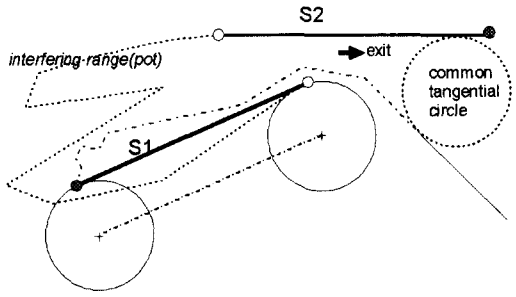

Fig. 8 Reverse-interfering relation

정의 6 Interfering relation): PWID-test 에서 1) 만약 S1 과 S2 모두 partial-interfering segment 라면 ***partial-interfering relation*** 이라 부른다. 그렇지 않고 2) 만약 S1 과 S2 둘 중의 어느 하나라도 reverse-interfering segment 라면 ***reverse-interfering relation*** 이라 부른다. 3) 위의 두 가지에 해당하지 않으면 ***full-interfering relation*** 이라 부른다.

Fig. 7(a)는 S1 과 S2 common tangential circle 을 가지기 때문에 둘 다 상대방에 대해 partial-interfering segment 가 되므로, partial-interfering relation 이 된다. Fig. 7(b)는 S1 이 full-interfering segment 이고 S2 가 partial-interfering segment 이므로 full-interfering relation 이 된다. Reverse-interfering relation 은 Fig. 8 에 나타나 있는데, S1 의 모든 tangential circle 이 S2 와 교차하지 않으므로 S1 이 reverse-interfering segment 가 된다. Fig. 8 의 reverse-interfering relation 에서 보여주고 있듯이, S1 을 뒤따르는 모든 reverse-interfering segment(S2 에 대하여)들은 동일한 local interfering range(LIR)에 속하고, 그 LIR 은 하나의 common tangential circle 로 막힐 수 있다.

4. PWID 오프셋 알고리듬

우선 PWID 오프셋 알고리듬의 핵심이라고 할 수 있는 **'PWID-test(Pair-Wise Interference Detection)'**를 설명하고 전체적인 PWID 오프셋 알고리듬을 설명하도록 한다.

4.1 PWID-test

본 장에서는 PWID-test 를 설명하도록 하며, 우선 다음과 같은 variable 들을 정의하도록 한다.

- f, b: 전/후진 추적을 위해 주어진 다각형상의 매개변수 값들이다.
- f_seg, b_seg: 전진 segment 와 후진 segment 를 나타낸다.

PWID-test 의 입력은 한 쌍의 매개변수 값 (f_s, b_s) 이다. 그러면 $f=f_s$ 에서 시작하는 전진 segment (f_seg)와 $b=b_s$ 에서 시작하는 후진 segment (b_seg) 를 가져온다. 그후 f_seg 와 b_seg 의 interfering relation 을 알아내고, 그 interfering relation 이 partial-interfering relation 이 될 때까지 다음의 두 operation 을 반복적으로 수행한다.

- **Replace-full-segment (f_seg, b_seg)**: 두 개의 segment 가 ***full-interfering relation*** 일 경우에 하나의 full-interfering segment 를 선택하여 (둘 다 full-interfering segment 이면 임의로 선택) 다음 segment 로 대치한다. 물론 다음 segment 는 f_seg 와 b_seg 의 추적 방향을 고려하여 선택한다. Fig. 9 에는 세 가지 경우가 나타나 있는데, 두 개의 segment 가 linear-linear segment 일 경우, linear-reflex segment 일 경우 그리고 reflex-reflex segment 일 경우를 각각 나타내고 있다. Fig. 9(a)에서는 S1 이 full-interfering segment 이므로 (S1 의 end tangential circle 이 S2 와 교차) 다음 segment 로 대치한다. Fig. 9(b)에서는 S1 이 reflex segment 인데 S1 이 full-interfering segment 이므로 S1 을 다음 segment 로 대치한다. Fig. 9(c)에서는 S1 과 S2 가 모두 reflex segment 인 경우이고, S2 가 full-interfering segment 이므로 S2 를 다음 segment 로 대치한다.

- **Replace-reverse-segment (f_seg, b_seg):** 두 개의 segment 가 ***reverse-interfering relation*** 일 때는 하나의 reverse-interfering segment 를 선택하여 (둘 다 reverse-interfering segment 이면 임의로 선택) 상대방에 대해서 non-reverse-interfering segment 가 나올 때까지 추적 방향을 따라 전진한다. 물론 이때 f_seg 혹은 b_seg 가 계속해서 update 되는 것이다. Fig. 8 에서 S1 이 reverse-interfering segment 이므로 S2 에 대해서 최초의 non-reverse- interfering segment 인 다음 segment 가 나올 때까지 전진하면 Fig. 10 처럼 된다.

만약 PWID-test 가 ***partial-interfering relation*** 으로

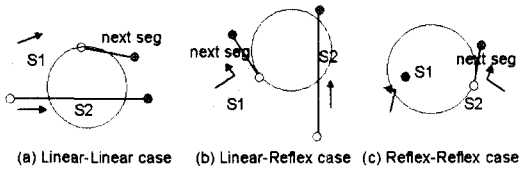

Fig. 9 Replacing a full-interfering segment

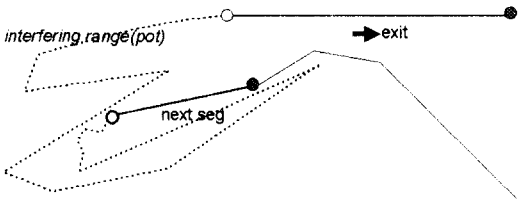

Fig. 10 Replacing all reverse-interfering segments

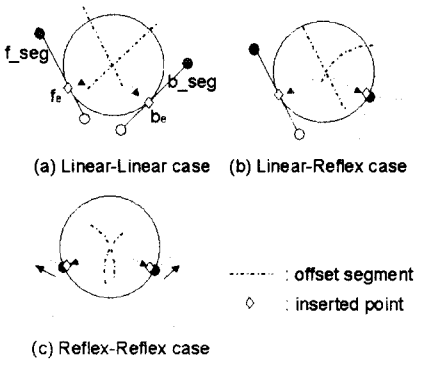

Fig. 11 Partial-interfering relation: end of tracing

끝나게 되면 다음 operation 이 수행되어 한 쌍의 "end-of-interfering" 매개변수 값을 출력하게 된다.

- **End-of-interfering (f_seg, b_seg, f, b):** 두 개의 segment 가 ***partial-interfering relation*** 인 경우에, 두 개의 segment 는 반드시 common tangential circle 을 가지게 되고, 접하는 점들을 계산할 수 있다. 계산된 매개변수 값을 f, b 에 할당함으로써 출력한다. Fig. 11 에서처럼 접하는 점들($f=f_e$, $b=b_e$)는 offset segment 들의 교차점을 계산함으로써 쉽게 찾을 수 있다.

그러나 f_seg 와 b_seg 가 partial-interfering relation 에 도달하지 않고 만나는 경우가 있을 수 있는데, 그러한 경우는 주어진 다각형이 유효한 오프셋 결과를 가지지 못하는 경우이다. 이점까지 고려하여 PWID-test 알고리듬을 기술하면 다음과 같다.

PWID-test (f, b)
// f: 전진 매개변수, b: 후진 매개변수 //
// f_seg: 전진 segment, b_seg: 후진 segment //

Begin
f, b 에 해당하는 segment 들을 f_seg 와 b_seg 에 각각 할당한다 ;
Repeat {
만약 f_seg 와 b_seg 가 동일한 end point 를 가지면 f=b; return (f, b);
f_seg 와 b_seg 의 interfering relation 을 알아낸다.
만약 ***Full-interfering relation*** 이면
Replace-full-segment (f_seg, b_seg);
만약 ***Reverse-interfering relation*** 이면
Replace-reverse-segment (f_seg, b_seg) ;
}Until (***Partial-interfering relation***)
End-of-interfering (f_seg, b_seg, f, b) ;
return (f, b) ;
End.

만약 위에 정의된 PWID-test 가 interfering 이 시작되는 ($f=f_s$, $b=b_s$)를 입력으로 하여 수행되었을 때, interfering 이 끝나는 ($f=f_e$, $b=b_e$)를 출력해 준다면 결정되는 interfering range(IR)는 IR = $[b_e, b_s] \cup [f_s, f_e]$가 된다. 주어진 다각형의 매개변수 영역은 segment 의 개수 n 을 주기로 하여 반복되므로, 만약 구간 [a, b]에서 a > b 이면 그 구간은 [a, n] ∪ [0, b]와 동일하다. 이제 PWID-test 를 이용하여 다음 절에서 PWID 오프셋 알고리듬을 기술하도록 한다.

4.2 PWID 오프셋 알고리듬

다각형 오프셋을 위한 PWID 오프셋 알고리듬은 다음 다섯 단계를 거친다:

1) 주어진 다각형에서 모든 local-interfering range(LIR)들을 제거한다.
2) 남은 segment 들을 offset 하여 raw offset curve 를 구성한다.
3) Raw offset curve 에서의 꼬인 점들을 모두 구한다.
4) Raw offset curve 의 꼬인 점들을 이용하여 모든 global-interfering range(GIR)들을 제거한다.
5) GIR 이 제거된 후 남은 것은 모두 valid offset curve 이다.

이러한 PWID 오프셋 알고리듬을 자세히 기술하면 다음과 같다. 그 후 예제를 통한 추가적인 설명을 덧붙이도록 한다.

PWID Offset Algorithm:
// Input: simple polygon, offset direction, offset distance (ρ) //
1. 모든 local-interfering range (LIR)들을 제거.
While (다각형에 convex vertex 남아 있으면) **do**
1) 남은 Convex vertex 를 하나 선정하여 그것의 매개변수 값을 t_c에 할당 ;
2) $f = t_c$; $b = t_c$;

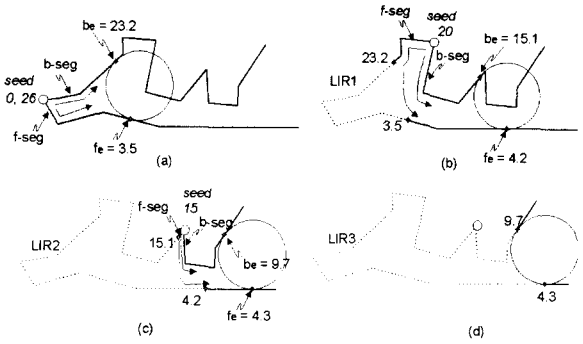

Fig. 12 Deletion of local interfering ranges

3) (f, b)=**Pair-wise-interference-detection**(f, b);
4) LIR = [b, f]를 다각형으로부터 제거한다.;
// End of While //
2. 남은 segment 들을 offset 하여 raw offset curve 를 구성한다.
3. Sweep line algorithm[2]을 이용하여 raw offset curve 에서의 꼬인 점들을 모두 구한다.
4. 모든 global-interfering range (GIR)들을 제거.
GIR = Φ ;
While (raw offset curve 에 처리되지 않은 꼬인 점이 있으면) **do**
1) 처리 되지 않은 꼬인 점을 선택하여 관련된 두 개의 매개변수 값을 (f_s, b_s)에 할당한다. 선택된 꼬인 점을 처리 표시한다.
//꼬인 점은 두 개의 segment 가 교차한 것이므로 관련된 매개변수 값이 두 개인데, 어떤 것을 f_s 에 할당하고 어떤 것을 b_s 에 할당할 것인가는, 아래에서 예제를 이용하여 설명한다.//
2) $f = f_s$; $b = b_s$;
3) (f, b)=**PWID-test**(f, b);
4) (f, b)에 해당하는 꼬인 점을 처리 표시한다.
// GIR 은 꼬인 점에서 시작하여 꼬인 점에서 끝난다. //
5) GIR = GIR ∪ $[f_s, f]$ ∪ $[b, b_s]$;
; // End of while //
6) GIR 을 raw offset curve 로부터 제거.
5. 남은 raw offset curve 를 이용하여 valid offset curve 를 구성한다.

위에 설명된 알고리듬의 Step 1 은 Fig. 6 에 그려진 LIR 을 제거하는 과정을 Fig. 12 를 통해 다시 자세히 예를 들어 설명하도록 한다. Fig. 12(a)에서 보여지듯이 Step 1-1 에서 t_c=0 인 convex vertex 가 local seed 로 선정한다. 그리고 Step 1-3 에서 **PWID-test**(f=0, b=0)이 실행되어 common tangential circle 이 생기는 점, ($f=f_e$=3.5, $b=b_e$=23.2)을 얻게 된다. Step 1-4 에서는 구해진 LIR1=[23.2, 3.5]를 다각형으로부터 제거한다(Fig. 12(b)). 그 다음에는 다시

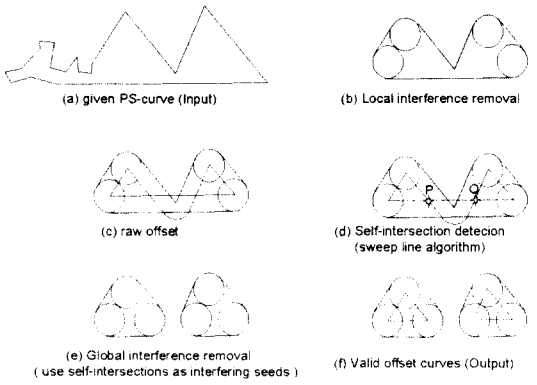

Fig. 13 Deletion of global interfering ranges

Step 1-1 로 되돌아가서 local seed 로 t_c=20 인 convex vertex 를 선택하여 Step 1-3 에서 **PWID-test**(f=20, b=20)을 부르게 되고 그 결과로 (f=f_c=4.2, b=b_c=15.1)을 얻게 된다. 그러면 Step 1-4 에서 LIR2=[15.1, 4.2]를 제거한다(Fig. 12(c)). 다시 한번 t_c=15 로 잡아 반복하면 LIR3=[9.7, 4.3]을 얻게 되고 그것을 제거한다(Fig. 12(d)).

이제 Step 4 를 Fig. 13 을 예로 들어 설명하고자 한다. Step 4-1 에서는 처리 되지 않은 꼬인 점으로 P (Fig. 13(d))를 선택한다. 이때 P 에서 interference 가 일어나는 방향으로 f_s 와 b_s 를 결정하여야 한다. 즉 꼬인 점 P 는 common tangential circle 의 중심 (Fig. 5)으로 볼 수 있고, 그 tangential circle 이 gouge 를 일으키는 방향으로 추적 방향을 잡아야 한다는 의미이다. 그러므로 예제에서는 전진 매개변수 f_s = 4.5 그리고 후진 매개변수 b_s = 8.2 (Fig. 5)가 되는 것이다. (점 P 를 기준으로 매개변수 값 4.5 에서는 앞으로 진행하고, 매개변수 값 8.2 에서는 뒤로 진행해야 간섭을 일으키는 부분이라는 것이다.) Step 4-3 에서는 결정된 매개변수들을 이용하여 **PWID-test**(f=4.5, b=8.2)를 수행하면, 다른 꼬인 점 Q 를 나타내는 (f=4.7, b=6.8)을 얻을 수 있다. Step 4-5 에서는 GIR = [4.5, 4.7]∪[6.8, 8.2]를 구하게 된다. 구해진 GIR 은 계속 누적 시키다가 모든 꼬인 점이 처리 되어야만 row offset curve 에서 제거시킨다. 왜냐하면 GIR 중의 일부가 중복되어서 추적될 가능성을 배제할 수 없기 때문이다.

Choi & Park[1]이 이미 밝혔듯이 PWID 오프셋 알고리듬의 time complexity 는 Step 3 의 time complexity 인 $O((r+s)\cdot logm)$에 (r: 다각형의 segment 개수, s: 꼬인 점의 개수, m: monotone chain 의 개수) 제약 받지만, 실제 실험결과로 볼 때는 계산시간은 주어진 다각형의 segment 개수에 linear 하게 변하는 것을 볼 수 있다[1]. 그 이유는 LIR 을 raw offset curve 를 구하기 전에 제거함으로 인해 실제로는 꼬인 점의 개수 (s)가 매우 작아져서 Step 3 에서 거의 계산 부하가 걸리지 않기 때문이다.

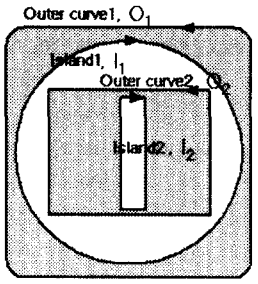

Fig. 14 A region defined by PS-curves

5. 영역 오프셋 알고리듬

본 장에서는 앞서 설명된 PWID 오프셋 알고리듬을 확장하여 영역 오프셋을 가능하게 하는 알고리듬을 제안하고자 한다. 본 논문에서 다루는 영역에 대해서 간단히 설명하면 다음과 같다. 외곽을 정의하는 다각형들과 내부 아일랜드를 정의하는 여러 개의 다각형이 있을 수 있다. 물론 영역을 정의하는 다각형들 사이에는 꼬임이 없어야 한다. Fig. 14 는 본 논문에서 다루는 영역의 예를 보여준다. PWID 오프셋 알고리듬의 확장을 설명하기 이전에 이 전에 이미 설명 되었던 PWID 오프셋 알고리듬의 전반적인 절차를 다시 한번 살펴보면 다음과 같다.

1) 주어진 다각형에서 모든 local-interfering range(LIR)들을 제거한다.
2) 남은 segment 들을 offset 하여 raw offset curve 를 구성한다.
3) Raw offset curve 에서의 꼬인 점들을 모두 구한다.
4) Raw offset curve 의 꼬인 점들을 이용하여 모든 global-interfering range(GIR)들을 제거한다.
5) GIR 이 제거된 후 남은 것은 모두 valid offset curve 이다.

이중에서 Step 1,2 는 하나의 다각형에 대해서 독립적으로 적용되므로 영역을 이루는 여러 개의 다각형에 대해서 각각 적용 시킬 수 있다. 즉 영역 오프셋인 경우에도 LIR 의 제거와 raw offset curve 의 생성은 하나의 다각형을 다룰 때처럼 동일하게 처리해도 된다는 것이다. 그리고 Step 3 인 경우는 여러 개의 raw offset curve 들을 모두 고려하여 꼬인 점을 찾도록 바꾸어야 하며, Park et al. 98[2]에서 제안된 알고리듬을 활용하면 효율적으로 꼬인 점들을 찾을 수 있다. 영역 오프셋으로의 확장을 위해서 가장 많이 바뀌어 할 부분은 Step 4) GIR 제거 부분이다.

Step 4 의 확장을 고려하면서 가장 먼저 생각해야 할 부분은 바로 GIR 의 특성이라 할 수 있을 것이다. GIR 의 특징은 바로 '**항상 두개의 꼬인 점**'으로 정의 된다는 것이다. GIR 의 되기 위해서는 한번 꼬임(interference)이 시작되었으면 반드시 그 꼬임이 풀리는 지점도 있어야 하기 때문이다. 다시 말하면 GIR 은 꼬인 두 점으로 정의되는 특정 구간일 뿐이지 하나의 다각형 내에 존재할 어떠한 필요도 없는 것이다. 결과적으로는 Step3 에서 여러 개의 raw offset curve 들 사이의 꼬임을 그대로 이용하여 각 꼬인 점에서 기존의 PWID-test 를 적용하면 꼬임이 풀리는 점까지 추적하게 되고 그 결과를 이용하면 쉽게 GIR 들을 찾아서 제거할 수 있다.

그러나 아쉽게도 영역 오프셋에서 꼬인 점들만을 이용하여 GIR 을 모두 제거 할 수는 없다. 왜냐하면 오프셋 거리에 따라서 어떤 raw offset curve 는 전혀 꼬인 점을 가지지 않음에도 불구하고 전체가 유효하지 않은 고리일 수 있기 때문이다. 예를 들어 설명하면 외곽이 하나의 사각형이고 내부에 원형의 아일랜드가 있다고 가정하자. 이 상황에서 영역 내부 방향으로 매우 큰 오프셋 거리를 주고 오프셋 하게 되면 우선 LIR 을 제거하고 raw offset curve 를 생성하게 된다. 이 때 raw offset curve 를 관찰하면 외부 사각형은 줄어들어 조그맣게 변하고 내부 원은 커져서 사각형을 포함하고 있을 수 있다. 그렇게 되면 두개의 raw offset curve 사이에는 꼬인 점이 없게 된다. 그럼에도 불구하고 이런 경우는 명백히 두개의 raw offset curve 모두 GIR 에 포함되어야 한다는 것은 알 수 있다. 이러한 문제를 해결하기 위해서는 결국 영역을 이루는 다각형들 간에 포함 관계를 검사해야 한다. 즉 raw offset curve 들을 구성한 후에는 각 raw offset curve 들에 대해서 원래의 포함 관계가 뒤집어지는지를 검사하고 만약 그것이 뒤집어지면 관련된 raw offset curve 들 전체를 GIR 로 보고 제거하여야 한다. 지금까지 설명된 사항들을 기반으로 PWID 오프셋 알고리듬을 확장한 영역 오프셋 알고리듬은 다음과 같이 기술될 수 있다.

■ **영역 오프셋 알고리듬**

// 입력: 여러 개의 다각형으로 구성된 영역,
오프셋 방향과 거리.

// 출력: 오프셋 결과

1) LIR 제거
주어진 다각형들에서 각각 독립적으로 PWID-test 를 적용하여 LIR 들을 제거한다. (Fig. 15(a))

2) Raw offset curve 들 구성

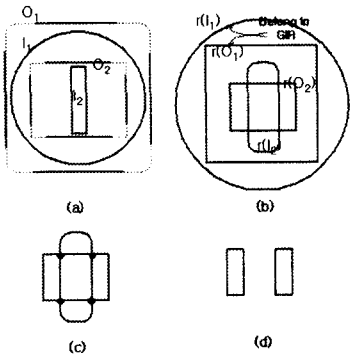

Fig. 15 Region offsetting procedure

남은 segment 들을 offset 하여 raw offset curve 들을 구성한다. (Fig. 15(b))

3) 포함관계를 이용한 GIR 제거
Raw offset curve 들 중에서 초기 영역의 포함관계를 뒤집는 것들을 모두 제거한다. (Fig. 15(c))

4) 꼬인 점 구하기
Raw offset curve 들 사이의 꼬인 점들을 모두 구한다.(Fig. 15(c))

5) 꼬인 점을 이용한 GIR 제거
Raw offset curve 의 꼬인 점들을 이용하여 모든 GIR 들을 제거한다. (Fig. 15(d))

6) GIR 이 제거된 후 남은 것은 모두 valid offset curve 이다. (Fig. 15(d))

제안된 확장 알고리듬을 예제를 이용하여 설명하도록 한다. Fig. 14 에 가공영역이 나타나 있다. 즉 그 내부를 채우는 오프셋 가공이 필요한 것이다. 이때 O1 이 I1 을 포함하고 O2 가 I2 를 포함하고 있음을 쉽게 알 수 있다. 확장 알고리듬에서 Step1 에서 LIR 을 제거하면 Fig. 15(a)처럼 점선 부분이 제거되고 실선 부분만 남게 된다. 그리고 남은 부분들로 Step 2 에서 raw offset curve 를 구성하게 되면 Fig. 15(b)와 같이 된다. 이때 Step 3 에서 포함 관계를 검사할 필요가 있는데, raw offset curve 들 중에서 $r(I_1)$이 원래의 포함 관계를 뒤집어서 오히려 $r(O_1)$을 포함하고 있음을 알 수 있다. 즉 $r(I_1)$과 $r(O_1)$ 모두 GIR 에 속하므로 Step 3 에서 제거하게 된다. 반면 $r(O_2)$와 $r(I_2)$는 서로 꼬임이 생겨 포함관계를 따질 수가 없으므로 원래 포함 관계를 뒤집었다고 보지 않는다. Step 4 에서 남은 raw offset curve 들의 꼬인점을 찾으면 Fig. 15(c)와 같이 되고 Step 5 에서 꼬인점을 이용하여 GIR 들을 제거하면 Fig. 15(d)와 같이 된다. 그 결과 남은 것이 유효한 오프셋 결과 인 것이다.

6. 공구경로 생성에의 활용

본 연구에서 제안된 영역 오프셋 알고리듬을

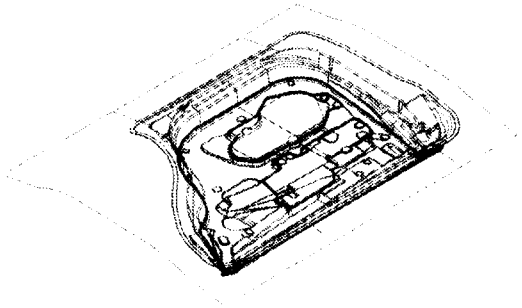

Fig. 16 Machining region by SSI

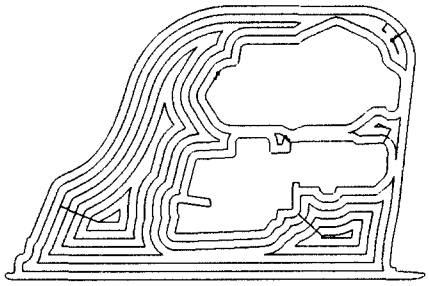

Fig. 17 An example of the tool-path

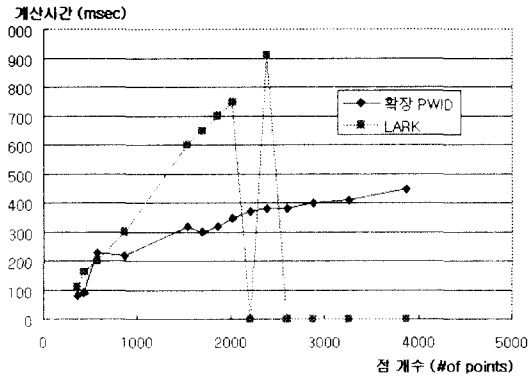

Fig. 18 Time performance of the suggested algorithm

이용하여 오프셋 방식의 공구 경로 생성에 활용한 예를 보여준다. 우선 가공 영역을 생성하기 위해서 가공 곡면을 평면으로 자르는 SSI(Surface-Surface Intersection)를 수행한다(Fig. 16). 이렇게 생성된 가공 영역을 제안된 알고리듬으로 오프셋 하여 생성된 공구경로가 Fig. 17 에 나타나 있다. 예제 모델의 치수는 320mm x 210mm 이며 생성된 가공경로의 간격은 20mm 이다.

본 논문에서는 제안된 알고리듬의 효용성을 입증하기 위해서 상용 시스템 Lark[6]와 계산 속도를 비교하는 실험을 수행하였다. Lark 는 Voronoi diagram 방식으로 오프셋을 수행하는 시스템이다. 실험을 위해서 가공 영역을 구할 때 SSI 오퍼레이션의 공차를 조절함으로써 14 가지 서로 다른 점 개수 (360 개 에서 3877 개)를 가지는 곡선들을 생성함으로써 데이터를 작성하였다. 실험은 리눅스 운영체제하에서 펜티엄 2-450 에서 수행되었다. 계산 시간은 밀리 세컨드 (msec) 단위로 측정 되었으며 Lark 에서 계산시간이 0 으로 나오는 부분은 시스템이 계산에 실패한 경우를 나타낸다. Fig. 18 의 실험결과에서 보이듯이 제안된 확장 PWID 오프셋 알고리듬이 최소한 해당 예제에 한하여는 더 효율적이고 강건함을 볼 수 있다. 하지만 주어진 데이터의 성격에 혹은 실험 환경에 따라 결과는 달라 질 수 있음을 밝힌다.

7. 결 론

본 연구에서는 PWID 오프셋 알고리듬을 확장함으로써 영역 오프셋 알고리듬을 개발하였다. PWID 오프셋 알고리듬을 채택한 이유는 기존의 다각형 오프셋 알고리듬의 가장 큰 어려움인 near circular singularity 를 해결하였을 뿐 아니라 빠른 계산 속도 때문이다. PWID 오프셋 알고리듬을 영역 오프셋으로의 확장을 위해서 GIR 의 개념을 확장하였는데, 우선 기존 GIR 이 두개의 꼬인 점으로 정의되면 이러한 꼬인 점들이 하나의 다각형 내에 국한될 필요는 없다는 사실을 밝혔다. 또한 영역 오프셋에서는 꼬인 점으로 찾을 수 없는 GIR 이 있다는 사실을 지적하였고 해결을 위해서는 원래 영역을 구성하는 다각형들 사이의 포함관계를 이용하여야 한다는 사실을 밝혔다.

이렇게 제안된 영역 오프셋 알고리듬을 구현하여 실제 공구경로 생성에 활용하여 그 유용성을 증명하였다. 하지만 본 연구가 오프셋 공구경로 생성문제를 모두 해결하였다고는 볼 수 없으며, 실제 가공의 기술적인 문제인 미삭 영역 탐지와 가공,[3] 공구경로의 연결 문제,[4] 가공 부담에 따른 가공속도 (feed rate) 조정 문제[7]를 잘 고려하여야 효율적인 NC-code 를 생성할 수 있다.

참고문헌

(1) Choi, B. K. and Park SangC., 1999, "A Pair-Wise Offset Algorithm for 2D Point-Sequence," *Computer Aided Design*, Vol. 31, No. 12, pp. 735~45.

(2) Park, SangC., Shin, H. and Choi, B. K., 1998, "A Sweep Line Algorithm for Polygonal Chain

Intersection and Its Applications," *In Proc. of IFIP WG5.2 GEO-6 Conference* in Tokyo University, Dec. 7-9, pp. 187~195.
(3) Park, SangC., Choi, B. K., 2001, "Uncut-Free Pocketing for Tool-Path Generation Using Pair-Wise Offset Algorithm," *Computer Aided Design*, Vol. 33, No. 10, pp. 739~46.
(4) Park, SangC., Chung, Y. C., 2001, "Offset Tool-Path Linking for Pocket Machining," accepted for publication in *Computer Aided Design*.
(5) Hansen, A. and Arbab, F., 1992, "An Algorithm for Generating NC Tool Path for Arbitrary Shaped Pockets with Islands." *ACM Transactions on Graphics*, Vol. 11, No. 2, pp. 152~182.
(6) Held, M., Lukacs, G., and Andor, L., 1994, "Pocket Machining Based on Contour-Parallel Tool Paths Generated by Means of Proximity Maps," *Computer Aided Design*, Vol. 26, No. 3, pp. 189~203.
(7) Choi, B. K. and Kim, B. H., 1997, "Die-Cavity Pocketing via Cutting Simulation," *Computer Aided Design*, Vol. 29, No. 12, pp. 837~846.
(8) Persson., H., 1978, "NC Machining of Arbitrary Shaped Pockets," *Computer Aided Design*, Vol. 10, No. 3, pp. 169~174.
(9) Kokichi Sugihara, 1998, "Degeneracy and Instability in Geometric Computation," *In Proc. of IFIP WG5.2 GEO-6 Conference* in Tokyo University, Dec. 7-9, pp. 5~15.
(10) Chiang, C. S., Hoffman, C. M. and Lynch, R. E., 1991, "How to Compute Offsets without Self-Intersection," *In SPIE Conf. Proc. Curves and Surfaces in Computer Vision and Graphics II*, Boston, MA, pp. 76~87.
(11) Kalmanovich G. and Nisnevich G., 1998, "Swift and Stable Polygon Growth and Broken Line Offset," *Computer Aided Design*, Vol. 30, No. 11, pp. 847~852.
(12) Rohmfeld, R. F., 1998, "IGB-Offset for Plane Curves Loop Removal by Scanning of Interval Sequences," *Computer Aided Geometric Design*, Vol. 15, pp. 339~375.
(13) Held, M., 1991, On the Computational Geometry of Pocket Machining, Springer-Verlag, Berlin, 1991.
(14) Suh Yong Seok and Lee Kunwoo, 1990, "NC Milling Tool Path Generation for Arbitrary Pockets Defined by Sculptured Surfaces," *Computer Aided Design*, Vol. 25, No. 5, pp. 273~284.
(15) 김태주, 이건우, 홍성의, 1994, "옥트리를 이용한 황삭 가공경로 생성," 대한기계학회논문집, 제 18 권, 제 1 호, pp. 53~64.