한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003년 5월 16일-17일 한동대학교(포항)

포켓가공을 위한 오프셋 및 공구경로 연결 알고리즘

허진헌 · 김영일 · 전차수

경상대학교 산업시스템공학부, 항공기부품기술연구소

Contour Parallel Offsetting and Tool-Path Linking Algorithm For Pocketing

Jin-Hun Huh · Young-Yil Kim · Cha-Soo Jun

Division of Industrial and Systems Engineering, Research Center for Aircraft Parts Technology

Presented in this paper is a new fast and robust algorithm generating NC tool path for 2D pockets with islands. The input shapes are composed of line segments and circular arcs. The algorithm has two steps: creation of successive offset loops and linking the loops to tool path. A modified pair-wise technique is developed in order to speed up and stabilize the offset process, and the linking algorithm is focused on minimizing tool retractions and preventing thin-wall cutting. The proposed algorithm has been implemented in C++ and some illustrative examples are presented to show the practical strength of the algorithm.

1. 서 론

CAD/CAM 시스템의 2차원 및 $2^1/_2$차원 밀링가공 분야에서 가장 기본적이면서 중요한 문제 중의 하나는 오프셋이다. 그러나 오프셋은 계산이 복잡한 기하학적인 문제를 내포하고 있어서 많은 연구를 통해 다양한 알고리즘이 제시된 바 있다. 이전에 연구되어진 윤곽의 오프셋 방법은 크게 네 가지 종류의 알고리즘으로 나눌 수 있다. 첫 번째는 윤곽을 직접 오프셋팅 하여 루프(Loop)를 형성시킨 후 루프의 방향이 반대로 형성된 꼬임(Self intersection)에 대해서 그 부분을 제거하는 방법이고[4,9,10], 두 번째는 Voronoi diagram을 이용한 방법[1,7]이다. 세 번째는 Pair wise intersection을 이용한 방법.[6,8] 그리고 마지막으로 Pixel을 이용한 방법[5]으로 구분할 수 있다. 첫 번째 일반적인 오프셋 방법은 구현하기는 쉬우나 오프셋팅 후 루프의 꼬임에 대한 완벽한 대처방법이 없다는 단점이 있고, 두 번째 방법인 Voronoi diagram을 이용한 방법은 꼬임이 발생하는 부분을 사전에 방지할 수는 있으나 비교적 구현이 어려우며 수치계산상의 오류가 발생하기 쉬운 단점이 있다. 그리고 Pair wise intersection 방법은 안정적인 결과를 얻을 수 있으나 계산시간이 많아진다는 단점이 있다. 마지막으로, Pixel을 이용한 오프셋 방법은 비교적으로 안정적인 결과를 보장할 수는 있지만 정밀도가 증가함에 따라 많은 량의 메모리를 필요로하게 되고 처리에 과도한 시간이 소비된다는 단점이 있다.

제시된 방법 중에서 구현하기가 비교적 쉽고 효율적이면서 CAD/CAM 시스템에서 널리 사용되고 있는 방법은 Pair wise intersection을 이용한 방법이다. 본 연구에서는 Pair wise intersection 방법에서 문제가 되는 교점계산부분에 소요되는 시간을 최소화하기 위한 개선된 Pair wise intersection 방법을 개발하고 이를 이용한 2차원 윤곽 형상의 오프셋팅 알고리즘을 제안하고자 한다.

또한, 본 연구에서는 개선된 Pair wise intersection offsetting 방법을 사용하여 주어진 2차원 형상을 연속적으로 오프셋팅하여 만들어진 루프들을 연결하여 최적의 공구경로로 만든다. 여기서는 첫째, 공구의 진입 및 후퇴구간을

최소화시키므로 드릴링 작업의 횟수를 최소화 시킨다. 둘째, 공구의 진행방향을 고려하여 절삭 방향을 일관성 있게 유지시켜 '얇은 벽 가공'(Thin wall cutting)이 발생되지 않도록 한다. 따라서 본 연구에서는 2차원 형상의 윤곽을 안정적으로 오프셋팅 하고 이를 효율적으로 연결할 수 있는 공구경로 연결 알고리즘을 제안하고자 한다.

2. 용어의 정의

본 연구의 내용은 그림 1에서 보듯이 크게 두 단계로 나누어질 수 있다. 첫 번째는 주어진 2차원 형상을 오프셋팅하여 공구경로의 근간이 되는 다수의 프로파일을 생성하는 단계이고, 두 번째는 생성된 프로파일을 연결시켜 최종적인 포켓팅 공구경로를 생성하는 단계이다.

이장에서는 본 연구에서 오프셋팅 알고리즘을 구현하는데 필요한 용어와 전제조건은 다음과 같다.

1) 루프(Loop)를 구성하고 있는 최소단위는 세그먼트(Segment)로 정의하고 세그먼트는 직선(Line Segment)과 원호(Circular Arc)로 구분한다.

2) 하나의 루프(Loop)는 3개 이상의 세그먼트로 구성되어야 한다.

3) 윤곽 프로파일(Contour Profiles)은 닫힌 루프(Closed loop)로 구성되어야 한다.

4) 루프의 방향(Loop Direction)은 그림 2에서 보는바와 같이 외부루프(Outer Loop)는 반시계 방향(Counter Clock Wise)으로 형성되어야 하며 내부루프는(Inner Loop or Islands) 시계방향(Clock Wise)으로 형성되는 것을 원칙으로 한다.

5) 하나의 루프(Single Loop)는 루프 자체가 꼬임(Self intersection)이 없어야 한다.

6) 하나의 외부루프는 여러 개의 내부루프(Islands)를 포함할 수 있다. 그러나 내부루프 안에는 또 다른 외부루프가 존재해서는 안된다.

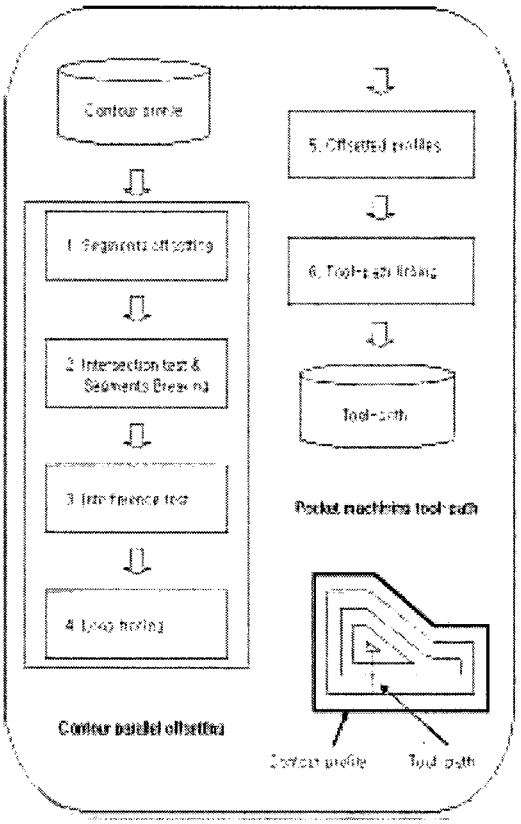


그림 1. 포켓팅 공구경로의 생성흐름

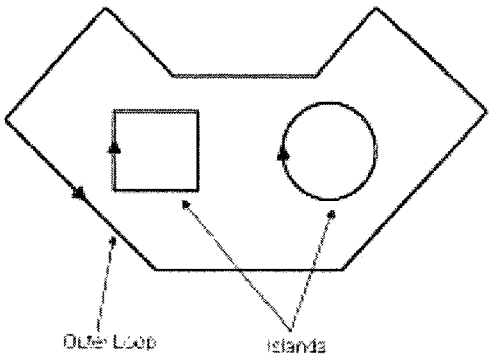


그림 2. 루프의 정의 및 방향

3. 2차원 오프셋팅 알고리즘

제안된 2차원 윤곽의 오프셋팅 알고리즘은 크게 네 개의 단계로 구분할 수 있다. 1) 세그먼트 오프셋팅: 입력된 윤곽 루프을 구성하고

있는 개별적인 세그먼트 요소를 각각 주어진 값으로 오프셋 시킨다. 2) 세그먼트들 간의 교점검사 및 분할: 오프셋팅된 모든 세그먼트 간에 교점을 검사하고 교점이 발생한 세그먼트는 분할시킨다. 여기서는 분할된 세그먼트 중 불필요한 부분에 플래그를 설정[2]하여 다음 작업에서 제외시키므로 간섭대상이 되는 많은 세그먼트들이 이 과정에서 여과된다. 3) 간섭검사: 이렇게 여과되고 남은 세그먼트 중에서도 독자적으로 간섭발생 요인이 되는 세그먼트를 찾아서 제거 시켜준다. 4) 루프추적: 간섭가능 세그먼트들을 모두 제거시키고 나서 남는 세그먼트들을 연결하여 닫힌루프로 구성시킨다.

3.1 세그먼트 오프셋

세그먼트의 오프셋은 루프를 구성하고 있는 개개의 세그먼트를 그림 3과 같이 원하는 오프셋 거리만큼 오프셋시켜 새로운 세그먼트를 생성한다. 만약 두 세그먼트 사이에 갭이 발생한다면 그 사이에는 원호가 삽입되며, 오프셋팅한 원의 반지름이 0이라면 그 세그먼트는 생성되지 않는다.

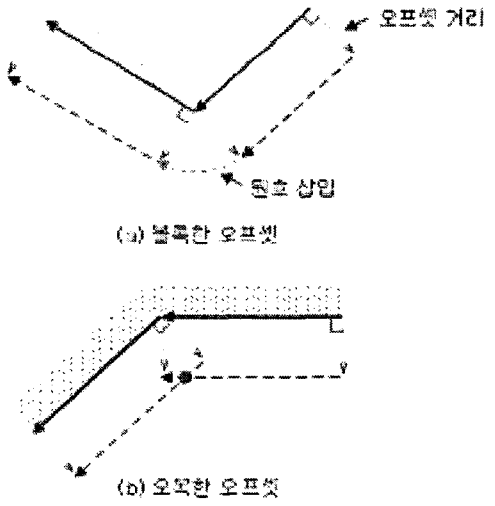


그림 3. 세그먼트 오프셋

3.2 교점테스트 및 분할

■ 교점테스트 알고리즘

Pair wise intersection 알고리즘에서 많은 부분을 차지하고 있는 과정이 교점처리 과정이다. 입력된 루프의 개수에 의해 루프를 구성하고 있는 세그먼트의 수가 많아지고 이를 모두 일반적인 교점계산 알고리즘을 사용한다면 $O(n^2)$ 시간이 걸리게 된다. 본 연구에서는 Hansen[6]이 제안한 교점처리 알고리즘을 개선하여 입력된 원시루프들을 지정된 거리만큼 오프셋팅하고 나서 오프셋된 세그먼트들간에 교점계산을 수행하게 된다. 여기서는 오프셋된 세그먼트의 바운딩 영역(MinX, MaxX)을 구하고 MinX값 기준으로 정렬시킨 후 이를 이용해 교점이 발생할 세그먼트를 검출하여 교점검사를 수행하게 된다. 그림 4에서는 x span 내에 3개의 다른 세그먼트가 걸쳐 있음을 나타내고 있다.

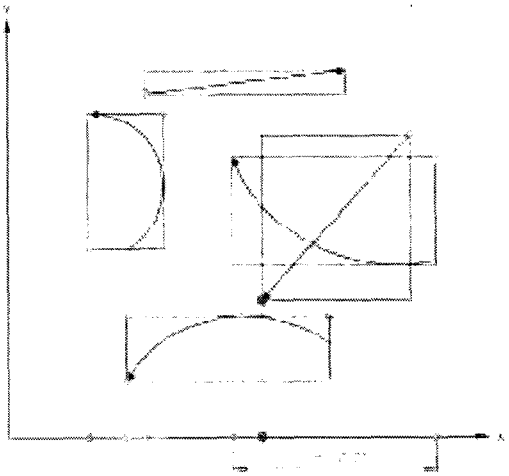


그림 4. 교점검사를 위한 범위 탐색

■ 요소분할 및 제거

Pair wise intersection 방법은 교점계산이 이루어진 후 교점이 발생한 세그먼트를 찾아서 그 교점을 중심으로 필요한 부분과 필요없는 부분을 구분해서 필요없는 부분을 잘라내는 과정을 거쳐야한다. 이 과정에서 한 세그먼트에 그림 5에서 보는 것과 같이 여러 개의 교점이 발

생한다면 세그먼트에 놓인 교점의 거리나 각도를 가지고 구해진 교점의 순서를 정렬시킨 후에 세그먼트로 분할시켜 교점이 발생한 개수만큼의 새로운 세그먼트가 순서를 유지해가며 생성되야 한다. 이 과정이 이루어진 다음에는 그림 6과 같이 없어질 부분을 판단하여 불필요한 부분에 플래그를 설정한다. 그리고는 플래그가 설정된 요소들을 리스트에서 제거시키게 된다.

(1) 세그먼트의 고유번호로 교점정렬

교점이 발생한 두 세그먼트를 분리하여 교점배열에 저장한다. 그리고, 두 세그먼트가 분할될 때 제거될 부분을 판단하여 플래그를 설정한다. 본 연구에서는 직선 세그먼트는 세그먼트의 시작점에서 교점까지의 거리로 원호는 중심점과 교점이 이루는 각도를 속성에 저장시켜 같은 세그먼트 고유번호를 비교하여 배열을 오름차순으로 정렬시킨다.

(2) 세그먼트 분할

같은 세그먼트의 고유번호만을 찾아서 시작점에서 교점까지의 거리나 각도에 의해 작은 순으로 정렬시킨다. 정렬된 순서를 가지고 순차적으로 새로운 세그먼트를 생성시킨다. 또한, 교점배열에서 정해진 플래그로 생성된 세그먼트의 앞/뒤 위치를 판단하여 세그먼트 리스트의 머리 또는 꼬리속성에 플래그를 부여한다.

(3) 플래그가 설정된 세그먼트 제거

세그먼트 리스트의 머리 또는 꼬리의 플래그 속성이 하나라도 1 값이 설정되어 있다면 세그먼트를 리스트에서 제거시킨다.

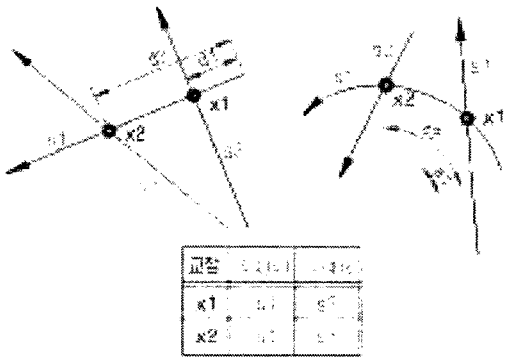


그림 5. 세그먼트 s1에서 발생한 교점의 순서

다음에 표시된 그림 7은 각각 교점에 의해 플래그가 설정되고 불필요한 세그먼트가 제거된 모습을 보여주고 있다.

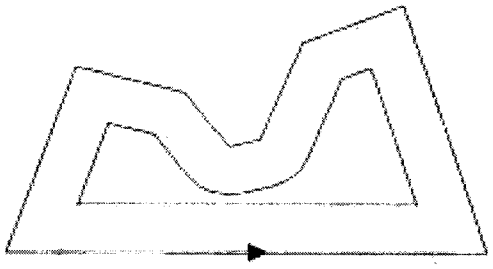
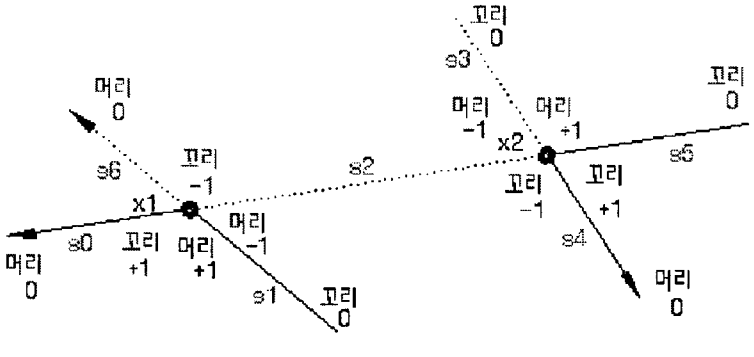


그림 7. 플래그가 제거된 오프셋 세그먼트



edge	머리	꼬리	제거/유지
s0	0	+1	유지
s1	+1	0	유지
s2	-1	-1	제거
s3	-1	0	제거
s4	0	+1	유지
s5	+1	0	유지
s6	0	-1	제거

그림 6. 세그먼트의 머리/꼬리 플래그가 설정된 예

3.3 간섭발생 요소의 제거

1차적으로 각각의 오프셋팅된 세그먼트들 간에 교점을 구하여 그 부분을 분할한 다음 플래그 설정방법을 통해 불필요한 세그먼트를 걸러내는 작업을 거쳐 간섭이 발생하는 많은 부분을 작업대상에서 제외시킬 수 있었다. 그러나 이 작업을 수행한 후에도 그림 8과 같이 여전히 원시루프에 영향을 주는 세그먼트는 남아 있을 수 있으며, 이는 세그먼트간의 거리가 짧아서 개수가 많아질 때 교점이 발생하지 않아 머리 및 꼬리 플래그가 초기값인 0로 설정되어 제거되지 않은 채 남아 있는 세그먼트가 이에 해당된다. 따라서 플래그 설정방법에서 걸러지지 않는 세그먼트들을 대상으로 간섭발생요소를 검출해 내어 제거시킬 필요가 있다.

남아 있는 오프셋된 세그먼트의 중점에서 그림 9.(a)와 같이 원시루프와의 최단거리 L을 계산하여 공구반경 내에 원시루프 세그먼트가 포함되는지를 판단하여 포함된다면 1 플래그를 설정한다. 마찬가지로 중점에서 그림 9.(b)와 같이 공구반경 내에 원시루프 세그먼트의 양쪽 끝점이 포함되는지 판단하여 포함된다면 오프셋된 세그먼트의 플래그 속성에 1을 설정한다. 오프셋된 세그먼트에 플래그가 설정되었다면 세그먼트를 리스트에서 제거시킨다.

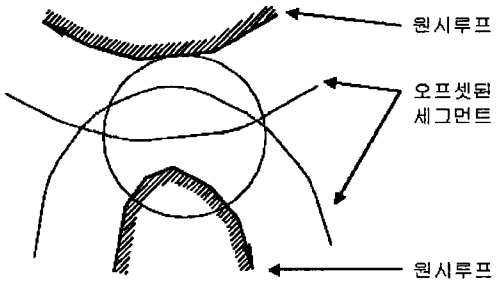


그림 8. 간섭이 발생하는 세그먼트

3.4 루프 추적

간섭이 발생할 수 있는 세그먼트를 모두 제거하고 나면 남아 있는 오프셋 세그먼트만을 대상으로 루프를 순차적으로 추적한다. 앞절에서 이미 교점이 분할될 때 세그먼트의 순서를 유지해가며 분할된 상태이므로 추적을 할 때도 많은 부분 시간을 절약할 수 있는 장점이 있다.

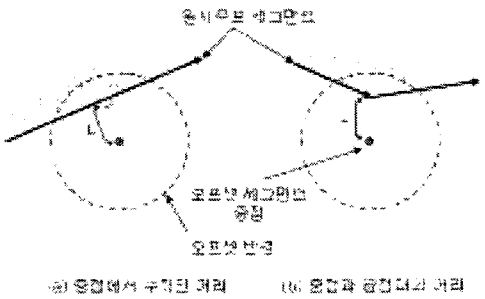
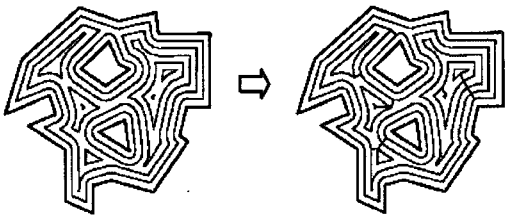


그림 9. 원시루프와의 간섭검사

4. 포켓팅 공구경로 연결

이 장에서는 윤곽 평행 오프셋팅 알고리즘을 이용하여 주어진 윤곽 프로파일의 내부영역을 가공할 수 있는 공구경로 알고리즘을 제시한다. 그림 10과 같이 연결과정에서는 오프셋된 루프의 가장 깊은 수준에서부터 시작해서 밖으로 가공해 나오는 방법을 택하여 가능한 공구의 후퇴 횟수를 줄이고 절삭방향을 일정하게 유지시키며 연결시켜나간다.



루프 오프셋 포켓팅 경로 연결

그림 10. 공구경로 연결 2단계

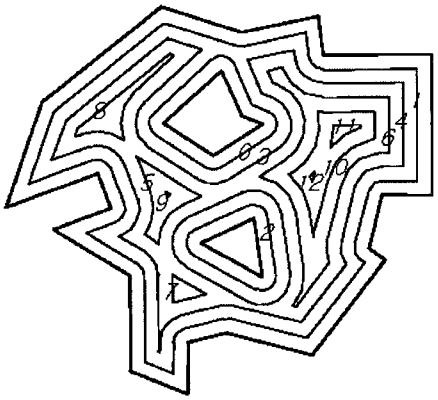
4.1 루프의 오프셋

(1) 1차 오프셋

윤곽 평행 오프셋팅을 통해 원시루프를 처음 옵셋한 수준(Level) 0의 루프(그림 11의 루프 번호 0, 1, 2)를 얻는다. 이 오프셋 루프는 최상위 수준의 부모루프이며 루프를 식별하기 위한 id가 부여된다.

(2) 연속 오프셋

1차 오프셋된 루프들을 가지고 다시 오프셋하여 2차 오프셋된 루프를 얻는다. 2차 오프셋부터는 1차 오프셋된 루프들과의 계층관계를 파악하여 자신이 몇 번째 최상위 부모루프의 자손인지를 결정한다. 계속해서 이전에 생성된 루프들을 가지고 루프가 생성되지 않을 때까지 반복한다.



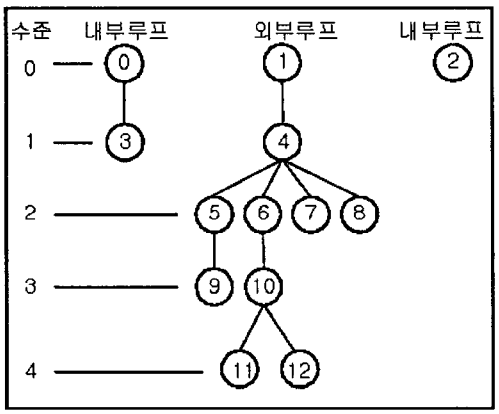


그림 11. 루프의 번호와 계층적 트리

4.2 루프의 연결

(1) 트리의 줄기 생성

루프의 그룹번호가 동일한 것만을 대상으로 그림 12와 같이 최하위 수준의 루프를 취하여 자신의 상위 루프의 가까운 점을 찾아서 끼워넣는다. 이 과정을 최상위 부모루프에 연결될 때 까지 반복한다.

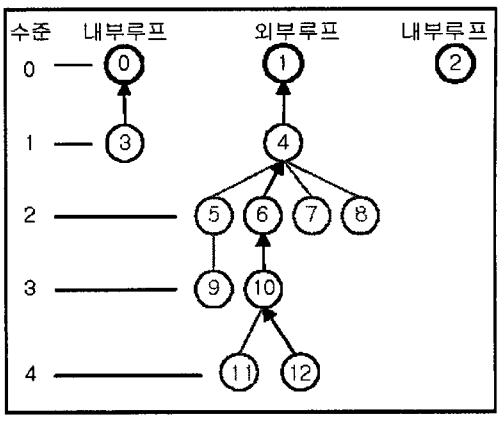


그림 12. 트리의 줄기 만들기

(2) 첫 번째 가지 연결

외부루프(Outer loops) 중에 뼈대에서 첫 번째로 뻗어나가는 가지루프만을 대상으로 최상위 부모루프(루프번호 1)에 제일 가까운 세그먼트를 찾아 루프의 시작점으로 설정하고 루프의 순서를 재정렬 시킨다. 루프가 재정렬되면 가지루프의 시작점에서 부모루프와 제일 가까운 점을 찾아서 연결시킨다.

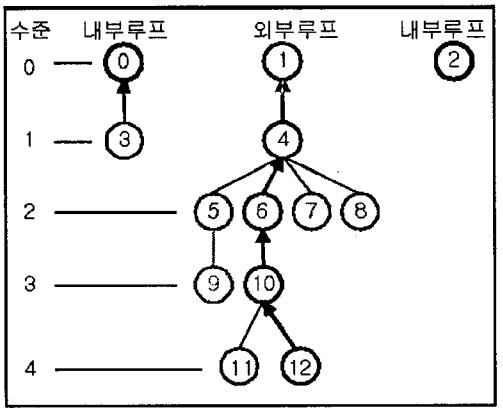


그림 13. 처음 뻗어나가는 가지 연결

(3) 나머지 가지 병합

외부루프 중에서 같은 그룹번호에 동일한 수준(Level)의 루프를 선택하여 선택된 루프 중 같은 수준의 루프가 한 개만 존재할 경우 부모루프의 시작점에서 현재의 루프와 제일 가까운 점을 찾아서 루프를 재정렬 시킨 뒤 연결시킨다. 만약 같은 수준의 루프가 여러 개 존재한다

면 그 중에서 부모루프의 시작점과 제일 가까운 루프만이 부모루프와 직접적으로 연결될 수 있으며 나머지는 부모루프에서 순서가 제일 빠른 세그먼트 순서에 의해 시작점을 재정렬 시킨 후 부모루프에 연결시킨다.

(4) 내부루프들의 병합

최상위 수준의 외부루프에 현재의 내부루프가 포함되는지를 판별하여 해당되는 외부루프에 연결한다. 내부루프를 외부루프에 병합하는 과정은 비교적 신중을 기해야 한다. 왜냐하면 외부루프를 대부분 가공하고 나서 내부루프로 옮겨 가공하지 않는다면 외부루프 가공도중 임의의 지점에서 내부루프를 가공하고 나올 수 있으므로 경우에 따라 '얇은 벽'이 생겨 가공성을 떨어뜨릴 수 있는 요인이 될 수도 있다.

(5) 연결 보류 루프의 재병합

외부루프에 속한 자식루프들 중 연결이 보류된 루프만을 대상으로 자신의 최상위 부모루프와 최근점을 구하여 간섭이 생기지 않는 범위에서 연결을 시도한다. 만약 간섭이 발생한다면 그 루프는 공구의 후퇴가 이루어진 후 따로 진입하여 가공되어야 할 루프로 속성이 설정된다. 그림 14는 연결이 보류된 루프들 중 더 이상 연결이 불가능하여 별도의 공구진입이 이루어져야 할 루프를 나타내고 있다. 또한, 그림 15는 이제까지 제시한 알고리즘을 바탕으로 최종적인 연결이 끝난 후의 모습을 보여 주고 있다.

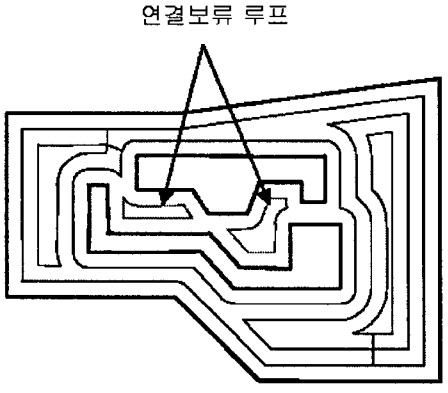


그림 14. 새로운 진입이 필요한 루프

5. 결론

본 연구에서는 2차원 윤곽 프로파일을 빠르고 안정적으로 오프셋팅하고 효과적으로 포켓팅 공구경로를 생성할 수 있는 알고리즘을 제안하였다. 오프셋팅 과정에서는 수정된 Pair wise intersection 방법을 이용하여 빠르고 안정적인 계산이 가능하도록 하였으며, 경로연결 과정에서는 공구진입 및 후퇴를 최소화하고 '얇은 벽 가공'이 발생되지 않도록 하였다. 본 연구의 결과가 기존의 상업용 시스템의 결과와 비교하였을 때 보다 우수한 결과를 나타내었다.

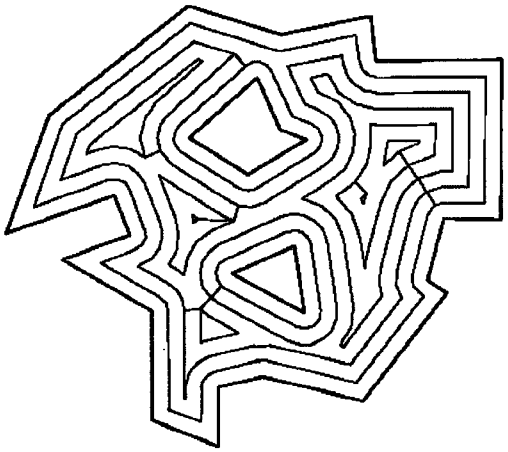


그림 15. 연결 완료 후의 경로

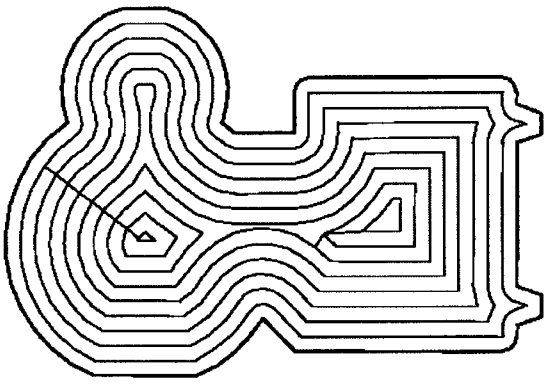


그림 16. 단일 윤곽 프로파일 포켓팅

그림 17. UniGraphics에서 산출한 포켓팅 경로(가공 중에 얇은 벽 현상이 발생하는 모습)

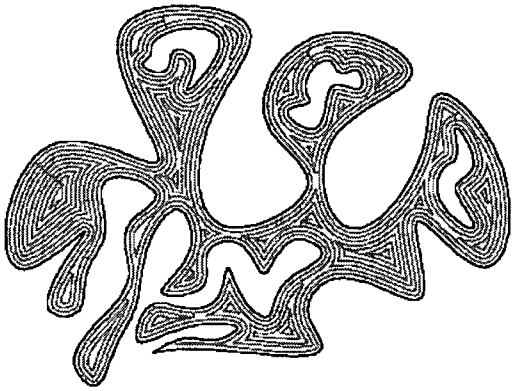


그림 18. 본 연구에서 산출한 얇은 벽 현상이 발생하지 않는 포켓팅 가공 경로

참고문헌

1) 김덕수, 「Polygon Offsetting for Pocket Machining of CAM software」, '94대한산업공학회논문집, (1994)
2) 김동수, 「다면체 모델 가공을 위한 곡선기반 방법」, 경상대학교대학원 박사학위논문, (2002)
3) 박상철, 정연찬, 「포켓가공을 위한 오프셋 공구경로 연결 알고리즘」, 한국 CAD/CAM 학회 논문집, 제6권 제3호, pp.169 173, (2001)
4) 정재훈, 「복합곡면의 설계 및 가공경로 생성에 관한 연구」, 포항공대 석사학위논문, (1994)
5) Choi, B. K., Kim, B. H., 「Die Cavity Pocketing via Cutting Simulation」, Computer Aided Design, 29(12), 837 846, (1997)
6) Hansen, A. and Arbab, F., 「An algorithm for generating NC tool paths for arbitrarily shaped pockets with islands」, ACM Transactions on Graphics, 11(2), 152 82, (1992)
7) Held, M., 「On the Computational Geometry of Pocket Machining」, Springer Verlag, (1991)
8) Tawfik, T. and Ahmed, E., 「A sweep line algorithm and its application to spiral pocketing」, society of CAD/CAM engineers, 2(3), (2002)
9) Tiller, W. and Hanson, EG., 「Offsets of two dimensional pofiles」, IEEE Computer Graphics and Applications, 36 46, (1984)
10) Yang, SN. and Huang, ML., 「A New Offsetting Algorithm Based on Tracing Technique」, '93 ACM Solid Modeling Symposium, (1993)