한국 CAD/CAM 학회 논문집
제 7권 제 4 호 2002년 12월 pp. 269-279

학술논문

# 민코프스키 덧셈 연산에 근거한 기하 도형의 모핑 제어 방법

이주행*, 이재열*, 김 현*, 김형선*

# Interactive Control of Geometric Shape Morphing based on Minkowski Sum

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

**ABSTRACT**

Geometric shape morphing is an interesting geometric operation that interpolates two geometric shapes to generate in-betweens. It is well known that Minkowski operations can be used to test and build collision-free motion paths and to modify shapes in digital image processing. In this paper, we present a new geometric modeling technique to control the morphing on geometric shapes based on Minkowski sum. The basic idea develops from the linear interpolation on two geometric shapes where the traditional algebraic sum is replaced by Minkowski sum. We extend this scheme into a Bézier-like control structure with multiple control shapes, which enables the interactive control over the intermediate shapes during the morphing sequence as in the traditional CAGD curve/surface editing. Moreover, we apply the theory of blossoming to our control structure, whereby our control structure becomes even more flexible and general. In this paper, we present mathematical models of control structure, their properties, and computational issues with examples.

***Key words*** : Morphing, Minkowski sum, Bézier curve and surface, Blossom

## 1. Introduction

Shape morphing is a special geometric operation that interpolates two geometric shapes to generate in-between shapes. Morphing is also known as metamorphosis, shape blending, and shape interpolation[3]. The result of morphing operation can be a single instance of intermediate shape or a sequence of in-between shapes. Morphing is a well-known technique in computer graphics. In many computer graphics applications, morphing interpolates not only geometric entities (such as vertices, lines, and surfaces) but also color information (such as vertex color and texture map). Our morphing method does not aim at an image morphing (or wrapping) for which better solutions exist[15].

Many previous researches have proposed efficient morphing methods[5,7,14] and some of them are implemented in commercial design systems such as Adobe Illustrator[1]. However, in most of the previous works, a control structure is not proposed for an interactive design system. Furthermore, it is rarely known to interpolate multiple shapes at once (We do not deal with detailed issues of other morphing techniques since they are out of the scope of this paper.).

The classic application of Minkowski sum is to test and build collision-free motion paths[10,11] and to modify shapes in digital image processing[8].

The possibility of morphing based on Minkowski sum can be found in some previous works[7,10,3]. The basic idea develops from the linear interpolation on two geometric shapes; however, we replace the traditional algebraic sum into Minkowski sum. The idea is very straightforward, but its property is not well analyzed in previous works. In this paper, we clearly describe the underlying mathematical models of Minkowski sum morphing, some of

*한국전자통신연구원 통사공학연구팀
- 논문투고일: 2002. 07. 09
- 심사완료일: 2002. 09. 23

their properties, and computational issues. In addition, we propose a new method to control the intermediate shapes of a morphing sequence. Our proposed method has a Bézier-like control structure with multiple control shapes, which enables the interactive control over the intermediate shapes as in the interactive drawing systems (The concept of control shape is generalized in Jüttler *et al.*[9]). Moreover, the proposed control structure is further generalized into a Blossom form.

The remainder of this paper is organized as follows. In section 2, we briefly describe the definition of Minkowski sum of geometric shapes and its properties. Section 3 introduces the concept of LIMS (Linear interpolation by Minkowski sum) and its application to geometric shape morphing. In section 4, we propose Bézier-like control structure based on LIMS, and deal with some of its properties such as the blossom and computational issues. In section 5, we present some experimental results. In the final section, we conclude this paper.

## 2. Minkowski Sum

### 2.1 Preliminaries

Let $\mathbb{E}^3$ and $\mathbb{R}^3$ be three-dimensional affine (or point) and linear (or vector) space, respectively. We denote a vector or a point by a lowercase boldface letter. A vector or a point is composed of three coordinate values: $\boldsymbol{a} = (a_x, a_y, a_z)$.

In this paper, a *geometric shape* in affine space is a set of points. We denote a geometric shape by an uppercase boldface letter: $\boldsymbol{A} = \{\boldsymbol{a} | \boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{a} \in \mathbb{E}^3\}$. In this paper, the center of a geometric shape does not mean the centroid or center of gravity. Rather it means a certain fixed point contained in a geometric shape.

Multiplication by a scalar number $s$ is defined on a geometric shape as follows:

$$s\boldsymbol{A} = \{s\boldsymbol{a} | s\boldsymbol{a} = (sa_x, sa_y, sa_z), \boldsymbol{a} \in \boldsymbol{A}\}.$$

It can be interpreted as a special kind of scaling operation. Translation and scaling of a geometric shape are defined using Minkowski sum in the next section.

### 2.2 Definition

For two geometric shapes A and B, *Minkowski sum* (or *Minkowski addition*) is denoted by $\oplus$ and defined as a binary operation as follows:

$$\boldsymbol{A} \oplus \boldsymbol{B} = \{\boldsymbol{a} + \boldsymbol{b} | \boldsymbol{a} \in \boldsymbol{A} \text{ and } \boldsymbol{b} \in \boldsymbol{B}\}. \tag{1.1}$$

Addition of points is not a well-defined operation. However, we can interpret the point addition in the above definition as a special vector sum which generates a point $\boldsymbol{c}$:

$$\boldsymbol{a} + \boldsymbol{b} \equiv (\boldsymbol{a} - \boldsymbol{0}) + (\boldsymbol{b} - \boldsymbol{0}) = \boldsymbol{c} + \boldsymbol{0}.$$

where $\boldsymbol{0}$ denote a zero point or an origin of the world: $\boldsymbol{0} = (0, 0, 0)$. Hence, based on geometric interpretation, the definition of Minkowski sum of equation (1.1) is refined as follows:

$$\begin{aligned} \boldsymbol{C} &= \boldsymbol{A} \oplus \boldsymbol{B} \\ &= \{\boldsymbol{c} | (\boldsymbol{c} + \boldsymbol{0}) = (\boldsymbol{a} - \boldsymbol{0}) + (\boldsymbol{b} - \boldsymbol{0}), \boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{b} \in \boldsymbol{B}\} \end{aligned} \tag{1.2}$$

### 2.3 Properties

The numbers of elements of Minkowski sum operands may not be equal: $|\boldsymbol{A}| \neq |\boldsymbol{B}|$. The number of elements of Minkowski sum result is bounded as follows:

$$|\boldsymbol{A} \oplus \boldsymbol{B}| \leq |\boldsymbol{A}| \cdot |\boldsymbol{B}|$$

since, for example, $\boldsymbol{a}_i + \boldsymbol{b}_l = \boldsymbol{a}_j + \boldsymbol{b}_m$ where $\boldsymbol{a}_i \neq \boldsymbol{a}_j$, $\boldsymbol{b}_m \neq \boldsymbol{b}_l$ and $\boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{b} \in \boldsymbol{B}$.

Note that the number of elements is meaningful only for a discrete set, but not for a continuous set as a geometric shape with volume or area.

In some cases, an operand set may be an empty set. We define that Minkowski sum with an empty set leaves the result empty:

$$\boldsymbol{A} \oplus \{\} = \boldsymbol{A} + \phi = \phi.$$

An empty set is interpreted as a null or erased geometric object in Boolean operations[11].

The *identity* of Minkowski sum is $\{\boldsymbol{0}\}$:

$$\boldsymbol{A} \oplus \{\boldsymbol{0}\} = \{\boldsymbol{0}\} \oplus \boldsymbol{A} = \boldsymbol{A}.$$

However, there is no proper *inverse* set which is not empty.

A point (or a vector) $\boldsymbol{p}$ in $n$-dimension is represented by $n$ scalar components $p_i$: $\boldsymbol{p} = (p_1, \ldots, p_n)$. The sum operation on two points (or a vector sum) is defined by scalar sums on each corresponding scalar component. Hence, Minkowski sum on two sets of points in any dimension is *commutative*:

$$A \oplus B = B \oplus A$$

In addition, Minkowski sum is associative:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

since an element in one set is added to every element in the other set.

It is an interesting property of Minkowski sum that the following simple equality does not hold for some geometric shapes such as non-convex ones:

$$A \oplus A = 2A$$

which may restrict the evaluation and computation of Minkowski sum morphing. (See section 4.) For example, this equality holds for a convex triangle A, but not for a simple star shape B (See Fig. 1.) If the shape is convex and has no internal hole, it is clear that the equality holds. Otherwise, it does not. (Generally, for two real numbers $\alpha$ and $\beta$, $\alpha A \oplus \beta A \neq (\alpha+\beta)A$ ). This property mainly inherits from the relationship between the convolution and Minkowski sum: The boundary curve of Minkowski sum of non-convex geometric shapes is computed as a trimmed convolution curve between them. Trimming is not required between two convex shapes[12].

Note that, in Fig. 1, dashed lines represent convex hulls of $2B$ and $B \oplus B$, which are two identical pentagons in this case. When $H_B$ denotes the convex hull of $B$, we can easily show that there exists a following *containment relation*:

$$2B \subseteq B \oplus B \subseteq 2H_B .$$

The key idea of proof is to show that $B \oplus B$ includes addition between every pair of points $B$ of whereas $2B$ includes addition between the same points.

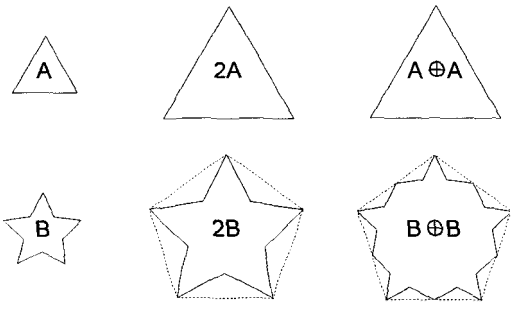



**Fig. 1.** $A \oplus A = 2A$ , but $B \oplus B \neq 2B$.

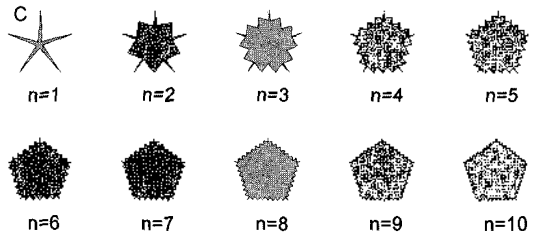



**Fig. 2.** Convergence property of self-addition.

Moreover, for the repeated self-addition of $B$, we can expect the following *convergence property*:

$$\lim_{n\to\infty} \frac{1}{n}\left( \bigoplus_{i=1}^{n} C \right) = H_C \tag{1.3}$$

where we define $\oplus_i^n$ as a following substitute of $\sum_i^n$:

$$\bigoplus_{i=1}^{n} A_i = \overbrace{A_1 \oplus A_2 \oplus \ldots \oplus A_n}^{n}$$

Fig. 2 shows an example of this convergence property: the given shape is a 5-star (say $C$) and each shape represents the result of $\frac{1}{n}\left( \bigoplus_{i=1}^{n} C \right)$ where $1 \le n \le 10$. As $n$ increases, the results converge on a pentagon which is the convex hull 5-star.

A barycentric combination of geometric shape using Minkowski sum is similarly defined as in affine point space. Based on definition of equation (1.2), a barycentric combination of $n$ geometric shapes is defined as follows:

$$A = \bigoplus_{i=1}^{n} \alpha_i A_i = \alpha_1 A_1 \oplus \alpha_2 A_2 \oplus \ldots \oplus \alpha_n A_n,$$
$$\sum_{i=1}^{n} \alpha_i = 1 \tag{1.4}$$

where $A_i$ is a geometric shape and $\alpha_i$ is a real number.

An affine map $\Phi$ on a geometric shape has a following form:

$$\Phi A = M \cdot A + v,$$

where $M$ is a 3×3 transformation matrix and $v$ is a vector in $\mathbb{R}^3$. We assume that the matrix multiplication on a geometric shape is applied to every point included in that shape. Likewise, the vector

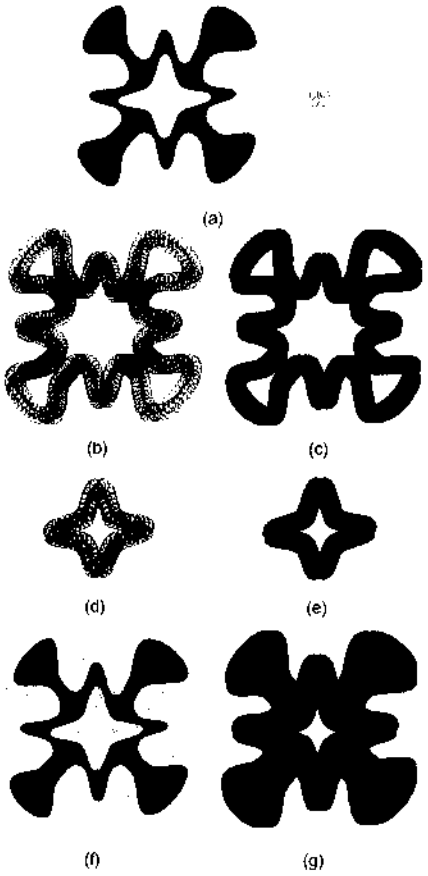



**Fig. 3.** An example of Minkowski sum computation.

sum is applied to every point. Hence, it is clear that any affine map defined in $\mathbb{E}^3$ can be applied to geometric shapes composed of points in $\mathbb{E}^3$. Moreover, a barycentric combination of geometric shapes is affine invariant.

When there is such relation as $v = p - 0$ (a vector $v$, a point $p$, and the origin $0$), the vector sum on a geometric shape can be expressed using Minkowski sum as $A + v = A \oplus \{p\}$. This is the translation of a geometric shape $A$ by the vector $v$.

### 2.4 Computation

The computation of Minkowski sum on geometric shapes is a relatively complex CAGD problem. Most of the computation methods are based on shape boundaries or convolution curves since it is redundant to compute on internal points[2,10,11].

Note that our morphing method is not dependent on a specific method for Minkowski sum computation. However, a certain method may increase the computation efficiency significantly[12].

Fig. 3 shows an illustrative example of Minkowski sum computation on two geometric shapes based on the method of Lee *et al.*[11]: (a) a flower and a butterfly; (b) sweep of a butterfly along the outer boundary of the flower; (c) the sweep boundary of (b); (d) sweep of a butterfly along the hole boundary of the flower; (e) the sweep boundary of (d); (f) union of the flower and (c) and (e); and (g) the result of Minkowski sum.

## 3. Minkowski Sum Morphing

This section describes the concept of *LIMS* (*Linear interpolation by Minkowski sum*) and its application to geometric shape morphing.

The basic idea develops from the linear interpolation (LI) on two geometric shapes, where the traditional algebraic sum $\oplus$ is replaced by Minkowski sumsince algebraic sum is not defined on sets. For given two geometric shapes $A$ and $B$, LIMS is defined as follows:

$$(1-t)A \oplus tB$$

where $t$ is the interpolation parameter. LIMS is a special case of barycentric combination of geometric shapes of equation (1.4). Let $C(t)$ represent LIMS parameterized by $t$:

$$C(t) = (1-t)A \oplus tB .$$

From the viewpoint of morphing, $C(t)$ at a certain value of $t$ denotes one of the intermediate morphing instances. Specially, when $t$ is 0 or 1,

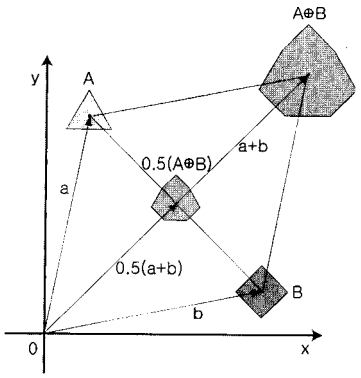



**Fig. 4.** LIMS and linear interpolation.

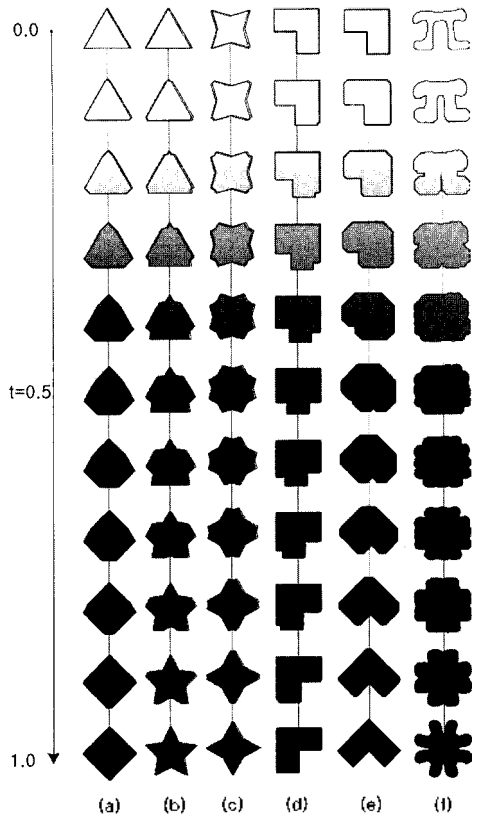



**Fig. 5.** Some examples of LIMS morphing.

$C(t)$ is identical to $A$ or $B$:

$$C(0) = 1 \cdot A \oplus 0 \cdot B = A \oplus \{\mathbf{0}\} = A,$$
$$C(1) = 0 \cdot A \oplus 1 \cdot B = \{\mathbf{0}\} \oplus B = B.$$

Fig. 4 shows an example of LIMS on a triangle $A$ and a square $B$, and linear interpolation between the centers of $A$ and $B$. Let $a$ and $b$ be vectors from the origin to the center of $A$ and $B$, respectively. $A \oplus B$ is Minkowski sum of $A$ and $B$, and $a + b$ is the vector sum of $a$ and $b$. We assume that the center of $A \oplus B$ be located at $a + b$. The linear interpolation between $a$ and $b$ is $(1-t)a + tb$, where the center of every LIMS instance is located. For example, the center of $0.5(A \oplus B)$ is located at $0.5(a + b)$.

Fig. 5 shows some examples of LIMS morphing between two geometric shapes (the top white and the bottom dark grey): (a) a pair of convex shapes, (b) a convex and a concave shapes, (c) rotated stars, (d) a pair of flipped shapes, (e) rotated non-symmetric shape, and (f) complex shapes (a Phi and a flower.)

In Fig. 5(c)-(e), the first shape has been transformed to the second via rotation or flip. Hence, the intermediate morphing instance may be one from appropriate transformation. However, LIMS morphing does not generate such kind of transformation. That kind of transformation is not the target of LIMS morphing operation. Some method may support similar feature in the specific situation but users have to specify the correspondences between geometric shapes manually[11].

In Fig. 5, the intermediate shapes seem somewhat "fatter" than two control shapes. This can be explained by the set-theoretic definition of the Minkowski sum and relationship to sweep operation. Minkowski sum can be defined as translational sweep operation between two control shapes. When one control shape sweep along the boundary of the other control shape, the resulting sweep boundary participate in the Minkowski sum boundary. Moreover, this boundary is a generalized offset of the other shape[11]. Hence, a non-convex boundary (i.e., a pocket) of the other control shape is rounded to form a fatter shape. This is one of the unique features of Minkowski sum morphing.

## 4. Control Structure

In this section, we propose a new control method which has a Bézier-like control structure with multiple control shapes. The proposed method enables interactive design and control over the intermediate shapes during the morphing as in the traditional CAGD techniques. In the following sections, we will describe the proposed method in detail. First, the simplest case of quadratic control is explained in the next section.

### 4.1 Quadratic Control

#### 4.1.1 Mathematical Model

Let $A_0$ and $A_2$ be start and end control shapes, and we will generate their in-betweens using *quadratic interpolation by Minkowski sum (QIMS)*. First, when using LIMS morphing of section 3 for $A_0$ and $A_2$, the generated sequence $C_0^0(t)$ is simply parameterized as follows:

$$C_0^0(t) = (1-t)A_0 \oplus tA_2 .$$

In QIMS, an additional control shape is required, which will affect the LIMS morphing between $A_0$ and $A_2$. Let $A_1$ be the new control shape. The LIMS morphing sequence from $A_0$ to $A_1$ and from $A_1$ to $A_2$ are $C_0^1(t)$ and $C_1^1(t)$, respectively:

$$C_0^1(t) = (1-t)A_0 \oplus tA_1,$$

$$C_1^1(t) = (1-t)A_1 \oplus tA_2.$$

Now, QIMS morphing is recursively defined as LIMS between $C_0^1(t)$ and $C_1^1(t)$, which has a de Casteljau-style parametric form as follows:

$$C_0^2(t) = (1-t)C_0^1 \oplus tC_1^1(t). \tag{4.1}$$

4.1.2 Computation

Since equation (4.1) is in a recursive form, we need to derive an explicit formulation for actual computation of QIMS. Before deriving this, we need to compare QIMS and a quadratic Bézier curve. Let $a_i$ be the center position of $A_i$. Note that the center of each intermediate shape of $C_0^2(t)$ is on the quadratic Bézier curve $a_0^2(t)$, which can be expressed either recursively or explicitly as follows:

$$a_0^2(t) = (1-t)a_0^1(t) + ta_1^1(t) = \sum_{i=0}^{2} a_i B_i^2(t)$$

where $B_i^2(t)$ denotes the Bernstein polynomial of degree 2[7].

Using the Bernstein polynomial, we can rewrite the above equation more explicitly as follows:

$$a_0^2(t) = (1-t)^2 a_0 + 2t(1-t)a_1 + t^2 a_2.$$

Explicit representation helps to evaluate and compute the value of $a_0^2(t)$ at a certain value of parameter $t$. This expression is derived directly from the recursive form above:

$$\begin{aligned} a_0^2(t) &= (1-t)a_0^1(t) + ta_1^1(t) \\ &= (1-t)((1-t)a_0 + ta_1) + t((1-t)a_1 + ta_2) \\ &= (1-t)^2 a_0 + 2t(1-t)a_1 + t^2 a_2 \\ &= a_0 B_0^2(t) + a_1 B_1^2(t) + a_2 B_2^2(t). \end{aligned}$$

Note that, during the above derivation, the simple equality has been used:

$$t(1-t)a_1 + t(1-t)a_1 = 2t(1-t)a_1.$$

However, as described in section 2, $A_1 \oplus A_1 = 2A_1$ may not hold in general. Hence, we have to modify Bernstein polynomial to evaluate intermediate shape at an arbitrary value of the parameter $t$. (In fact, the binomial coefficient part of Bernstein polynomial is not used any more). $C_0^2(t)$ of equation (4.1) is expanded as follows:

$$\begin{aligned} C_0^2(t) &= (1-t)^2 A_0 \oplus t(1-t)(A_1 \oplus A_1) \oplus t^2 A_2 \\ &= A_0 B_0^2(t) \oplus (A_1 \oplus A_1)\frac{B_1^2(t)}{2} \oplus A_2 B_2^2(t) \end{aligned} \tag{4.2}$$

We can reduce the equation (4.2) into Bernstein form only if equality $A_1 \oplus A_1 = 2A_1$ holds:

$$\begin{aligned} C_0^2(t) &= (1-t)^2 A_0 \oplus 2t(1-t)A_1 \oplus t^2 A_2 \\ &= \bigoplus_{i=0}^{2} A_i B_i^2(t). \end{aligned} \tag{4.3}$$

To recursively evaluate one geometric shape at a certain value of $t$ using equation (4.1), three Minkowski sum computations are required: each one for $C_0^1(t)$ and $C_1^1(t)$, and then last one for $(1-t)C_0^1(t) \oplus tC_1^1(t)$. Hence, when we evaluate $N$ in-between shapes we have to compute $3N$ Minkowski sums based on recursive equation (4.1). (We do not consider multiplication by real numbers since it takes relatively less computational load in overall computation.)

When using equation (4.2), 3 Minkowski sum computations are required for evaluating a single in-between shape. Note that, however, $A_1 \oplus A_1$ is not affected by a certain value of $t$. Hence, we can store the result of $A_1 \oplus A_1$ and reuse it regardless of the value of $t$. When we evaluate $N$ geometric shapes, 3 computations are required for the first evaluation, and $2(N-1)$ computations for additional $(N-1)$ geometric shapes. As a result, total $(2N+1)$ Minkowski sum computations are required in QIMS evaluation.

When the control shape $A_1$ is convex, the equality $A_1 \oplus A_1 = 2A_1$ holds. Hence, using equation (4.3), total $2N$ Minkowski sum computations are required.

4.1.3 Example

In Fig. 6, there are three control shapes: a triangle, a circle, and a square. Three LIMS morphing sequences are illustrated in white shapes: $C_0^0(t)$,

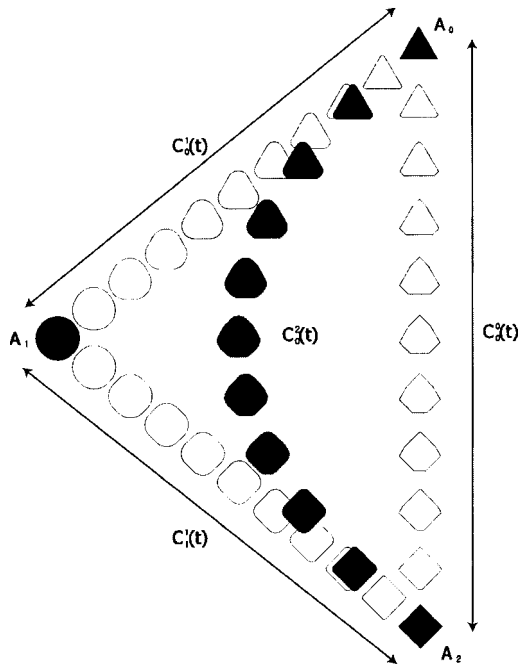



**Fig. 6.** Examples of QIMS morphing.

$C_0^1(t)$, and $C_1^1(t)$. The final QIMS morphing sequences is $C_0^2(t)$. Note that each shape in $C_0^0(t)$ is a polygon with angular corners. However, shapes in the control sequences, $C_0^1(t)$ and $C_1^1(t)$, have rounded corners. As a result, final shapes in $C_0^2(t)$ have round corners while they gradually change their shapes from a triangle to a square along the parameter $t$ changes. Therefore, in this case, the effect of a control shape is rounding each shape in $C_0^0(t)$.

### 4.2 Bézier Control of Degree *n*

4.2.1. Mathematical Model

For $(n+1)$ control shapes, Minkowski sum morphing has a following recursive form:

$$C_0^n(t) = (1-t)C_0^{n-1}(t) \oplus tC_1^{n-1}(t). \tag{4.4}$$

When $n = 3$ in equation (4.4), there are 4 control shapes: $A_0, \ldots, A_3$. The recursive form of cubic interpolation is expressed as follows:

$$C_0^3(t) = (1-t)C_0^2(t) \oplus tC_1^2(t) \tag{4.5}$$

We call this a cubic interpolation by Minkowski sum (CIMS). We can expand the equation (4.5) to get explicit form of CIMS as follows:

$$\begin{aligned} C_0^3(t) &= (1-t)\{(1-t)^2A_0 \oplus t(1-t)(A_1 \oplus A_1) \oplus t^2A_2\} \\ &\quad \oplus t\{(1-t)^2A_1 \oplus t(1-t)(A_2 \oplus A_2) \oplus t^2A_3\} \\ &= (1-t)^3A_0 \oplus t(1-t)^2((A_1 \oplus A_1) \oplus A_1) \\ &\quad \oplus t^2(1-t)((A_2 \oplus A_2) \oplus A_2) \oplus t^3A_3 \end{aligned} \tag{4.6}$$

It is clear that, when $A_i \oplus A_i \neq 2A_i$, $(A_i \oplus A_i) \oplus A_i \neq 3A_i$. Therefore, CIMS can be expressed with modified Bernstein polynomial $L_i^n = t^i(1-t)^{n-i}$ as follows:

$$\begin{aligned} C_0^3(t) &= A_0L_0^3(t) \oplus ((A_1 \oplus A_1) \oplus A_1)L_1^3(t) \\ &\quad \oplus ((A_2 \oplus A_2) \oplus A_2)L_2^3(t) \oplus A_3L_3^3(t). \end{aligned} \tag{4.7}$$

Although the explicit parameterization becomes more complex when $n > 3$ (since we cannot simplify the expanded expression using the equality $A_1 \oplus A_1 = 2A_1$), it is still possible to give an explicit algorithm for computation[13]. When the equality holds, the expression is simplified using Bernstein polynomial:

$$C_0^n(t) = \bigoplus_{i=0}^{n} A_iB_i^n(t). \tag{4.8}$$

4.2.2. Computation

To recursively evaluate one geometric shape at a certain value of $t$ using CIMS morphing equation (4.5), seven Minkowski sum computations are required: each three for $C_0^2(t)$ and $C_1^2(t)$, and then last one to compute $(1-t)C_0^2(t) \oplus tC_1^2(t)$. Hence, when we evaluate $N$ in-between shapes, we have to compute $7N$ Minkowski sums based on the recursive form of equation (4.5).

When using the explicit form of equation (4.7), seven Minkowski sum computations are required for evaluating a single in-between shape also. Considering the reusability of $(A_1 \oplus A_1) \oplus A_1$ and $(A_2 \oplus A_2) \oplus A_2$, the subsequent evaluation requires three more computations saving 4 each evaluation. Hence, $7 + 3(N-1) = 3N + 4$ computations are required. When both control shapes, $A_1$ and $A_2$, are convex, $3N$ Minkowski sum computations are required.

Generally, to compute $N$ intermediate shapes with $(n+1)$ control shapes in pure recursive way, $(2^n - 1)$ $N$ Minkowski sum computations are required, which is easy to verify using basic series computations.

However, when we evaluate the recursion in bottom-up with storing computed results for reuse,

we can expect some speed-up. For example, $C_1^1(t) = (1-t)A_1 \oplus tA_2$ can be reused in both $C_0^2(t)$ and $C_1^2(t)$. In this way, for $N$ intermediate shapes with $(n+1)$ control shapes, $n(n+1)N/2$ Minkowski sum computations required.

When we use explicit form such as equation (4.7), we have to compute terms like $(((A_i \oplus A_i) \oplus A_i)L \oplus A_i)$ which requires $\left(\binom{n}{i}-1\right)$ of Minkowski sums. Moreover, these terms can be reused regardless of the value of $t$. To compute single instance at a certain value of $t$, $n+\sum_{i=1}^{n-1}\left(\binom{n}{i}-1\right) = n+(2^n-n-1)$ of Minkowski sums are required. Considering reusability, $nN+(2^n-n-1)$ Minkowski sums are required for $N$ intermediate shapes.

When all the control shapes are convex, just $nN$ Minkowski sum computations are required. Note that the difference from non-convex case is $(2^n-n-1)$, which will affect overall performance when some of control shapes are complex or the number of control shapes is large.

Table 1 summarizes the number of Minkowski sum computations required for some computational methods. Considering the generality of non-convex control shapes, explicit evaluation is most efficient since the number of in-betweens $N$ does not multiply the last exponential part $(2^n-n-1)$.

As a computational approximation of non-convex control shapes, we can use the convergence property of equation (1.3). For example, we can replace the actual computation with the convex hull of a control shape when large number of self-addition is required.

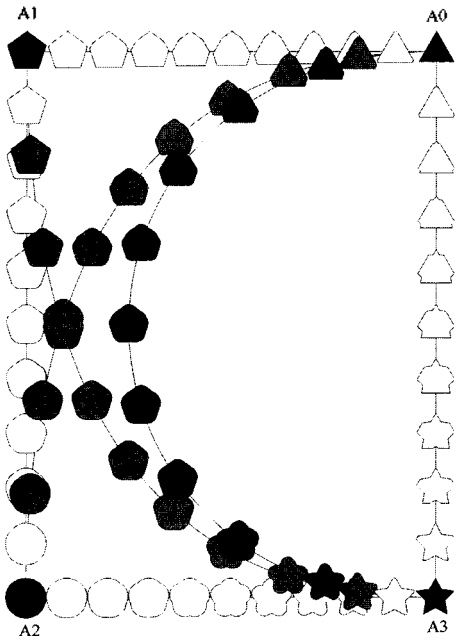



**Fig. 7.** An example of CIMS morphing.

4.2.3. Example

Fig. 7 shows an example of CIMS morphing. Let us assume that the initial control shapes are a triangle and a star (dark grey shapes in Fig. 7).

In an interactive drawing system, a designer first generates an LIMS morphing sequence between a triangle and a star. If she/he thinks the generated sequence is too angular, she/he can add an additional control shape to round the boundaries of intermediate shapes from the first LIMS morphing. In this case, a circle has been chosen as the third control shape. After generating the second sequence using QIMS morphing, she/he wants for the triangle to transform into a pentagon in an early stage. Hence, she/he can add the other control shape near a triangle. Finally, four control shapes has been used to generate the CIMS morphing sequence that satisfies the designer's intention.

### 4.3 The Blossom

In this section, the Bézier style control structure

**Table 1.** Number of computations of Minkowski sum on various situations

| Method | Number of Control Shapes | | | |
|---|---|---|---|---|
| | 2(LIMS) | 3(QIMS) | 4(CIMS) | n+1 |
| Pure Recursion | $N$ | $3N$ | $7N$ | $(2^n-1)N$ |
| Bottom-up Recursion | $N$ | $3N$ | $6N$ | $n(n+1)N/2$ |
| Explicit(Non-convex) | $N$ | $2N+1$ | $3N+4$ | $nN+(2^n-n-1)$ |
| Explicit(Convex) | $N$ | $2N$ | $3N$ | $nN$ |

in the previous section is generalized into a multi-affine form based on the principle of blossoming[7].

For the cubic case with 4 control shapes, we have the following blossom expression:

$$\begin{array}{llll} A_0 & & & \\ A_1 & A_0^1[t_1] & & \\ A_2 & A_1^1[t_1] & A_0^2[t_1,t_2] & \\ A_3 & A_2^1[t_1] & A_1^2[t_1,t_2] & A_0^3[t_1,t_2,t_3] \end{array} \quad (4.10)$$

In the above equation, $A_0^3[t_1,t_2,t_3]$ represents the resulting shape at certain values of three independent parameters $t_1$, $t_2$, and $t_3$. The example of evaluation steps is as follows:

$$A_0^1[t_1] = (1-t_1)A_0 \oplus t_1 A_1,$$

$$A_0^2[t_1,t_2] = (1-t_2)A_0^1[t_1] \oplus t_2 A_1^1[t_1],$$

$$A_0^3[t_1,t_2,t_3] = (1-t_3)A_0^2[t_1,t_2] \oplus t_3 A_1^2[t_1,t_2].$$

Note that, when $t = t_1 = t_2 = t_3$, equation (4.10) is identical to equation (4.5) of CIMS morphing.

Referring to Fig. 8, let

$$B_s = (1-s)A_0 \oplus sA_1,$$

$$C_s = (1-s)A_1 \oplus sA_2,$$

$$B_t = (1-t)A_0 \oplus tA_1,$$

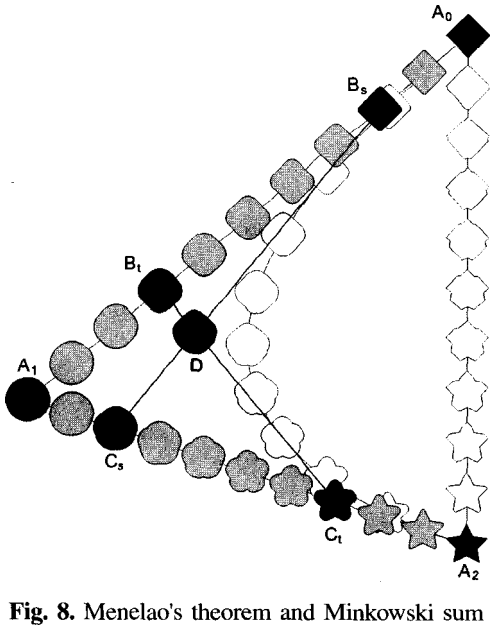



**Fig. 8.** Menelao's theorem and Minkowski sum

$$C_t = (1-t)A_1 \oplus tA_2.$$

The local origins (or the center) of each shape of equation (4.11) are as follows:

$$\begin{aligned} b_s &= (1-s)a_0 + sa_1, \\ c_s &= (1-s)a_1 + sa_2, \\ b_t &= (1-t)a_0 + ta_1, \\ c_t &= (1-t)a_1 + ta_2. \end{aligned} \quad (4.11)$$

where $a_i$ is the center of each control shape $A_i$

According to the Menelao's theorem the following holds:

$$(1-s)b_t + sc_t = (1-t)b_s + tc_s = d$$

where $d$ means the intersection point between two lines connecting $B_sC_s$ and $B_tC_t$, respectively (See [7] for details). With equation (4.11), it is easy to verify below:

$$(1-s)B_t \oplus sC_t = (1-t)B_s \oplus tC_s = D \quad (4.12)$$

since Minkowski sum is associative and commutative. If we expand the Moreover, it is clear that the center of $D$ is located at $d$.

Equation (4.12) helps to prove the symmetric function property of the blossom. The symmetric property simplifies the order of blossom evaluation[7]. For example, the following equation holds based on equation (4.12):

$$\begin{aligned} A_0^2[t_1,t_2] &= (1-t_2)A_0^1[t_1] \oplus t_2 A_1^1[t_1] \\ &= (1-t_1)A_0^1[t_2] \oplus t_1 A_1^1[t_2] \\ &= A_0^2[t_2,t_1]. \end{aligned}$$

With the blossom control structure proposed above, we have the more general and flexible controllability over Minkowski sum morphing. Consequently, the generated morphing sequences more varying and interesting.

## 5. Experiments

The control methods in section 4 have been implemented and tested in a Microsoft Windows XP Professional platform with a single Intel Pentium III-2GHz CPU and 1G RAM. The computational algorithm has been implemented using ANSI C++ and all the output has been generated in both Adobe PostScript and Macromedia Flash.

(Actually, output is not restricted to special format.) Our proposed algorithm and its implementation have generated all the figures presented in this paper.

The execution speed is quite fast. For example, CIMS morphing sequence in Fig. 6 is generated in near real-time with our test implementation. With some refinement, it will be possible to embed in a GUI-based design system that requires less response time in a drag-and-drop modification.

Since the performance of our morphing method is highly dependent on Minkowski sum execution, we have to refine the current method for Minkowski sum computation or adapt other algorithms for better performance[5,11]. (Actually, there has been a significant improvement in the algorithm of computing Minkowski sum morphing, which is not explained in this paper[12].)

Because of the available implementation of Minkowski sum computation, we have presented only 2D examples. However, our method is not restricted to 2D. It can be easily extended to 3D when coupled with 3D Minkowski sum computation.

## 6. Conclusion

In this paper, we have introduced a new and convenient geometric modeling technique to interactively control and generate the morphing sequence of geometric shapes based on Minkowski sum operations. The basic idea has developed from the linear interpolation on two geometric shapes where the algebraic sum has been replaced by Minkowski sum.

Moreover, we have extended this scheme into a Bézier-like control structure with arbitrary number of control shapes, which enables the interactive control over the intermediate shapes during the morphing sequence as in the traditional CAGD curve/surface editing. As an important contribution, the proposed control structure is further generalized into a Blossom form.

As a further research, we are interested in following topics: (1) coupling with Minkowski difference, general sweep and Boolean operations; (2) further analysis of mathematical properties of our control structure such as influence of control shapes and continuity of generated shape; (3) piecewise control structure; (4) algorithmic optimization and approximation for a high degree interpolation; (5) considering constraints on shape morphing such as area or volume preservation; (6) special control shapes which influence a certain part of the area for subtle control; (7) 3D extension based on 3D Minkowski sum; and (8) comparison and benchmark with other morphing methods.

## Acknowledgement

This research has been partly supported by Korean MOIC through Development of Collaborative Product Commerce Technologies project (2001). The preliminary version of this paper has been presented in Korean at Korea CADCAM Society Conference, Jan 30, 2002.

## References

1 Adobe Systems Inc. *Adobe Illustrator 9.0*, http://www.adobe.com/products/illustrator.

2. A. Kaul and R.T. Farouki, Computing Minkowski Sums of Plane Curves, *International Journal of Computational Geometry & Applications*, Vol. 5, No. 4., pp. 413-432, 1995.

3. H. Alt and L.J. Guibas, 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.

4. P.K. Agarwal, E. Flato, and D. Halperin, Polygon decomposition for efficient construction of Minkowski sums," *Computational Geometry*, Vol. 21, No. 1/2, pp.39-61, 2002.

5. D. Cohen-Or, D. Levein, D, and A. Solomovici, "Three-Dimensional Distance Field Metamorphosis," *ACM Transactions on Graphics*, Vol. 17, No. 2, pp. 116-141, 1998.

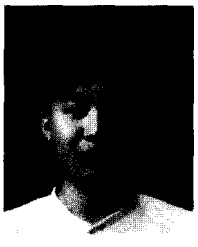
6. G. Farin. *Curves and Surfaces for CAGD*, 5th edition, Academic Press, 2002.

7. E. Galin and S. Akkouche, "Blob Metamorphosis based on Minkowski Sums," *Computer Graphics Forum*, Vol. 15 No. 3, pp. 143-153, August 1996.

8. P.K. Ghosh, "A Mathematical Model for Shape Description Using Minkowski Operators", *CVGIP*, Vol. 44, pp. 239-269, 1988.

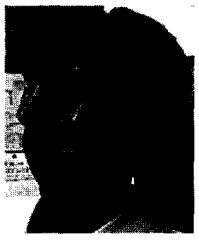
9. B. Jüttler, "Spatial Rational Motions and Their

Application in Computer Aided Geometric Design," *Mathematical Methods for Curve and Surfaces*, M. Dæhlen, T. Lyche, and L.L. Schumaker (Eds.), Vanderbit University Press, La Vergne, Tennessee, pp. 271-280, 1995.
10. I.-K. Lee, M.S. Kim, and G. Elber, "Polynomial/Rational Approximation of Minkowski Sum Boundary Curves," *GMIP*, Vol. 60, No. 2, pp. 136-165, 1998.
11. J.-H. Lee, S.J. Hong, and M.-S. Kim, "Polygonal boundary approximation for a 2D general sweep based on envelope and Boolean operations," *The Visual Computer*, Vol. 16, No. 3/4, pp. 208-240, 2000.
12. J.-H. Lee, "Notes on Minkowski Sum and Difference Operations: Properties and Applications," Technical Reports, ETRI (http://www.etri.re.kr), 2002.
13. J. Rossignac and A. Kaul, "AGRELs and BIPs: Metamorphosis as a Bézier curve in the space of polyhedra," *EUROGRAPHICS '94*, M. Dæhlen and L. Kjelldhal (Eds.), Blackwell Publishers, pp. C179-C184.
14. G. Turk and J. O'Brien, "Shape Transformation Using Variational Implicit Functions," *SIGGRAPH 99*, August 1999, pp. 335-342
15. J. Gomes, L. Darsa, B. Costa, and L. Velho, *Warping and Morphing of Graphical Objects*, Morgan Kaufmann Publishers, San Francisco, 1999.



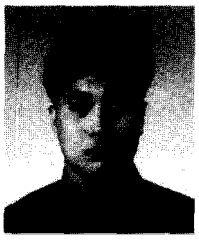
**이 주 행**

1994 포항공과대학교 전자계산학과 학사
1996 포항공과대학교 전자계산학과 석사
1999 포항공과대학교 전자계산학과 박사
1999~현재 한국전자통신연구원 동시공학연구팀 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
1990년~현재 한국전자통신연구원 동시공학연구팀장, 책임연구원
관심분야: Concurrent Engineering, Virtual Engineering, CAD/CAM/CAE/PDM



**이 재 열**

1992년 포항공과대학교 산업공학과 학사
1994년 포항공과대학교 산업공학과 석사
1998년 포항공과대학교 산업공학과 박사
1998년~현재 한국전자통신연구원 동시공학연구팀 선임연구원
관심분야: Internet-based CAD and Graphics, Collaborative Virtual Prototyping, Product and Process Integration Using Agents, Collaborative Product Commerce

**김 형 선**

1982년 상지대학교 경영학과 학사
1992년 광운대학교 컴퓨터공학과 석사
2001년~현재 대전대학교 컴퓨터공학과 박사과정
1985년~현재 한국전자통신연구원 선임연구원
관심분야: CAD, Collaborative Product Commerce, 분산컴퓨팅, 전자상거래, 정보보호, 데이터베이스