한국 CAD/CAM 학회 논문집
제4권 제3호 1999년 9월 pp. 269-283

학술논문

곡면간의 교차곡선 계산을 위한 개선된 Tracing 알고리즘

조두연*, 이규열**, 임중현***

An Improved Tracing algorithm for surface/surface intersection

Doo-Yeoun Cho*, Kyu-Yeul Lee** and Joong-Hyun Rhim***

ABSTRACT

Surface/surface intersection is a common and important problem in geometric modeling and CAD/CAM. Several methods have been used to approach this problem. All possible intersection curves can be obtained by using the subdivision algorithm, while it requires a great deal of memory and is somewhat inefficient. The tracing algorithm is much faster than the subdivision algorithm, and can find points on the intersection curve sequentially. But, the tracing algorithm has some problems in the intersection curves on surface boundaries. In this paper, an Improved tracing algorithm that includes some ideas such as a new trace-terminating condition for the intersection curves on surface boundaries, detecting closed intersections and extension for composite surfaces is suggested. This algorithm consists of three steps: generating start points for curve tracing, tracing intersection curves and sorting pieces of the intersection curves. The results of this algorithm and comparisons to the 'DESIGNBASE' and 'ACIS' system are presented.

Key words : Surface/surface intersection, Tracing, Subdivision, Composite surface

1. 서 론

곡면간의 교차곡선을 구하는 것은 형상모델링, CAD/CAM, robotics와 같은 많은 공학적 응용분야에서 종종 발생하는 문제로, 빠르고 정확하게 교차곡선을 구해내는 것이 중요하다.

자유곡면(free-form surface)사이의 교차곡선을 계산하는 기존의 알고리즘으로는 크게 subdivision 방법[1]과 tracing 방법[2-4], 그리고 두 방법의 장점을 취한 hybrid 방법[5] 등이 있다. Subdivision 방법은 곡면을 남김없이 분할함으로써 곡면사이에 존재하는 모든 교차곡선을 구할 수 있는 반면에 많은 저장공간이 필요하고 수행시간이 느린 단점이 있다. 이에 반해 tracing 방법은 수행시간이 빠르고 교차곡선상의 점들을 순차적으로 구해낼 수 있지만, 그러기 위해서는 먼저 교차곡선상의 최소 한 개이상의 초기점(starting point)을 구해내야 하는 문제점이 있었다. 또한, 곡면의 경계에서 교차곡선이 존재하는 경우에 기존의 추적 종결조건(tracing terminating condition)을 적용하면 올바른 교차곡선을 구할 수 없었다(2.4.5절 참고). 따라서, 본 논문에서는 기존의 subdivision 방법과 tracing 방법을 개선, 확장하여 곡면사이의 교차곡선을 계산하는 보다 안정적인 알고리즘을 제안하였고, 이를 구현해서 그 효율성을 입증하고자 하였다.

본 논문에서 개선, 확장한 내용은 크게 다음과 같다.

- 복합곡면(composite surface)간의 교차곡선 계산을 위하여 기존의 알고리즘을 확장
- 곡면경계에서 교차곡선이 존재하는 경우에도 올바른 교차곡선을 구할 수 있도록 기존의 추적 방법을 개선, 이를 위해서 기존의 추적 종결조건을 새로운 추적 종결조건으로 대체
- 닫힌 교차곡선(closed intersection curve)을 처리하는 알고리즘 제안

2. 개선된 tracing 알고리즘

본 논문에서는 subdivision 방법과 tracing 방법을

*학생회원, 서울대학교 조선해양공학과 대학원
**종신회원, 서울대학교 조선해양공학과
***정회원, 서울대학교 조선해양공학과 대학원

혼합한 Barnhill[4]의 방법을 토대로, 기존의 Barnhill[4]의 알고리즘이 곡면경계에서 교차곡선이 존재하는 경우에 올바르게 처리해 주지 못하는 문제점을 개선한 추적알고리즘(tracing algorithm)을 제안, 구현하였다.

2.1 알고리즘 개요(Algorithm outline)

본 논문의 알고리듬은 다음과 같은 곡면사이의 교차곡선을 계산할 수 있도록 작성되었다.

- 모든 (u, v)값에 대해서 곡면상의 점과 u, v방향의 1차 미분값을 계산할 수 있는 rectangular parametric C^1 surface 또는 rectangular parametric C^1 patch들로 이루어진 복합곡면(composite surface)
- 곡면을 Bezier 곡면이나 B-spline 곡면과 같이 특별한 구조로 가정하지 않는다.

일반적으로 두개의 곡면이 교차할 때 일어날 수 있는 상황은 교차곡선이 발생하는 경우(접선 포함), 접점이 발생하는 경우, 접면이 발생하는 경우이다. 본 논문의 알고리즘은 접면이 발생하는 경우는 고려하지 않았다.

본 논문의 알고리즘은 초기점(starting point)을 구해내는 단계, 초기점으로부터 교차곡선 위의 점을 순차적으로 찾아내는 추적(tracing) 단계, 그리고 구해낸 교차곡선상의 연속된 점들을 올바르게 이어주는 정렬(sorting) 단계로 나누어진다.

다음은 본 논문의 알고리즘의 개략적인 과정을 나타낸 것이다.

과정 I. 입력받은 두 개의 복합곡면을 구성하는 단일곡면들에 대해서 II에서 III까지의 과정을 반복한다.

과정 II. 초기점을 계산한다(2.3절 참고).

1. 두 곡면이 곡면분할 기준(편평도와 곡면의 경계곡선이 직선에 가까운 정도)을 만족하지 않으면 각 곡면을 4조각으로 분할한다(2.3.1절 참고).
2. 분할된 각 곡면조각을 포함하는 bounding box를 생성한다(2.3.2절 참고).
3. 생성된 bounding box간의 간섭계산을 통해서 교차가능성이 있는 곡면조각들의 쌍을 저장한다.
4. 다시 1번부터 3번까지의 과정을 되풀이한다.
5. 4번의 과정을 거치면 주어진 곡면분할 기준을 만족하고 교차가능성이 있는 곡면조각들의 쌍을 얻게 된다.
6. 교차가능성이 있는 2개의 곡면조각의 꼭지점(각 u,v 파라미터의 경계값에서의 점, 예를 들면 (0,0), (0,1), (1,0), (1,1) 파라미터에서의 점) 8개의 평균점을 계산한다(2.3.3절 참고).
7. 6번에서 계산한 평균점을 가지고 교차곡선상의 점을 구한다. 이 점이 교차곡선을 추적하기 위한 초기점이 된다.
8. 6에서 7번까지의 과정을 모든 교차가능성이 있는 곡면조각들의 쌍에 대해서 수행하여 교차곡선상의 초기점들을 계산한다.

과정 III. 구해낸 초기점을 시작점으로 교차곡선을 trace한다(2.4절 참고).

1. 추적벡터(trace vector)의 방향과 크기를 구한다.
2. 추적벡터의 끝점을 가지고 교차곡선상의 점들을 차례대로 구한다.
3. 2번의 과정 중에 중복된 초기점이 존재할 경우 이를 제거한다.
4. 2번의 과정을 수행하면서 추적하고 있는 교차곡선이 닫힌 교차곡선(closed intersection curve)인지 검사한다. 닫힌곡선이라면 그에 따른 처리를 하고 추적을 마친다.
5. 2번의 과정을 수행하면서 추적하고 있는 교차곡선이 추적종결조건을 만족하는지 검사해서 만족하면 이 방향으로의 교차곡선 추적을 끝내고, 다시 초기점에서 반대방향으로 1번부터의 과정을 반복한다. 반대방향으로의 교차곡선 추적을 종결되게 되면, 이 초기점에 해당하는 교차곡선상의 점들을 모두 구한 것이다.
6. 모든 초기점에 대하여 1번부터 5번까지의 과정을 반복한다.

과정 IV. 과정III에서 구한 교차곡선상의 점들을 정렬한다.

2.2 복합곡면(composite surface)의 처리

복합곡면이란 주어진 기하학적 연속조건을 만족하는 단일 곡면(surface patch)들로 이루어진 곡면을 말한다[6,7]. 선박의 선수, 선미부분은 매우 복잡한 형상을 가지고 있어서 단일 곡면으로는 표현하기 힘들기 때문에 복합곡면으로 표현하는 경우가 많다. 선박의 선수나 선미부분을 복합곡면으로 표현하는 것에 관련된 연구는 임[8]에 의해 수행된바 있다. 본 논문에서는 기존의 단일 곡면간의 교차곡선계산 알고리즘을 반복하여 복합곡면간의 교차곡선도 처리할 수 있도록 하였다(Fig. 1참고).

복합곡면간의 교차곡선계산은 내부적으로는 단일 곡면간의 교차곡선계산 알고리즘을 이용하기 때문에 이를 수행한 결과는 단일 곡면간의 교차곡선들이다. 하

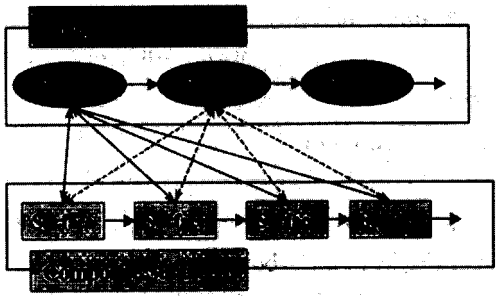


Fig. 1. Intersection between composite surfaces.

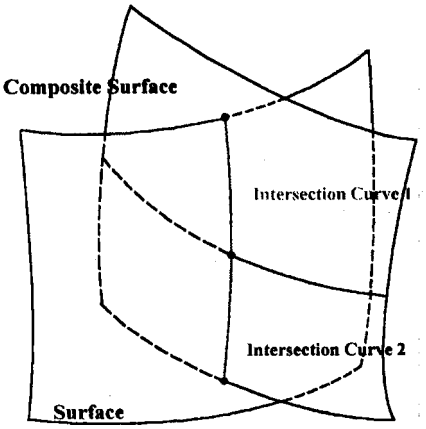


Fig. 2. Necessity of sorting.

지만 우리가 원하는 것은 복합곡면간의 교차곡선이기 때문에 정렬(sorting) 단계에서, 필요하다면 하나의 곡선으로 이어주는 과정이 필요하다(Fig. 2참고).

2.3 초기점 생성(starting point generation)

본 알고라듬에서는 곡선을 추적하기 위한 초기점을 구하기 위해서 곡면을 삼각형 평면들로 근사하지 않는다[1][3][9] 곡면을 근사하게 되면, 접점이나 접선 등에 해당하는 초기점을 계산하지 못하는 경우가 있었다. 따라서 본 논문에서는 Barnhill[4]의 방법을 채택해서 곡면을 완전히 포함하도록 bounding box를 생성하고, bounding box간의 간섭체크를 통한 곡면간의 교차가능성을 판단하여 접선, 접점에 해당하는 초기점을 최대한 누락하지 않도록 하였다.

2.3.1 곡면분할(surface subdivision)

곡면을 분할하는 방법은 크게 uniform subdivision 방법과 adaptive(non-uniform) subdivision 방법으로 나눌 수 있다. 또한 두 개의 곡면사이의 교차가능성을 곡면분할의 조건으로 추가한 선택적 uniform subdivision 방법과 선택적 adaptive subdivision 방법이 있다.

이러한 여러 가지 분할 방법들은 Fig. 3에서 그

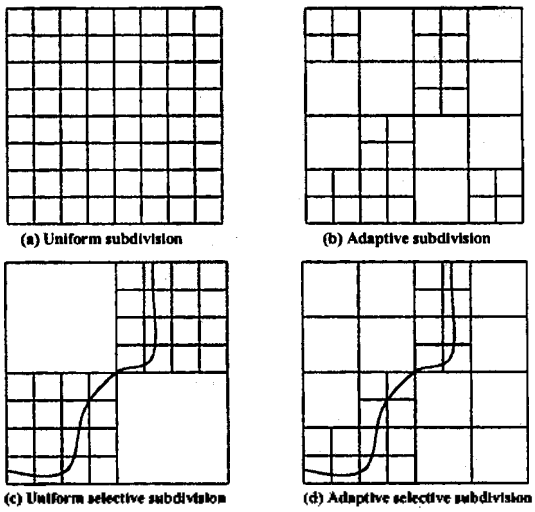


Fig. 3. Various subdivision methods.

차이점을 살펴볼 수 있다. Fig. 3(a)는 3단계까지 uniform subdivision을 수행한 결과를 곡면의 파라미터 공간에 도시한 그림이다. Fig. 3(c)와 Fig. 3(d)는 교차가능성이 있는 부분에 대해서 선택적 uniform subdivision과 adaptive subdivision을 수행한 결과를 나타낸 것이다.

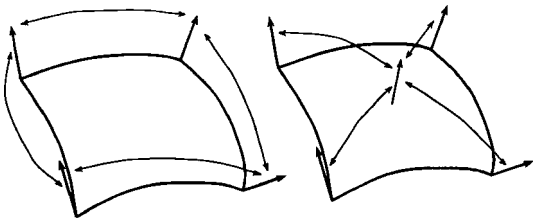
본 논문에서는 알고리즘의 효율을 위해서 곡면을 미리 정해진 단계까지 일정하게 분할하는 uniform subdivision 방법대신, 곡면의 편평도(flatness)와 곡면경계곡선이 직선과 가까운 정도(edge linearity)를 분할조건으로 하고 서로 교차할 가능성이 있는 부분만을 분할하는 선택적 adaptive subdivision 방법을 사용하였다.

곡면의 분할과정에서 사용되는 곡면의 편평도와 곡면경계곡선이 직선과 가까운 정도는 Barnhill[4]의 연구에서 사용했던 식을 이용하였다.

(1) 편평도 계산: 곡면의 4개의 꼭지점(예를 들면 (0,0), (1,0), (0,1), (1,1) 파라미터에 해당하는 곡면상의 점)과 파라미터의 평균값(예를 들면 (0.5,0.5))에 해당하는 점의 곡면 법선벡터(surface normal vector)사이의 각도를 기반으로 편평도를 계산한다. 각 꼭지점에서의 곡면 단위법선벡터(unit surface normal vector)를 Ni라고 하면, 곡면의 편평도를 다음의 식으로 나타낼 수 있다[4](Fig. 4, 부록 I 참고).

$$N_i \cdot N_j = |N_i||N_j| \cos\theta = \cos\theta \qquad (1)$$
$$1 - N_i \cdot N_j = 1 - \cos\theta < \textit{Flatness Tolerance} = 0.1$$

4개의 꼭지점끼리 계산한 편평도값과, 파라미터의 평균값에 해당하는 점과 4개의 꼭지점끼리 계산한 편평도



(a) 4번의 편평도 계산 (b) 4번의 편평도 계산

Fig. 4. The calculation of surface flatness.

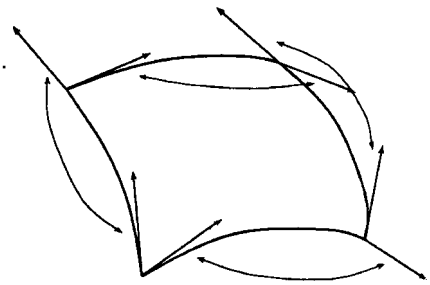


Fig. 5. The calculation of edge linearity.

값 중 최대 값을 그 곡면의 편평도값으로 간주한다.

(2) 곡면경계가 직선과 가까운 정도(Edge linearity Test): 곡면의 경계를 직선으로 볼 수 있는지 판단하는데 기본이 되는 값으로 각 꼭지점에서의 u, v 파라미터 방향의 접선 벡터사이의 각도를 기반으로 계산한다. 각 꼭지점에서의 곡면 단위 법선벡터를 Ti라고 한다면 곡면경계가 직선과 가까운 정도를 나타내는 식은 다음과 같다[4](Fig. 5, 부록 II 참고).

$$T_i \cdot T_j = |T_i||T_j| \cos\theta = \cos\theta \qquad (2)$$
$$1 - T_i \cdot T_j = 1 - \cos\theta < \textit{Edge Linearity Tolerance} = 0.1$$

2.3.2 Bounding box

본 논문에서는 효율적인 곡면분할을 위해서 교차가능성이 있는 곡면들만 분할하는 선택적 adaptive subdivision을 수행한다. 따라서 2개의 곡면사이의 교차가능성을 판단하여야 하는데, 이 때 bounding box를 사용하게 된다. 즉 분할된곡면조각을 포함하는 bounding box가 서로 교차한다면 그에 해당하는 곡면조각끼리 교차할 가능성이 있다고 판단하는 것이다. 주[5]는 Bezier 곡면 또는 B-spline 곡면과 같이 볼록포(convex hull)성질이 존재하는 곡면의 bounding box를 조정점을 포함하는 최소최대상자로 생성하였다. 그러나 최소최대상자를 이용하는 방법은 2개의 곡면이 방향이 비슷하고 가까운 거리에 있다면 너무나 많은 분할과정을 거쳐야만 실제로 교차하는지 아닌지를 알 수 있게 되어 비효율적이고[11], 볼록포 성질이 존재하지 않는 곡면에 적용하기 힘든 단점이 있다. 따라서 본 논문에서는 Houghton[11], Barnhill[4]에 의해서 제시된 볼록포 성질이 없는 곡면에 대하여도 적용할 수 있는 곡면방향을 반영하는 bounding box(surface oriented bounding box)를 사용하였다.

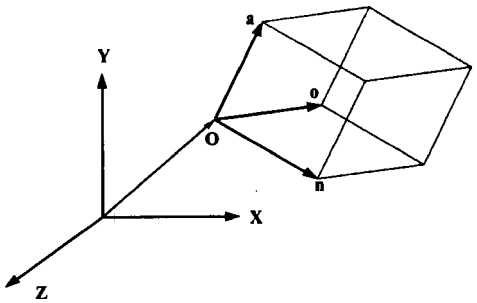


Fig. 6. The representation of bounding box.

곡면방향을 반영하는 bounding box를 쉽게 표현하기 위해서 곡면방향을 반영하는 지역 좌표계(surface oriented local coordinate)를 설정할 수 있다. 그러면 bounding box를 생성하는 것은 곡면방향을 반영하는 지역 좌표계의 n,o,a축의 성분을 전체좌표계에 대해서 계산하는 것과 같은 작업이 된다(Fig. 6참고).

Bounding box를 생성하는 데 있어서 가장 중요한 것은 bounding box가 곡면조각을 완전히 포함하게 하는 것이다. 본 논문에서 bounding box를 생성하는 과정은 다음과 같다.

(1) 초기생성: 먼저 곡면조각의 4개의 꼭지점을 포함하는 bounding box를 만들어 낸다.

① n=V_1-V_0, o=V_3-V_0, a의 방향은 n×o 방향과 같고, 크기는 1로 결정한다(Fig. 8의 a참고).

② ①에서 만들어진 n, o, a는 V_2는 고려하지 않은 것이므로 V_2를 포함하도록 n, o, a를 늘린다(Fig. 8의 b참고).

(2) 최종생성: (1)에서 만들어진 bounding box는 일반적으로 곡면경계와 곡면의 내부를 완전히 포함한다고 할 수 없다. 따라서 식 (1),식 (2)에 의해서 계산되는 곡면의 편평도와 곡면경계가 직선에 가까운 정도에 근거해서, 각 좌표축방향으로 bounding box를 확장해서 곡면조각을 완전히 포함할 수 있는 가능성을 높여야 한다(Fig. 8의 c참고).

① a축은 곡면내부가 위 또는 아래로 튀어나온 높이를 고려해서 늘려야 한다. 그러나 곡면내부의 모든 점을 조사할 수는 없으므로 다음 식을 이용해서 확장할 길이 2Ha를 근사적으로 계산해서 a축을 늘린다(Fig. 7, 부록 III참고).

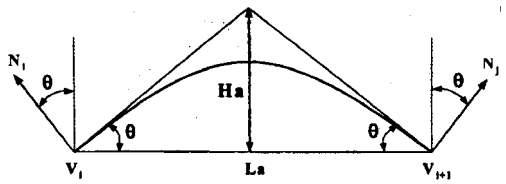


Fig. 7. The extension of bounding box.

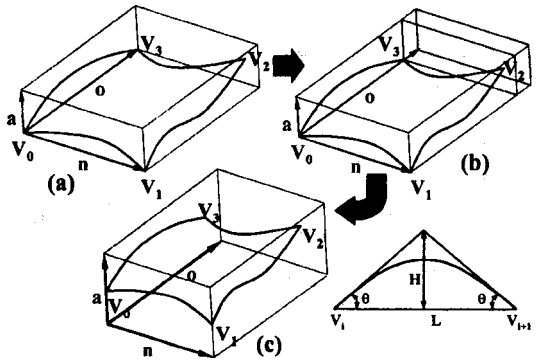


Fig. 8. The generation of surface-oriented bounding box.

$$L_a=\max\ (\|V_2-V_0\|,\ \|V_3-V_1\|)$$

$$\alpha=\max(N_i\cdot N_j,\ T_i\cdot T_j)\ \doteq \cos 2\theta$$

$$H_a=\frac{L_a}{2}\sqrt{\frac{1-\alpha}{1+\alpha}}=\frac{L_a}{2}\sqrt{\frac{1-\cos 2\theta}{1+\cos 2\theta}}$$

$$=\frac{L_a}{2}\sqrt{\frac{\sin^2\theta}{\cos^2\theta}}=\frac{L_a}{2}\tan\theta \qquad (3)$$

② 좌표축 o와 n의 경우에도 ①과 비슷하게 Lo= max($\|V_0-V_3\|$, $\|V_2-V_1\|$), Ln=max($\|V_1-V_0\|$, $\|V_2-V_3\|$)으로 하여 2Ho, 2Hn가 되도록 늘린다(부록 III참고).

③ 곡면방향을 반영하는 지역 좌표계(surface oriented local coordinate)의 원점을 (-Ha,-Ho,-Ha)로 옮긴다(부록 III참고).

Fig. 8은 bounding box를 생성하는 과정을 나타낸 것이다.

2.3.3 초기점(starting point) 계산

2.3.1에서 선택적 adaptive subdivision을 수행하고 나면, 서로 교차할 가능성이 있고, 분할허용기준을 만족하는 곡면조각 쌍들을 얻을 수 있다. 이제 이들로부터 교차곡선을 추적하기 위한 초기점을 계산해야 한다.

두개의 교차가능성이 있는 곡면조각으로부터 초기점을 구하는 방법은 다음과 같다.

일단 각 곡면조각들의 8개의 꼭지점(곡면조각 1개당 4개의 꼭지점, 곡면조각 쌍이므로 총 8개)의 평균점을 구한다(Fig. 9참고). 이점은 일반적으로 교차곡선 위에 존재하는 점이 아니다. 하지만 교차곡선 근처에

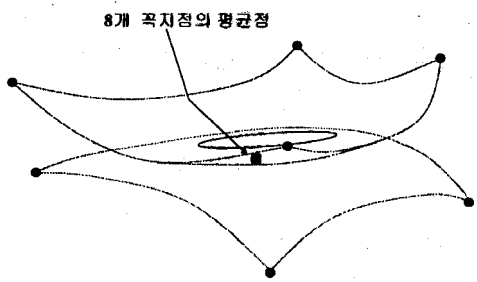


Fig. 9. The calculation of initial value for starting point.

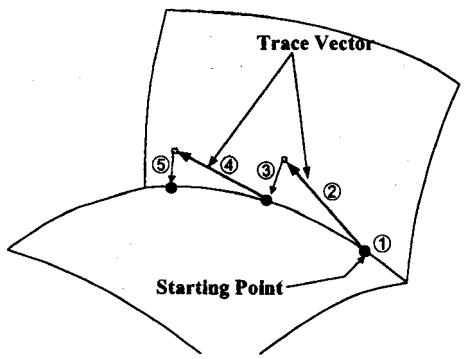


Fig. 10. The procedure of curve tracing.

존재하는 점일 것이다. 이 평균점을 가지고 교차곡선상의 초기점을 계산한다.(자세한 과정은 2.4.3과 2.4.4에 설명되어 있다.)

2.4 교차곡선을 추적(tracing)하는 과정

추적단계에서는 곡면분할단계로부터 계산한 초기점에서부터 교차곡선 위의 점들을 차례로 구하게 된다. 교차곡선을 추적하기 위해서는 먼저 추적벡터의 방향과 크기를 결정하고, 추적벡터의 끝점을 초기값으로 교차곡선상의 점을 계산해야 한다. 그 후 계산된 교차곡선상의 점에서 다시 추적벡터를 구해서 교차곡선상의 또 다른 점을 구하는 과정을, 추적 종결조건을 만족할 때까지 반복하여 교차곡선상의 점들을 순차적으로 구한다. 한 쪽 방향으로 추적을 마친 후에는 초기점으로부터 반대 방향으로 추적해서 완전한 교차곡선을 계산할 수 있다(Fig. 10참고).

2.4.1 추적벡터(Trace vector) 방향 결정

교차곡선 위의 어떤 점에서의 추적벡터의 방향은 그 점에서의 각 곡면 법선벡터의 와적으로 정의할 수 있다.

Trace vector 방향=Surface 1 normal×Surface 2 normal

그러나 곡면의 법선벡터가 서로 평행하다면 trace

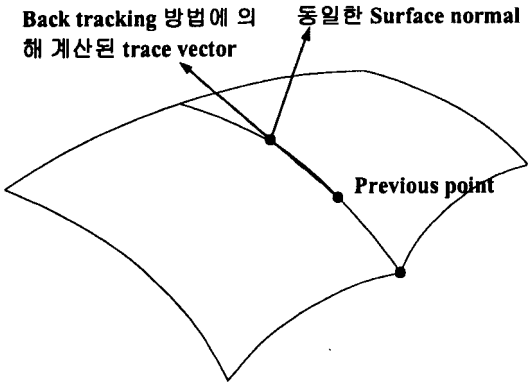


Fig. 11. Back tracking method.

vector의 방향을 결정할 수가 없다. 예를 들면, 교차곡선이 공통의 surface tangent plane을 가지는 접선인 경우에 이러한 상황이 발생한다. 만약 바로 전에 구해놓은 교차곡선상의 점이 존재한다면, 현재의 점과 바로 이전 점의 차이를 추적벡터의 방향으로 삼는 back tracking 방법을 사용한다[3,4](Fig. 11참고).

그러나 초기점에서 두 개의 곡면의 법선벡터가 평행할 때에는 back tracking 방법을 사용할 수 없다. 왜냐하면 바로 이전 점이 존재하지 않기 때문이다. 이 경우에는 어느 한 곡면의 파라미터 공간에서 초기점의 파라미터값 주위로 sampling circle을 생성해서 이 원상의 파라미터값에 해당하는 곡면상의 점을 초기값으로 교차곡선상의 점을 구한다. 그 후 구해진 교차곡선상의 점과 초기점과의 차이를 추적벡터의 방향으로 결정하는 sampling method를 사용한다[4](Fig. 12참고).

만약에 sampling방법으로도 추적벡터의 방향을 구해내지 못하면, 이 점 근처의 연속된 점을 더 이상 구해낼 수 없는 것으로 해석하고 이 점을 접점으로 판단하게 된다.

2.4.2 추적벡터(Trace vector)의 크기 결정

추적벡터의 방향을 알게 되면, 이제 남은 것은 추적벡터의 크기를 결정하는 일이다. Barnhill[3]은 CRT (Curve Refinement Tolerance)라는 고정적인 사용자 입력 값을 사용하여 추적벡터의 크기를 계산하였다. 그러나 CRT는 길이에 기반을 둔 값이기 때문에 입력된 곡면의 크기에 영향을 받아서 올바르지 못한 결과를 줄 수 있다. 예를 들면, 고정적으로 입력된 CRT의 크기보다 훨씬 큰 곡면에 대해서는 하나의 교차곡선을표현하기 위해 너무나 많은 교차곡선상의 점을 계산해야 한다. 또한 CRT라는 고정적인 추적벡터의 크기는 구하려는 교차곡선의 곡률을 반영하지 않기 때문에, 교차곡선상의 점들을 이은 piecewise linear intersection curve와 실제 교차곡선사이의 오차가 커지는 단점이 있다.

Barnhill[4]은 이러한 이유 때문에 CRT를 사용하는 대신, 근사적으로 계산한 교차곡선의 곡률과 angle tolerance $\Delta\theta$를 기반으로 해서 추적벡터의 크기를 결정하였다. 그리고 기존의 CRT는 추적벡터 크기의 최대값을 제한하는데 사용하였다. 본 논문에서는 Barnhill[4]의 방법을 이용해서 추적벡터의 크기를 결정하였다(Fig. 13참고).

본 논문에서 CRT의 값은 1e-1로 결정하였다.

Angle tolerance $\Delta\theta$가 주어졌을 때, 추적벡터 의 크기는 다음과 같은 식으로 계산한다.

$$L=\rho\Delta\theta$$

where ρ is the radius of curvature

본 논문에서는 $\Delta\theta=3^\circ$ (4)

이 식으로 계산하게되면 곡률이 큰 부분에서는 추적벡터의 크기가 작아져서 교차곡선상의 점을 보다 많이 계산하게되며, 곡률이 작은 부분에서는 추적벡터

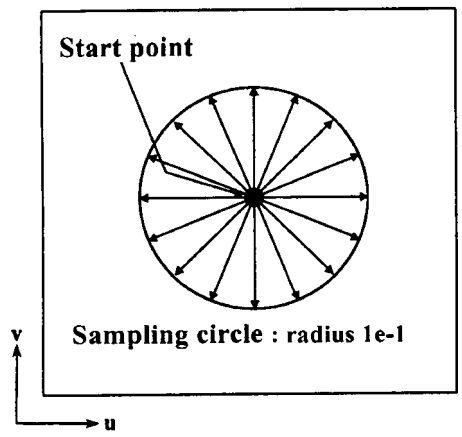


Fig. 12. Sampling method.

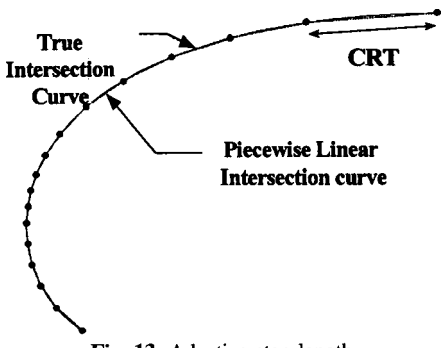


Fig. 13. Adaptive step length.

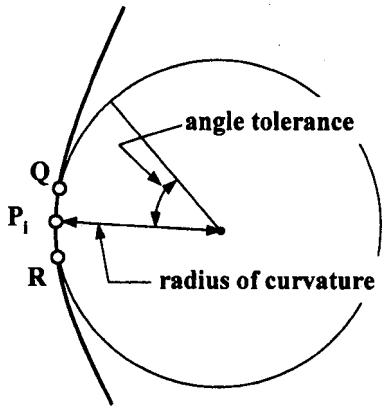


Fig. 14. The Calculation of curvature.

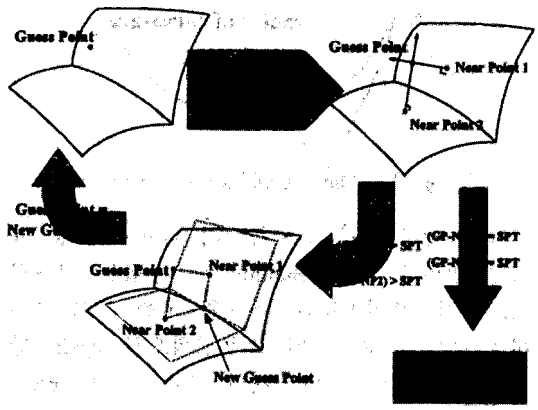


Fig. 15. The geometric iteration method for intersection point.

의 크기가 커져서 교차곡선상의 점을 보다 작게 계산하게된다.

이제 남은 것은 교차곡선의 곡률 반경을 계산하는 일인데, 역시 Barnhill[4]의 방법을 이용하였다. 먼저, 추적벡터의 방향을 구한 점을 Pi라고 하면, 이 점 Pi로부터 전후방향으로 epsilon만큼 떨어진 교차곡선상의 2개의 점을 구한다(Fig. 14의 점 Q, R) 그 다음에 Q, Pi, R 세 점으로 결정되는 원을 구하면, 이원은 근사적으로 점 P에서의 osculating circle로 생각할 수 있고, 이 원의 반지름은 점 P에서의 곡률 반경으로 사용할 수 있다. 주어진 세 점 Q, Pi, R로 이루어지는 원의 반지름은 다음의 식으로 계산할 수 있다[10].

$$\rho = \frac{|a||b||a-b|}{2|a\times b|}$$
$$where\ a=Q-P_i,\ b=R-P_i \qquad (5)$$

2.4.3 교차곡선상의 점을 구하는 방법(point refinement)

본 논문에서는 교차곡선 근처의 점을 가지고 교차곡선상의 초기점를 구하기 위해서, 그리고 추적 단계에서 추적벡터의 끝점을 초기값으로 연속적인 교차곡선상의 점을 계산하기 위하여 point refinement 과정을 필요로 한다.

결국 point refinement는 두 개의 곡면의 교차곡선상의 점을 찾아내는 과정이다. 그런데, surface/surface intersection식은 미지수의 개수가 식의 개수보다 많은 under-determined system of equations이기 때문에 해를 구하기 위해서는 수치적인 방법을 사용해야 한다. surface/surface intersection식의 해를 구하기 위한 대표적인 방법으로는Newton-Raphson iteration 방법과 이를 기하학적으로 해석한 Houghton[1], Barnhill[3,4] 등의 방법이 있다.

본 논문에서는 기본적으로는 Barnhill[4]에 의해서 연구된 surface/surface intersection을 계산하는 기하학적인 iteration 방법을 이용하였다.

기하학적인 iteration 방법을 이용해서 교차곡선상의 점을 계산하는 과정은 다음과 같다(Fig. 15참고).

먼저, 입력된 교차곡선 근처에 있는 추정점(guess point)에 대해서 Jacobian Inversion을 수행하여 추정점에 가장 가까운 각 곡면 위의 점(near point)을 계산한다[7]. 그러면 각 곡면 위의 점에서의 곡면 법선벡터로 정의되는 2개의 평면을 구할 수 있다. 그리고 추정점과 추정점으로부터 가장 가까운 곡면위의 점(near point) 2개로 정의되는 또 하나의 평면을 계산할 수 있다.

이 3개의 평면간의 교점은 일반적으로 처음의 추정점보다 교차곡선에 더 근접한 점이 된다. 이 세 평면간의 교점을 새로운 추정점으로 생각해서 위의 과정을 계속해서 반복하면 점점 더 교차곡선에 가까운 점을 구할 수 있다. 이러한 반복적인 과정 중에 추정점에서 가장 가까운 각 곡면위의 두 점(near point1, near point2) 사이의 거리가 허용오차(SPT: Same Point Tolerance)보다 작아서 같은 점이라고 생각할 수 있게 되면, 그 점을 교차곡선상의 점(intersection point)이라고 판단한다.

본 논문에서 SPT는 1e-7으로 결정하였다.

2.4.4 개선된 곡면경계에서의 교차곡선상의 점을 계산하는 방법

2.4.3에서 설명한 기하학적인 방법으로는 곡면경계에서 교차곡선이 존재하는 경우에 교차곡선 위의 점을 정확하게 계산하지 못하는 단점이 있다[4]. 이에 Barnhill[4]은 곡면의 경계에서 교차곡선이 존재하는 경

우에는, 곡면경계 곡선과 또 다른 곡면사이의 curve/surface intersection식을 이용해서 교차곡선 위의 점을 계산하였다.

두 개의 곡면 $S_0(s, t)$와 $S_1(u, v)$이 주어지고 $s=s^*$ (예를 들면 s=0, 1)인 곡면경계에 교차곡선이 존재한다면, 교차곡선상의 점을 구하는 식은 $S_1(u, v) - S_0(s^*, t) = 0$이다. 이 식을 선형화하면 다음의 iteration 식 (6)을 구할 수 있다[4]. 따라서 입력된 교차곡선근처의 추정점을 초기값으로 식 (6)의 iteration식을 이용하면 교차곡선상의 점을 구할 수 있게된다.

$$\left[-\left|\frac{\partial S_0}{\partial t}\right|_{s^*,t_i} \left|\frac{\partial S_1}{\partial u}\right|_{u_i,v_i} \left|\frac{\partial S_1}{\partial v}\right|_{u_i,v_i}\right] \begin{bmatrix} t_{i+1}-t_i \\ u_{i+1}-u_i \\ v_{i+1}-v_i \end{bmatrix} = [S_0(s^*,t_i)-S_1(u_i,v_i)] \quad (6)$$

Iteration terminates

when $|S_0(s^*,t_{i+1})-S_1(u_{i+1},v_{i+1})|<SPT$

본 논문에서는 기본적으로 위에서 설명한 Barnhill[4]의 식 (6)을 이용하여 곡면경계에서의 교차곡선상의 점을 계산한다.

그러나 Fig. 16과 같이 곡면경계에 교차곡선이 존재하고, 교차곡선에서 각 곡면의 법선벡터가 평행한 경우에는 식 (6)의 왼쪽의 3×3 행렬의 역행렬을 계산할 수 없게 되어 iteration을 수행할 수 없다.

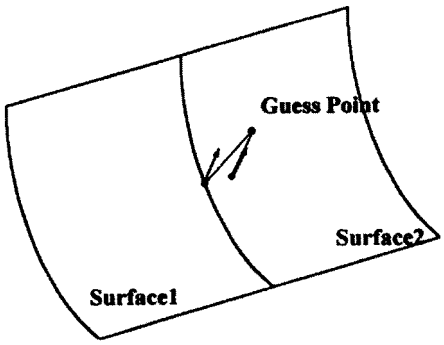


Fig. 16. The case that has the problem with formula (6).

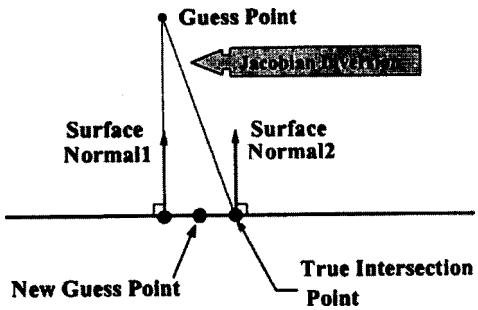


Fig. 17. New geometric iteration method for intersection point on surface boundaries.

따라서 본 논문에서는 이러한 경우에 입력된 추정점으로부터 가장 가까운 곡면상의 2점(near point)을 Jacobian Inversion을 통해 구한 뒤, 이 2점(near point)의 중점을 새로운 추정점으로 계산하여 iteration하는 방법을 제안한다(Fig. 17참고).

이러한 방법을 적용해서 새로운 추정점을 반복해서 계산하게되면, 추정점에서 가장 가까운 곡면상의 점(near point)이 곡면경계를 향해 수렴하게 되어 곡면경계에서의 교차곡선상의 점을 구할 수 있다.

2.4.5 새로운 추적 종결조건

기존의 추적 알고리즘에서 교차곡선 추적(trace)을 종결하는 조건은 크게 다음의 2가지 경우이다.

(1) 추적도중에 두 곡면 중 어느 한 곡면의 곡면경계와 만났을 때

(2) 추적을 하다가 다시 시작점으로 돌아오게 되어 닫힌 교차곡선(closed intersection curve)이 발생하는 경우

그런데 기존의 추적 종결조건 중에서 (1)은 문제를 발생시킬 수 있는데, 바로 곡면경계에서 교차곡선이 발생할 때이다(Fig. 18참고) 이런 경우에는 초기점에서 한쪽 방향으로 trace를 해서 교차곡선상의 한 점을 구한 뒤에, 추적 종결조건 (1)을 만족하기 때문에 이 방향으로의 추적을 끝내고, 다시 반대방향으로 추적을 해서 또 하나의 점을 구한 뒤, 또 다시 추적 종결조건 (1)을 만족하기 때문에 교차곡선 추적이 종결되어서, 교차곡선 위의 3개의 점밖에는 구해내지 못한다.

따라서 본 논문에서는 기존의 추적 종결조건 중 문제를 발생시키는(1)을 다음과 같은 새로운 추적 종결조건으로 대체하였다.

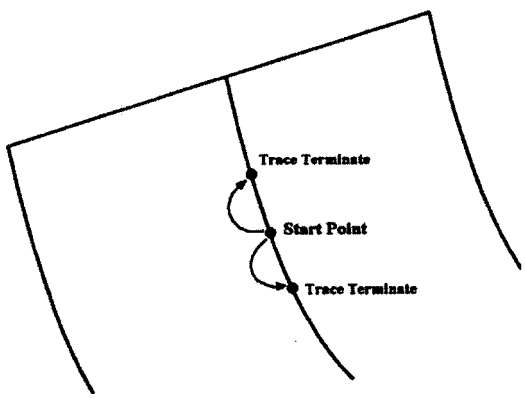


Fig. 18. The problem with previous tracing terminating condition.

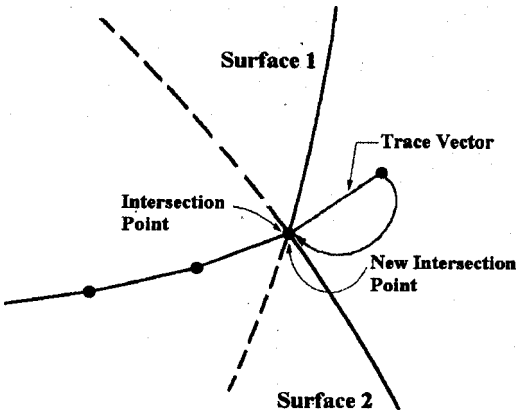


Fig. 19. The result with new tracing terminating condition (1).

(1) 교차곡선 추적을 수행하다가 어느 한 곡면의 경계와 만날 때 종결하는 것이 아니라, 새롭게 계산한 교차곡선상의 점과 바로 전에 구해놓은 교차곡선상의 점이 일치하게 되면 추적을 종결한다.

바로 전에 구한 교차곡선상의 점과 새롭게 계산한 교차곡선상의 점이 일치하게 되면, 계속해서 무한루프를 돌게 되므로 교차곡선 추적을 종결해주는 것은 당연한 조건이다.

Fig. 19는 곡면경계에서 교차곡선이 존재하지는 않지만, 어느 한 곡면의 경계와 만나게 되어 교차곡선을 더 이상 추적할 수 없는 경우이다. 기존의 추적 종결조건(1)을 적용한 경우 올바르게 교차곡선 추적을 종결할 수 있다. 그리고 본 논문에서 제안한 새로운 추적 종결조건으로도 올바르게 추적이 종료되므로 새로운 종결조건은 기존의 종결조건을 대체해도 아무런 문제가 없다.

Fig. 20은 기존의 추적 종결조건에서는 올바르게 처리할 수 없었던 경우로, 새로운 추적 종결조건을 적용하면 곡면경계에 존재하는 교차곡선상의 점들을 순차적으로 구해가다가, 교차곡선의 끝에서 새로 구한 교차곡

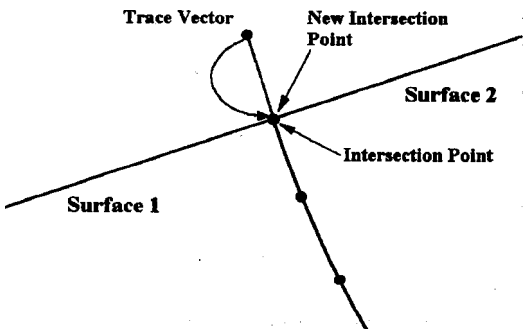


Fig. 20. The result with new tracing terminating condition (2).

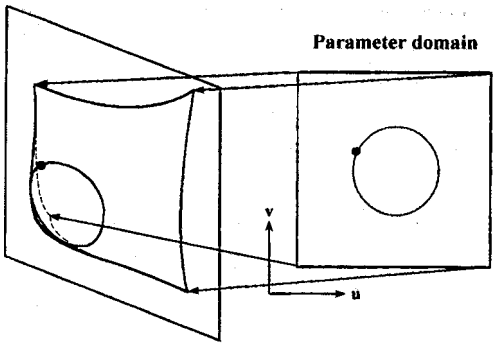


Fig. 21. Closed intersection.

선상의 점이 이전에 구한 점과 일치하게 되어 추적을 종료하게 되어 올바른 교차곡선을 얻을 수 있다.

2.4.6 닫힌 교차곡선의 처리방법

곡면사이에 닫힌 교차곡선이 존재 할 경우에는 이를 정확히 판단하여 곡선의 시작점과 끝점을이어주고 추적을 끝내서 교차곡선을 끊임없이 추적하는 것을 막아야 한다.

교차곡선상의 점을 순차적으로 계산하는 추적 알고리즘에서는 교차곡선이 닫힌 곡선이라 할지라도 일반적으로 끝점이 시작점과 정확하게 일치하도록 구해지는 경우는 흔치않다. 따라서 교차곡선을 추적하는 과정에서 닫힌 교차곡선(closed intersection curve)인지 아닌지를 정확히 판단하는 일이 중요하다.

본 논문에서는 닫힌 교차곡선이 존재하면 곡면의 파라미터 공간에서도 닫힌 곡선이 발생한다는 점에 착안하여 닫힌 교차곡선의 존재를 판단하고 시작점과 끝점을 하나로 이어주었다(Fig. 21참고). 그 과정은 다음과 같다.

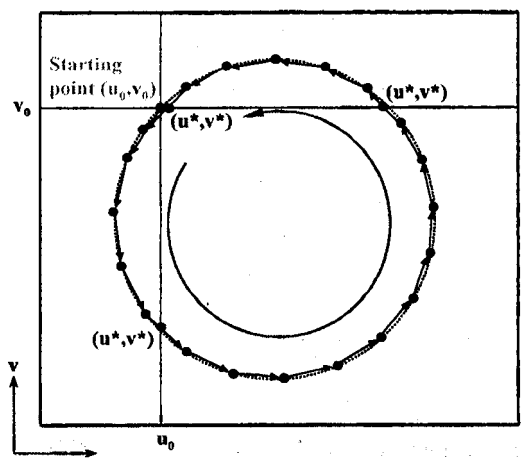


Fig. 22. Checking closed intersection during curve tracing.

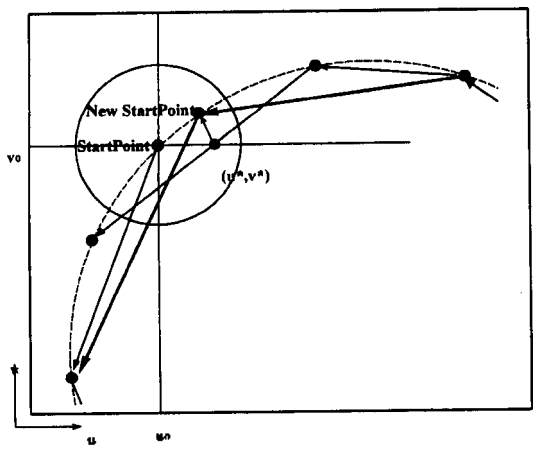


Fig. 23. Process for start/end point of closed intersection curve (Close-up of Fig. 22).

(1) 교차곡선 추적을 시작하면서 초기점(starting point)의 파라미터 값(u_0, v_0)을 저장한다.

(2) 교차곡선을 추적하면서, 파라미터 공간의 직선 $u=u_0$, $v=v_0$와 교차곡선을 추적하면서 구한 점의 파라미터 값을 이은 직선조각이 서로 교차하는지 검사한다(Fig. 22참고).

(3) (2)에서 만약 교차한다면, 정확히 교차하는 파라미터 값(u*,v*)을 구한다. Fig. 22를 보면 이러한 점이 3번 발생하게 된다. 하지만 교차한다고 해서 모두 닫힌 교차곡선이 발생하는 경우는 아니다.

(4) (3)에서 구한 파라미터 값(u*,v*)이 초기점의 파라미터 값 (uo,vo)와 주어진 tolerance circle내에 존재하는지를 검사한다. tolerance circle내에 존재한다면 비로소 닫힌 교차곡선이 발생한다고 판단하고 (u*,v*)값을 초기값으로 2.4.3의 방법으로 교차곡선상의 점을 계산한다. 이 점을 교차곡선의 시작점과 끝점으로 대체해서 교차곡선을 닫아준다(Fig. 23참고).

본 논문에서 tolerance circle의 반경은 1e-2로 결정하였다.

3. Examples

본 논문의 알고리즘을 이용한 surface/surface intersection library는 C/C++으로 작성되었으며 개발환경은 인텔 펜티엄II 333Mhz PC/ 64Mb RAM/ Visual C++ 6.0 이다.

교차곡선을 계산한 결과를 가시화 하는 프로그램은 제1회 CAD/CAM software 공모전에서 배포한 WinDraw version 1.0을 사용하였다.

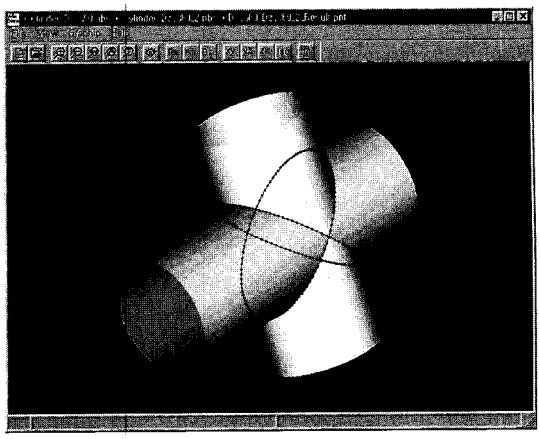


Fig. 24. Intersection between two cylinders (1).

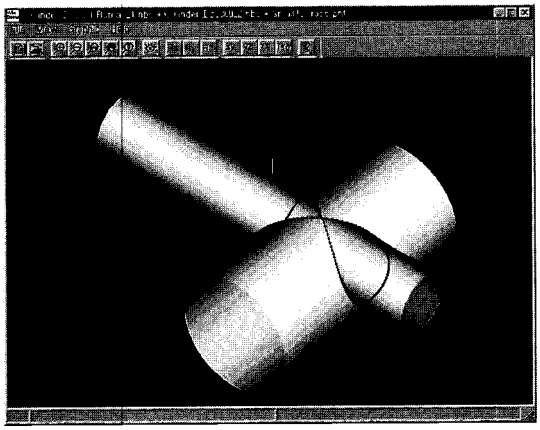


Fig. 25. Intersection between two cylinders (2).

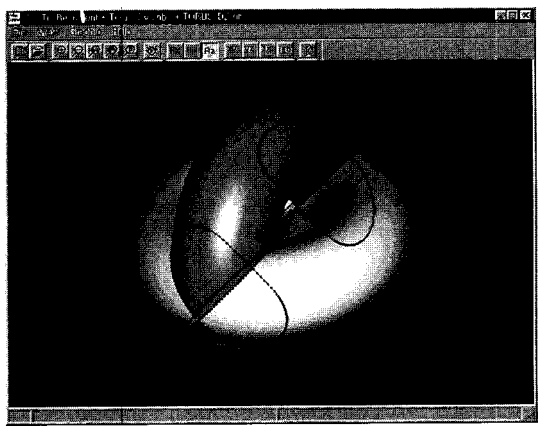


Fig. 26. Intersection between two toruses.

3.1 기본 예제

원기둥, torus, 평면과 같은 기본물체(primitive object) 끼리의 교차곡선 계산을 수행한 결과이다. Fig. 24는 반지름이 같은 두 개의 원기둥사이의 교차곡선계산의

Fig. 27. Intersection between two free-form surfaces (1).

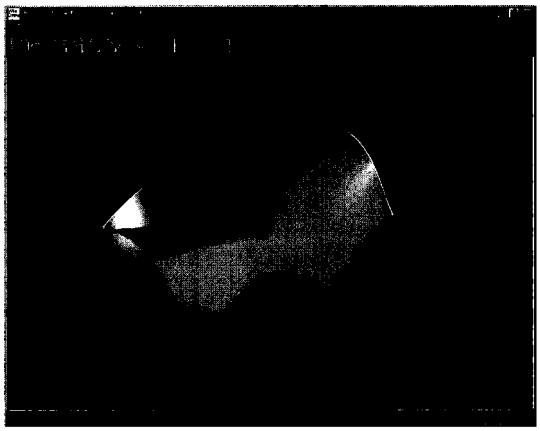


Fig. 28. Intersection between two free-form surfaces (2).

결과이고 Fig. 25는 반지름이 서로 다른 원기둥끼리의 교차곡선 계산의 예이다. Fig. 26은 torus끼리의 교차곡선 계산을 수행한 결과이다.

3.2 자유곡면간의 교차곡선 계산의 예

Fig. 27과 Fig. 28은 두 개의 자유곡면간의 교차곡선을 계산한 결과를 보여주고 있다.

3.3 복합곡면(composite surface)간의 교차곡선 계산의 예

Fig. 29는 3개의 단일 곡면으로 이루어진 복합곡면과 원기둥사이의 교차곡선을 계산한 결과를 보여주고 있다. 이 경우 단일 곡면간의 교선 계산을 통해서 2

[1] 미국 Spatial Tech-nology에서 개발된 Solid Modeling kernel로서 전세계적으로 사용하는 CAD/CAM system core

[2] 일본 Ricoh에서 개발, 상용화된 Solid Modeling CAD/CAM system

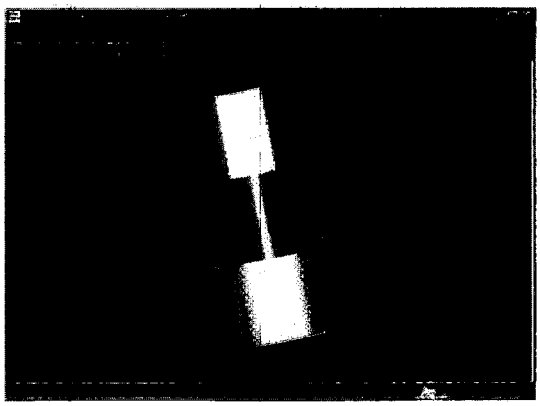


Fig. 29. Intersection between composite surface and cylinder.

개의 곡선이 구해지게 되는데, 정렬과정을 거쳐서 닫힌 교차곡선으로 만들어 줌을 볼 수 있다.

3.4 곡면경계에서 교차곡선이 발생하는 경우

본 논문의 알고리즘을 이용한 곡면경계에서 발생하는 교차곡선 계산결과를 ACIS[1]의 surface/surface intersection library와 DESIGNBASE[2]를 이용한 결과와 함께 비교하였다. 그 결과 DESIGNBASE는 올바른 교차곡선을 찾아내지 못하는 경우가 많았고, ACIS는 비교적 올바른 결과를 보여주었으나, 일부의 경우에는

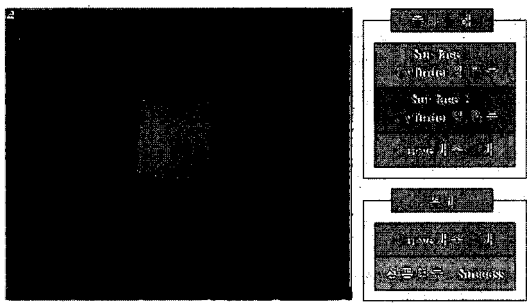


Fig. 30. Intersection result with improved tracing algorithm of this paper.

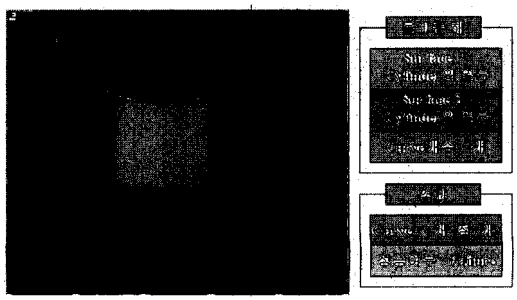


Fig. 31. Intersection result with DESIGNBASE.

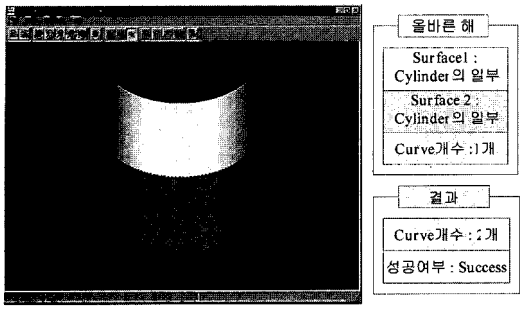


Fig. 32. Intersection result with ACIS.

하나의 교차곡선을 2개로 구해내는 오류가 있었다. 그러나 상용 library는 내부적으로 어떠한 알고리즘을 사용하였는지 겉으로 드러나는 바가 없기 때문에 직접적인 비교는 어렵다고 할 수 있다.

Fig. 30, 31, 32는 두 개의 원기둥의 일부를 나타내는 곡면이 경계에서 교차하는 예인데, 본 알고리즘

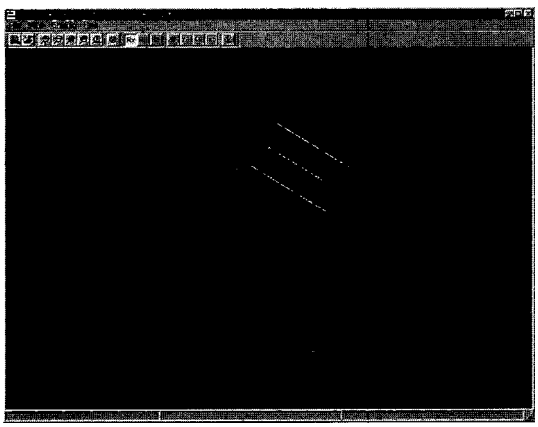


Fig. 33. Intersection between hull and cylinder (before shading).

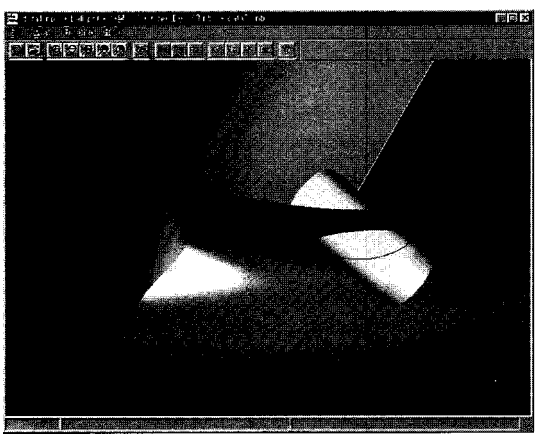


Fig. 34. Intersection between hull and cylinder (shading).

과 ACIS를 이용한 결과는 올바르게 나왔으나, DESIGNBASE는 점 1개만을 구해내는 올바르지 못한 결과를 주었다.

3.5 Hull surface intersection

Fig. 33,34는 선수부분의 닻을 내리고 올리는 통로인 bell mouth의 입구부분을 곡면간의 교차계산을 통하여 구해낸 것이다.

4. 결 론

본 연구에서는 기존의 추적 알고리즘의 추적 종결조건에 의해서 곡면경계에서의 교차곡선을 올바르게 구하지 못하는 문제점을 해결한 개선된 추적 알고리즘을 제안하였다. 개선된 추적알고리즘의 특징을 설명하면 다음과 같다.

1) 곡면(parametric surface)을 특정한 형태(예를 들면, Bezier surface 혹은 B-Spline surface)로 가정하지 않고, 곡면상의 점과 1차 미분값만 사용한다.

2) 단일 곡면을 다루는 기존의 방법들을 확장하여 복합곡면간의 교차계산을 수행할 수 있다.

3) 새로운 추적 종결조건은 도입하여 곡면경계에서의 교차곡선을 보다 안정적으로 계산할 수 있다.

5) 닫힌 교차곡선을 처리할 수 있는 알고리즘을 제안하였다.

본 논문에서 제안한 개선된 추적 알고리즘은 아직 접면과 trimmed 곡면을 처리할 수 없다. 따라서 이를 올바르게 계산할 수 있도록 보완이 필요하며, 보다 안정적인 계산을 위해서는 곡면의 크기에 따라서 CRT를 자동적으로 결정하는 알고리즘도 구현되어야 한다.

앞으로의 연구과제로 개발된 surface/surface intersection 알고리즘을 이용하여 계산된 교차곡선으로 trimmed 곡면을 구성하는 것과 선박의 구획배치 시스템의 개발, 장비배치 등의 간섭문제 해결, 그리고 선박계산 프로그램 등이 개발되어야 할 것이다.

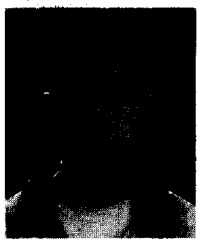
감사의 글

ACIS와 DESIGNBASE를 이용한 비교 테스트를 할 수 있도록 도와주신 서울대 이건우 교수님과 경북대 전차수 교수님, 이시열씨께 감사드립니다.

참고문헌

1. Houghton, E.G., Emnett, R.F., Factor, J.D. and Sabharwal, C.L., "Implementation of a divide-and conquer

method for intersection of parametric surfaces", Computer Aided Geometric Design, Vol. 2, pp. 173-183, 1985.
2. Timmer, H.G., "Analytic background for computational of surface intersections", Douglas Aircraft Company Technical Memorandum, C1-250-CAT-77-036, 1977.
3. Barnhill, R.E., Farin, G., Jordan, M. and Piper, B.R., "Surface/surface intersection", Computer Aided Geometric Design, Vol. 4, pp. 3-16, 1987.
4. Barnhill, R.E., and Kersey, S.N., "A marching method for parametric surface/surface intersection", Computer Aided Geometric Design, Vol. 7, pp. 257-280, 1990.
5. 주상윤, 이상헌, "곡면간의 교선에서 Step Size 결정 및 접점탐지 방법", 한국 CAD/CAM 학회 논문집 제 3권 제 2호, pp. 121-126, 1998.
6. Mortenson, M.E., Geometric Modeling, John Wiley & Sons, 1985.
7. Choi, B.K., Surface Modeling for CAD/CAM, Elsevier, 1991.
8. 임중현, 이규열, "베지에 곡선모델(드 카스텔죠 알고리듬)을 이용한 곡면 통합 모델링 기법", 대한조선학회논문집, 제34권, 제4호, 1997.
9. 이시열, 전차수, 곡면간의 교선에 대한 복합적 접근, 경상대학교 석사학위논문, 1998.
10. Faux, I.D. and Pratt, M.J., Computer Geometry for Design and Manufacture, Ellis Horwood, 1979.
11. Lasser, D. and Hoschek, J., Fundamentals of Computer Aided Geometric Design, A K Peters, 1993.
12. 이건우, 컴퓨터 그래픽과 CAD, 영지문화사, 1990.

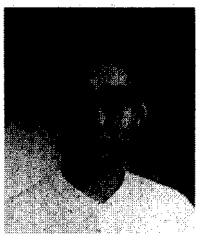


조 두 연

1997년 서울대학교 조선해양공학과 학사
1999년 서울대학교 조선해양공학과 석사
1999년~현재 서울대학교 조선해양공학과 박사과정
관심분야: Computer-Aided Geometric Design, Surface/surface Intersection

이 규 열

1971년 서울대학교 공과대학 조선공학과 학사
1975년 독일 하노버 공과대학 조선공학 석사
1975년~1983년 독일 하노비 공과대학 선박설계 및 이론연구소, 주정부 연구원
1983년~1994년 한국기계연구원 선박해양공학연구센터, 선박설계, 생산자동화 연구사업(CSDP)단장
1994년~현재 서울대학교 공과대학 조선해양공학과 부교수
관심분야: 최적설계, 형상모델링, CALS



임 중 현

1995년 서울대학교 조선해양공학과 학사
1997년 서울대학교 조선해양공학과 석사
1999년 서울대학교 조선해양공학과 박사수료
관심분야: Computer-Aided Geometric Design, Computer Graphics, Solid Modeling, Mesh Generation

부록 I. 곡면의 편평도를 계산하는 식

곡면이 편평한 정도를 계산하는 식은 상황과 기준에 따라서 다르게 결정할 수 있다. 본 논문에서는 간단한 계산으로 곡면의 편평도를 비교적 잘 나타낼 수 있는 Barnhill[4]의 식을 이용하였다.

일반적으로 곡면의 편평한 정도를 알아내기 위해서, 곡면내부의 모든 점들을 조사하는 것은 불가능하다. 따라서 한정된 개수의 정보를 가지고 곡면의 편평도를 잘 나타낼 수 있는 식을 찾아야 한다. 이러한 성질을 잘 만족시키는 것으로, 곡면 코너에서의 법선벡터를 생각할 수 있다. 만약 곡면 코너에서의 법선벡터들의 방향이 많이 다르다면, 그 곡면은 내부에서 굴곡이 심할 것이라고 생각할 수 있고, 반대로 법선벡터들의 방향이 거의 같다면, 그렇지 않은 경우보다

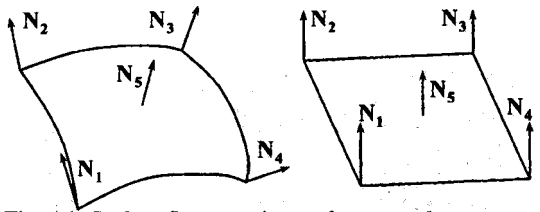


Fig. A1. Surface flatness using surface normal vectors on corner vertices.

곡면은 편평할 것이라고 추정할 수 있다(Fig. A1).

그러나, 곡면 코너에서의 법선벡터들의 방향이 거의 같을지라도 곡면내부에서 굴곡이 심한 경우도 있을 수 있으므로, 곡면 파라미터의 중점에서의 법선벡터도 고려하는 것이 좋다. 벡터의 내적을 이용하면, 곡면의 법선벡터들 사이의 방향을 계산할 수 있으므로, 곡면

의 편평도를 근사적으로 계산하는 식은 다음과 같이 나타낼 수 있다.

$$Flatness=1-N_i \cdot N_j=1-|N_i||N_j|\cos\theta$$
where N_i is a unitsurface normal vector

위에 식에 따르면, Flatness값이 클수록, 곡면코너에서의 법선벡터들의 사이각이 큼을 뜻하므로 곡면이 굴곡이 심할 것이라고 추정할 수 있다.

부록 II. 곡면 경계가 직선과 가까운 정도를 계산하는 식

곡면의 경계가 직선과 가까운 정도를 나타내는 값은, Bounding Box의 확장 길이를 계산할 때나, 곡면을 분할할 때 분할기준으로 사용된다. 곡면의 경계가 직선과 가까운 정도를 계산하는 식은 부록 I의 곡면의 편평도를 계산하는 식과 유사하게 결정된다[4].

Fig. A2처럼 곡면의 한 경계곡선이 있을 때, 경계곡선 양끝의 접선벡터 T_1, T_2사이의 각도가 크다면, 그 경계곡선은 굴곡이 심하다고 추정할 수 있다. 반대로 접선벡터 사이의 각도가 0도에 가깝다면, 그 경계곡선은 직선에 가깝다고 생각할 수 있다.

따라서, 곡면 경계가 직선과 가까운 정도를 근사적으로 계산하는 식은 다음과 같이 나타낼 수 있다.

$$\text{Edge Linearity}=1-T_i \cdot T_j$$
$$=1-|T_i||T_j|\cos\theta$$
$$=1-\cos\theta$$
where T_i is a unit tangent vector

위에 식에 따르면, Edge Linearity값이 클수록, 곡면경계 접선벡터들의 사이각이 큼을 뜻하므로 곡면경계곡선의 굴곡이 심할 것이라고 추정할 수 있다.

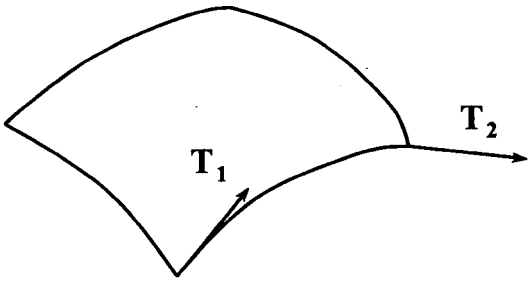


Fig. A2. Edge linearity using tangent vectors on corner vertices.

부록 III. Bounding box의 확장길이를 계산하는 식

Bounding box를 생성하는 데 있어서 가장 중요한 것은, bounding box가 곡면을 완전히 포함할 수 있도록 하는 것이다. 일단 bounding box가 곡면조각의 네 코너에서의 점을 포함하도록 생성되었다고 하자. Bounding box가 곡면조각을 완전히 포함하려면 곡면의 네 코너에서의 점 이외에도, 곡면의 경계곡선과 곡면내부를 포함하도록 확장되어야 한다.

하지만, 곡면상의 모든 점에 대해서 조사할 수는 없으므로, 보다 적은 정보로 만족할 만한 확장길이를 얻기 위한 식을 결정하여야 한다.

먼저, bounding box가 곡면의 경계곡선을 포함하도록 하는 확장길이를 계산해보자. 곡면경계에 관해서 가장 얻기 쉬운 정보는 양끝에서의 접선벡터이다(Fig. A3의 T_1, T_2).

이 두 접선벡터 사이의 각도가 2θ라면, 곡면의 경계곡선의 모양을 Fig. A4와 같이 좌우대칭으로 가정해서, 확장길이 H는 다음 식으로 계산한다[4].

$$L=\|V_2-V_1\|,\ \alpha=T_1 \cdot T_2=\cos 2\theta,$$
where T_1, T_2 are unit tangent vectors

$$H=\frac{L}{2}\tan\theta=\frac{L}{2}\sqrt{\frac{\sin^2\theta}{\cos^2\theta}}$$
$$=\frac{L}{2}\sqrt{\frac{1-\cos 2\theta}{1+\cos 2\theta}}=\frac{L}{2}\sqrt{\frac{1-\alpha}{1+\alpha}}$$

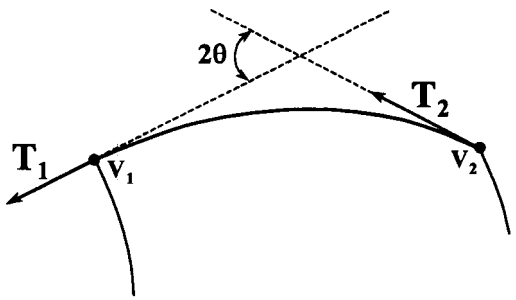


Fig. A3. Surface boundary curve with tangent vector T_1, T_2.

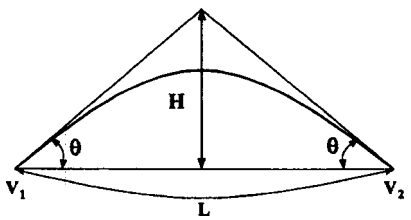


Fig. A4. Boundary curve assumed to be symmetric.

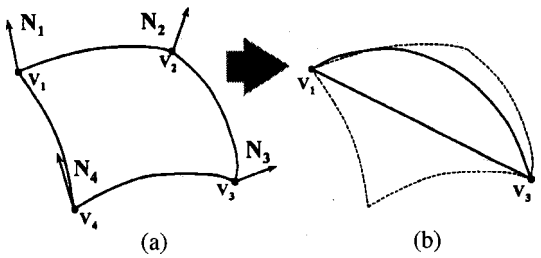


Fig. A5. The estimation of inner surface, using surface normal vectors.

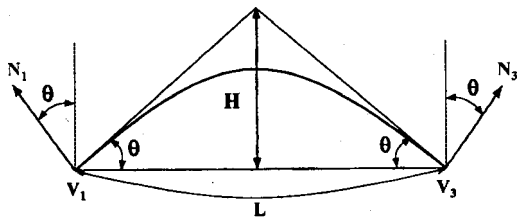


Fig. A6. The Close-up of Fig. A5.

그 다음으로, bounding box가 곡면 내부를 포함하도록 확장하는 식을 결정해보자. 부록 I에서 언급한 바와 같이 곡면의 네 코너에서의 법선벡터사이의 각도를 통해, 곡면 내부에서의 굴곡을 추정할 수 있다. Fig. A5(a)와 같이 곡면의 두 코너 V_1, V_3에서의 법선벡터를 N_1, N_3라고 하고 두 벡터사이의 각이 2θ라면, 곡면 내부의 모양을 Fig. A5(b), Fig. A6과 같이 가정할 수 있다.

그러면, 곡면경계에서와 같은 방법으로, bounding box가 곡면내부를 포함하기 위한 확장길이 H_{13}을 다음 식으로 계산할 수 있다[4].

$L=\|V_3-V_1\|$, $\alpha=N_1 \cdot N_3=\cos 2\theta$,

where N_1, N_3 *are unit tangent vectors*

$$H=\frac{L}{2}\tan\theta=\frac{L}{2}\sqrt{\frac{\sin^2\theta}{\cos^2\theta}}$$

$$=\frac{L}{2}\sqrt{\frac{1-\cos 2\theta}{1+\cos 2\theta}}=\frac{L}{2}\sqrt{\frac{1-\alpha}{1+\alpha}}$$

또, 벡터 N_2, N_4에 대해서도 확장길이 H_{24}를 계산할 수 있다. 이 중 큰 확장 길이를 택한다.