한국 CAD/CAM학회 논문집
제15권 제2호 2010년 4월 pp. 115-123

평면곡선에 대한 Hausdorff 거리 계산의 가속화 기법에 대한 연구

김용준*, 오영택*, 김명수**

Efficient Hausdorff Distance Computation for Planar Curves

Yong-Joon Kim*, Young-Taek Oh* and Myung-Soo Kim*

ABSTRACT

We present an efficient algorithm for computing the Hausdorff distance between two planar curves. The algorithm is based on an efficient trimming technique that eliminates the curve domains that make no contribution to the final Hausdorff distance. The input curves are first approximated with biarcs within a given error bound in a pre-processing step. Using the biarc approximation, the distance map of an input curve is then approximated and stored into the graphics hardware depth-buffer by rendering the distance maps (represented as circular cones) of the biarcs. We repeat the same procedure for the other input curve. By sampling points on each input curve and reading the distance from the other curve (stored in the hardware depth-buffer), we can easily estimate a lower bound of the Hausdorff distance. Based on the lower bound, the algorithm eliminates redundant curve segments where the exact Hausdorff distance can never be obtained. Finally, we employ a multivariate equation solver to compute the Hausdorff distance efficiently using the remaining curve segments only.

Key words : Hausdorff distance, biarc approximation, distance map, depth buffer, trimming, solution of multivariate equation

1. 서 론

인류는 유사이래 거리측정을 정확하게 하기 위해서 뿐만 아니라 거리의 개념을 일반화하기 위하여 많은 노력을 들여왔다. 상대성원리의 이론도 기존의 Euclidean 거리개념을 Riemann 거리의 개념으로 일반화하는 과정에서 개발되었다.

기하모델링 분야의 연구에 있어서 두 물체들 사이의 최단거리계산에 관한 효율적인 알고리즘들이 지금까지 많이 개발되었다[1-7]. 이에 비하여 두 물체들 사이의 유사성을 측정하는데 적합한 Hausdorff 거리계산에 관한 알고리즘의 개발은 아직도 거의 초보적인 단계에 머물러 있다[8-12]. Hausdorff 거리 계산에 대한 이전의 연구결과들에 비하여 Tang 등[13]의 최근 연구결과는 아주 획기적인 방법으로서 정확한 Hausdroff 거리의 상한값과 하한값을 효율적으로 찾는 새로운 기법에 근거를 두고 있다. 하지만, 삼각형의 특성에 기반을 둔 이 결과를 곡선이나 곡면으로 둘러싸인 자유 형상의 물체들에 적용하기는 쉽지 않다.

최근 Elber and Grandine[14]은 두 곡선사이의 Hausdorff 거리를 계산하는 알고리즘을 개발하였다. 하지만, 알고리즘의 중간단계에서 계산되는 많은 연립다항식들의 해들이 최종결과에는 영향을 주지 않는 경우가 많으므로, 전반적으로 알고리즘의 효율성이 크게 떨어진다. 본 연구에서는 최종 결과에 영향을 미치는 부분곡선들을 빨리 찾아내어 알고리즘의 효율성을 높이는 가속화 기법을 제안한다.

Aichholzer 등[15]은 최근의 연구에서 평면 곡선분들의 Voronoi diagram을 아주 안정적임과 동시에 오차가 적게 계산하는 알고리즘을 제시하였다. 이 새로운 접근방법은 주어진 입력곡선들을 biarc들로 근사하여 arc 들에 대한 Voronoi diagram 계산을 안정적으로 수행하는 아주 획기적인 결과이다. 본 논문에서는 이에 착안하여, Hausdorff 거리계산의 중간단계에서 biarc들을 이용하여 알고리즘의 효율성과 안정성을 동

*학생회원, 서울대학교 전기컴퓨터공학부
**비회원, 서울대학교 컴퓨터공학부
- 논문투고일: 2009. 03. 03
- 논문수정일: 2010. 01. 19
- 심사완료일: 2010. 02. 09

시에 높이는 접근방법을 제시한다.

Voronoi 영역의 계산에 비하여 Hausdorff 거리계산은 보다 간단한 문제일 수 있다. 왜냐하면, Hausdorff 거리는 하나의 실수 값으로 주어지는데 비하여, Voronoi 영역은 평면을 여러 개의 복잡한 영역으로 분할하고 이를 데이터구조로 표현하는 일련의 과정을 거쳐서 계산되기 때문이다. 반면 Hausdorff 거리는 본질적으로 Voronoi 영역을 구하지 않고는 그 계산을 제대로 수행하기 어렵다는 관계로 서로 복잡하게 얽혀 있다.

본 연구에서는 이와 같은 문제를 해결하기 위한 현실적인 방안으로, 그래픽스 하드웨어 렌더링 기능을 이용하여 Voronoi 영역의 근사계산을 효율적으로 수행하는 깊이 buffer 기반의 알고리즘[16]을 중간단계에서 이용한다. 이에 기반하여 Hausdorff 거리계산의 최종해를 효율적으로 구하는 가속화 기법을 개발하며, 그 구체적인 방법은 아래의 단계에 따라 진행된다(여기서 주어진 입력곡선들은 전처리 단계에서 biarc 들로 이미 근사되어 있다고 가정한다.)

1. 각 arc의 거리 함수를 나타내기 위하여, 이 원호를 포함하고 xyz-공간상에서 정의된 circular cone을 만든다.
2. 거리함수를 나타내는 circular cone들을 z축의 $-\infty$ 시점으로부터 양의 방향으로 보면서, 그래픽스 깊이 버퍼에 렌더링한다.
3. 그래픽스 깊이 버퍼의 각 pixel에 저장된 깊이값이 바로 그 점으로부터 주어진 곡선까지의 최단거리를 나타낸다.
4. 깊이 버퍼의 값으로 예측한 Hausdorff 거리를 이용하여 곡선의 불필요한 부분을 트리밍하고 나서 후보로 남은 곡선 구간에 대해서만 Elber and Grandine[14]에서 사용된 연립방정식을 풀어서 정확한 Hausdorff 거리를 효율적으로 계산한다.

본 논문의 구성은 다음과 같다. 먼저 제 2절에서 본 연구의 기초가 되는 기본 개념들을 설명한다. 제 3절에서는 biarc로 근사된 곡선의 거리 함수를 깊이 버퍼에 렌더링하여 저장하고 이를 이용하여 Hausdorff 거리의 하한값을 계산하는 기법을 소개한다. 그리고 이러한 하한값을 이용하여 곡선 구간 중 실제 Hausdorff 거리에 기여하지 않는 부분을 안전하게 트리밍하는 방법에 대해 논의한다. Hausdorff 거리를 정확하게 계산하는 방법은 제 4절에서 설명하고, 본 논문에서 제안하는 접근방법의 효율성을 입증하기 위하여 여러 가지의 실험 결과를 제 5절에서 보인다. 마지막으로 제 6절에서 본 연구의 결론을 내리고 추후 연구과제에 대해서 논한다.

2. Preliminaries

2.1 Hausdorff 거리

임의의 두 집합 A, B와 이들에 속한 임의의 원소 $x \in A$ 와 $y \in B$ 사이의 거리함수 $d(x,y)$가 주어져 있을 때, 집합 A로부터 B까지의 Hausdorff 거리 $h(A,B)$는 다음과 같이 정의된다[14].

$$h(A,B) = \max_{a \in A,\, b \in B} \min d(a,b)$$

집합 B로부터 A까지의 Hausdorff 거리 $h(B,A)$도 마찬가지로 정의되며 일반적으로 $h(A,B) \neq h(B,A)$이다.

이로부터 A와 B 사이의 Hausdorff 거리 $H(A,B)$는 양쪽의 거리 $h(A,B)$와 $h(B,A)$를 모두 고려하여

$$H(A,B) = \max h\{(A,B), h(B,A)\}$$

로 정의된다.

2.2 Biarc 근사

두 점과 각 점에서의 접선 방향 벡터가 주어 졌을 때 각 점을 지나고 점에서 주어진 접선방향을 가지는 곡선을 두 개의 G^1 연속인 원호를 이용하여 만들 수 있는데 이를 biarc라고 한다. 본 연구는 주어진 곡선을 여러 개의 biarc로 근사한 뒤 이를 이용하여 곡선간의 Hausdorff거리를 계산한다. 곡선을 biarc로 근사하는 여러 가지의 방법들이 있다[17-20]. 본 논문에서는 Sir 등[17]이 제안한 방법을 이용하여 임의의 오차 범위 내로 주어진 곡선을 근사한다. Fig. 1은 biarc의 한 예를 나타낸다. 그림에서와 같이 두 원호의 교점을 joint라고 하자.

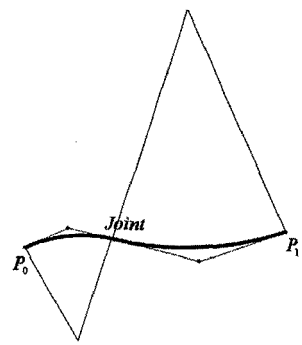


Fig. 1. Biarc curve construction.

Fig. 2와 같이 어떤 curve segment의 양 끝점과 접선방향이 주어졌을 때 이러한 경계조건을 만족하는 biarc에는 여러 가지가 있는데 Sir 등[17]은 이러한 biarc중 joint가 curve segment 위에 있는 biarc가 적어도 하나 존재함을 보였다. Fig. 3은 이러한 biarc의 한가지 예를 나타낸다.

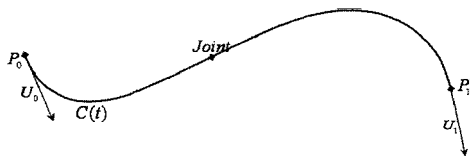


Fig. 2. End points and their tangent directions.

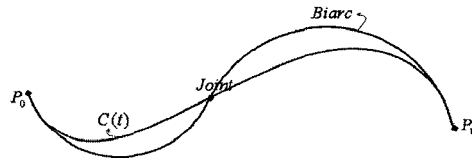


Fig. 3. Biarc approximation of Sir *et al.*[17].

이와 같은 Sir 등[17]의 biarc근사 방법은 다른 결과들에 비해 실제곡선과의 오차가 작고, 각각의 원호마다 이에 대응하는 curve segment의 구간이 잘 정의되어 곡선을 원호 단위로 처리하기가 편리하다. 이러한 성질은 뒤에 소개할 곡선 트리밍에 아주 유용하게 사용된다.

3. Hausdorff 거리 예측

3.1 원호까지의 거리 함수

Hoff 등[16]은 깊이 버퍼를 이용하여 Voronoi diagram을 계산하는 방법을 제안하였다. 평면 상의 한 점 P를 생각하자. Fig. 4와 같이 점 P를 꼭지점으로 하고 중심축이 평면에 수직인 원뿔을 그리면 평면상의 임의의 점 X로부터 점 P까지의 거리 $\overline{XP}$는 $\overline{XX'}$의 길이와 같게 된다.

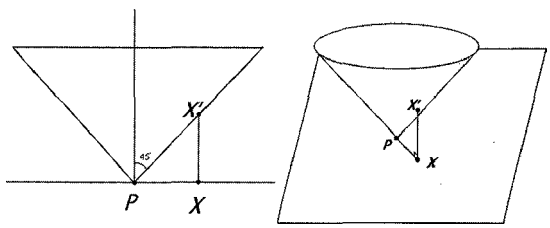


Fig. 4. Distance from an arbitrary point X to the point P.

마찬가지로 P와는 다른 점 Q에서도 원뿔을 그리자. 이 두 원뿔을 아래 쪽에서 평면에 수직인 방향으로 보면 Fig. 5의 오른쪽과 같은 형태로 보이게 된다. 두 원뿔이 평면상에 투영된 영역은 두 점 P와 Q사이의 bisector 직선에 의하여 이등분되는데 만일 점 X가 P의 원뿔이 만든 영역에 속한다면 점 P에 더 가까운 것이고 Q의 원뿔이 만든 영역에 속한다면 점 Q에 더 가까운 것이다. 즉 원뿔을 평면에 수직인 방향으로 정사영 시킨 그림이 바로 Voronoi diagram이 된다.

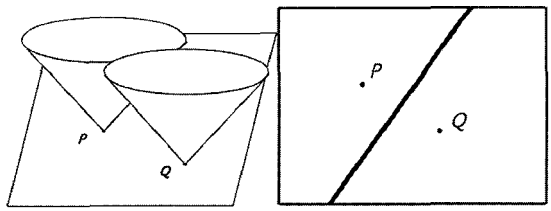


Fig. 5. Voronoi diagram of two points P, Q.

이러한 사실에 착안하여 Hoff 등[16]은 각 Voronoi site에 해당되는 원뿔을 렌더링하여 Voronoi diagram을 근사하였다. 이렇게 렌더링을 하면 Voronoi diagram을 근사할 수 있을 뿐만 아니라 각 픽셀에 저장된 깊이 값이 픽셀에 해당하는 점으로부터 가장 가까운 Voronoi site까지의 최단 거리가 된다. 본 연구에서는 이러한 거리 함수를 원호에 적용시킨다.

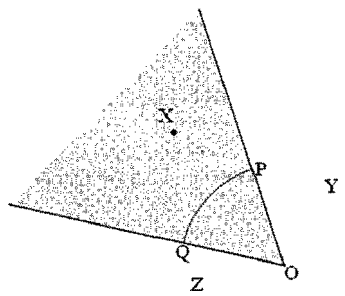


Fig. 6. Minimum distance between a point and an arc.

Fig. 6과 같이 점 O를 중심으로 하는 원호 $\overset{\frown}{PQ}$를 생각해보자. 두 개의 반직선 $\overrightarrow{OP}$와 $\overrightarrow{OQ}$ 사이의 영역에 있는 점 X와 원호 사이의 최단거리는 원호의 양 끝점이 아닌 P와 Q 사이에 있는 원호 상의 한 점에서 구해진다. 반면에 반직선의 밖에 있는 점들 Y, Z와 원호 사이의 최단거리는 각각 끝 점인 P와 Q에서 구해진다.

우리는 원래의 곡선을 G^1연속인 biarc들로 근사했기 때문에 다른 두 개의 원호들이 부드럽게 연속적으

로 연결된 원호의 연결점에서 일어나는 최단거리에는 신경을 쓰지 않아도 된다. 왜냐하면 원호의 한 끝점에서 최단거리가 구해진다는 것은 우리가 지금 측정하고 있는 원호가 아닌 다른 인접한 원호에서 최단거리가 구해진다는 의미이기 때문이다.

반직선 $\overrightarrow{OP}$와 $\overrightarrow{OQ}$ 사이의 색칠된 부분에 해당하는 거리함수를 그려보면 Fig. 7와 같이 잘려진 원뿔의 부분곡면들로 이루어져 있으며 이들은 각각 NURBS 곡면으로 정확하게 표현 가능하다. 따라서, 이를 이용하여 거리함수를 렌더링 하는 것도 오차가 아주 작게 할 수 있다.

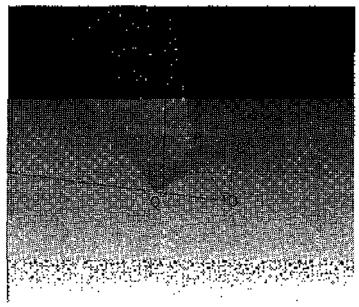


Fig. 7. Distance field of an arc $\widehat{PQ}$.

3.2 깊이 버퍼 값 읽기

이제 biarc로 근사 된 곡선에 속한 모든 원호에 대해서 앞에서 설명한 거리함수를 렌더링 한다. 이때 임의의 점이 어느 원호로부터 최단거리를 갖는지 알기 위해서 각각의 원호마다 다른 색을 이용하여 그려준다. Fig. 8은 이렇게 렌더링 된 한가지 예를 보여준다.

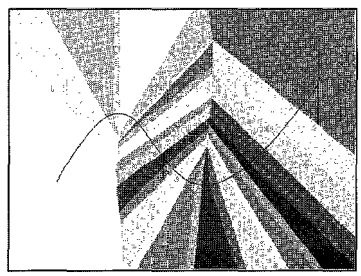


Fig. 8. Distance field rendering.

렌더링이 끝나면 상대 곡선을 따라가면서 Fig. 9과 같이 그 곡선 위의 점을 sampling 하면서 그 점에 해당하는 pixel의 깊이 버퍼를 읽어나간다. 이 값이 원래 주어진 곡선까지의 최단거리의 근사값이 된다.

한쪽 곡선에 대한 최단거리 근사가 끝나면 반대 곡선에 대해서도 똑 같은 방법으로 거리 함수를 렌더링 하고 최단거리의 근사값을 계산한다. 이와같이 두 곡선을 차례로 따라가면서 상대방 곡선까지 측정한 최단거리 중에서 가장 큰 값을 Hausdorff 거리의 근사값으로 사용 한다.

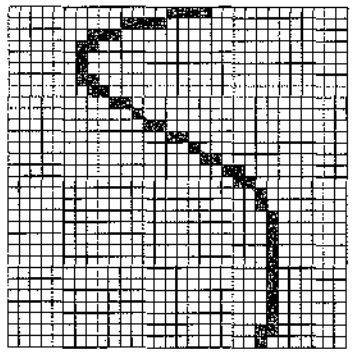


Fig. 9. Pixels on a curve.

4. Hausdorff 거리계산

설명의 편의를 위해서 두 개의 곡선을 각각 $C(t)$, $D(s)$라 하고 이들의 biarc 근사 곡선을 각각 $BC(t)$, $BD(s)$라고 하자. 그리고 $h(C(t), D(s))$을 구하는 방법과 $h(D(s), C(t))$를 구하는 방법이 동일하므로 여기서는 $h(D(s), C(t))$를 계산하는 경우만 설명한다.

4.1 계산오차의 처리

위에서 구한 Hausdorff 거리의 근사값을 이용하여 실제 Hausdorff 거리의 범위를 쉽게 구할 수 있다. 본 연구에서 사용하는 방법에서 발생하는 오차는 크게 세 가지로 분류할 수 있다. 한가지는 실제 곡선을 biarc로 근사 할 때 발생하는 오차 ε_1, 두 번째 오차는 거리 함수를 렌더링 할 때 발생하는 오차 ε_2, 그리고 마지막으로 거리 함수를 읽을 때 곡선의 실제 좌표가 아닌 픽셀 단위로 읽음으로 인하여 발생하는 오차 ε_3가 있다.

첫 번째 오차 ε_1는 사용자가 원하는 만큼 줄일 수 있다. 하지만 오차를 줄이면 biarc segment의 개수가 증가하므로 그만큼 계산 시간이 더 걸리게 된다. 두 번째 오차 ε_2는 원뿔을 렌더링하는 과정에서 발생하는 triangulation 오차이다. 본 연구에서는 Hoff 등[16]이 제안한 오차분석법을 사용하였다. 원뿔은 꼭지점을 공유하는 여러 개의 삼각형으로 triangulation된다. 또한 삼각형의 개수를 늘려서 오차를 원하는 만큼 줄일 수 있다.

그리고 마지막 오차 ε_3는 픽셀의 한 변의 길이를 l이라고 했을 때 다음과 같은 부등식을 만족시킨다.

$$\varepsilon_3 \le \sqrt{2}l \quad (1)$$

따라서, 깊이 버퍼를 이용하여 구한 Hausdorff 거리의 근사값을 *app_HD* 라고 하면 실제 Hausdorff 거리 *HD*의 범위는 다음과 같다.

$$app\_HD-(\varepsilon_1+\varepsilon_2+\varepsilon_3) \le HD \quad (2)$$

즉 $app_HD-(\varepsilon_1+\varepsilon_2+\varepsilon_3)$는 Hausdorff 거리의 lower bound가 된다. 이 값을 *LB_HD* 라고 하자.

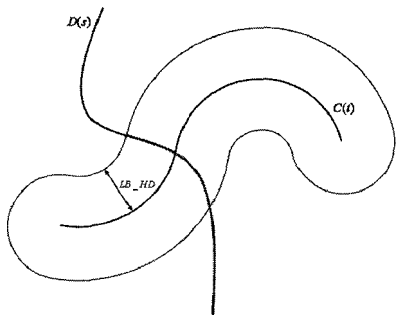


Fig. 10. Offset curve of $C(t)$.

4.2 곡선 트리밍

위에서 구한 Hausdorff 거리의 lower bound *LB_HD* 를 이용하여 $D(s)$의 구간 중 Hausdorff 거리에 기여하지 않는 구간을 찾아내어 제외시킴으로써 불필요한 계산을 최소한으로 줄일 수 있다.

반지름 *LB_HD* 인 $C(t)$의 offset 영역을 생각해보자(Fig. 10). 위의 그림에서 $D(s)$의 구간 중에 붉은색으로 표시된 부분은 Hausdorff 거리에 기여를 하지 않는다. 왜냐하면 붉은 부분에 있는 모든 점들은 $C(t)$와의 최단거리가 Hausdorff 거리의 lower bound인 *LB_HD* 보다 작기 때문이다. 그러면 우리는 이 구간을 안전하게 제거할 수 있다.

이와 같이 제거할 수 있는 구간을 찾기 위해서 $D(s)$가 지나는 모든 pixel을 검사하는 것은 비효율적이다. 따라서, 계산의 효율성을 높이기 위하여 $C(t)$와 $D(s)$의 biarc 곡선을 이용한다. $BC(t)$와 $BD(s)$는 여러 개의 원호로 이루어져 있다. 이 원호들 중에서 불필요한 원호들은 비교적 간단한 기하학적 검증을 통하여 쉽게 제거할 수 있다는 것을 보이는 것이 본 논문의 핵심이다. $BD(s)$를 이루고 있는 원호중 하나를 $ArcD(s)$라고 하자. 원호는 2차 NURBS curve이므로 이에 해당하는 control points를 찾을 수 있다. Fig. 11과 같이 원호의 control point에 해당하는 픽셀의 색상정보를 읽으면 $BC(t)$의 원호들 중 어느 원호가 그 control point와 가장 가까운지 알 수 있다.

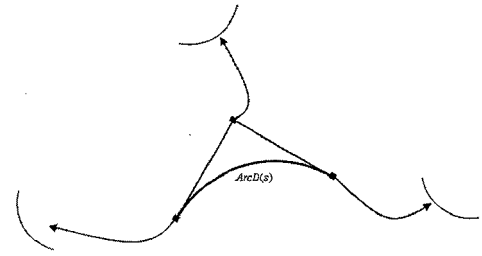


Fig. 11. Arcs of $BC(t)$ close to the $ArcD(s)$.

원호의 control point가 세 개이므로 $BC(t)$의 원호들중 최대 세 개의 원호가 가까운 원호로 선택된다. 이중에서 거리가 가장 가까울 것으로 보이는 원호를 $ArcC(t)$라고 하자. $ArcC(t)$는 $BC(t)$의 원호들 중 비교적 $ArcD(s)$와 가까우므로 $ArcC(t)$를 *LB_HD*만큼 offset 해서 구해진 영역은 $ArcD(s)$를 그 내부에 완전히 포함할 가능성이 상대적으로 높다(Fig. 12 참조).

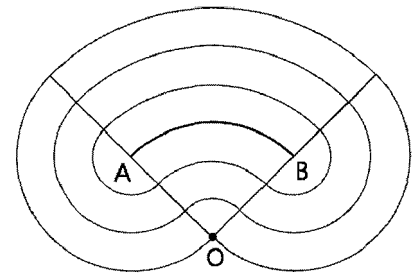


Fig. 12. Offset curves of an arc.

$ArcC(t)$의 offset 영역은 세가지 부분으로 나뉜다. Fig. 13에서 점 A를 중심으로 하고 반지름이 *LB_HD* 인 반원 ①번 영역, 그리고 반직선 $\overline{OA}$ 와 $\overline{OB}$ 사이

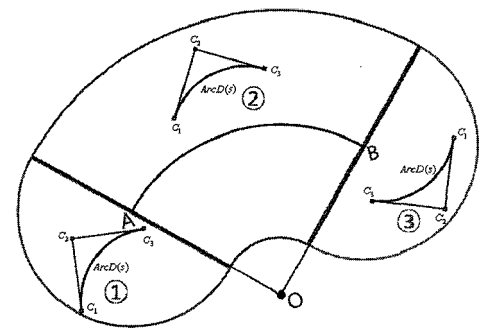


Fig. 13. Regions of an offset curve.

에 있는 ②번 영역, 그리고 점 B를 중심으로 하고 반지름이 LB_HD인 반원 ③번 영역이 있다.

우선 $ArcD(s)$의 control point가 모두 반직선 $\overline{OA}$와 $\overline{OB}$ 사이에 있는지 검사한다. 만약 모든 control point가 두 반직선 사이에 있다면 control point가 ②번 영역에 들어가는지 확인한다. Control point를 C_1, C_2, C_3라고 했을 때 control point가 ②번 영역에 속하는지 여부는 아래와 같이 확인해 볼 수 있다. (아래 수식에서의 $\overline{XB}$ 기호는 점 X와 점 Y를 잇는 선분의 길이를 나타낸다.)

$$\max\{|\overline{OC_1}-\overline{OA}|,|\overline{OC_2}-\overline{OA}|,|\overline{OC_3}-\overline{OA}|\}<LB\_HD \tag{3}$$

하지만 control point가 모두 ②번 영역 안에 있다고 하더라도 $ArcD(s)$는 Fig. 14와 같이 offset curve에 포함되지 않을 수도 있다.

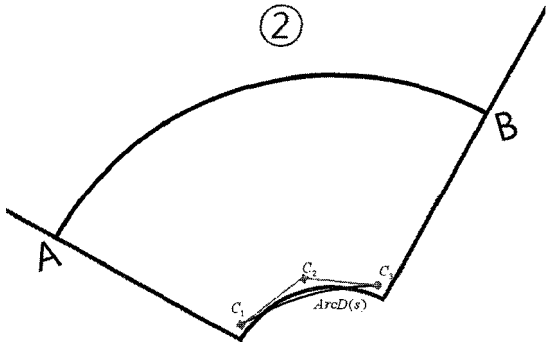


Fig. 14. Region ② does not completely contain $ArcD(s)$, but it contains control points of $ArcD(s)$.

이러한 경우를 고려하기 위해서 biarc를 생성할 때 Fig. 15처럼 원호의 ε 값을 측정해서 저장해 둔다.

그러면 ε'의 값은 다음과 같이 구할 수 있다.

$$\varepsilon' = \varepsilon \times \frac{\overline{OA'}}{\overline{OA}} \tag{4}$$

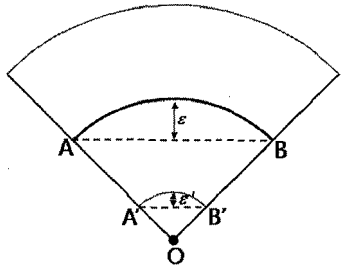


Fig. 15. Concave part of an offset curve.

Control point가 모두 ②번 영역 안에 들어갈 경우 원호를 포함하는 control polygon은 최대 ε'만큼 ②번 offset 영역을 벗어날 수 있다. 만약 모든 control point가 Fig. 16의 붉은 영역과 같이 반지름이 $LB\_HD-\varepsilon'\times 1/\cos\theta$ 인 offset 영역 안에 들어가면 control polygon은 반드시 ②번 offset 영역 내에 속하게 된다.

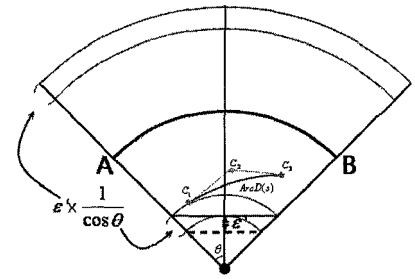


Fig. 16. Underestimating offset radius for the concave part.

$$\max\{|\overline{OC_1}-\overline{OA}|,|\overline{OC_2}-\overline{OA}|,|\overline{OC_3}-\overline{OA}|\}< LB\_HD-\varepsilon'\times\frac{1}{\cos\theta} \tag{5}$$

따라서 control point가 (5)을 만족시키면 원호의 offset 영역이 $ArcD(s)$를 완전히 포함한다.

이제 control point중 반직선 $\overline{OA}$와 $\overline{OB}$ 사이의 영역에 속하지 않는 것이 존재할 경우를 생각해보자. 이 영역에서 벗어난 최초의 점이 $\overline{OA}$의 왼편에 존재하면 $ArcD(s)$의 control point들이 점 A를 중심으로 하고 반지름이 LB_HD인 원안에 들어가는지 검사한다. 즉 다음 식을 계산한다.

$$\max\{\overline{AC_1},\overline{AC_2},\overline{AC_3}\}<LB\_HD \tag{6}$$

마찬가지로 주어진 영역에서 벗어난 최초의 점이 $\overline{OB}$의 오른편에 존재하면 다음 식을 체크한다.

$$\max\{\overline{BC_1},\overline{BC_2},\overline{BC_3}\}<LB\_HD \tag{7}$$

만약 $ArcD(s)$의 control point들이 수식 (6)나 (7)를 만족하면 $ArcC(t)$의 offset 영역은 convex hull 성질에 의하여 $ArcD(s)$를 모두 포함한다.

위에서 제시한 방법만으로는 $ArcC(t)$의 offset 영역 내에 $ArcD(s)$가 포함되는 모든 경우를 완벽하게 검사했다고 말할 수는 없다. 예를 들어서 control point가 ①, ②, ③에 각각 하나씩 걸쳐 있으면 비록 $ArcD(s)$가 $ArcC(t)$의 offset 영역안에 포함된다고 하더라도 위의 알고리즘으로는 이를 정확하게 판단할 수 없다. 하지

만 $ArcD(s)$는 매우 작은 원호이기 때문에 이러한 예외적인 경우는 거의 일어나지 않는다고 볼 수 있으므로 전반적인 계산 효율성을 높이기 위하여 이러한 경우들을 검사하지 않는다.

마지막으로 $ArcD(s)$가 아닌 이에 대응하는 실제 $D(s)$의 구간을 제거하기 위해서는 수식 (5), (6), (7)에서 biarc 근사할 때 사용된 오차 tolerance인 ε_1만큼을 빼주어야 한다.

따라서 수식 (5), (6), (7)은 다음과 같이 변형된다.

$$\max\{|\overline{OC_1}-\overline{OA}|,|\overline{OC_2}-\overline{OA}|,|\overline{OC_3}-\overline{OA}|\}< LB\_HD-\varepsilon'\times\frac{1}{\cos\theta}-\varepsilon_1 \quad (8)$$

$$\max\{\overline{AC_1},\overline{AC_2},\overline{AC_3}\}<LB\_HD-\varepsilon_1 \quad (9)$$

$$\max\{\overline{BC_1},\overline{BC_2},\overline{BC_3}\}<LB\_HD-\varepsilon_1 \quad (10)$$

위의 트리밍조건들을 $BD(s)$의 원호 각각에 적용하면 $D(s)$에서 안전하게 제거될 수 있는 구간을 쉽게 찾아낼 수 있다. Fig. 17은 이러한 곡선 트리밍의 예를 보여준다.

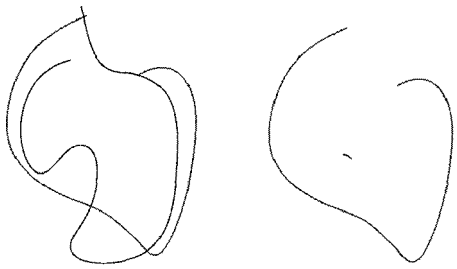


Fig. 17. Before and after the curve trimming.

4.3 방정식의 해

우선 곡선을 Hausdorff거리를 결정할 수 있는 중요한 후보구간들만 남기고 나머지는 트리밍하여 제거한다. 이렇게 트리밍 되고 남은 두 곡선을 각각 $C(t)$와 $D(s)$라고 하자. 두 곡선들 사이의 Hausdorff거리는 다음과 같이 세가지 경우들 중 한 가지에서 구해진다[14]. 첫 번째 경우는 곡선 끝점으로부터 다른 곡선 까지의 최단거리가 Hausdorff거리인 경우이다(Fig. 18(a)). 이를 방정식으로 표현하면 아래와 같다.

$$\begin{aligned}&<C(t_0)-D(s),D'(s)>=0 \ or\\&<D(s_0)-C(t),C'(t)>=0\end{aligned} \quad (11)$$

두 번째 경우는 두 곡선이 법선을 공유할 때이다(Fig. 18(c)와 18(d)). 이를 마찬가지로 방정식으로 표현해 보면 다음과 같다.

$$\begin{aligned}&<C(t)-D(s),C'(t)>=0,\ and\\&<C(t)-D(s),D'(s)>=0\end{aligned} \quad (12)$$

마지막 경우는 한 곡선이 다른 곡선의 self-bisector와 만나는 경우이다(Fig. 18(b)). 이를 마찬가지로 방정식으로 표현하면 다음과 같다.

$$\begin{aligned}&<C(t)-D(r),D'(r)>=0,\ and\\&<C(t)-D(s),D'(s)>=0,\ and\\&<C(t)-D(s),C(t)-D(s)>-<C(t)-D(r),C(t)-D(r)>\\&=0\end{aligned} \quad (13)$$

Hausdorff 거리를 가질 수 있는 후보 곡선들에 대해서 위의 (11), (12), (13)에 해당하는 해들을 모두 찾은 후 이 중에서 가장 거리가 큰 값을 택하여 정확한 Hausdorff 거리를 결정한다.

5. 실험 결과

본 실험은 2.40GHz Intel Pentium4 CPU와 2.0GB RAM의 시스템에서 수행되었다. 제 4.3절의 알고리즘에서 소개한 연립방정식을 푸는 부분은 IRIT solid modeling system의 multi-variate equation solver[1,2]를 이용하여 구현되었다.

Fig. 18은 이 실험에서 사용한 입력 곡선들을 각각 노란색과 파란색으로 나타낸 것이고 두 곡선간의 Hausdorff거리는 붉은 선분으로 표시하였다. Fig. 18(a), 18(b), 18(c), 18(d)의 곡선들은 모두 3차 B-Spline 곡선들이며 각 그림의 곡선들은 각각 5개, 10개, 15개, 30개의 control points를 가진다. Fig. 18(a)는 Hausdorff 거리가 곡선의 끝점과 다른 곡선의 내부 점에서 일어나는 경우를 보이고 있고, Fig. 18(b)는 Hausdorff 거리가 한 곡선의 self-bisector와 나머지 곡선의 교점에서 구해지는 경우를 보이고 있다. Fig. 18(c)와 18(d)는 일반적으로 Hausdorff 거리가 내부 점들 사이에서 구해지는 경우를 보이고 있다.

Table 1은 Hausdorff 거리를 Elber and Grandine[14]과 같이 트리밍 과정을 거치지 않고 처음부터 방정식들을 모두 푸는 방법으로 계산했을 때와 본 논문에서 제안한 알고리즘으로 트리밍을 하고 난 후에 남은 부분곡선분들에 대해서만 방정식을 풀었을 때의 계산시간을 비교하여 나타낸다.

실험 결과에서 알 수 있듯이 본 논문에서 제안한 알고리즘을 사용하면 속도의 향상을 적게는 5배에서

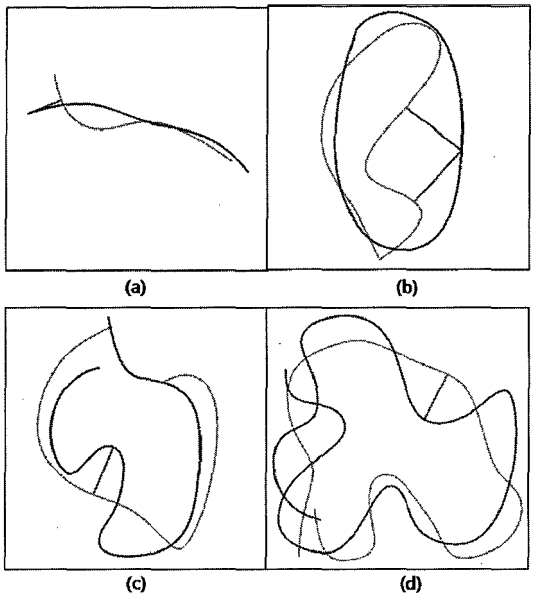


Fig. 18. Test examples.

Table 1. Computation time comparison with Elber and Grandine[14] (ms)

Control point #	Elber and Grandine[14]	Our Algorithm
5	2,953	146
10	10,156	1,572
15	15,502	1,306
30	35,725	2,215

많게는 20배까지 얻을 수 있음을 알 수 있다.

6. 결 론

본 논문에서는 평면곡선의 biarc근사와 이들의 circular cone으로 주어지는 거리함수의 근사값을 깊이 버퍼에 저장하여 Hausdorff 거리를 효율적으로 근사하고 이를 이용하여 Elber and Grandine[14]의 알고리즘을 가속화 시키는 기법을 소개하였다. 또한 이 기법을 실제 평면곡선들에 적용하여 알고리즘의 수행시간이 크게 개선되었음을 보였다.

향후 연구에서는 본 결과를 공간 곡선들과 곡면들에 대한 결과로 확장할 계획이다. 하지만 이러한 경우들에는 본 연구에서 사용한 평면곡선의 기하학적 특성에 기반을 둔 개념들 즉 biarc 근사와 이의 거리함수를 렌더링하여 깊이 버퍼에 저장하는 방식 등을 그대로 적용하기 어려워 이들을 대체할 수 있는 새로운 접근방법을 개발해야 할 것으로 보인다.

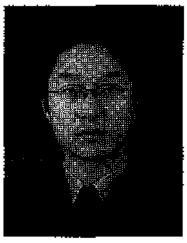
참고문헌

1. Elber, G. and Kim, M. S., "Geometric Constraint Solver using Multivariate Rational Spline Functions", *Proc of the Sixth ACM Symposium on Solid Modeling and Applications*, pp. 1-10, 2001.
2. Lennerz, C. and Schomer, E., "Efficient Distance Computation for Quadric Curves and Surfaces", *Proc. of Geometric Modeling and Processing*, pp. 60-69, 2002.
3. Sohn, K. A., Juttler, B., Kim, M. S. and Wang, W., "Computing Distances between Surfaces using Line Geometry", *Pacific Conference on Computer Graphics and Applications*, pp. 236-245, 2002.
4. Kim, K. J., "Minimum Distance between a Canal Surface and a Simple Surface", *Computer-Aided Design*, Vol. 35, No. 10, pp. 871-879, 2003.
5. Chen, X. D., Yong, J. H., Zheng, G. Q., Paul, J. C. and Sun, J. G., "Computing Minimum Distance between Two Implicit Algebraic Surfaces", *Computer-Aided Design*, Vol. 38, No. 10, pp. 1053-1061, 2006.
6. Chen, X. D., Chen, L., Wang, Y., Xu, G. and Yong, J. H., "Computing the Minimum Distance between Bezier Curves", *Journal of Computational and Applied Mathematics*, Vol. 230, No. 1, pp. 294-310, 2009.
7. Rabl, M. and Juttler, B., "Fast Distance Computation Using Quadratically Supported Surfaces", *Proc of Computational Kinematics (CK 2009)*", pp. 141-148, 2009.
8. Juttler, B., "Bounding the Hausdorff Distance of Implicitly Defined and/or Parametric Curves", *Mathematical Methods in CAGD*: Oslo 2000, pp. 223-232, 2001.
9. Cignoni, P., Rocchini, C. and Scopigno, "Metro : Measuring Error on Simplified Surfaces", *Computer Graphics Forum*, Vol. 17, No. 2, pp. 167-174, 1998.
10. Aspert, N., Santana, D. and Ebranhimi, T., "MESH: Measuring Errors between Surfaces using the Hausdorff Distance", *Proc. of the IEEE International Conference on Multimedia and Expo 2002 (ICME)*, Vol. 1, pp. 705-708.
11. Alt, H. and Scharf, L., "Computing the Hausdorff Diatance between Sets of Curves", *Proc. of the 20th European Workshop on Computational Geometry*, pp. 233-236, 2004.
12. Alt, H. and Scharf, L., "Computing the Hausdorff Distance between Curved Objects", *International Journal of Computational Geometry and Applications*, Vol. 18, No. 5, pp. 307-320, 2008.
13. Tang, M., Lee, M. and Kim, Y. J., "Interactive Hausdorff Distnace Computation for General Polygonal Models", *Proc. of SIGGRAPH 09, Computer graphics Annual Conference Series*, Article No. 74,

2009.
14. Elber, G. and Grandine, T., "Hausdorff and Minimal Distances between Parametric Freeforms in R^2 and R^3", *Lecture Notes in Computer Science: Advaces in Geometric Modeling and Processing*, Proc. of 5th Int. Conf. GMP 2008, Vol. 4975, pp. 191-204, 2008.
15. Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Oberneder, M. and Juttler, B., "Medial Axis Computation for Planar Free-Form Shapes", *Computer-Aided Design*, Vol. 41, No. 5, pp. 339-349, 2009.
16. Hoff, K., Culver, T., Keyser, J., Lin, M. and Manocha, D., "Fast Computation of Generalized Voronoi Diagrams Using Graphic Hardware", *Proc. of SIGGRAPH 99, Computer Graphics Annual Conference Series*, pp. 277-286, 1999.
17. Sir, Z., Feichtinger, R. and Juttler, B., "Approximating Curves and Their Offsets Using Biarcs and Pythagorean Hodograph Quintics", *Computer-Aided Design*, Vol. 38, No. 6, pp. 608-618, 2006.
18. Nutbourne, A. and Martin, R., "Differential Geometry Applied to Curve and Surface Design", Vol. I, Chichester, UK, Ellis Horwood, 1988.
19. Parkinson, D. and Moreton, D., "Optimal Biarc-curve Fitting", *Computer-Aided Design*, Vol. 23, No. 6, pp. 411-419, 1991.
20. Rossignac, J. and Requicha, A., "Piecewise Circular Curves for Geometric Modeling", *IBM Journal of Research and Development*, Vol. 31, No. 3, pp. 296-313, 1987.
21. IRIT 9.5 User's Manual, Technion. http://www.cs.technion.ac.il/~irit.

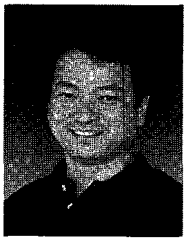
김 용 준

2008년 서울대학교 수학교육학과 학사
2008년~현재 서울대학교 전기컴퓨터공학부 석사과정
관심분야: 컴퓨터그래픽스, 기하모델링, 사용자인터페이스



오 영 택

2006년 서울대학교 화학생물공학부/컴퓨터공학부 학사
2009년 서울대학교 전기컴퓨터공학부 석사
2009년~현재 서울대학교 전기컴퓨터공학부 박사과정
관심분야: 컴퓨터그래픽스, 기하모델링, 사용자인터페이스



김 명 수

1980년 서울대학교 수학교육학과 학사
1982년 서울대학교 수학과 석사
1985년 Purdue University 응용수학 석사
1987년 Purdue University 전자계산학 석사
1988년 Purdue University 전자계산학 박사
1988년~1999년 포항공과대학교 교수
1999년~현재 서울대학교 컴퓨터공학부 교수
관심분야: 컴퓨터그래픽스, 애니메이션, 기하모델링