한국 CAD/CAM 학회 논문집
제 3 권 제 2 호 1998년 6월 pp. 113-120

학술논문

Bézier 클리핑을 이용한NURBS 곡선간의 교점 계산

민병녕*, 김재정**

Calculation of NURBS Curve Intersections using Bézier Clipping

Byungnyung Min* and Jay Jung Kim**

ABSTRACT

Calculation of intersection points by two curves is fundamental to computer aided geometric design. Bézier clipping is one of the well-known curve intersection algorithms. However, this algorithm is only applicable to Bézier curve representation. Therefore, the NURBS curves that can represent free form curves and conics must be decomposed into constituent Bézier curves to find the intersections using Bézier clipping. And the respective pairs of decomposed Bézier curves are considered to find the intersection points so that the computational overhead increases very sharply.

In this study, extended Bézier clipping which uses the linear precision of B-spline curve and Grevill's abscissa can find the intersection points of two NURBS curves without initial decomposition. Especially the extended algorithm is more efficient than Bézier clipping when the number of intersection points is small and the curves are composed of many Bézier curve segments.

Key words : Computer aided geometric design, Bézier curve, NURBS, Linear precision, Grevill's abscissa

1. 서 론

CAD 모델러에서 곡선간의 교점 계산은 곡선과 곡면 또는 두 곡면간의 교점이나 교선을 구하는 알고리즘의 기초가 된다. 또한 일반적인 모델링 방법인 불리언 연산에서도 주요한 역할을 수행하는 핵심적인 기능 중 하나이다.

교점 계산 알고리즘은 수치적으로 정확하고, 모든 해를 구할 수 있어야 하며, 계산 속도가 빨라야 하고, 사용자의 도움 없이 문제를 해결할 수 있어야 한다. 그러나 하나의 알고리즘이 이를 모두 만족시키기는 어렵고 곡선의 표현 형태에 따라 적절한 알고리즘을 적용해야 한다[1].

NURBS 곡선은 자유 곡선뿐만 아니라 원추 곡선까지 한 방정식의 형태로 나타낼 수 있으므로 수학적으로 여러 가지 장점을 갖고 있다. 예를 들어 곡선과 곡선의 교차점을 계산하는 프로그램을 개발하는 경우, 원호와 원호, 원호와 자유 곡선, 원호와 포물선 등 모든 가능한 경우에 대해서 교점 계산 프로그램을 만드는 대신, 이들 원추 곡선을 NURBS 곡선 방정식으로 표현한다면 NURBS 곡선간의 교점을 계산하는 프로그램 하나로 위의 모든 경우를 해결할 수 있다[2]. 이와 같은 수학적인 여러 가지 장점에 힘입어 최근의 CAD 모델러에서 NURBS 곡선을 많이 채택하고 있다. 따라서 NURBS 곡선간의 교점 계산 알고리즘의 개발이 필요하다고 하겠다.

곡선간의 교점 계산 알고리즘 중 하나인 Bézier 클리핑은 4차 까지는 Sederberg와 Anderson[3]의 암시적 형태로 바꾸는 방법보다 느리지만 5차 이상에서는 더 빠르다[4,5].

Bézier 클리핑은 렌더링 기법 중 하나인 레이 트레이싱(ray tracing)에서 패치(patch)와 레이(ray)의 교점을 계산하기 위해서도 사용된다[6].

그런데 Bézier 클리핑을 이용하여 NURBS 곡선간의 교점을 구할 경우, 먼저 NURBS 곡선을 여러 개

*기아자동차
**한양대학교 기계설계학과

의 Bézier 곡선으로 분할한 후 각각의 Bézier 곡선쌍에 대해서 교점을 구해야 하므로 계산량이 많아지는 단점이 있다.

본 연구에서는 NURBS 곡선간의 교점을 효율적으로 계산할 수 있도록 B-spline 곡선의 linear precision 과 Grevill의 횡좌표를 응용하여 Bézier 클리핑을 확장하였다.

2. Bézier 클리핑의 단점

Bézier 클리핑은 Bézier 곡선의 방정식에만 적용할 수 있으므로 NURBS 곡선간의 교점을 구하기 위해서는 먼저 NURBS 곡선을 여러 개의 Bézier 곡선으로 분할해야 한다. p차인 NURBS 곡선의 매듭 벡터에서 각 매듭값을 $p+1$개로 중복시키면 Bézier 곡선으로 분할할 수 있다. 예를 들어 다음과 같은 매듭 벡터

{0,0,0,0,1/4,2/4,3/4,1,1,1,1}

에 대해서 정의된 3차 NURBS 곡선을 매듭값 삽입[7]으로

{0,0,0,0,1/4,1/4,1/4,1/4}
{1/4,1/4,1/4,1/4,2/4,2/4,2/4}
{2/4,2/4,2/4,2/4,3/4,3/4,3/4,3/4}
{3/4,3/4,3/4,3/4,1,1,1,1}

과 같이 네 개의 매듭 벡터를 갖는 3차 Bézier 곡선들로 분할할 수 있다. 분할된 각각의 Bézier 곡선쌍에 Bézier 클리핑을 적용하여 교점을 구해야하므로 초기에 많은 계산을 하게 되어 비효율적이다. 만약 Fig. 1과 같이 각각 7개와 8개의 Bézier 곡선으로 구성된 두 NURBS 곡선의 교점을 구할 경우 56쌍(7×8)의 곡선에 대해서 교점을 구해야 한다.

본 연구에서는 NURBS 곡선을 Bézier 곡선으로 분할하지 않고 NURBS 곡선의 교점을 직접 계산할 수 있도록 Bézier 클리핑을 확장하였다.

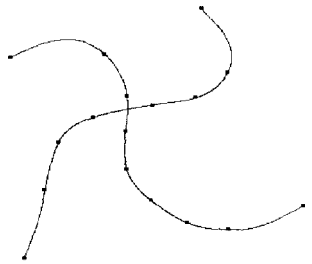


Fig. 1. Decomposition of NURBS curves into Bézier curves.

3. Bézier 클리핑을 NURBS 곡선으로 확장

3.1 클리핑 라인 계산

NURBS 곡선 중에서도 비주기 NURBS 곡선은 조정점의 양 끝점을 지나기 때문에 Bézier 클리핑을 확장하기 편리하다. 따라서 이 곡선에 대해서 다루기로 한다.

각각 p차와 q차인 두 개의 비주기 NURBS 곡선

$$C_1(t)=\frac{\sum_{i=0}^{l} N_{i,p}(t)w_i P_i}{\sum_{i=0}^{l} N_{i,p}(t)w_i} \quad t_{\min}\le t\le t_{\max} \tag{1}$$

$$C_2(u)=\frac{\sum_{j=0}^{n} N_{j,q}(u)w_j Q_j}{\sum_{j=0}^{n} N_{j,q}(u)w_j} \quad u_{\min}\le u\le u_{\max} \tag{2}$$

은 각각 매듭 벡터

$$\{\underbrace{t_{\min},\cdots,t_{\min}}_{p+1},\cdots,\underbrace{t_{\max},\cdots,t_{\max}}_{p+1}\}$$

와

$$\{\underbrace{u_{\min},\cdots,u_{\min}}_{q+1},\cdots,\underbrace{u_{\max},\cdots,u_{\max}}_{q+1}\}$$

에 대해서 정의되며, 매듭 벡터의 처음과 끝에서 매듭값이 차수보다 1개 많은 개수로 중복된다.

호모지니어스 좌표를 사용하여 식 (1)과 식 (2)를 나타내면

$$C_1^w(t)=\sum_{i=0}^{l} N_{i,p}(t)\mathbf{P}_i^w$$
$$\text{where } \mathbf{P}_i^w=(w_i P_{xi}, w_i P_{yi}, w_i) \tag{3}$$

$$C_2^w(u)=\sum_{j=0}^{n} N_{j,q}(u)\mathbf{Q}_j^w$$
$$\text{where } \mathbf{Q}_i^w=(w_j Q_{xi}, w_j Q_{yi}, w_j) \tag{4}$$

이다.

Bézier 클리핑에서는 알고리즘의 구현에 대해서 정확한 언급이 없으므로, 본 연구에서는 두 NURBS 곡선 중에서, 매개 변수 구간이 좁은 곡선의 길이가 더 짧다는 가정을 하고, 이 곡선을 기준으로 시작한다. 길이가 짧은 곡선을 선택했을 때 수렴 속도가 증가하기 때문이다. 따라서 매개 변수 구간이 더 좁은 곡선 $\mathbf{C}_1(t)$를 선택한다. Bézier 클리핑과 마찬가지로

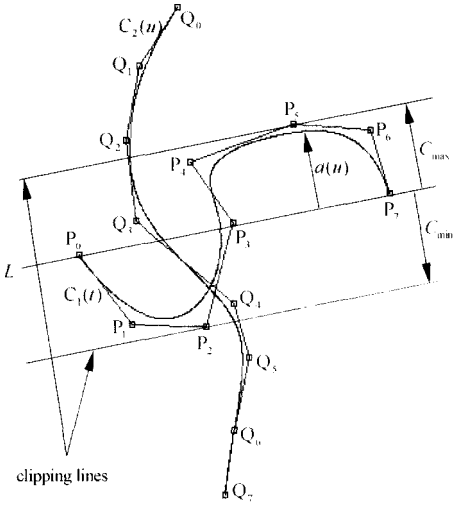


Fig. 2. Extended Bézoer clipping.

$\mathbf{C}_1(t)$의 시작점 $\mathbf{C}_1(t_{min})$과 끝점 $\mathbf{C}_1(t_{max})$를 지나는 직선

$$L = ax + by + c \quad (a^2+b^2=1,\ b \geq 0) \tag{5}$$

를 찾는다. 비주기 NURBS 곡선은 조정점의 시작점과 끝점을 지나기 때문에 이 점들로 L을 쉽게 구할 수 있다. 직선 L에서 임의의 점까지의, 부호가 있는 거리는

$$ax + by + c \quad (\because \sqrt{a^2+b^2}=1) \tag{6}$$

이다. 직선 L에서 $\mathbf{C}_1(t)$의 조정점들 까지의 거리가 최소(C_{min})인 점과 최대(C_{max})인 점을 구하고, 이 점들에서 L에 평행한 직선을 그린다.

그런데 곡선을 분할하기 위해 평행선을 사용하기 때문에 이를 클리핑 라인(clipping lines)으로 정의한다. $\mathbf{C}_1(t)$는 클리핑 라인 안에 존재하므로 클리핑 라인 바깥에 있는 $\mathbf{C}_2(u)$의 구간은 $\mathbf{C}_1(t)$와 만날 수 없다. 따라서 이 클리핑 라인으로 $\mathbf{C}_2(u)$를 분할하여 매개 변수 구간을 줄인다.

3.2 곡선 분할 과정

$\mathbf{C}_2(u)$를 클리핑 라인으로 분할하는 과정은 다음과 같다. 직선 L에서 $\mathbf{C}_2(u)$까지의 거리는 식 (2)와 식 (6)을 이용하여

$$\hat{d}(u) = a\left(\frac{\sum_{j=0}^{n} N_{j,q}(u) w_j Q_{xj}}{\sum_{j=0}^{n} N_{j,q}(u) w_j}\right) + b\left(\frac{\sum_{j=0}^{n} N_{j,q}(u) w_j Q_{yj}}{\sum_{j=0}^{n} N_{j,q}(u) w_j}\right) + c$$

$$= \frac{\sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{yi} + c)}{\sum_{j=0}^{n} N_{j,q}(u) w_j} \tag{7}$$

와 같이 구할 수 있다. 식 (7)을 이용하여 클리핑 라인 바깥에 있는 부분

$$\sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{yj} + c - C_{max}) > 0$$

$$\sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{xj} + c - C_{min}) < 0 \tag{8}$$

을 잘라 내어 곡선을 분할할 수 있다. 식 (8)에서 식 (7)의 분자만을 고려하므로 계산의 편의를 위해 거리를

$$d(u) = \sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{yj} + c)$$

$$= \sum_{j=0}^{n} N_{j,q}(u) d_j \tag{9}$$

로 정의한다. 식 (9)에서 $aQ_{xj}+bQ_{yj}+c$는 L에서 $\mathbf{C}_2(u)$의 조정점들 까지의 거리이다. $\mathbf{C}_2(u)$와 클리핑 라인이 만나는 매개 변수 값을 구하기 위해

$$\sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{yj} + c) = d_{max}$$

$$\sum_{j=0}^{n} N_{j,q}(u) w_j (aQ_{xj} + bQ_{yj} + c) = d_{min} \tag{10}$$

where $d_{max} = C_{max} \sum_{j=0}^{n} N_{j,q}(u) w_j$

$$d_{min} = C_{min} \sum_{j=0}^{n} N_{j,q}(u) w_j$$

을 계산해야 한다. 식 (10)은 곡선과 직선의 교점을 구하는 문제이므로 계산이 복잡하다. 따라서 식 (10)을 이용하지 않고 다음과 같은 방법으로 근사적으로 해를 구하여 효율을 증가 시키도록 하였다.

$d(u)$를 u와 d의 도메인에서 나타내면 $d(u)$는 u의 함수이지만 B-spline 곡선의 다음과 같은 특성을 이용하여 $d(u)$를 곡선의 형태로 나타낼 수 있다.

임의의 직선 $l(u)=\alpha u+\beta$에 대하여

$$\sum_{j=0}^{n} l(\xi_j) N_{j,q}(u) = \alpha u + \beta$$

$$\xi_j = \frac{1}{q}(u_{j+1} + \cdots + u_{j+q}) \tag{11}$$

를 만족한다[8]. 식 (11)은 Grevill이 유도한 식이고,

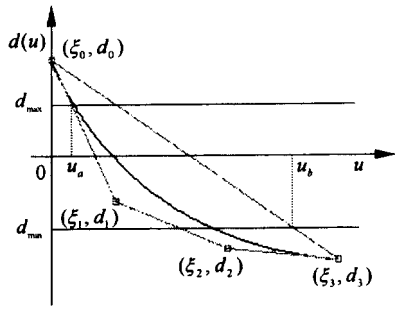


Fig. 3. Intersections between convex hull and clipping lines.

여기서 ξ_j는 Grevill의 횡좌표(abscissa)이다. 식 (11)에서 α=1이고 β=0이면

$$\sum_{j=0}^{n} N_{j,q}(u)\xi_j = u \tag{12}$$

이고 식 (9)와 식 (12)를 벡터로 나타내면 곡선의 형태인

$$\mathbf{D}(u) = \begin{bmatrix} u \\ d(u) \end{bmatrix} = \sum_{j=0}^{n} N_{j,q}(u) \begin{bmatrix} \xi_j \\ d_j \end{bmatrix} \tag{13}$$

를 얻을 수 있다. 여기서 (ξ_j, d_j)는 $\mathbf{D}(u)$의 조정점이 된다. Bézier 클리핑에서는 $d(u)$를 곡선의 형태로 나타내기 위해서

$$\sum_{j=0}^{n} B_{j,n}(u)\left(\frac{j}{n}\right) = u \tag{14}$$

를 이용하였고[5], 본 연구에서는 B-spline 곡선의 linear precision과 Grevill의 횡좌표를 이용하여 식 (14)와 유사한 식 (12)를 유도하였다.

$\mathbf{D}(u)$의 조정점을 구하였으므로 convex hull을 만들 수 있고 이 convex hull 안에 $\mathbf{D}(u)$가 포함된다. 클리핑 라인을 u와 $d(u)$의 도메인에서 그리면 u축에 평행한 선이 되고, 이 클리핑 라인과 $\mathbf{D}(u)$의 convex hull간의 교점을 구하여 그 중에서 최소값(u_a)와 최대값(u_b)에서 $\mathbf{C}_2(u)$를 분할하면 클리핑 라인 밖에 있는 구간을 버릴 수 있다. 이 때 직선간의 교점을 구하므로 계산이 간단해 진다. 여기서 구한 u는 $\mathbf{D}(u)$와 클리핑 라인간의 교점이 아니고 $\mathbf{D}(u)$의 convex hull과 클리핑 라인간의 교점 중에서 최소값과 최대값이므로 분할된 곡선은 끝점이 정확하게 클리핑 라인 위에 있지 않고 클리핑 라인을 바깥쪽으로 조금 벗어나게 된다.

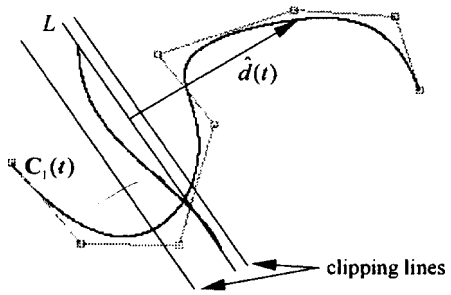


Fig. 4. After first extended Bézier clipping.

$\mathbf{C}_2(u)$를 u_a와 u_b에서 분할하기 위하여 매듭 벡터에 u_a와 u_b를 차수 q만큼 중복 삽입하여

$$\{\cdots, \underbrace{u_a, \cdots, u_a}_{s_a}, \underbrace{u_a, \cdots, u_a}_{q-s_a}, \cdots, \underbrace{u_b, \cdots, u_b}_{s_b}, \cdots, \underbrace{u_b, \cdots, u_b}_{q-s_b}, \cdots\} \tag{15}$$

를 구하는데 s_a와 s_b는 각각 u_a와 u_b가 삽입되기 전에 중복되어 있는 개수이다. $\bar{u} \in [u_k\ u_{k+1})$를 r번 중복했을 때 j번째 조정점은 호모지니어스 좌표를 사용하여

$$\mathbf{R}_{j,r}^{w} = \alpha_{j,r}\mathbf{R}_{j,r-1}^{w} + (1-\alpha_{j,r})\mathbf{R}_{j-1,r-1}^{w}$$

$$\alpha_{j,r} = \begin{cases} 1 & j \le k-q+r-1 \\ \dfrac{\bar{u}-u_j}{u_{j+q-r+1}-u_j} & k-q+r \le j \le k-s \\ 0 & j \ge k-s+q \end{cases} \tag{16}$$

로 구하고, 식 (15)에서 매듭 벡터를

$$\{\underbrace{u_a, \cdots, u_a}_{q+1}, \cdots, \underbrace{u_b, \cdots, u_b}_{q+1}\} \tag{17}$$

로 취하면 u_a와 u_b에서 분할된 곡선을 얻을 수 있고[7], 매개 변수 u의 구간이 줄어든다.

분할된 곡선의 클리핑 라인에 대해서 $\mathbf{C}_1(t)$를 분할하고 클리핑 라인 바깥에 있는 $\mathbf{C}_1(t)$의 부분을 제거하여 매개 변수 t의 구간을 줄인다.

곡선 분할 과정을 반복하여 오차 한계 까지 두 곡선의 매개 변수 구간이 줄어들면 이들 구간의 중점으로 교점을 구한다.

3.3 클리핑 라인과 convex hull의 위치에 따른 클리핑 방법

클리핑 라인과 convex hull의 교점을 구할 때 Fig. 5와 같은 경우가 생긴다.

Fig. 5에서 (a)와 (f)는 곡선이 클리핑 라인의 바깥쪽에 놓여 있기 때문에 두 곡선간의 해가 없음을 나타내고, (d)의 경우에는 클리핑 라인으로 곡선을 분

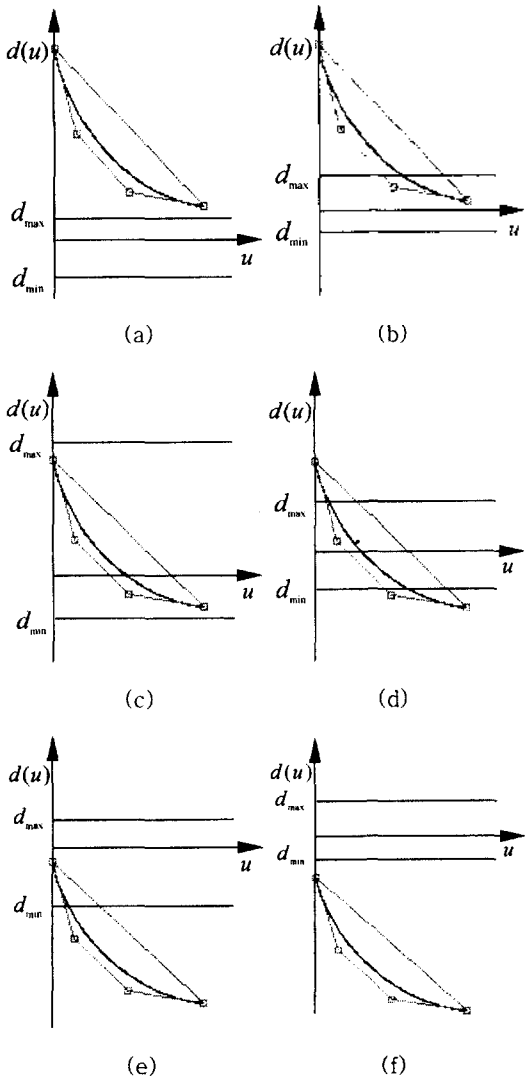


Fig. 5. Various configuration of convex hull and clipping lines.

합하고 (b), (c), (e)의 경우에는 매개 변수 구간이 좁은 곡선을 반으로 분할하고 분할된 곡선들과 다른 곡선의 교점을 구한다. 곡선을 분할하여 매개 변수 구간이 20% 이상 줄지 않으면 클리핑된 곡선을 반으로 분할한다. 20%는 Bézier 클리핑에서 나온 경험적인 수치로 이 때 가장 좋은 성능을 발휘하며, 10%보다 작은 값이면 Bézier 클리핑이 지나치게 많이 수행되고, 40%보다 클 경우에는 불필요한 이진 분할(binary subdivision)이 발생한다[6]. 확장된 알고리즘에서도 20%를 사용하였다. 이와 같은 방법으로 교점이 여러 개인 경우에도 해를 구할 수 있다.

```
C1.findIntersection(C2)
  if(C1.Δu < ε and C2.Δu < ε)
    (C1.u_min + C1.u_max)/2 , (C2.u_min + C2.u_max)/2
    ⇒ intersection point
  endif

  if(C1.Δu > C2.Δu)
    C2.findIntersection(C1)
  endif

  if(C1 is outside of C2.clippingLines)
    no intersection
  endif

  if(C1 is inside of C2.clippingLines)
    firstHalfOfC1.findIntersection(C2)
    secondHalfOfC1.findIntersection(C2)
  endif

  clippingLinesOfC1.decompose(C2)

  if(decomposedC1.Δu ≤ 0.2×C1.Δu)
    decomposedC1.findIntersection(C2)
  else
    firstHalfOfDecomposedC1.findIntersection(C2)
    secondHalfOfDecomposedC1.findIntersection(C2)
  endifelse
endfindIntersection
```

Fig. 6. Pseudo code of extended Bézier clipping.

이상의 알고리즘을 Fig. 6에 나타내었다.

4. 적용 사례

확장된 알고리즘을 Java 언어를 사용하여 IBM RS/6000 580에서 구현하였다. Java는 객체 지향 프로그래밍 언어로서 플랫폼에 관계없이 프로그램을 실행시킬 수 있으며, 네트워크 기능이 우수하므로 지리적 한계를 극복하고 멀리 떨어진 곳에서도 네트워크를 통한 설계를 할 수 있다. Bézier 클리핑과 확장된 알고리즘의 속도를 비교하기 위해 JIT를 사용하지 않고 인터프리터 방식을 사용하였다.

Fig. 7은 조정점이 각각 15개와 17개인 두 곡선이 한 점에서 만나는 경우이다. 차수가 증가할 때 확장

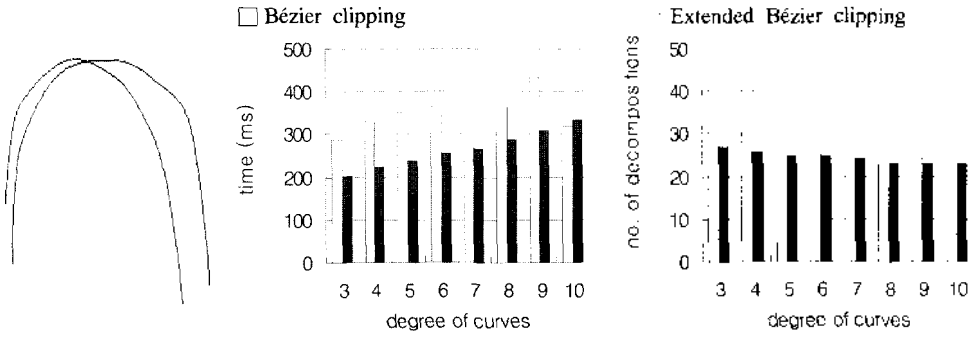


Fig. 7. Example 1

된 알고리즘의 계산 시간은 거의 선형으로 증가하는 성향을 보이고, Bézier 클리핑도 차수가 높아질 때 계산 시간이 증가하지만 7차와 10차에서는 오히려 감소한다. 분할 횟수를 보면 확장된 알고리즘은 차수가 높아지면서 분할 횟수가 감소하고, Bézier 클리핑은 5차부터 감소 경향을 보이다 9차에서 크게 증가한다.

Fig. 8은 조정점이 각각 21개와 17개인 두 곡선의 교점을 구하는 경우이다. 확장된 알고리즘은 차수가 높아지면서 선형으로 증가하고, Bézier 클리핑도 증가하는 경향을 보이는데 8차와 10차에서는 약간 감소했다. 확장된 알고리즘은 4차부터 분할 횟수가 거의 일정하고, Bézier 클리핑은 차수가 높아지면서 감소 경향을 보인다.

Fig. 9는 조정점이 각각 95개와 25개인 두 곡선의 교점을 구하는 경우이다. 확장된 알고리즘이 Bézier 클리핑보다 월등히 우수한 경우이다. 확장된 알고리즘은 차수가 높아지면서 계산 시간의 증가가 작은 반면 Bézier 클리핑은 선형으로 증가한다. 분할 횟수 또한 확장된 알고리즘은 감소 추세를 보이고 Bézier 클리핑도 7차와 9차를 제외하고 감소 추세를 나타낸다.

Fig. 10은 조정점이 각각 78개와 85개인 두 곡선이

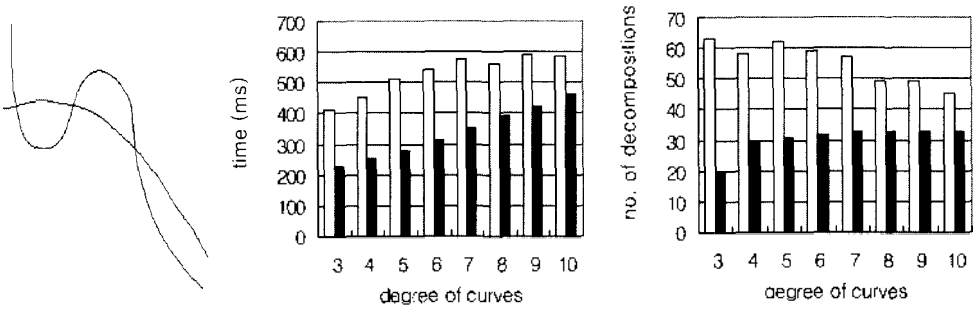


Fig. 8. Example 2

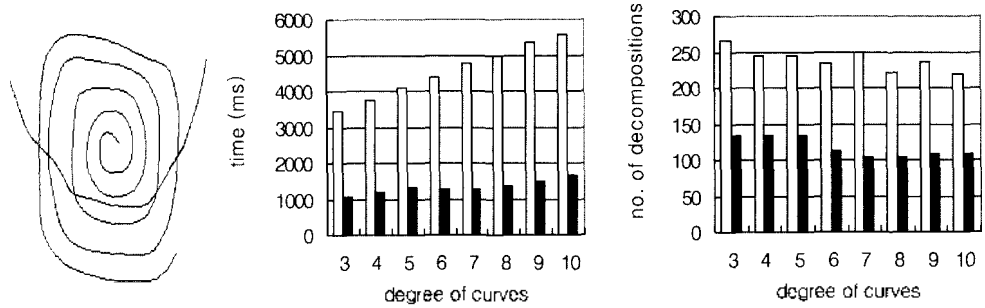


Fig. 9. Example 3

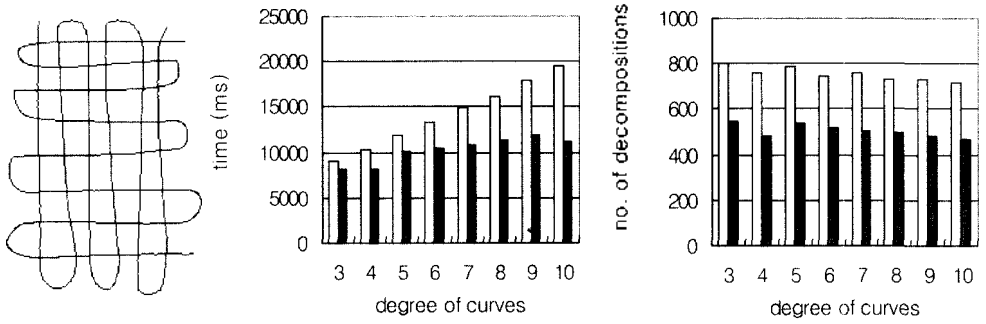


Fig. 10. Example 4

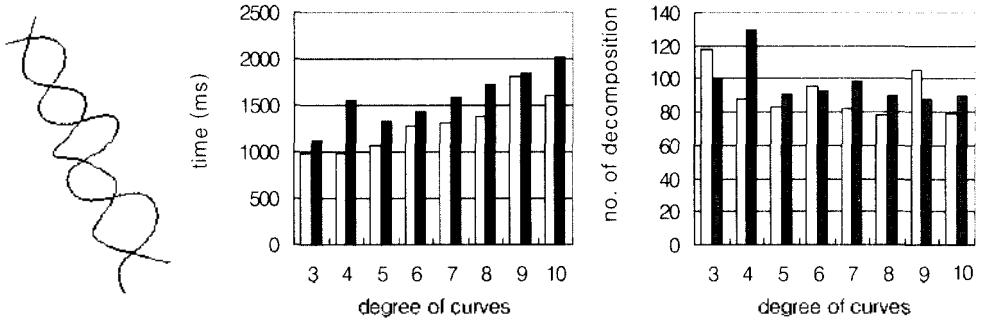


Fig. 11. Example 5

만나는 경우이다. 차수가 증가하면서 Bézier 클리핑이 확장된 알고리즘보다 가파른 증가세를 보인다. 분할 회수는 차수가 증가하면서 두 알고리즘 모두 약간의 감소 경향을 나타낸다.

Fig. 11은 조정점이 각각 19개와 20개인 두 곡선의 교점을 구하는 경우이다. 확장된 알고리즘이 Bézier 클리핑보다 느린 경우로서 특히 4차에서 Bézier 클리핑보다 많은 계산 시간이 소요된다. 그러나 전체적으로 두 알고리즘의 계산 속도가 비슷한 경향을 보인다. 분할 횟수 또한 4차에서 큰 차이가 난다. 두 곡선이 같은 방향으로 놓여 있으면서 교점의 개수가 많을 때 확장된 알고리즘의 분할 횟수가 증가하여 계산 시간이 많이 소요되기 때문이다. 그러나 Bézier 클리핑은 여러 개의 교점을 찾기 위한 분할 과정을 초기에 했기 때문에 그 만큼 시간적 보상을 받은 것이다.

5. 결 론

본 연구에서는 B-spline 곡선의 linear precision과 Grevill의 횡좌표를 이용하여 NURBS 곡선을 초기에 여러 개의 Bézier 곡선으로 분할하지 않고, Bézier 클리핑을 확장하여 필요한 경우에만 분할하도록 하는 알고리즘을 개발하였다. 본 알고리즘은 NURBS 곡선이 여러 개의 Bézier 곡선으로 구성되어 있고, 교점의 개수가 적은 경우에 Bézier 클리핑보다 계산 속도가 더 빠르고 효율적이다. 그러나 두 곡선간의 교점 수가 많고 두 곡선이 같은 방향으로 놓여 있을 경우에는 Bézier 클리핑 보다 분할 횟수가 많아지는 단점이 있다.

본 연구의 알고리즘을 Fig. (6)과 같이 구현하였지만, 두 곡선 중에서 어느 곡선을 기준으로 클리핑할 것이며, Fig. (5)의 (b), (c), (e)와 같은 경우에 어느 곡선을 어떤 비율로 분할할 것인지에 대해서는 여러 가지 선택이 있을 수 있다. 또한 클리핑된 곡선의 구간이 20%이상 줄어 들지 않을 경우에 이 곡선을 반으로 분할하였는데, 이때 20% 수치는 Bézier 클리핑의 수치를 그대로 적용하였다. 따라서 수치가 확장된 알고리즘에도 적합한지에 대한 통계가 필요하다.

비주기 NURBS 곡선이 아닌 경우에는 매듭값 변환[7]을 통하여 비주기 NURBS 곡선으로 고친 후 알고리즘을 적용할 수 있을 것이다.

본 연구에서 제안한 알고리즘은 곡선의 기하학적 특성을 이용하여 교점을 구하므로 이 알고리즘의 수

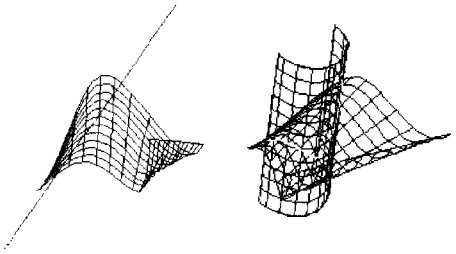


Fig. 12. Curve/surface and surface/surface intersections.

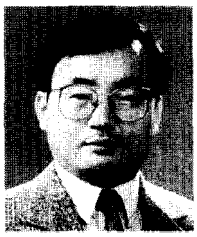
력에 대한 수학적 연구와 두 곡선의 일부분이 겹쳐 있는 경우에 그 구간을 찾아내는 방법 등의 개발이 향후 연구 과제로 필요하다. 또한 Fig. 12와 같이 곡선과 곡면, 곡면과 곡면간의 교점이나 교선을 계산하는 알고리즘으로 확장하는 연구가 필요하다.

참고문헌

1. Hoschek, Josef and Lasser, Dieter, "Fundamentals of Computer Aided Geometric Design", A K Peters, 1989.
2. 이건우, 컴퓨터 그래픽과 CAD, 영지문화사, 1994.
3. Sederberg, T.W. and Anderson, D.C., "Implicit Representation of Parametric Curves and Surfaces", *Computer Vision, Graphics, and Image Processing*, Vol. 28, pp. 72-84, 1984.
4. Sederberg, T.W. and Parry, S.R., "Comparison of three curve intersection algorithms", *Computer-Aided Design*, Vol. 18, No. 1, pp. 58-63, 1986.
5. Sederberg, T.W. and Nishita, T., "Curve Intersection using Bézier Clipping", *Computer-Aided Design*, Vol. 22, No. 9, pp. 538-549, 1990.
6. Nishita, Tomoyuki, Sederberg, T.W. and Kakimoto, Masanori, "Ray Tracing Trimmed Rational Surface Patches", *Computer Graphics*, Vol. 24, No. 4, pp. 337-345, 1990.
7. Piegl, L and Tiller, W., The NURBS Book, Springer, 1995.
8. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design A Practical Guide*, 3rd edition, Academic Press, 1993.

민 병 녕

1996년 한양대학교 자동차공학과 학사
1998년 한양대학교 기계설계학과 공학석사
1998년 ~ 현재 기아자동차 중앙기술연구소 근무
관심분야 : CAD/CAM 응용, Computational Geometry



김 재 정

1981년 한양대학교 정밀공학과 학사
1983년 미국 George Washington 대학 공학석사
1983년 ~ 1984년 미국 National Food Processors Association 연구원
1989년 미국 MIT 공학박사
1989년 ~ 1991년 미국 IBM T.J. Watson 연구소 연구원
1991년 ~ 1993년 한국 IBM 소프트웨어 연구소 연구원
1993년 ~ 현재 한양대학교 기계공학부 부교수
관심분야 : Geometric Modeling, CAD/CAM 응용