한국 CAD/CAM 학회 논문집
제 3 권 제 2 호 1998년 6월 pp. 121-126

학술논문

# 곡면 간의 교선에서 Step Size 결정 및 접점탐지 방법

주상윤*, 이상헌**

# Methods on Determination of Step Sizes and Detection of Tangential Points for SSI

Sang-yoon Ju* and Sang-heon Lee**

**ABSTRACT**

It is one of important issues to find intersection curves in representation of complex surfaces on a computer. Three typical methods, i.e. the tracing method, the subdivision method, and hybrid method, are often applied to find intersection curves between sculptured surfaces. In this paper two topics are dealt with for efficiency and robustness of the hybrid method. One topic is about how to determine step sizes variably during tracing, the other is about how to find tangential points between surfaces. Tracing by variable step size finds intersections rapidly and requires less memory size. Some illustrations show tangential points between surfaces.

***Key words*** : Intersection, Tracing, Hybrid, Step size, Tangential point

## 1. 서 론

컴퓨터를 이용하여 복잡한 곡면을 표현할 때 곡면간의 교선을 구하는 것은 가장 기본적인 작업이며 또한 중요한 문제 중에 하나이다. 곡면모델링에서 교선을 구하는 방법으로는 대수학적 방법, 래터스 탐색법, tracing방법, subdivision방법 등이 있으나 자유곡면들간의 교선을 구하는 방법으로는 tracing방법과 subdivision방법이 자주 이용되고 있다[1]. Barnhill[2], Timmer[3] 등에 의하여 제안된 tracing방법은 일단 교선 상의 출발점이 탐지된 경우에는 수행속도가 매우 빠르지만, 출발점탐지를 위한 mesh 생성단계에서 mesh의 간격이 작은 경우에는 출발점탐지에 많은 시간이 소요되고, mesh의 간격이 넓은 경우에는 접점이나 미소한 폐곡선 같은 일부 교선들을 찾지 못하는 문제점이 있다. 그에 반하여 subdivision방법은 분할된 곡면을 둘러싼 최대최소상자를 이용하여 두 곡면의 교차여부를 판별하게 되며, 이 때 교차하는 최대최소상자들은 존재하는 모든 교선들을 포함하게 된다[4-6]. 그러나 subdivision 알고리즘의 수행속도는 상대적으로 느릴 뿐만 아니라 얻어진 교선이 매끄럽지 못하다는 단점이 있다[7]. 그 밖에 자유곡면간의 교선을 구하기 위하여 tracing방법과 subdivision방법을 절충한 hybrid SSI(surface/surface intersection)방법이 연구된 바 있다[7-9]. hybrid 방법은 subdivision방법을 이용하여 교선들을 탐지하고 발견된 초기점으로부터 tracing기법을 이용하여 교선을 구하게 되는데, 접점들을 포함한 교선들을 안정적으로 찾을 수 있을 뿐만 아니라 양질의 교선을 얻을 수 있는 장점이 있다.

본 연구에서는 hybrid 알고리즘을 기초로 하여 교선과 접점을 효과적으로 찾을 수 있는 2가지 내용을 다루고자 한다. 그 가운데 하나는 교선을 추적할 때 step size를 교선의 곡률에 따라 가변적으로 결정하는 방법에 관한 것이고 나머지 다른 하나는 곡면간에 존재하는 접점을 효과적으로 찾는 방법에 관한 것이다. step size를 결정하는 첫번째 내용은 SSI방법들 가운데 tracing방법과 관련된 것이고 곡면간의 접점을 찾는 두 번째 내용은 tracing방법이 접점을 제대로 탐지하지 못하는 한계로 인하여 subdivision방법을 필요로 한다. 따라서 본 논문에서는 이 두가지

*종신회원, 울산대학교 산업공학과
**학생회원, 울산대학교 산업공학과 대학원

내용을 한꺼번에 수용할 수 있는 hybrid방법을 이용하여 연구내용을 전개하고자 한다. 본 논문에서는 2장에서 hybrid 알고리즘에 대하여 소개하고, 3장에서 step size를 결정하는 방법에 관하여 그리고 4장에서는 두 곡면 간에 존재하는 접점을 찾는 방법을 소개하고자 한다.

## 2. hybrid SSI알고리즘

hybrid 알고리즘은 입력된 두 곡면 f(s, t)와 g(u, v)에 대하여 교선 상에 존재하는 일련의 점들을 다음의 3단계를 거쳐 구한다.

• subdivision단계

곡면들을 분할하여 교차 가능한 subpatch쌍들을 저장한다.

• 탐지단계

저장된 subpatch 쌍으로부터 교선상의 한 점을 찾아낸다. 여기서 발견된 점은 교선의 추적을 위한 시작점이 된다.

• 추적단계

시작점으로부터 출발하여 교선상에 존재하는 점들을 연속적으로 찾는다.

### 2.1 subdivision 단계

입력된 두 곡면 f(s, t), g(u, v)를 Bezier곡면이라고 하자. Fig. 1과 같이 Bezier곡면의 조정점을 이용하여 각각의 곡면을 둘러싼 최소최대상자를 결정하여 두 최소최대상자의 교차여부를 조사한다.

두 최소최대상자가 교차하면 de Casteljau 알고리즘을 이용하여 Bezier곡면들을 각각 4개의 subpatch로 분할한다. 분할된 subpatch들의 최소최대상자들에 대하여 다시 교차여부를 조사하여 교차하는 subpatch들만 다시 분할한다. 최소최대상자의 교차테스트와 subpatch의 분할은 종료조건에 도달할 때까지 반복되며 최소최대상자가 교차하는 subpatch쌍들은 교차 가능한 subpatch쌍으로서 저장된다. 입력된 곡면이 NURB곡면인 경우에는 우선 Bezier곡면으로 변환시킨 다음 동일한 과정을 수행한다.

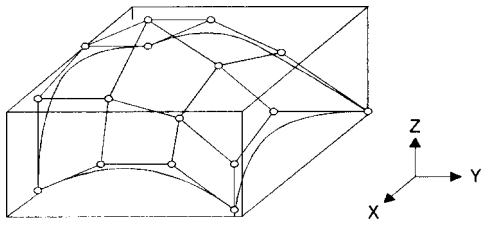


Fig. 1. minmax box.

### 2.2 탐지단계

교차 가능한 subpatch쌍의 중점을 $f(s_m, t_m)$, $g(u_m, v_m)$이라고 하자. 두 점 $f(s^*,t^*)=f(s_m, t_m)$, $g(u^*, v^*)=g(u_m, v_m)$으로부터 다음과 같은 과정에 의하여 교선상의 한 점을 구할 수 있다.

2.2.1 탐지과정(detection procedure)

과정 1. 두점 f(s*, t*), g(u*, v*)으로부터 Fig. 2와 같이 추정점 G*를 정한다.

과정 2: 부록의 Jacobian inversion에 의하여 추정점 G*에 가까이 위치한 곡면 상의 두 점 f(s*, t*)와 g(u*, v*)을 찾는다.

과정 3: 두 점 사이의 거리 |f(s*, t*)-g(u*, v*)|가 SPT(same point tolerance)보다 크면 과정 1로 가고 그렇지 않으면 종료한다.

발견된 교점은 교선 추적을 위한 시작점으로 사용되기 위하여 두 곡면에 대한 domain 값 (s*, t*), (u*, v*)를 저장한다.

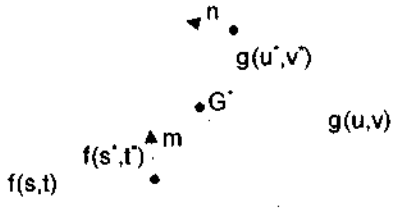


Fig. 2. Determination of a guess point G* from two points on surfaces.

### 2.3 추적단계

두 곡면 f(s, t)와 g(u, v)에 대하여 교선 상의 한 점이 얻어지면 교선 상의 이웃한 점들을 연속적으로 구할 수 있다. 이웃한 교점을 결정하기 위한 추정점 G는 Fig. 3과 같이 현재의 교점 $f(s_0, t_0)=g(u_0, v_0)$에서 교선에 대한 접선방향으로 결정된다. 접선의 단위벡터 T는 현재의 교점에서 두 곡면의 법선벡터에 동시에 수직한 방향으로 구해지므로 추정점 G는 식(1)과 같이 얻어진다.

$$G = f(s_0, t_0) \pm \delta T \quad (1)$$

여기서

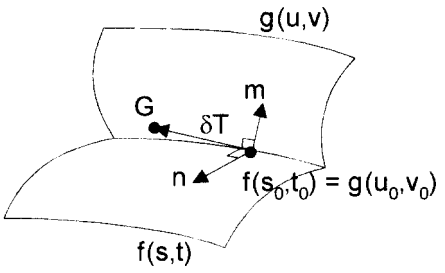


**Fig. 3.** A guess point G for finding an adjacent intersection point.

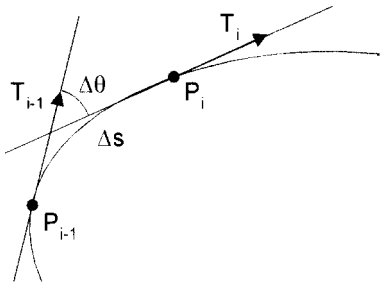


**Fig. 4.** Determination of a curvature.

δ: step size

**T**=(**m**×**n**)/|**m**×**n**|: 단위접선벡터

**m**, **n**: **f**($s_0$, $t_0$), **g**($u_0$, $v_0$)에서의 단위법선벡터

교선 상의 한점으로부터 이웃한 교점은 다음의 과정에 의하여 얻어진다.

2.3.1 추적과정(trace procedure)

과정 1: 현 교점에서 식 (1)에 의하여 추정점 **G**를 정한다.

과정 2: 부록의 Jacobian inversion에 의하여 추정점 **G**에 가까이 위치한 곡면상의 두점 **f**(s*, t*)와 **g**(u*, v*)를 찾는다.

과정 3: Fig. 2와 같이 추정점 **G***를 정한다.

과정 4: 추정점 G*에 대하여 Jacobian inversion 에 의하여 곡면 상의 두 점 **f**(s*, t*)와 **g**(u*, v*)을 얻는다.

과정 5: 두 점 사이의 거리 |**f**(s*, t*)-**g**(u*, v*)|가 SPT(same point tolerance)보다 큰 경우 과정 3으로 가고 그렇지 않으면 종료한다.

이상의 과정을 반복하면 교선 상에 존재하는 일련의 점들을 얻게 된다. 교선의 추적은 다음의 경우에 중단된다.

- 교선이 자신의 시작점을 만날 때
- 교선이 단위곡면 밖으로 벗어났을 때

교선이 곡면 밖으로 벗어나면 시작점으로 돌아와 반대방향으로 교선을 추적한다.

## 3. step size 의 결정

곡면간의 교선을 찾는데 있어서 step size는 수행속도와 교선의 정밀도에 큰 영향을 미친다[7]. 만약 step size가 작은 값으로 설정된 경우, 정밀한 교선을 구할 수 있으나 교선을 찾는 시간은 많이 소요된다. 반면에 step size가 큰 값인 경우, 오차가 증가하여 교선의 정밀도에 문제가 된다. 특히 step size가 지나치게 큰 경우에는 이웃한 교점을 찾는데 실패하거나 혹은 추적 도중에 다른 교선을 찾아가기도 한다. 따라서 교선을 추적해 나갈 때 step size를 적절하게 정하는 것은 매우 중요하다.

Barnhill[2], 이치근[7]은 고정된 step size를 사용하여 교선을 추적하고 있다. 이런 경우 지정된 허용오차를 만족하는 step size는 교선의 곡률이 큰 곳에서 결정되므로 교선의 곡률이 작은 곳에서는 불필요하게 많은 교점들을 샘플링하게 된다. 따라서 본 연구에서는 교선의 곡률에 근거하여 step size를 가변적으로 결정하는 방법을 소개하고자 한다.

곡선의 곡률은 곡선의 길이에 대한 접선각도의 순간변화율로 정의된다[10]. 따라서 교선상에 이웃한 두 점이 가까이 존재하는 경우 Fig. 4에 도시된 바와 같이 두 점 $\mathbf{P}_{i-1}$, $\mathbf{P}_i$와 접선 $\mathbf{T}_{i-1}$, $\mathbf{T}_i$를 이용하여 곡률 k는 식(2)와 같이 추정할 수 있다.

$$k=\Delta\theta/\Delta s \tag{2}$$

여기서

$\Delta s$: 이웃한 두 교점 $\mathbf{P}_{i-1}$, $\mathbf{P}_i$간의 교선의 길이

$\Delta\theta$: 두 접선 $\mathbf{T}_{i-1}$, $\mathbf{T}_i$의 사잇각

식 (2)에서 교선의 길이 $\Delta s$를 두 교점 간의 현의 길이 $|P_i-P_{i-1}|$로 대신하면 보다 빠르게 곡률을 추정할 수 있다.

구해진 곡률 k와 허용오차 ε로부터 step size δ를 계산해 보자. Fig. 5는 현재의 점 $\mathbf{P}_i$에서 접선방향으로 step size δ만큼 떨어져 있는 추정점 **G**를 구하는 방법을 나타내고 있다. 추정점 **G**에서 원호에 가장 가까이 위치한 점을 **Q**라고 할 때 두점 $\mathbf{P}_i$와 **Q**를 연결한 현과 원호사이의 오차가 허용오차 ε의 크기를 갖도록 추정점 **G**를 정할 수 있다. 바꾸어 말하여 추정점 **G**는 원호의 중심점 **C**와 점 **Q**를 연결한 직선과 점 $\mathbf{P}_i$에서의 접선이 만나는 점으로 결정된다.

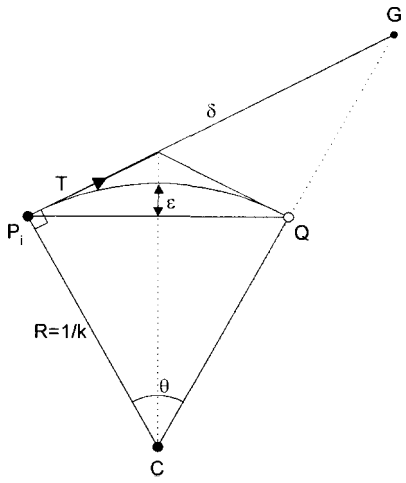


**Fig. 5.** Determination of a step size δ.

Fig. 5로부터 다음의 두 관계식을 얻을 수 있다.

$$\delta = R \tan\theta \quad (3)$$

$$\varepsilon = R(1-\cos(\theta/2)) \quad (4)$$

여기서

R=1/k: 곡률반지름

식 (3)과 식 (4)에서 θ를 소거하면 step size δ와 곡률 k의 관계식을 식 (5)와 같이 얻는다.

$$\delta = \frac{2(1-k\varepsilon)\sqrt{2k\varepsilon - k^2\varepsilon^2}}{k(1-4k\varepsilon+2k^2\varepsilon^2)} \quad (5)$$

일반적으로 곡률이 작은 값을 취하는 경우 step size의 크기는 커지게 된다. 그러나 식 (5)는 step size가 작은 크기일 때 교선을 원호에 근사시켜 얻은 것이므로 step size가 큰 값을 취하는 경우에는 적절하지 못하다. 따라서 step size의 최대값 $\delta=\delta_{max}$을 지정하는 것이 필요하다. 즉 식 (5)로부터 얻어진 step size의 크기가 $\delta > \delta_{max}$인 경우 $\delta=\delta_{max}$로 정한다.

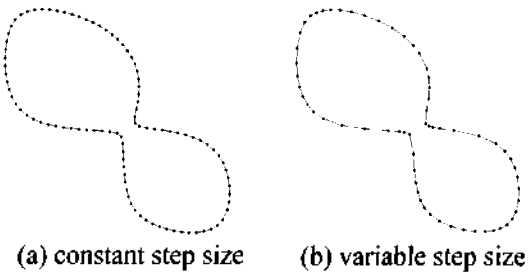


**Fig. 6.** Tracing with a constant step size and variable step sizes.

Fig. 6은 두 곡면 간의 교선을 구하는 과정에 있어서 step size를 고정시킨 경우와 step size가 변하는 경우에 대하여 각각 얻어진 교점들을 도시하고 있다. 두 경우에 있어서 각각 97개와 64개의 교점들이 얻어졌으며, 166MHz, 64M RAM을 가진 펜티엄급 컴퓨터에서 각각 0.16초와 0.11초가 소요되었다. 따라서 가변의 step size를 사용하는 것이 고정된 step size를 사용하는 것보다 컴퓨터 메모리나 소요시간 면에서 유리함을 알 수 있다.

## 4. 접점의 결정

곡면들간에는 교선들만 아니라 접점들도 자주 발생한다. 여기서 접점은 수학적으로 정의된 면적이 없는 한점이 아니라 두 곡면이 허용오차 이내로 매우 가까이 접근했거나 혹은 미세하게 교차한 상태로 가정한다. Fig. 7과 같이 두 곡면에 접점이 존재할 경우 교선의 출발점을 탐지하기 위한 두 곡면 상의 두 추적점 **P**와 **Q**는 교점 **A**를 향하여 수렴하게 된다. 두점 **P**와 **Q**의 거리 d가 허용오차(SPT)보다 작아지는 경우, 점 **P**(혹은 **Q**)를 교선 상의 한점으로 인식하고 이를 출발점으로 시작하여 접점 **A**의 주위를 추적하게 된다. 이같은 추적의 결과 Fig. 8과 같이 접점 **A**를 둘러싼 loop 모양의 점군들을 얻게 된다.

일단 loop이 존재하면 두곡면은 교차하거나 혹은 충분히 근접해 있는 상태이므로 접점이나 교선이 존재하게 된다. 이때 loop의 크기가 매우 작으면 접점으로 가정하고 그렇지 않은 경우에는 얻어진 loop으로부터 접점 혹은 교선의 여부를 판단해야 한다. 접점 **A**를 둘러싼 loop모양의 교점들을 $\mathbf{P}_0$, $\mathbf{P}_1$, ..., $\mathbf{P}_{n-1}$

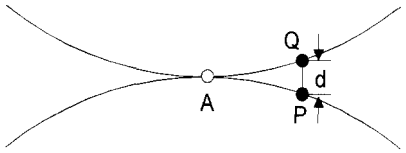


**Fig. 7.** A trouble caused by the same point tolerance.

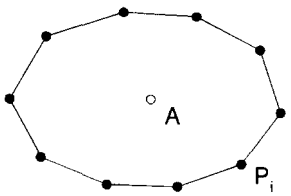


**Fig. 8.** Intersected points around a tangential point.

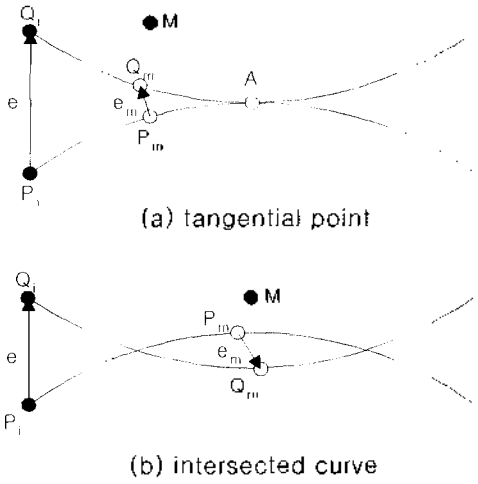


**Fig. 9.** Discrimination between a tangential point and an intersected curve.

혹은 $\mathbf{Q}_0$, $\mathbf{Q}_1$, .... $\mathbf{Q}_{n-1}$ 이라고 하고 이들의 중점 **M**을 다음과 같이 정의하자.

$$\mathbf{M}=\Sigma\mathbf{P}_i/n \text{ 혹은 } \Sigma\mathbf{Q}_i/n \quad (6)$$

중점 **M**에 대한 두 곡면 상의 파라메터($s_m$, $t_m$), ($u_m$, $v_m$)을 Jacobian inversion으로 구하여 대응하는 곡면 상의 점을 $\mathbf{P}_m=\mathbf{f}(s_m, t_m)$, $\mathbf{Q}_m=\mathbf{g}(u_m, v_m)$이라고 하자. Fig. 9와 같이 loop 상에서 정의된 벡터 $\mathbf{e}=\mathbf{Q}_i-\mathbf{P}_i$와 중점에서 정의된 벡터 $\mathbf{e}_m=\mathbf{Q}_m-\mathbf{P}_m$로부터 접점의 존재 여부를 다음과 같이 판정한다.

2.3.2 판정과정(decision procedure)

과정 1: 식 (6)으로부터 교점들의 중점 **M**을 구한다.

과정 2: Jacobian Inversion에 의하여 중점 **M**에 대한 곡면상의 두점 $\mathbf{P}_m=\mathbf{f}(s_m, t_m)$와 $\mathbf{Q}_m=\mathbf{g}(u_m, v_m)$를 구한다.

과정 3: loop상의 임의의 점 $\mathbf{P}_i=\mathbf{f}(s_i, t_i)$, $\mathbf{Q}_i=\mathbf{g}(u_i, v_i)$에 대하여 방향벡터 $\mathbf{e}=\mathbf{Q}_i-\mathbf{P}_i$를 정의하고 중점에 대한 방향벡터 $\mathbf{e}_m=\mathbf{Q}_m-\mathbf{P}_m$을 구한다.

과정 4: 만약 $\mathbf{e}.\mathbf{e}_m$의 값이 양의 값이면 접점이 존재하고 그렇지 않으면 교선이 존재한다고 판정한다.

이상의 과정을 통하여 일단 두 곡면 사이에 접점이 존재한다고 판단되면 두 곡면 상의 점 $\mathbf{Q}_m$과 $\mathbf{P}_m$의 중점을 접점으로 정한다.

Fig. 10은 지금까지 기술한 절차를 적용하여 두 곡면간에 존재하는 접점들을 찾아내 도시하고 있다. Fig. 10(a)는 실린더와 cone이 한 점에서 접하는 경우이며, Fig. 10(b)는 복잡한 자유곡면과 평면이 다수의 접점에서 만나는 경우를 보여주고 있다.

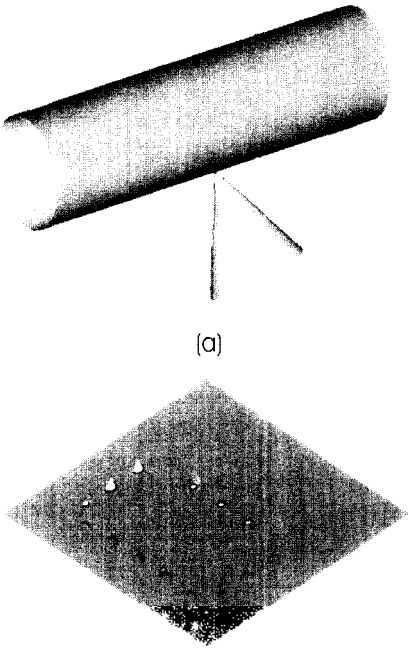


**Fig. 10.** Tangential points between surfaces.

## 5. 결 론

본 연구에서는 곡면간의 교선을 찾기 위한 hybrid 알고리즘에서 가변적으로 step size를 결정하는 방법과 접점을 찾는 방법을 소개하였다. tracing과정 중에 교선의 곡률에 따라 step size를 가변적으로 변화시킨 결과 저장할 교점의 수가 줄었으므로 적은 메모리를 사용하였을 뿐만 아니라 교선을 찾는데 소요되는 시간도 단축되었다. 가변적으로 step size를 결정하는 본 연구방법은 단지 hybrid알고리즘에서만 이용될 수 있는 것이 아니라 tracing방법으로 교선을 찾는 모든 SSI알고리즘에 대하여 동일하게 적용될 수 있다. 또한 본 연구에서는 허용오차로 인하여 교선으로 혼동되기 쉬운 곡면간의 접점을 찾는 방법을 제시하였다. 본 연구에서는 접점을 두 곡면이 허용범위 내에 근접하거나 미세하게 교차하는 것으로 가정하였으나 수학적으로 정확한 접점이 존재하는 경우 이를 식별해 내는 연구가 필요하다.

## 참고문헌

1. Pratt, M.J. and Geisow, A.D., *The Mathematics of*

*the Surface*, Clarendon Press, Oxford, pp. 117-142, 1986.
2. Barnhill, R.E., Farin, F., Jordan, M. and Piper, B.R., "Surface/Surface Intersection", *Computer Aided Geometric Design*, Vol. 4, No. 4, pp. 3-16, Dec., 1987.
3. Timmer, H.G., "Analytical Background for Computation of Surface Intersection", *Douglas Aircraft Company Technical Memorandum*, C1-250-CAT-77-036, Apr., 1977.
4. Peng, Q.P., "An Algorithm for Finding the Intersection Lines between two B-Spline Surfaces", *Computer-Aided Design*, Vol. 16, No. 4, pp. 191-196, July, 1984.
5. Lasser, D., "Intersection of Parametric Surfaces in the Bernstein-Bezier Representation", *Computer-Aided Design*, Vol. 18, No. 4, pp. 186-192, May, 1986.
6. Haughton, E.G., Emnett, R.F., Factor, J.D., and Sabharwal, C.L., "Implementation of an Divide and Conquer Method for Intersection of Parametric Surfaces", *Computer Aided Geometric Design*, Vol. 2, pp. 173-183, 1985.
7. 이치근, 매개변수곡면 간의 교선을 구하는 알고리즘의 각종 여유공차의 효과에 관한 연구, KAIST 석사학위 논문, 1992.
8. Aziz, N.M. and Bata, R., "Bezier Surface/Surface Intersection", *IEEE CG&A*, pp. 50-58, Jan, 1990.
9. Choi, B.K., *Surface Modeling for CAD/CAM*, Elsevier, New York, pp. 297-312, 1990.
10. Faux, I.D. and Pratt, M.J., *Computational Geometric for Design and Manufacture*, Ellis Horwood, Chichester, pp. 40-42, 1979.
11. Choi, B.K. and Ju, S.Y., W., "Constant-radius Blending in Surface Modeling", *Computer- Aided Design*, Vol. 21, No. 4, pp. 213-220, May, 1989.

## 부록. 곡면에 대한 Jacobian inversion

Fig. A1과 같이 추정점 **G**에 대하여 곡면 **r**(u, v)에 가장 가까운 점 $\mathbf{r}(u_g, v_g)$을 찾고자 한다. 곡면 상의 한 점을 $\mathbf{r}(u_i, v_i)$라 할 때 이는 벡터 $\mathbf{e}=\mathbf{G}-\mathbf{r}(u_g, v_g)$의 크기가 최소값을 갖도록 $\Delta u=u_g-u_i$, $\Delta v=v_g-v_i$를 구하는 문제와 동일하다. 벡터 $e$는 식 (A1)과 같이 바꿀 수 있다.

$$\mathbf{e}=\mathbf{d}-[\mathbf{r}(u_g, v_g)-\mathbf{r}(u_i, v_i)] \quad \text{(A1)}$$

여기서 $\mathbf{d}=\mathbf{G}-\mathbf{r}(u_i, v_i)$

$\mathbf{r}(u_g, v_g)$를 $\mathbf{r}(u_i, v_i)$에 관하여 Taylor 전개한 후 2차 이상의 미분항을 무시하면 식 (A2)와 같다.

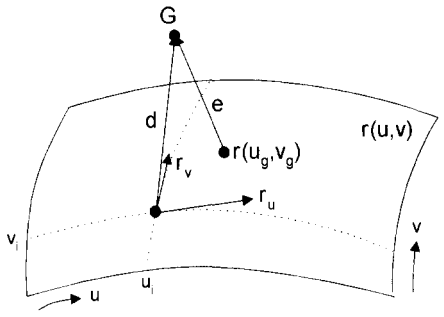


**Fig. A1.** A point $\mathbf{r}(u_g, v_g)$ on a surface nearest to **G**.

$$\mathbf{r}(u_g, v_g) \cong \mathbf{r}(u_i, v_i) + \Delta u\mathbf{r}_u + \Delta v\mathbf{r}_v \quad \text{(A2)}$$

식 (A2)를 식 (A1)에 대입하면 식 (A3)을 얻는다.

$$\mathbf{e}=\mathbf{d}-\mathbf{D}u\mathbf{r}_u-\Delta v\mathbf{r}_v \quad \text{(A3)}$$

F=**e**,**e**라고 정의할 때 함수 F를 최소로 하는 $\Delta u$와 $\Delta v$는 $\partial F/\partial(\Delta u)=0$와 $\partial F/\partial(\Delta v)=0$의 연립방정식으로부터 식 (A4)와 같이 얻을 수 있다[11].

$$\Delta u=\frac{(\mathbf{r}_u\times\mathbf{r}_v)\cdot(\mathbf{d}\times\mathbf{r}_v)}{|\mathbf{r}_u\times\mathbf{r}_v|^2};\quad \Delta v=\frac{(\mathbf{r}_u\times\mathbf{r}_v)\cdot(\mathbf{r}_u\times\mathbf{d})}{|\mathbf{r}_u\times\mathbf{r}_v|^2} \quad \text{(A4)}$$

여기서 $\mathbf{d}=\mathbf{G}-\mathbf{r}(u_i, v_i)$

**주 상 윤**

1977년 서울대학교 산업공학과 학사
1979년 한국과학원 산업공학과 석사
1989년 한국과학원 산업공학과 박사
1979년 ~ 현재 울산대학교 산업공학과 교수
관심분야 : 곡면모델링, CAD/CAM, NC 가공, 제조시스템자동화

**이 상 헌**

1997년 울산대학교 산업공학과 학사
1997년 ~ 현재 울산대학교 산업공학과 석사과정
관심분야 : 곡면모델링, CAM시스템개발, 데이터통신