한국 CAD/CAM 학회 논문집
제8권 제4호 2003년 12월 pp. 243-253

학술논문

방향지도 기반 기하모핑의 효율적인 계산 및 제어 방법

이주행*, 김 현*, 김형선*

Efficient Computation and Control of Geometric Shape Morphing based on Direction Map

Lee, J.-H.*, Kim, H.* and Kim, H.-S.*

ABSTRACT

This paper presents a new geometric morphing algorithm for polygons based on a simple geometric structure called direction map, which is mainly composed of a circular list of direction vectors defined by two neighboring vertices of a polygon. To generate a sequence of intermediate morphing shapes, first we merge direction maps of given control shapes based on a certain ordering rule of direction vectors, and scale the length of each direction vectors using Bezier or blossom controls. We show that the proposed algorithm is an improvement of the previous methods based on Minkowski sum (or convolution) in th aspects of computational efficiency and geometric properties.

Key words : Geometric Shape Morphing, Direction Map, Polygon, Bezier and blossom controls

1. 서 론

기하 도형의 모핑은 주어진 두 제어도형의 모양을 보간하는 중간도형들을 생성하는 특별한 기하연산이다[1]. 일반적으로 모핑 연산의 결과는 모양이 연속적으로 변화하는 일련의 중간 도형들이다. 모핑의 대상이 되는 것은 점, 선분, 곡면 등의 기하학적인 특성과 색상 등의 이미지 정보가 있는데, 본 논문에서는 특히 2차원 다각형의 기하학적인 모양의 변화를 대상으로 한다.

일반적으로 모핑의 계산 방법은 두 단계로 구성되어 있다. 첫 번째 단계에서는 주어진 도형들 간에 대응되는 특징의 쌍을 찾는다. 이를 특징쌍 대응단계로 부른다. 두 번째 단계에서는 사용자가 선택한 보간법을 사용하여 대응되는 한 특징을 다른 특징으로 변환시킨다. 이를 특징 보간 단계라고 부른다. 첫 단계에서는 명시적으로 정의된 기하학적 (또는 수학적) 규칙을 이용하여 대응 되는 모든 특징쌍을 자동적으로 찾을 수도 있고, 사용자 상호작용을 통해 사용자가 직접 대응관계를 기술할 수 있다. 두 방법을 함께 사용하는 방법도 가능할 것이다. 두 번째 단계에서는 대응되는 특징쌍을 보간하는 중간특징을 생성하는 것인데, 그 결과는 대응관계가 올바르게 찾아 졌는지에 먼저 영향을 받으며, 구체적인 보간 방법의 선택에 따라 생성된 중간 도형의 모양이 결정된다.

본 논문에서 새롭게 제시하는 기하도형의 모핑 방법은 대응관계 설정과 보간 방법에 있어서 이전 연구[2,3]에서 제시한 민코프스키 덧셈 또는 콘볼루션에 근거한 기하 도형의 모핑 방법과 많은 유사점을 갖는다. 특히, 민코스키 덧셈 기반의 모핑의 효과를 정확히 재현함과 동시에 그 계산방법의 효율성과 기하학적인 특성 면에서 많은 개선을 이루었다.

본 논문에서는 방향지도(direction map)라는 간단한 기하학적인 자료 구조를 이용한다. 방향지도는 방향벡터라고 하는 벡터의 원형 배열로 이루어져 있다. 방향지도는 도형을 표현할 수 있는 편리한 방법이며 다양한 연산을 정의하고 그 연산의 결합을 수학적으로 표현 할 수 있다는 점에서 유용하다.

주어진 기하도형의 경계선이 다각형이라고 가정할 때, 먼저 도형의 경계선으로부터 방향지도를 추출하여, 두 방향지도를 병합(merge)하게 된다. 방향지도를 병합하는 것은 서로 다른 기하도형으로부터의 두

*한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 인터넷컴퓨팅 연구부, 분산협업기술연구팀
- 논문투고일: 2003. 02. 17
- 심사완료일: 2003. 07. 04

방향벡터의 기하학적인 관계를 고려하여 그 순서를 재배열하는 것인데, 이는 특징상 대응 단계에 해당한다. 본 논문에서는 사용자가 선택하거나 명시할 수 있는 기하학적 규칙으로 특징쌍을 대응시키고자 하였다.

두 번째 단계에서는 방향벡터의 길이를 재조절하여 새로운 경계선을 만들어 낸다. 생성된 새로운 경계선은 모핑에 의한 중간도형을 표현한다. 방향벡터의 길이를 재조절하는 것은 경계선에서 나타나는 특징형상의 크기 또는 정도를 조절하는 것이다. 특히, 다중 제어도형을 사용할 수 있도록 하여 베지어 곡선이나 블러섬(blossom)에서처럼 사용자 상호작용성을 높였다.

방향지도 기반 모핑 방법의 중요한 특징은 첫 번째 단계에서 특징쌍의 대응관계가 결정되면 그 결과를 두 번째 단계에서 재사용할 수 있다는 점이다. 이점은 기존의 민코프스키 덧셈 기반의 모핑 방법에서 고려하지 못한 주요한 성능 향상 요소이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 개선하고자 하는 민코프스키 덧셈 기반의 기하도형 모핑의 방법을 설명하고 문제점을 제시한다. 3장에서는 본 논문에서 제시하는 방법의 기본이 되는 방향지도와 이와 관련된 연산들을 정의하고 설명한다. 4장에서는 주어진 기하도형을 보간하는 구체적인 방법을 설명하며, 다중 제어 도형의 적용방법도 함께 설명한다. 마지막에서 결론에서 논문을 정리하고 추후 연구방향을 제시한다.

2. 관련 연구

2.1 민코프스키 덧셈과 콘볼루션

민코프스키 덧셈은 $\oplus$로 표기하며, 두 기하도형 A_0와 A_1에 대해서 다음과 같이 정의되어 새로운 기하도형 A를 생성한다.

$$A_0 \oplus A_1 = \{a_0 + a_1 | a_i \in A_i\} = A \tag{1}$$

위식에서 a_i는 기하도형 내부의 점을 나타내는 벡터이다. 민코프스키 덧셈 연산은 로봇의 충돌 없는 동작 경로의 생성이나 기하 도형의 모양의 변형을 위해서 사용될 수 있는 매우 유용한 연산이다[4]. 하지만, 두 도형의 대응 가능한 모든 점들의 쌍에 대한 벡터 덧셈으로 정의 되어 있어서 그 계산은 매우 복잡한 편이다[4,5].

이에 비해 콘볼루션은 도형의 내부가 아닌 경계선 상의 점에 대해서 정의가 되고 연산의 결과로 새로운 곡선 C를 생성한다.:

$$\partial A_0 * \partial A_1 \{a_0 + a_1 | a_i \in \partial A_i, T(a_0) // T(a_1), \langle T(a_0), T(a_1) \rangle > 0\} = C. \tag{2}$$

∂A_i는 도형 A_i의 경계선을, $\langle\ \rangle$ 연산은 벡터의 내적을, $T(a_i)$는 a_i에서의 접선벡터를 의미한다[5]. 콘볼루션에서 벡터의 덧셈은 접선벡터의 방향이 같은 두 점들에 대해서만 이루어진다. 이 조건을 콘볼루션의 접선일치조건이라고 한다. 콘볼루션 연산의 결과인 곡선 C는 ∂A_i의 형태적 특성에 따라, 자기 교차를 갖게 될 수도 있다.

민코프스키 덧셈 연산과 콘볼루션 연산은 다음과 같은 관계가 있음이 잘 알려져 있다[5]:

$$TRIM(\partial A_0 * \partial A_1) = \partial(A_0 \oplus A_1) \tag{3}$$

TRIM 연산은 곡선에 대한 트리밍을 의미한다. 즉, 콘볼루션 연산의 결과에 대해서 트리밍 연산을 통해 자기 교차를 제거하고 외각 경계선만 추출하면, 그 결과는 민코프스키 덧셈의 결과에 해당하는 기하 도형의 경계선과 동일하다. 실제로 이 관계를 이용하여 민코프스키 덧셈 연산을 계산하는 방법이 제시되었다[5].

민코프스키 덧셈 연산에서 피연산자 기하도형이 오목한 모양인 경우 다음의 비선형적인 특성이 있다[2].

$$\alpha A_0 + \beta A_0 \neq (\alpha + \beta) A_0. \tag{4}$$

α와 β는 임의의 실수이다. (피연산자가 모두 볼록한 기하도형이라면 선형성을 갖는다.) 이 성질에 의해 민코프스키 덧셈 기반 모핑의 계산식은 매우 복잡해진다. 이는 다음절에서 언급하기로 한다.

2.2 민코프스키 덧셈 기반의 모핑

민코프스키 덧셈 기반 모핑의 가장 간단한 방법은 다음과 같은 선형 보간 방법이다:

$$A(t) = (1-t)A_0 \oplus tA_1. \tag{5}$$

$A(t)$는 실수 인수 t에 대해서 두 도형 A_0와 A_1을 보간하는 중간 기하 도형을 나타낸다. 위 식이 두 실수 또는 두 벡터의 선형 보간법과 다른 점은 대수적인 덧셈 연산 대신 민코프스키 덧셈 연산을 적용했다는 점이다. 이 방법을 민코프스키 덧셈에 의한 선형보간(LIMS, linear interpolation by Minkowski sum)이라고 한다[2].

여기서 한 가지 주목할 점은 LIMS에서 대응 특징쌍의 결정방법이다. 앞 절에서 식 (3)의 민코프스키 덧셈과 콘볼루션의 관계에 의해, 민코프스키 덧셈의 경계선은 접선벡터의 방향이 같은 두 점들 사이의 대응

으로 결정된다는 것을 알 수 있다. 이 대응관계는 매우 간단하며 모든 대응관계를 사용자의 개입이 없이 찾을 수 있다. 하지만, 모양의 특징을 점 단위로 판단하는데 한계가 있다는 점에서 문제가 있다. LIMS의 중간 도형이 특징을 잃고 뚱뚱해지는 현상은 이와 관계가 있다.

베지어나 블러섬 곡선에서의 벡터 덧셈 연산을 민코프스키 덧셈 연산으로 교체할 수 있다. 이 방법은 LIMS와 유사하게 주어진 두 기하도형을 보간할 뿐만 아니라 복수개의 제어도형으로 중간 도형의 모양을 좀 더 정밀하고 유연하게 제어할 수 있도록 해 준다[2].

예를 들어 베지어 곡선 방식에 민코프스키 덧셈을 적용하면, 먼저 재귀적인 표현 방법에 의해 다음과 같이 표현된다.

$$A_i^r(t) = (1-t)A_i^{r-1}(t) \oplus tA_{i+1}^{r-1}(t). \tag{6}$$

위식에서 $A_i^0(t)$ 는 초기 제어도형 A_i이며, $r = 1, \ldots, n$, $i = 0, \ldots, (n-r)$일 때, $A_0^n(t)$ 는 재귀식의 최종단계로 보간 인수 t에서의 중간도형을 의미한다.

일반적으로 재귀식에 바탕을 둔 알고리즘의 계산 수행의 비효율성을 극복하기 위해 명시적인 형태로 전개된 등가식을 대신 사용한다. 예를 들어 위의 재귀식을 $n = 3$에 대해서 전개를 하면 다음과 같다.

$$A_0^3(t) = (1-t)^3 A_0 \oplus t(1-t)^2((A_1 \oplus A_1) \oplus A_1) \oplus t^2(1-t)((A_2 \oplus A_2) \oplus A_2) \oplus t^3 A_3 \tag{7}$$

위 식에서 주목할 점은 더 이상 간단한 식으로 전개가 불가능하다는 점이다. 즉, 민코프스키 덧셈의 비선형성에 의해 $(A_1 \oplus A_1 \oplus A_1)$ 가 항상 $3A_1$으로 간략화 될 수 없기 때문이다.

일반적으로 베지어 제어에서는 $\binom{n}{i}$개의 A_i가 민코프스키 덧셈으로 더해진다[2]. 제어도형이 복잡한 오목다각형일 경우 이것은 상당한 계산량을 필요로 한다는 것을 의미한다.* 전체 제어 도형의 개수가 많다면 더 큰 부담이 될 수 있다. 그런데, 여기서 중요한 문제는 반복해서 같은 제어도형을 더했을 때, 그 계산의 복잡도에 비해 발생하는 효과가 크지 않는다는 것이다. 반복적으로 더하는 것은 NURBS 곡선에서 knot을 삽입하는 것과 같이 제어 도형의 비중을 높이거나 그 제어도형으로의 수렴 속도를 빨리 할 수 있다. 하지만, 한 오목 다각형을 연속적으로 더하게 되면 경계선이 매우 복잡해지고 기하학적 특징이 사라지는 현상이 발생한다. 이는 콘볼루션에 의한 대응조건의 특성상 불가피한 현상이다. Fig. 1은 6각 별을 반복적으로 민코프스키 덧셈한 경우 이 현상이 발생하는 예를 보여준다.

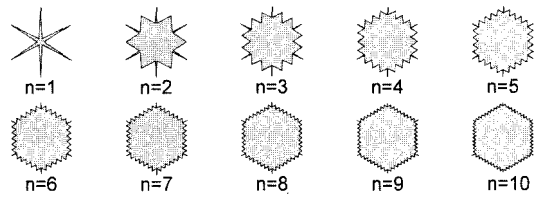


Fig. 1. 6각별 A에 대해서 $\frac{1}{n}(\oplus_{i=1}^{n} A)$.

만약, 주어진 제어도형이 모두 볼록 다각형이라면 선형성이 성립하기 때문에 간단한 표현이 가능하다.

$$\begin{aligned} A(t) &= \oplus_{i=0}^{n} B_i^n(t) A_i \\ &= B_0^n(t)A_0 \oplus B_1^n(t)A_1 \oplus \ldots \oplus B_n^n(t)A_n \\ &= A_0(t) \oplus A_1(t) \oplus \ldots \oplus A_n(t), \\ A_i(t) &= B_i^n(t)A_i. \end{aligned} \tag{8}$$

위식에서 $B_i^n(t)$ 는 i번째 n차 번슈타인 다항식이고, $A_i(t)$는 인수 t에서 번슈타인 다항식으로 크기조절을 한 i번째 제어도형이다.

하지만 선형성의 성립이후에도 계산상의 부담은 사라지지 않는다. 즉, 서로 다른 값의 t에 대해서 여전히 민코프스키 덧셈을 반복해야 한다는 것이다. 크기조절 연산은 매우 간단한 연산이므로 서로 다른 값의 t에 대해서 반복되어도 큰 계산량이 아니지만, 민코스키 덧셈을 t 값에 따라 계속 반복해야 한다는 것은 계산 부담이 상대적으로 크다.

블러섬은 베지어 제어의 일반적인 형태로 복수개의 보간인수가 사용되어 자유도가 높다[6]. 앞에서 설명한 분석은 블러섬에 대해서도 유사하게 적용된다. 즉, (1) 특징쌍 대응이 점 대응에 의존하므로 중간도형이 뚱뚱해 지는 현상, (2) 제어도형의 반복적인 덧셈의 효과가 상대적으로 적거나 오히려 좋지 않을 수 있다는 점, 및 (3) 계산량이 상대적으로 큰 민코프스키 덧셈 연산을 반복해서 계산해야한다는 것이다. 다음절에서 이러한 문제를 극복할 수 있는 방향지도 기반 기하도형 모핑 방법을 제시한다.

3. 방향지도

3.1 정의

방향지도(direction map)는 벡터들의 원형 배열

*4개의 제어도형이면 모핑을 표현하기에 가장 적합하다. 하지만, 제어도형이 모두 오목하다면 계산량이 매우 커서 펜티엄4 2GHz급 PC에서 실시간 수행을 기대하기 어렵다.

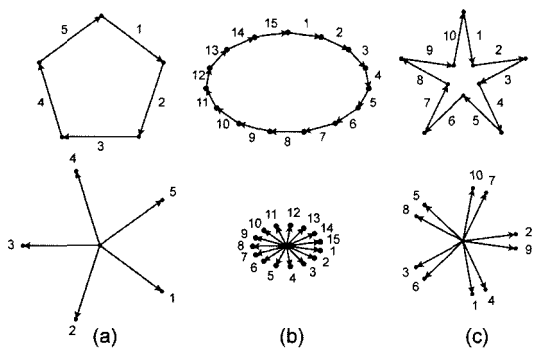


Fig. 2. 방향지도의 예.

(circular list)이다. (일반 벡터와의 구별을 위해 방향지도의 벡터를 방향벡터라고 부른다.) 어떤 다각형 A_i에 대해서 방향지도 D_i을 추출하는 것을 방향지도 연산 DM을 사용하여 다음과 같이 표기한다.

$$D_i = DM(\partial A_i). \tag{9}$$

다각형의 경우, 인접한 두 꼭짓점 p_k와 p_{k+1}는 하나의 방향벡터 $d_k = (p_{k+1} - p_k)$ 을 정의하며 경계선의 방위를 따라 순차적으로 원형 배열에 저장된다.

Fig. 2는 세 가지 도형(오각형, 15각 타원, 별)과 각 방향지도를 보여준다. 숫자는 선분의 시계방향 방위를 따른 순서와 그에 해당하는 방향벡터의 순서를 의미한다. 별의 방향지도에서 방향벡터의 실제 순서는 시계 방향을 따라 배열된 순서와 일치하지 않는다. 이는 오목한 특징형상 때문이다.

방향지도 D의 모든 방향벡터의 길이를 1로 하는 연산을 단위방향지도연산이라고 하고 $N(D)$으로 표기한다. 4절에서는 방향지도기반 모핑으로 생성된 중간도형의 단위방향지도의 불변에 대해서 설명한다.

방향벡터는 방향지도의 병합과 그룹단위 크기조절을 위해 그룹 식별자가 값을 갖는다. (이는 3.5절에서 보충 설명하기로 한다.) 그룹 식별자 값은 초기에 방향지도의 식별자 값으로 설정된다. 즉, D_i의 방향벡터 d_k의 그룹식별자 값은 i로 설정된다. 특별한 경우에는 방향지도의 식별자와 다른 값을 설정해야 하는데, 예를 들어, D_i의 모든 방향벡터에 j라는 그룹 식별자를 설정하는 경우를 $D_{i,j}$라고 표기한다. 특별한 언급이 없다면 D_i의 그룹 식별자는 i이다.

3.2 방향지도의 역연산

방향벡터 D가 다각형으로부터 정의되었다면, 방향지도로부터 원래의 다각형을 다시 생성하는 것은 쉽다. 즉, 방향벡터가 다각형의 경계선의 방위와 순서를 따라 생성된 것을 고려하여, 순차적으로 다각형의 꼭짓점을 생성할 수 있다.

$$p_0 = p_{init},$$
$$p_{k+1} = p_k + d_k. \tag{10}$$

위식에서 p_k는 다각형의 꼭짓점이고, p_{init}은 첫 꼭짓점의 실제 좌표로서 방향지도에 저장되는 추가적인 정보이다. p_{init}이 설정되지 않았을 때는 임의의 값을 사용할 수 있지만, 이 때는 다각형의 모양은 원래와 같이 복원되지만 그 절대위치가 달라진다. 이는 방향지도가 상대적인 위치를 표현하는 일련의 벡터로 구성되어 있기 때문이다.

위와 같이 방향지도로부터 다각형을 재구성하는 것을 방향지도의 역연산이라고 하고 다음과 같이 표기한다.

$$DM^{-1}(D) = DM^{-1}(DM(\partial A)) = \partial A. \tag{11}$$

이때 방향지도 D는 다각형 A을 나타낸다고 말한다.

방향지도의 역연산에 의해서 생성되는 경계선에 동일직선상의 선분이 연결되는 것을 방지하기 위해, 이웃한 동일방향의 방향벡터들을 모두 더해 하나의 벡터로 대체할 필요가 있다. 이 연산을 공선(共線, collinearity)제거라고 하고 CC로 표기한다. 다음에서 A의 경계선에는 인접한 선분들이 동일직선상에 놓이는 경우가 없다.

$$DM^{-1}(CC(D)) = \partial A. \tag{12}$$

3.3 방향지도의 병합

두 방향지도 $D_0 = DM(\partial A_0)$ 와 $D_1 = DM(\partial A_1)$ 의 방향벡터들을 어떤 규칙에 따라 혼합하고 재배열하여 얻는 방향지도 D는 새로운 도형을 표현할 수 있다. 이것을 방향지도의 병합이라고 하고, $\uplus$로 표기한다.

$$DM(\partial A_0) \uplus DM(\partial A_1) = D_0 \uplus D_1 = D. \tag{13}$$

병합된 방향지도의 역연산 결과에 트리밍을 통해 자기교차를 제거하면 새로운 도형 A의 경계선을 얻게 된다.*

$$TRIM(DM^{-1}(CC(D))) = \partial A. \tag{14}$$

이처럼 방향지도 병합은 새로운 도형을 만들 수 있는 중요한 방향지도 연산이다. 방향지도 연산과 3.5절의 그룹단위 크기조절 연산을 결합하여 4절에서는 방

*본 논문에서 제시하는 병합방법중 최소볼록집합 병합은 트리밍이 필요하지 않다.

향지도 기반의 모핑 알고리즘을 제시할 것이다.

방향지도 병합에서 가장 중요한 것은 두 방향지도로부터의 방향벡터를 혼합하고 재정렬 하는 방법이다. 다양한 방법이 가능한데, 다음절에서는 콘볼루션 병합과 최소볼록집합(convex hull) 병합방법을 소개한다.

3.3.1 콘볼루션 병합

콘볼루션 병합은 다음과 같은 결과를 유도하기 위한 특수한 병합방법이다.

$$DM^{-1}(DM(\partial A_0) \uplus DM(\partial A_1)) = DM^{-1}(D_0 \uplus D_1) = \partial A_0 * \partial A_1 \quad (15)$$

다른 병합방법과 구별하기 위해 콘볼루션 병합을 $\uplus_*$로 표기한다.

서로 다른 방향지도로부터의 방향벡터들이 혼합되어 재정렬 될 때, 다각형에 대한 콘볼루션의 접선일치 조건을 만족시키도록 한다. 다각형에서의 접선은 단일 벡터 대신 두 벡터에 의해 정의된다. 즉, 다각형 A_i의 k번째 꼭짓점 $p_{i,k}$가 정의하는 두 방향벡터 $d_{i,k} = (p_{i,k} - p_{i,k-1})$ 와 $d_{i,k+1} = (p_{i,k+1} - p_{i,k})$ 에 의한 접선구간으로 정의한다.

또한, 접선구간은 $d_{i,k}$에서 $d_{i,k+1}$로의 방위를 갖게 된다. 방위를 나타내는 식은 $d_k \times d_{k+1}$로 결정한다. 본 논문에서 도형의 경계선이 (구멍을 나타내지 않을 때) 시계방향이라고 가정하며, $p_{i,k}$가 국소적으로 볼록한 특징을 나타내는 경계상의 점일 때 접선구간의 방위는 0보다 작으며, 반대의 경우는 0보다 크다.

$d_{i,k}$에서의 접선구간을 간단히 $\angle T(p_{i,k})$라고 표기한다. $\angle T(p_{0,j})$와 $\angle T(p_{1,k})$의 방위가 같고 사잇각이 겹칠 때, $p_{0,j}$와 $p_{1,k}$는 콘볼루션의 접선일치조건을 만족시킨다고 한다. Fig. 3에서 두 접선구간의 콘볼루션 접선일치에 의해 병합된 방향지도에서 방향벡터는 다음의

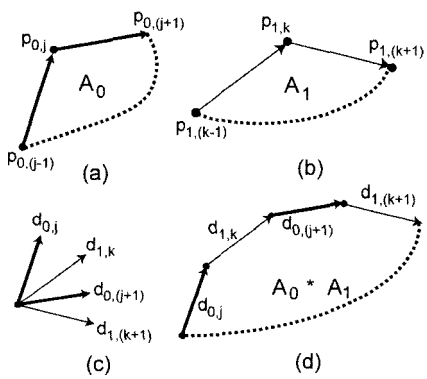


Fig. 3. 콘볼루션 병합의 적용 방법.

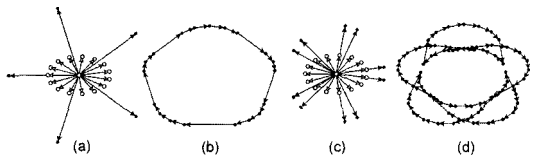


Fig. 4. 콘볼루션 병합의 예.

순서를 갖는다. $(d_{0,j}, d_{1,k}, d_{0,j+1}, d_{1,k+1})$

Fig. 4는 Fig. 2의 도형들에 대해서 (a) 오각형과 타원의 방향지도를 콘볼루션 병합하여 얻은 (b) 새로운 도형이다. (c)와 (d)는 타원과 별에 대해서이다. 콘볼루션의 특성상 오목한 특징과의 접선일치조건은 한 개 이상 발생할 수 있으므로, 한 방향벡터가 중복해서 삽입될 수도 있다. 이 때문에 (d)에서는 복잡한 자기교차가 발생한다.

콘볼루션 병합을 방향지도 기반 모핑에 사용하면, 식 (3)과 (15)로부터 다음과 같이 민코프스키 덧셈을 계산할 수 있다.

$$TRIM(DM^{-1}(D_0 \uplus_* D_1)) = TRIM(\partial A_0 * \partial A_1) = \partial(A_0 \oplus A_1). \quad (16)$$

3.3.2 최소볼록집합 병합

콘볼루션 병합을 사용하면 기존의 LIMS가 갖는 문제점이 그대로 발생한다. 즉, 비선형성에 의한 계산의 복잡함과 중간도형이 다소 뚱뚱해 지는 현상이다. 또한 드리밍이 필요하게 된다. 이 문제를 상당히 해결할 수 있는 간단한 방법은 여기서 소개하는 최소볼록집합 병합방법을 사용하는 것이다.

방향지도 $D = DM(\partial A)$ 에 대한 최소볼록집합 연산은 다음과 같이 정의 된다.

$$H(D) = H(DM(\partial A)) = DM(\partial H(A)). \quad (17)$$

표기상 편의를 위해 연산 H는 다형성(polymorphism)을 갖는다. 즉, 다각형에 대해서도 최소볼록집합을 나타낸다. 단, 방향지도에 대해서 연산을 적용한 후에는 기존의 방향벡터에 대한 정보를 잃지 않고 다음과 같이 역연산이 가능하도록 한다.

$$H^{-1}(H(D)) = D = DM(\partial A). \quad (18)$$

만약 최소볼록집합을 적용한 방향지도들을 병합한 경우에는 다양하게 역연산을 정의할 수 있다. Fig. 7은 역연산에 따라 생성되는 도형의 모습이 달라짐을 보여준다.

이러한 특성을 갖는 방향지도의 병합을 최소볼록집합 병합이라고 하고 다음과 같이 정의한다.

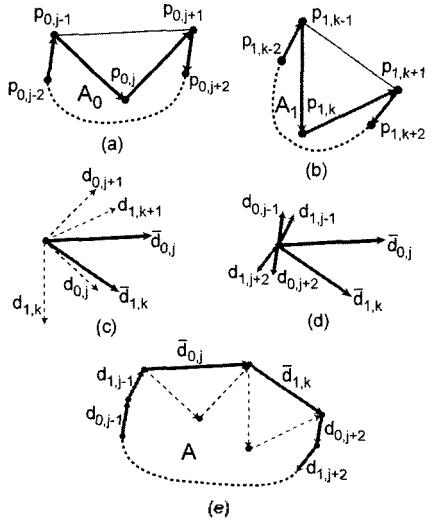


Fig. 5. 최소볼록집합 병합의 적용 방법.

$$D_0 \uplus_h D_1 = H^{-1}(H(D_0) \uplus_* H(D_1)). \quad (19)$$

특히, 자기병합(self-merge)에서 선형성이 유지되는 성질은 다음과 같이 표현할 수 있다.

$$CC(\alpha_0 D \uplus_h \ldots \uplus_h \alpha_n D) = (\alpha_0 + \ldots + \alpha_n)D. \quad (20)$$

Fig. 5에서 (a)와 (b)는 굵은 선으로 표현된 오목한 부분을 특징으로 갖고 있다. (c)와 (d)는 각각의 방향지도를 나타낸다. 방향지도에 대한 최소볼록집합 연산은 다각형의 최소볼록집합의 선분에 해당하는 방향벡터를 만들어 낸다. 예를 들어, (c)에서 원래방향벡터 $d_{0,j}$와 $d_{0,j+1}$ 대신 두 벡터의 합에 해당하는 $\bar{d}_{0,j}$가 생성된다. (d)에서도 유사하다. (c)와 (d)의 최소볼록집합 방향벡터들을 병합할 때는 (e)에서처럼 콘볼루션 병합을 수행하고, 최소볼록집합의 역연산에서 최소볼록집합으로 생성된 방향벡터를 원래 방향벡터들로 치환한다.

Fig. 6은 Fig. 2의 도형들에 대해서 (a) 오각형과 타원의 방향지도를 최소볼록집합 병합하여 얻은 (b) 새로운 도형이다. (c)와 (d)는 타원과 별에 대해서이다. 특히, (d)에서처럼 오목한 제어도형에 대해서도 자기교차가 발생하지 않기 때문에 트리밍이 필요 없다. 그럼에도 불구하고 두 도형의 특징이 모두 살아 있다.

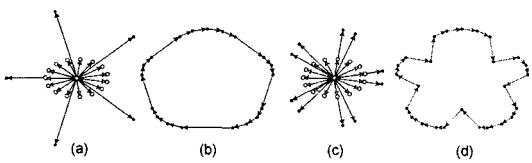


Fig. 6. 최소볼록집합 병합의 예.

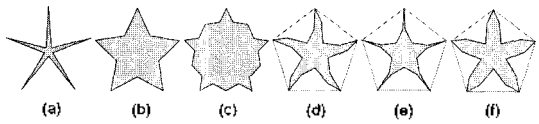


Fig. 7. 최소볼록집합 병합의 예.

Fig. 7은 두 도형 (a)와 (b)에 대한 병합의 결과를 보여준다. (c)는 콘볼루션 병합의 결과이므로 민코프스키 덧셈의 결과와도 같다. 중간 도형의 경계선이 이상하게 뚱뚱해짐을 볼 수 있다. (d)-(f)는 최소볼록집합 병합의 예로, 최소볼록집합 병합의 결과에 대한 역연산을 사용자가 다양하게 명시할 수 있음을 보여주며, 또한 주어진 두 도형의 특징형상이 잘 살아 있음을 볼 수 있다. (d)-(f)에서의 오각형 실선은 최소볼록집합을 의미한다.

최소볼록집합 병합은 혼합되는 특징형상이 콘볼루션 병합에서보다 명확하게 나타나는 점과 선형성을 유지할 수 있는 장점이 있지만, 역시 점 단위 대응 관계를 벗어날 수 없다는 한계가 있다.

3.4 그룹단위 크기조절

본 논문에서 제시하는 모핑 알고리즘에 주요하게 사용되는 연산으로 병합연산과 더불어 그룹단위 크기조절 (group scaling)연산이 있다. 그룹단위 크기조절은 *GS*로 표기하며, 병합된 방향지도와 일련의 스칼라 값을 인수로 하여 다음과 같이 정의된다.

$$GS(\uplus_{j=0}^{m} D_{*,j}, \{s_j\}_{j=0}^{m}) = GS(\uplus_{j=0}^{m} (s_j D_{*,j}), \{1\}_{j=0}^{m}) = \uplus_{j=0}^{m} (s_j D_{*,j}). \quad (21)$$

위식에서 $\uplus_{j=0}^{m} D_{*,j}$는 그룹 식별자 j을 갖는 방향지도들을 병합한 것이고, 수열 $(s_j)_{j=0}^{m} = \{s_0, s_1, \ldots, s_m\}$은 크기조절을 위한 스칼라 값이다. ($s_j$는 $D_{*,j}$에 곱해져서 그룹 식별자가 j인 방향벡터의 길이를 조절한다.)

4절에서 제시하는 알고리즘에서 사용하기 위해 그룹단위 크기조절 연산의 간단한 성질을 정리한다. 이는 식 (20)을 통해 쉽게 증명할 수 있다.

정리 3.4.1 (방향지도의 선형성) 볼록 다각형에 대한 콘볼루션 병합이나 임의 다각형에 대한 최소볼록집합 병합을 사용하는 경우 다음의 등식이 성립한다.

$$CC(GS(\uplus_{i=0}^{n} D_{*,i}, \{s_i\}_{i=0}^{n})) = \left(\sum_{i=0}^{n} s_i\right) D_0.$$

위 정리에 따르면, 선형성이 성립하는 경우의 n번

의 자기병합과 뒤이은 공선제거는 자기병합을 거치지 않고 크기를 n배로 하는 것과 같다.

4. 방향지도기반 모핑

4.1 방향지도에 의한 선형제어

방향지도 기반의 가장 간단한 모핑 방법으로, 3절에서 살펴한 방향지도와 관련 연산들을 이용하여 방향지도에 의한 선형보간(LIDM, linear interpolation by direction map) 알고리즘을 설명한다.

알고리즘 4.1.1 (LIDM) 두 제어 도형 A_0와 A_1와 보간 인수 t에 대해서, 다음과 같은 단계로 중간 도형 $A(t)$을 계산할 수 있다:

1. 각 제어 도형의 방향지도 계산: $D_i = DM(\partial A_i)$
2. 특정 병합 방법을 사용하여 두 방향지도를 병합: $D = D_0 \uplus D_1$
3. t에 초기값 설정
4. 그룹단위 크기조절을 통해 인수화된 방향지도를 계산: $\tilde{D}(t) = GS(D, \{1-(t,t)\})$
5. 공선제거: $\hat{D}(t) = CC(\tilde{D}(t))$
6. 방향지도의 역연산을 수행하여 경계선 생성: $\partial\hat{A}(t) = DM^{-1}(\hat{D}(t))$
7. 자기교차가 발생한 경우 트리밍을 수행: $\partial A(t) = TRIM(\partial\hat{A}(t))$
8. 또 다른 중간도형 생성을 위해서 새로운 t 값에 대해서 4단계(그룹단위 크기조절)부터 반복

위 알고리즘에서 4단계에서의 그룹단위 크기조절 연산의 중요한 역할은 병합과 크기조절을 분리한 것이다. 즉, 인수 t에 영향을 받지 않는 병합연산을 미리 계산하고 그 결과에 크기조절을 수행했다. 이 방법을 사용하면 모핑 중간도형을 여러 개 생성할 때 LIMS보다 계산이 효율적이다. 그 이유는 계산량이 상대적으로 큰 병합단계를 초기에 한번 계산하고, 그 결과를 상대적으로 계산량이 적은 크기조절 단계에서 반복하여 재사용할 수 있기 때문이다. 하지만 LIMS에서는 병합과 크기조절이 분리되지 않아 반복해야할 계산량이 더 크다.

식 (5)와 (16)을 이용하여 LIMS를 방향지도로 표현하면 다음과 같다.

$$\begin{aligned} D_0(t) &= DM(\partial((1-t)A_0)), \\ D_1(t) &= DM(\partial(tA_0)), \end{aligned}$$

$$\begin{aligned} \partial A(t) &= \partial((1-t)A_0 \oplus tA_1) \\ &= TRIM((1-t)\partial A_0 {}^{*t}\partial A_1) \\ &= TRIM(DM^{-1}(D_0(t) \uplus D_1(t))) \end{aligned} \quad (22)$$

위 식에 따르면 LIMS에서는 인수 t 값에 따라 크기가 달라진 도형 $A_i(t)$에 대해서 방향지도 $D_i(t)$의 병합연산을 (원하는 중간도형의 개수만큼) 반복해야 한다.

하지만, 병합의 기준이 되는 벡터의 방향은 크기조절에 의해서 바뀌지 않는다는 사실을 이용하여 알고리즘 4.1.1에서는 (생성하는 중간도형의 개수에 상관없이) 다음과 같이 한 번의 병합만을 필요로 한다.

$$\begin{aligned} \partial A(t) &= TRIM(DM^{-1}(GS(D_0 \uplus D_1), \{1-t, t\})) \\ &= TRIM(DM^{-1}(GS(D, \{1-(t,t)\}))) \end{aligned} \quad (23)$$

위 식에서 병합 과정은 인수 t와 독립적으로 이루어진다. (식 (22)와 (23)에서 CC 연산을 생략하였으나 설명하는데는 영향을 주지 않는다.) 따라서, 알고리즘 4.1.1의 LIDM은 LIMS에 비해 그 효율이 우수하다고 할 수 있다.

Fig. 8은 3각형과 8각형에 대한 LIDM 모핑의 결과를 보여준다. (a)는 위 알고리즘의 6단계 수행으로 얻은 일련의 모핑 중간도형들이고, (b)는 5단계에서 얻은 실제 방향지도이고, (c)는 (b)에 대한 단위방향벡터이다. (b)에서 볼 수 있듯이, 각 도형의 방향벡터는 그룹단위 크기조절 연산에 의해 크기가 조절됨을 볼 수 있다. 즉, 어떤 제어 도형 쪽으로 가까울수록 그 제어 도형에서 나온 방향벡터들의 길이가 상대적으로 길어진다. 하지만 각 방향벡터의 방향은 변하지 않기 때문에 (c)에서처럼 단위방향벡터는 불변이다.

Fig. 9는 귀자와 원모양사이의 LIDM 모핑의 결과를 보여준다. (a)는 콘볼루션 병합 결과에 트리밍을 적용하지 않은 것이고, (b)는 (a)에 트리밍을 적용한 예

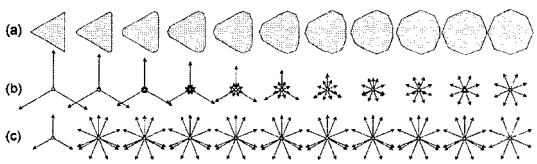


Fig. 8. 간단한 LIDM의 예.

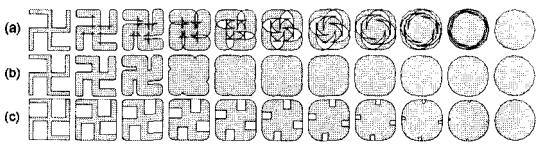


Fig. 9. 병합방법에 따른 LIDM 결과의 차이.

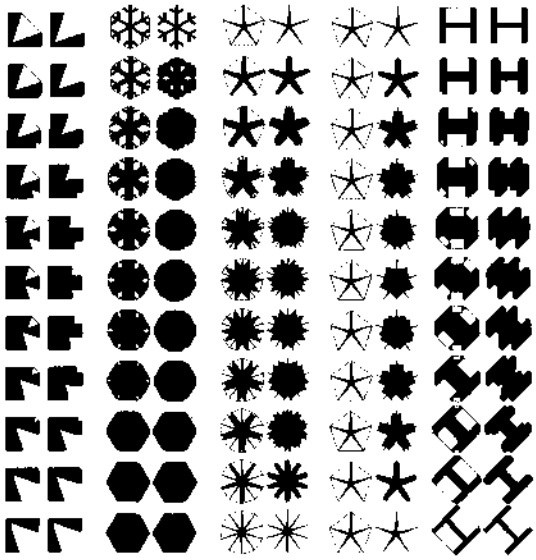


Fig. 10. 다양한 LIDM의 적용 예.

이고, (c)는 최소볼록집합 병합을 사용한 예이다. (b)는 콘볼루션 병합의 특성상 접선일치가 중복하여 발생하여 복잡한 자기교차 생기고 경계선이 뚱뚱해져서 제어도형의 특징형상이 일찍 사라져 버리는 현상이 있다.* 하지만, (c)에서는 두 도형의 특징형상이 전체 과정에서 잘 유지됨을 보인다. (오목한 특징형상 부위의 경계선에는 원모양의 특징이 적용되지 않은 것은 최소볼록집합 병합의 특성이다.)

Fig. 10은 콘볼루션 병합과 최소볼록집합 병합의 차이를 잘 보여준다. (a)에 V모양의 특징형상이 최소볼록집합 병합으로 보간된 경우 중간 단면에서 V 포켓 형상이 잘 유지된다. 하지만, (b) 콘볼루션 병합을 한 경우 V 특징형상 외에 L 형태의 특징형상이 발생한다. (c)에서 눈송이가 육각형으로 바뀌는 경우 최소볼록집합 병합을 사용한 경우와 (d)에서 콘볼루션 병합을 사용하여 특징형상이 너무 빨리 사라진 경우. (e) 최소볼록집합을 사용한 경우와 (f) 콘볼루션 병합으로 중간 도형의 경계선이 지나치게 복잡해진 경우. (g) 최소볼록집합 사용으로 선형성이 유지된 경우와 (h) 콘볼루션 병합으로 선형성이 성립하지 않는 경우. (i) H자가 회전한 경우로 이 경우의 회전에 대한 rigid body motion으로 가장 최적의 중간도형을 만들 수 있겠지만, 병합의 방법으로는 불가능하다. 최소볼록집합 병합을 수행한 경우 특징간의 간섭이 발생하여 자기교차제거에 의해 두개 이상의 도형이 발생할 수 있다. 이런 문제는 접 대응 방법의 한계이다. 이러한 문제의 해결을 위해서는 특징형상 단위 대응이 고려되어야 한다. (j) 콘볼루션 병합에서는 중간도형의 모양이 이상해지더라도 이런 현상은 발생하지 않는다.

*사용자의 인지나 기하학적인 규칙으로 이러한 현상을 발견하거나 예측하여, 재인수화를 통해 모핑 속도를 조절하여 문제를 해결할 수 있지만, 추가적인 계산의 부담이 따른다.

4.2 방향지도에 의한 베지어 제어

$(n+1)$개의 제어도형에 대해서 베지어 곡선의 재귀적 정의(de Casteljau 알고리즘)와 유사하게 LIDM 알고리즘을 일반화할 수 있다.

알고리즘 4.2.1 (베지어 LIDM) $(n+1)$개의 제어도형 A_i와 보간 인수 t에 대해서 중간 도형 $A(t)$을 다음과 같은 단계로 계산할 수 있다:

1. 각 제어 도형의 방향지도 계산:
   $D_i = DM(\partial A_i)$
2. t에 초기값 설정
3. 재귀적으로 $\tilde{D}(t) = D_0^n(t)$ 을 계산:
   $D_j^r(t) = (1-t)D_j^{r-1}(t) \uplus tD_{j+1}^{r-1}(t)$
   $= GS(D_{j,0}^{r-1}(t) \uplus D_{j+1,1}^{r-1}(t), \{1-(t,t)\})$
   위식에서 $D_{i,*}^0(t) = D_i$ 이고 $r = 1, \ldots, n$이고 $j = 0, \ldots, (n-r)$이다.
4. 알고리즘 4.1.1의 5-8단계를 수행.

위 알고리즘 3단계의 재귀적 수행의 비효율성을 개선하기 위해 다음과 같이 명시적인 방법으로 표현할 수 있다.

알고리즘 4.2.2 (베지어 LIDM의 빠른 계산) 알고리즘 4.2.1의 재귀적 3단계를 다음의 단계로 대체하면 빠른 계산이 가능하다.

1. 특정 병합 방법을 사용하여 자기병합:
   $\overline{D}_i = \uplus_{j=1}^{m_i} D_i, m_i = \binom{n}{i}$
2. 자기병합한 방향지도를 다시 병합:
   $\overline{D} = \uplus_{i=1}^{n} \overline{D}_i$
3. 그룹단위 크기조절을 통해 인수화된 방향지도를 계산: $\tilde{D}(t) = GS(\overline{D}, \{t^i(1-t)^{n-i}\}_{i=0}^{n})$

위 알고리즘에서 2단계의 병합된 방향지도는 3단계에서 재사용이 가능하다는 점에서 알고리즘 4.2.1에 비해 수행성능이 개선되었다고 할 수 있다. 1단계에서 자기 병합되는 방향벡터들은 모두 같은 그룹 식별

자를 갖게 된다. 이 그룹 식별자는 2단계 병합 후에도 유지가 되며, 이 그룹 식별자 값에 따라 3단계에서 크기 조절인수가 결정된다.

정리 3.4.1이 성립하는 경우에는 선형성을 이용하여 크기조절 인수에 변수타인 다항식을 사용할 수 있다. 이 경우 알고리즘 4.2.2의 첫 번째 단계와 같은 자기 병합을 하지 않아도 되므로 수행성능이 더욱 향상된다.

알고리즘 4.2.3 (베지어 LIDM의 더 빠른 계산) 정리 3.4.1이 성립하는 경우 알고리즘 4.2.1의 재귀적 3단계를 다음의 단계로 대체하면 더 빠른 계산이 가능하다.

1. 정리 3.4.1이 성립하는 병합방법을 사용하여 병합: $D = \biguplus_{i=1}^{n} D_i$
2. 변수타인 다항식을 이용하여 그룹단위 크기조절: $\tilde{D}(t) = GS(D, \{B_{i=0}^{n}(t)\}_{i=0}^{n})$

제어도형 및 병합방법의 특성에 따라 알고리즘 4.2.2과 4.2.3을 선택하여 일련의 모핑 중간도형 생성하면 민코프스키 덧셈 기반 방법(LIMS)에 비해 수행성능이 우수하다.

Fig. 11은 4개의 제어도형에 대해서 3차 베지어 LIDM을 적용한 결과이다. 모든 경우에 최소볼록 집합 병합을 사용하였다. 따라서 단 한번의 트리밍 연산도 없이 계산할 수 있었다. Fig. 12는 Fig. 11의 각 도형의 단위방향지도를 보여준다. 단 하나의 병합된 방향지도를 재사용하여 방향벡터의 크기만 조절하였다. 그러므로 모든 단위방향지도는 불변이다. 단, 크기조절 인수가 0인 경우의 방향벡터는 표현되지 않는다.

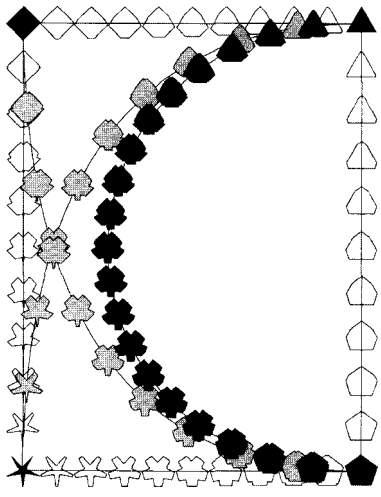


Fig. 11. 베지어 LIDM의 적용 예.

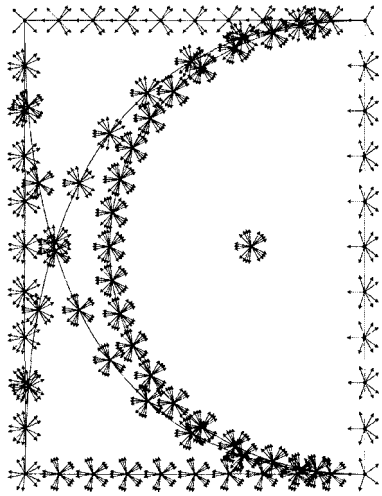


Fig. 12. Fig. 11의 단위방향지도.

4.3 방향지도에 의한 겹선형 제어

LIDM의 확장으로 겹선형(bilinear) 제어가 가능하다. 겹선형 제어는 두 독립인수 s, t에 대해서 다음과 같이 정의 된다.

$$\begin{aligned} D(s,t) &= (1-t)((1-s)D_0 \uplus sD_1) \\ &\quad \uplus t((1-s)D_2 \uplus sD_3) \\ &= (1-t)(1-s)D_0 \uplus (1-t)sD_1 \\ &\quad \uplus t(1-s)D_2 \uplus tsD_3 \\ &= \biguplus_{i=0}^{3} (\alpha_i(s,t)D_i) = GS(D, \{\alpha_i(s,t)\}_{i=0}^{3}), \\ D &= D_0 \uplus D_1 \uplus D_2 \uplus D_3 \end{aligned}$$

위 식을 베지어 LIDM과 유사하게 알고리듬에 적용

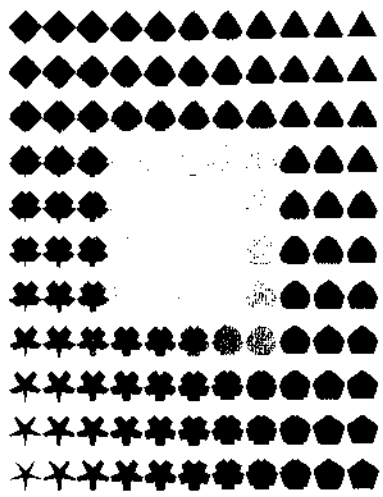


Fig. 13. 겹선형 LIDM의 적용 예.

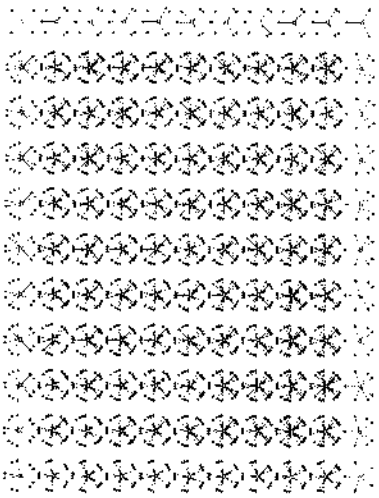


Fig. 14. Fig. 13의 단위방향지도.

할 수 있다.

Fig. 13은 Fig. 11과 같은 제어도형에 대해서 겹선형 제어를 수행한 결과이다. 베지어 제어로는 표현할 수 없는 도형도 표현이 가능함을 알 수 있다. 겹선형 LIDM에서도 방향지도를 재사용하기 때문에 Fig. 14에서처럼 단위방향지도는 불변이다. 더욱이 Fig. 13은 Fig. 11의 병합된 방향지도를 재사용할 수 있다. 즉, 베지어 또는 겹선형 등의 특정 보간 방법에 상관없이 방향지도 병합에 의한 특징쌍 대응의 결과는 재사용될 수 있다.

4.4 방향지도에 의한 블러섬 제어

블러섬(blossom)은 베지어 제어의 일반화된 형태로 $(n+1)$개의 제어점에 대해서 n개의 독립변수를 이용하는 다중선형(multiaffine) 제어방법이다[6]. 이전 연구에서 블러섬 제어를 민코프스키 덧셈에 적용 가능함을 보였다[3]. 베지어나 겹선형 LIDM처럼 블러섬 LIDM 제어 방법도 유사하다. (구체적인 방법은 지면 관계상 생략하기로 한다.) 방향지도의 특성상 블러섬 LIDM는 블러섬 LIMS보다 계산이 효율적이다.

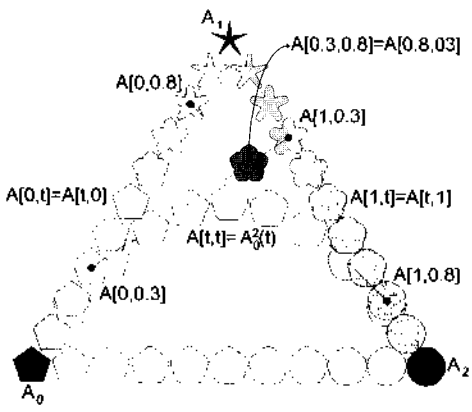


Fig. 15. 블러섬 LIDM의 적용 예.

Fig. 15는 블러섬 LIDM의 대칭성(symmetry)을 보여준다. 더불어, 2차 베지어 LIDM으로는 표현이 불가능한 $A[0.3, 0.8]$ 도형이 가능함을 보여준다.

5. 결 론

본 논문에서는 이전 연구에서 제시한 민코프스키 덧셈 또는 콘볼루션에 기반을 둔 기하도형의 모핑 방법을 개선한 방향지도 기반의 기하도형 모핑 방법을 제시하였다.

일반적으로 모핑 방법은 특징쌍 대응과 특징 보간에 의한 중간도형 생성으로 구성되어 있다. 기존의 방법에서는 이 두 단계가 섞여 있어서 수행성능에 문제가 되었다. 본 논문에서는 계산량이 상대적으로 큰 특징쌍 대응 단계를 방향지도 병합으로 계산하였고, 그 결과를 크기조절 시에 반복적으로 재사용할 수 있도록 함으로써 수행성능 문제를 해결하였다.

또한, 민코프스키 덧셈 기반 모핑은 콘볼루션 접선 일치조건에 의한 점 대응관계 설정으로 인해 중간도형이 다소 뚱뚱해지고, 오목한 특징형상이 있는 제어도형의 반복적인 덧셈에서 선형성이 보장되지 않는 단점이 있다. 본 논문에서 제시한 콘볼루션 병합 방법은 민코프스키 덧셈에 의한 기존의 방법과 같은 효과를 낼 수 있으며 그 계산은 더 효율적이다. 최소볼록 집합 병합은 중간도형의 특징이 잘 유지되고 선형성이 보장되도록 하여 콘볼루션 병합이나 민코프스키 덧셈 기반 방법의 문제점을 개선할 수 있다. 그 외에도, 병합 단계의 방향벡터 정렬 규칙을 사용자가 다양하게 명시할 수 있도록 함으로써 사용자 상호작용성을 높이고 다양한 모핑 효과를 기대할 수 있다.

추후 연구로 방향지도를 3차원 다면체 및 2차원 곡선에 적용할 계획이다. 또한, 점 대응 기반의 특징쌍 대응을 특징형상 기반으로 확장할 필요가 있다.

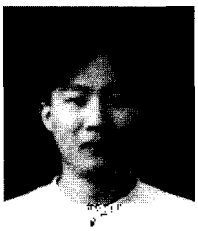
감사의 글

본 논문은 2002년 정보통신부의 "협업적 제품거래 기술 개발" 과제 및 2002-2003년 산업자원부의 "웹 기반 제품정보 통합 관리기술 개발" 과제의 지원을 받아 수행된 연구에 기초하고 있습니다. 본 논문은 2003

한국 CAD/CAM 학회 학술발표회에 같은 제목으로 발표된 저자들의 논문[8]을 확장하였습니다.

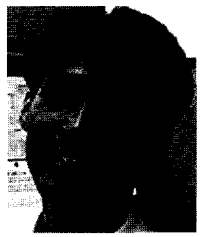
참고문헌

1. Alt, H. and Guibas, L. J., Discrete Geometric Shapes: Matching, Interpolation, and Approximation in Handbook of Computational Geometry, eds. J.-R. Sack and J. Urrutia (Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999). pp. 121-153.
2. 이주행, 이재열, 김현, 김형선, "민코프스키 덧셈 연산에 근거한 기하노형의 모핑 제어 방법," 한국 CAD/CAM 학회 논문집, Vol. 7, pp. 269-279, 2002.
3. Rossignac, J. and Kaul, A., "AGRELs and BIPs: Metamorphosis as a Bezier Curve in the Space of Polyhedra," EUROGRAPHICS '94, M. Dæhlen and L. Kjelldhal (Eds.), Blackwell Publishers, pp. C179-C184, 1994.
4. 이주행, "앤빌롭 및 집합연산에 근거한 일반스위의 경계선 생성," 박사 학위 논문, 포항공과대학교, 1999.
5. Lee, I.-K., Kim, M.S. and Elber., G., Polynomial/Rational Approximation of Minkowski Sum Boundary Curves. GMIP, Vol. 60, No. 2, pp. 136-165, 1998.
6. G. Farin. Curves and Surfaces for CAGD, 5th edition, Academic Press, 2002.
7. 이주행, "Direction Map: Properties and Applications," Technical Report, ETRI (http://www.etri.re.kr), 2002.
8. 이주행, 이재열, 김 현, 김형선, "방향지도 기반 기하모핑의 효율적인 계산 및 제어 방법," 2003 한국 CAD/CAM 학회 학술발표회 논문집, pp. 371-380, 2003.



이 주 행

1994년 포항공과대학교 전자계산학과 학사
1996년 포항공과대학교 전자계산학과 석사
1999년 포항공과대학교 전자계산학과 박사
1999년~현재 한국전자통신연구원 인터넷 컴퓨팅연구부 분산협업기술연구팀 선임연구원
관심분야: Geometric Modeling and Processing, Computer-Aided Design, Computer Graphics, Virtual Reality, Information Visualization, Distributed Computing



김 현

1984년 한양대학교 기계설계학과 학사
1987년 한양대학교 기계설계학과 석사
1997년 한양대학교 기계설계학과 박사
1998년~1999년 한양대학교 산업공학과 겸임교수
1900년~현재 한국전자통신연구원 분산협업기술연구팀장, 책임연구원
관심분야: Concurrent Engineering, Virtual Engineering, Distributed Collaborative Design, Internet-enabled CAD, Engineering Design Process, Engineering Knowledge Management

김 형 선

1982년 상지대학교 경영학과 학사
1992년 광운대학교 컴퓨터공학과 석사
2001년~현재 대전대학교 컴퓨터공학과 박사과정
1985년~현재 한국전자통신연구원 분산협업기술연구팀 책임연구원
관심분야: 분산컴퓨팅, Collaborative Product Commerce, 정보보호, 분산데이터베이스