The Visual Computer

Polygonal boundary approximation for a 2D general sweep based on envelope and boolean operations

Joo-Haeng Lee[1], Sung Je Hong[2], Myung-Soo Kim[3]

[1] Computer Software Technology Lab., ETRI, Taejon 305-350, South Korea
[2] Dept. of Computer Science, POSTECH, Pohang 790-784, South Korea
[3] Dept. of Computer Eng., Seoul National Univ., Seoul 151-742, South Korea
e-mail: joohaeng@etri.re.kr, sjhong@postech.ac.kr, mskim@comp.snu.ac.kr

This paper presents an algorithm that approximates (using polygons) the boundary of a general sweep for an arbitrary 2D curved object (possibly with holes). Based on set-theoretic properties of the general sweep, our algorithm generates the polygonal sweep boundary incrementally, where envelope approximations and union operations are repeatedly applied to intermediate boundaries of the sweep and consecutive instances of the moving object at sampled locations of the motion. For approximation, each instance of the object is polygonized along the motion, where the object may experience dynamic shape transformation with topological changes such as creating and/or destroying internal holes. The incremental nature of the proposed algorithm makes the boundary construction of a general sweep useful for applications in interactive shape design, collision detection, and mechanical part design. Our algorithm generates a precise approximation of the boundary of a general sweep with real-time performance in computing unsweeps, Minkowski sums and differences, and constant radius offsets. Some experimental results are also given in this paper.

Key words: General sweep – Unsweep – Minkowski sum and difference – Offset – Interactive shape design

Correspondence to: J.-H. Lee

1 Introduction

The *sweep* operation is useful for the generation of a complex object by sweeping a simple object along a trajectory curve. The simplest form of sweep is *linear extrusion* where a 2D object moves along its normal direction to generate a volume. Another simple sweep is the *rotational sweep* where a 2D object rotates about a fixed axis. A rotational sweep generates a volume bounded by a surface of revolution. A *general sweep* is the most general form of sweep where the generator (i.e., the moving object) changes its size, orientation, and shape [9]. Figure 1 shows an example of a general sweep where a 2D object moves along a path in the plane while changing its size and orientation. A simple form of the general sweep is *generalized cylinder*, which has been actively used for applications in computer vision [2]. Early generalized cylinders assume that a 2D cross-sectional curve moves in space while its containing plane is kept orthogonal to the tangent direction of the trajectory curve. Recently, some authors allow generalized cylinders defined by more general spatial motions [7]. General sweeps of solids are useful in modeling the swept region of a numerically controlled (NC) machining tool or that of a robot following a path. However, it is non-trivial to construct the boundary of a general sweep. Two of the main reasons for this difficulty can be summarized as follows: (a) there may be complex self-intersections in the sweep boundary; and (b) the sweep surfaces are usually algebraic surfaces of high degree that are difficult to deal with (efficiently and robustly) in contemporary CAD systems. In this paper, based on a repeated application of boolean operations on simple polygons, we suggest an efficient and robust algorithm that precisely approximates (using polygons) the boundary of a general sweep for an arbitrary 2D curved moving object (possibly with holes).

The general sweep of a 3D solid has attracted considerable research attention in various fields of engineering. Wang and Wang [25] generate a scan-rendered image of a 3D swept volume. Weld and Leu [26] represent the sweep boundary of a polyhedron with ruled and developable surface patches. Based on the B-spline motion of polyhedra, Jüttler and Wagner [14] represent these boundary surfaces as rational B-spline surfaces. Martin and Stephenson [19] suggest envelope theory as a theoretical basis for computing swept volumes. Blackmore et al. [5, 6] present a sweep-envelope differential equation that characterizes the boundary of a swept volume. Abdel-Malek and Yeh [1] use rank-deficiency

The Visual Computer (2000) 16:208–240

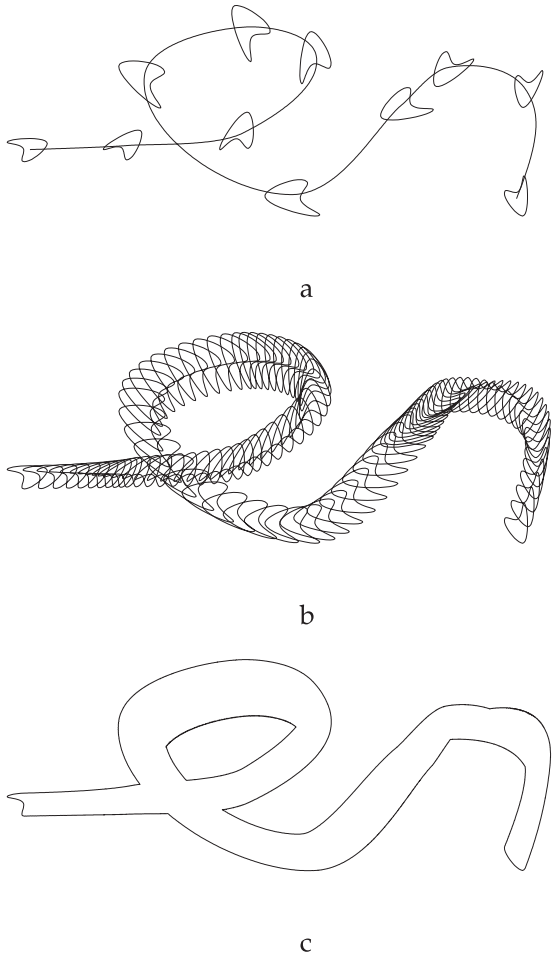


Fig. 1a–c. A general sweep and its boundary curve: **a** an object, changing its shape dynamically, moves along a trajectory curve; **b** sampled instances of the moving object: **c** the boundary curve of a general sweep

conditions to determine the interior and exterior of a swept volume. Recently, Pottmann [21] investigates various important geometric properties of general offset surfaces. Ilies and Shapiro [13] suggest the unsweep as an effective tool for designing mechanical parts that are collision-free from their surrounding packages. In these previous approaches, the sweep boundary is computed as a union of envelope surface patches. Since the envelope surfaces usually have a very high degree and quite often they have complex shapes (possibly with self-intersections), it is nontrivial to deal with these surfaces in constructing the boundary of a swept volume. Thus previous algorithms restrict moving objects to those having simple shapes and motions only.

The general sweep of a 2D curved object provides a handy design tool for modeling complex shapes such as oriental characters. Ahn et al. [3] develop an algorithm that approximates the boundary of a 2D general sweep for a planar curved object. Their algorithm is efficient and robust; however, redundant edges are eliminated in the last step of the sweep construction, which is inconvenient for an interactive shape design. Figure 2 illustrates the major steps of Ahn et al. [3] for constructing the boundary of a general sweep. Figure 2a shows polygons approximating the moving object at sampled locations; Fig. 2b shows the sweeps of polygon edges; Fig. 2c shows edges remaining after the elimination of some redundant edges; Fig. 2d shows the intersection points among these remaining edges; Fig. 2e is an intermediate result of eliminating redundant edges based on local conditions; Fig. 2f is the final result of eliminating all redundant edges based on global conditions; and finally Fig. 2g shows the result of fitting cubic Bézier curve segments to the polygonal approximation of the sweep boundary.

Kim et al. [15] represent the bristles of a brush stroke using variable-radius offset curves. The variable-radius offset is closely related to the general sweep where a line segment (of variable length) moves along a trajectory curve. Parida and Mudur [20] sweep a line segment along a 2D spine curve and design the outlines of Indian fonts and calligraphy. Blackmore et al. [5] propose sweep differential equations and a boundary-flow method for computing the boundary of a 2D general sweep. They consider simple polygons moving under linear deformation.

In this paper, we present an incremental algorithm that computes the boundary of a general sweep for an arbitrary 2D curved moving object (with holes). Based on set-theoretic properties of the general sweep, our algorithm generates the sweep boundary incrementally as the motion proceeds; that is, we repeatedly apply a union operation to an intermediate sweep boundary and a neighboring instance of the object at each sampled location of the motion trajectory. For approximation, we polygonize each instance of the moving object with dynamic shape change. Our algorithm is based on boolean operations on simple polygons, which can be implemented robustly using line/line intersections and inclusion tests for line segments with respect to simple poly-

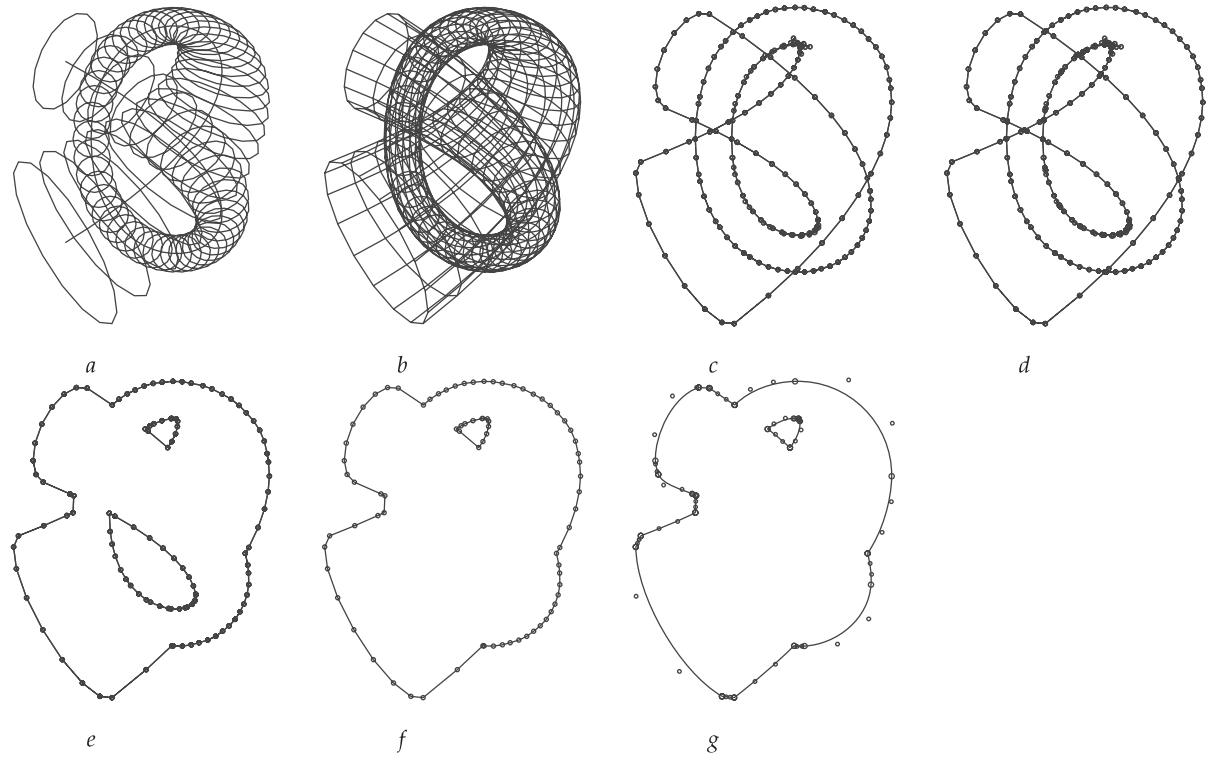


Fig. 2a–g. Major construction steps – Ahn et al. [3]

gons [11, 18, 22, 27]. The algorithms of Ahn et al. [3] and Blackmore et al. [5] generate complex polygons in intermediate steps; thus it is nontrivial to eliminate redundant edges robustly. (In particular, when the envelope curves have tangential and/or multiple intersections clustered together, a correct arrangement of envelope curves is very difficult to compute robustly.) It is clear that our approach is simple and leads to a robust implementation; nevertheless, previous work has not taken this approach due to efficiency considerations. In this paper, we propose various computational shortcuts that greatly improve the efficiency of our algorithm. The implemented result gives real-time performance. Moreover, the incremental nature of our algorithm makes the general sweep very useful for applications in interactive shape design, collision detection, and mechanical part design.

Our algorithm solves some important problems in geometric and solid modeling; e.g., we compute unsweeps, Minkowski sums/differences, and constant radius offsets using the general sweep. When unions are replaced by intersections, we can construct a dual of a sweep, called an unsweep. Ilies and Shapiro [13] propose the unsweep as an effective computational tool for detecting the collision of a moving mechanical part with its surrounding package. Figure 3 illustrates the design steps that remove colliding parts from the original shape and redesign a collision-free final shape. After designing an initial object and its surrounding package as shown in Fig. 3a, we can test the object to see if it collides with its surrounding package during the motion in Fig. 3b; the swept area can be computed and tested to see if there is an overlap with the package (see Fig. 3c). However, the sweep test is not an effective way of redesigning a collision-free object. Instead of sweeping an object, the package can be unswept as shown in Fig. 3d and e. After that, the unswept area is intersected with the initial object so as to construct

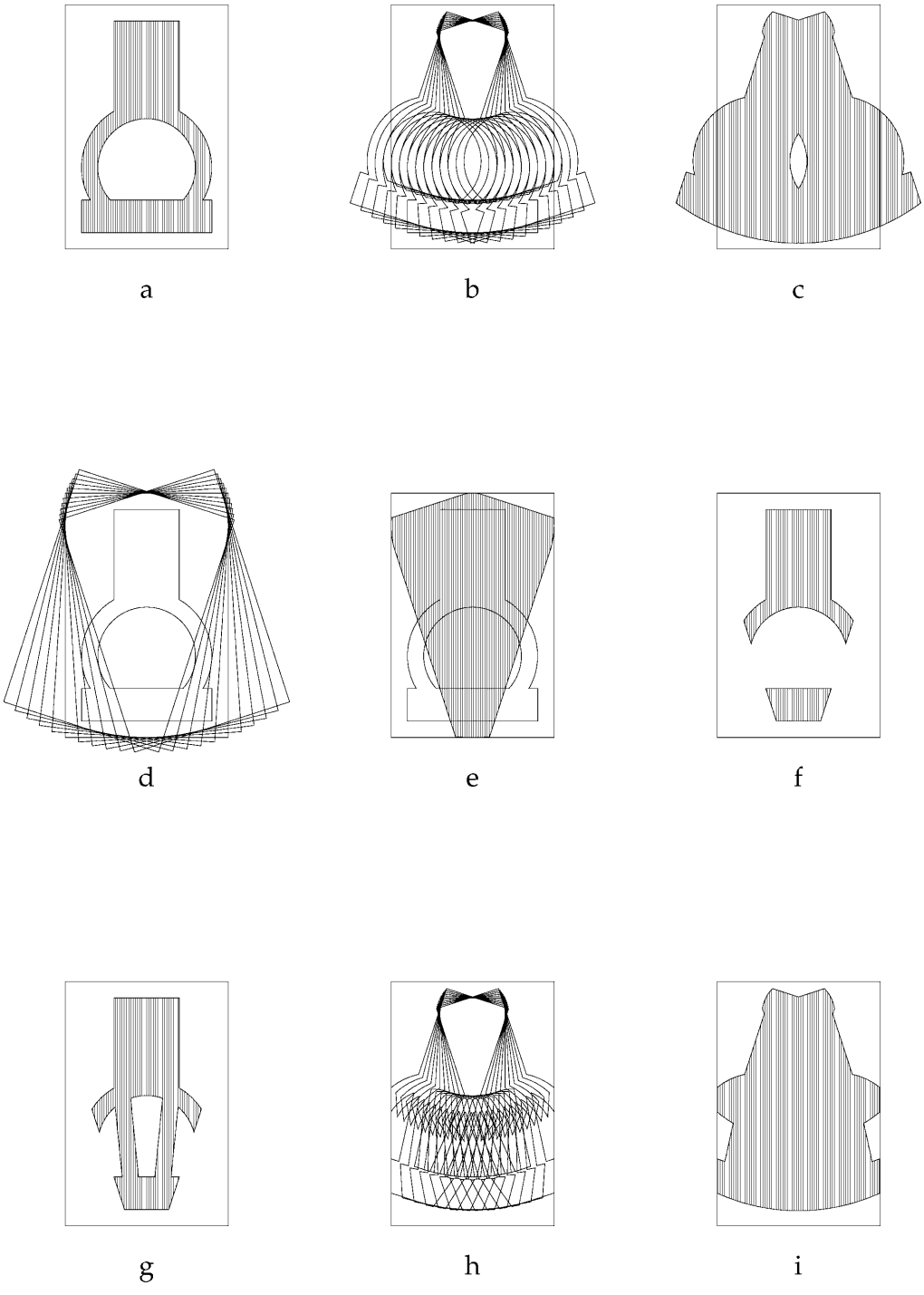


Fig. 3a–i. Application of the unsweep operation in the design of mechanical parts: **a** an initial shape O_1 and its surrounding package P; **b** rotational sweep of O_1 about a fixed axis; **c** the swept area S_1 overlaps the package; **d** unsweep of the package; **e** unswept area U; **f** collision-free parts: $O_1 \cap U$; **g** a collision-free new shape; **h** the sweep of the new shape; **i** the package contains a new sweep area S_2. (See Ilies and Shapiro [13].)

a collision-free object (see Fig. 3f). To connect the two separate components of Fig. 3f, new parts are added within the unswept area of Fig. 3e. The final shape shown in Fig. 3g is then guaranteed to be collision-free with its package (see Fig. 3h and i).

Minkowski sums and differences (or decomposition) are morphological operations for shape transformation such as dilation and erosion [10, 24]. They are important in computing collision free paths/areas in robot motion planning and swept areas/volumes in mechanical part design [4, 10, 12, 13, 17, 24]. The Minkowski sum of two planar objects is closely related to the *convolution curve* of the boundary curves of two input objects. (Given two planar curves, their convolution curve is defined as the set of all vector sums generated by all pairs of curve points that have the same curve normal direction [4].) The convolution curve is a superset of the boundary of a Minkowski sum [4]. Lee et al. [16] compute the boundary of a Minkowski sum based on convolution curve approximation, where redundant convolution curves are later eliminated and the remaining curves generate the boundary of a Minkowski sum. (An illustrative example of Lee et al. [16] is given in Fig. 24 of Sect. 5.) It is also possible to compute the boundary of a Minkowski difference by reversing the orientation of an operand curve in the convolution operation.

The boundaries of the Minkowski sum and difference are closely related to the translational sweep and unsweep of a moving object with a fixed shape and orientation. Our general sweep algorithm can be applied to the computation of the Minkowski sum and difference. We present algorithms for approximating the Minkowski operations. When constructing the boundaries of Minkowski sums and differences, our method does not introduce any redundancies in the intermediate steps of computation (see Sect. 5).

Constant radius offset, an important special case of the translational sweep, has been extensively used in shape design and NC tool path generation [23]. In general, when a circular disk moves along a given path, its sweep generates two offset curves of the path, one on each side of the path.

The rest of this paper is organized as follows. Section 2 presents some preliminary material. In Sect. 3, we show how to approximate the sweep-envelope boundary of a moving object between two consecutive locations. In Sect. 4, we construct the boundary of a general sweep using a sequence of unions coupled with envelope approximation. Section 5 presents algorithms that solve some problems related to the general sweep: unsweeps, Minkowski sums/differences, and constant radius offsets. Finally, Sect. 6 concludes this paper.

2 Preliminaries

In this section, we introduce basic notations and terminology that are used in the rest of the paper. A 2D *object* is defined as a closed set that includes its boundary as well as interior points. Let ∂A denote the boundary of an object A. We allow any number of holes in the object A, thus ∂A may consist of many components: an exterior boundary and the boundaries of holes. For notational convenience, the exterior boundary and the ith hole boundary are represented as $\partial_0 A$ and $\partial_i A$, respectively:

$$\partial A = \{\partial_0 A\} \cup \{\partial_i A\}, \quad 1 \le i \le \text{the number of holes.}$$

For the sake of simplicity, we exclude the following types of moving objects from consideration: (1) points, (2) open/closed curves, and (3) open sets. That is, we consider regular objects only.

Special types of regular sets include *null objects* and *infinite objects*: the null object O_ϕ represents an empty set of points; and an infinite object O_∞ contains points at infinity. These two special types of regular objects are used for defining regularized boolean operations. When regular polygonal objects have boundary representation, the null object has no vertex or edge, and an infinite object has a system-defined special polygon as its exterior boundary.

Given two objects A and B, boolean set operations are applied to both the interior and boundary points of A and B. In many conventional CAD systems, simple objects are usually represented by their boundaries [8]. To extract the boundary from the result of boolean set operations, we define *boolean boundary operations*:

$$(\partial A)^c \equiv \partial(A^c) \tag{1}$$

$$A \overset{\partial}{\cup} B \equiv \partial(A \cup B)$$

$$A \overset{\partial}{\cap} B \equiv \partial(A \cap B)$$

$$A \overset{\partial}{-} B \equiv \partial(A - B),$$

where A^c is the complement of A, and $(\partial\cdot)^c$, $\cup^\partial$, $\cap^\partial$, and $-^\partial$ denote the boolean boundary operations. An

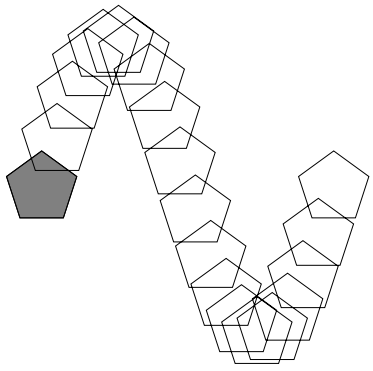
important procedure for implementing each boolean boundary operation is the elimination of redundant boundary segments (of the input objects) that belong to the interior of the resulting object. Boolean boundary operations need to be regularized to ensure that the results are also regular. Moreover, we need to deal with some special cases; e.g., (a) the intersection of two disjoint objects A and B must be a null object: $A \cap B = O_\phi$; (b) the complement of a bounded nonempty object A with holes is a set of regular objects, where one component must be an infinite object; and (c) the difference of an infinite object O_∞ and a bounded regular object A is another infinite object, $O_\infty - A = O_\infty$.

The result of a set operation may result in irregular objects. For example, when two objects have contact along their boundaries, their intersection contains some disconnected curves. In this case, each curve must be reduced to a null object by regularization. Compared with sweep operations, these boolean operations are relatively easy to implement in a robust way [11, 18, 22, 27]. Based on the robustness of these boolean operations, we propose a robust algorithm that constructs the boundary of a general sweep.

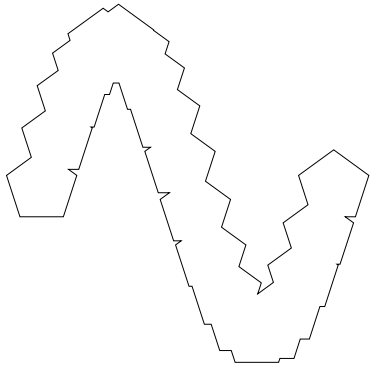
Regardless of representation schemes for input and output objects, boolean operations are useful for constructing a new object from existing ones [9]. Selectively combining some operations, a piece of an object can be added to or subtracted from other objects, and a hole can be created or destroyed. Some well-defined sequences of boolean operations generate even more complex and interesting shapes. The general sweep is one of these operations based on union operations sequentially applied to each instance of a moving object.

3 Envelope approximation

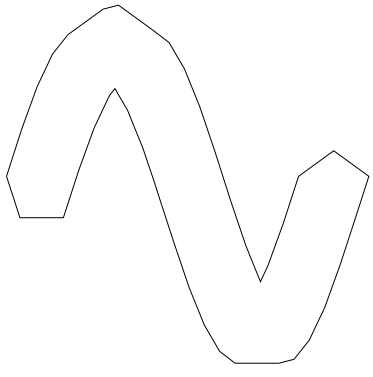
In this section, we consider how to approximate the sweep-envelope boundary of a moving object. For a simple approximation of the swept volume, a sequence of unions may be applied to densely sampled instances of a moving object along its motion trajectory. This approach generates a rough sweep boundary. As shown in Figs. 4b and 5b, such a sweep boundary has many jagged edges and redundant holes depending on the sampling density. To generate a smooth boundary curve, a large number of unions are required, which would make the algorithm inefficient. Moreover, when a moving object is



a



b



c

Fig. 4a–c. Sweeping a convex object: **a** the moving object is a pentagon; **b** a sweep boundary construction without envelope approximation; **c** a sweep boundary construction with envelope approximation

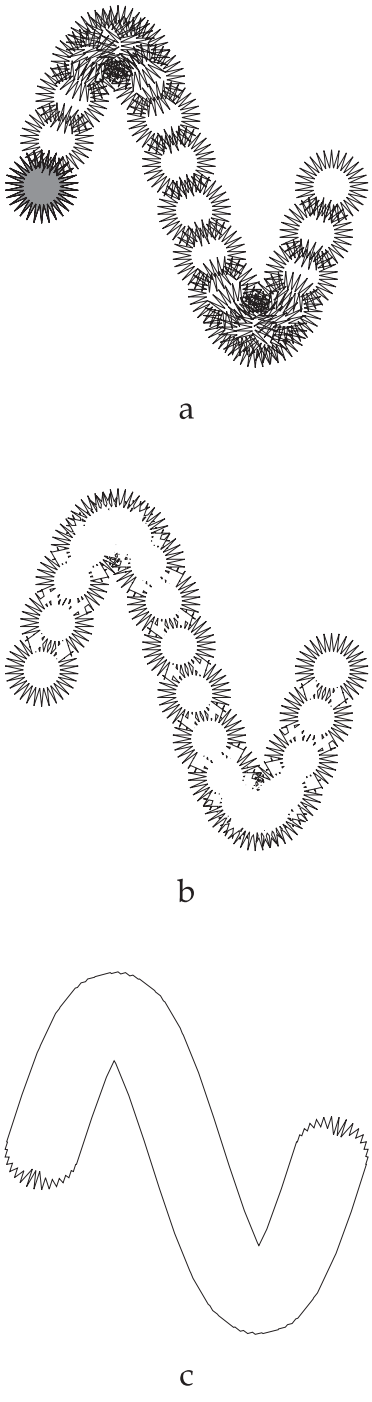


Fig. 5a–c. Sweeping a star-shaped object: **a** the moving object is a 30-star; **b** a sweep boundary construction without envelope approximation; **c** a sweep boundary construction with envelope approximation

nonconvex or star-shaped, a jagged sweep boundary is unavoidable (see Fig. 5b). To improve the precision of approximation and the efficiency of algorithm, we propose an envelope-based technique that approximates the sweep between two consecutive instances of a moving object by polygons. Our envelope approximation is based on line sweeps. That is, after polygonizing a moving object, we approximate its envelope boundary using the union of the sweeps of polygon edges. This simple approach provides an efficient construction of a smoother boundary of the general sweep. The boundaries of Figs. 4c and 5c are constructed with envelope approximation. Note that they are much less jagged than those of Figs. 4b and 5b.

Given an object A, its boundary ∂A can be represented as a union of closed boundary curves $\partial_0 A = C_{A,0}(u)$ and $\partial_i A = C_{A,i}(u)$ (each parameterized by u, for $0 \le u \le 1$):

$$\partial A = \{C_{A,0}(u)\} \cup \{C_{A,i}(u)\}, \tag{2}$$

where i enumerates the number of holes of A. Note that the boundary of an object is oriented in such a way that the object interior is to the left of the advancing direction of the boundary curve; i.e., the exterior boundary is oriented counterclockwise, and each hole boundary is oriented clockwise. By sampling points $\{p_k\}$ along the boundary curve ∂A, the object A is approximated by a polygon A^p. When the boundary curve is polygonized, it is represented by a list of oriented line segments $\{l_k\}$, where $l_k = (p_k, p_{k+1})$.

Let A_j and A_{j+1} be two consecutive instances of a moving object A, and $l_{j,k} = (p_{j,k}, p_{j,k+1})$ and $l_{j+1,k} = (p_{j+1,k}, p_{j+1,k+1})$ be the corresponding kth line segments of A_j^p and A_{j+1}^p, respectively. For approximation, we define an area that is swept from $l_{j,k}$ to $l_{j+1,k}$ (see the first and third columns of Fig. 6): (a) if $l_{j,k}$ and $l_{j+1,k}$ do not intersect, they define a triangle or a quadrangle (which is not necessarily convex) (see Fig. 6a–d); (b) if they intersect, they define one or two triangles (see Fig. 6e–g); and (c) if they are collinear, their sweep generates a line segment that has no contribution to the envelope computation since a line segment is not a regular object. Consequently, each line sweep is approximated by one or two simple polygons. Figure 6 shows some examples of line sweeps, where intermediate lines are generated by linear interpolation (as shown in the second column). The case of Fig. 6d needs some explanation. Note that a linear interpolation generates

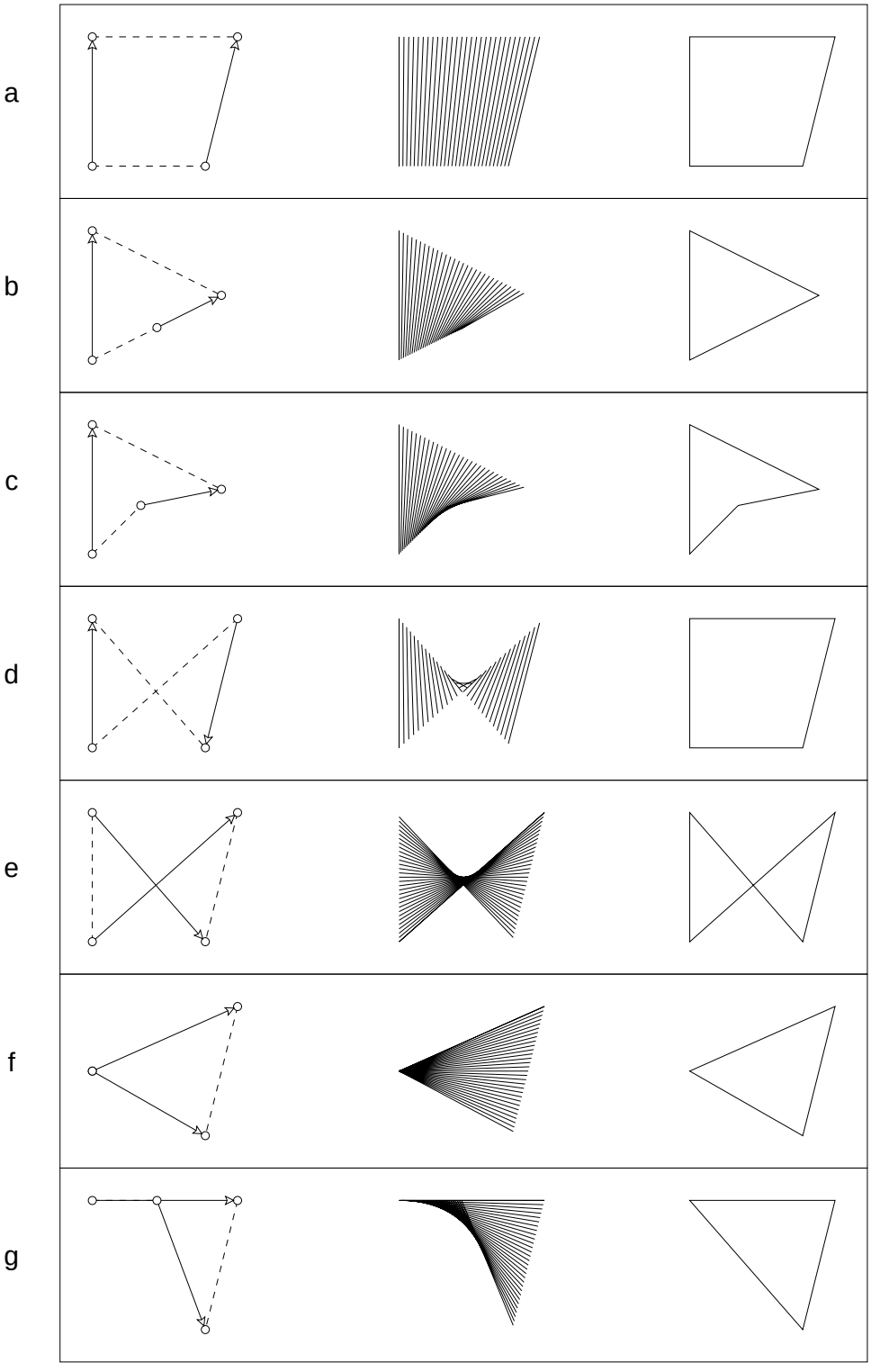


Fig. 6a–g. A line sweep is approximated by: **a** a quadrangle; **b** a triangle; **c** a quadrangle; **d** a quadrangle; **e** two triangles; **f** a triangle; **g** a triangle

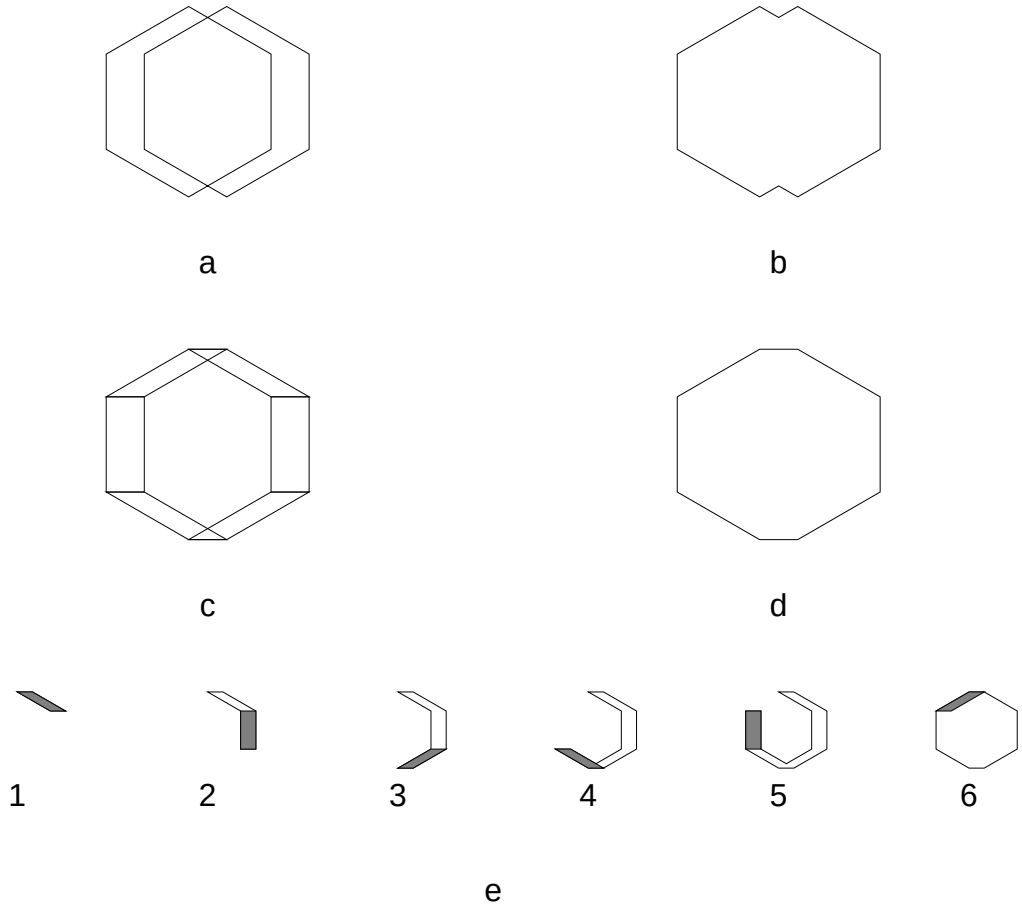


Fig. 7a–e. Line sweeps where $l_{j,k}$ and $l_{j+1,k}$ do not intersect: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** each $L_{j,k}$ is a quadrangle; **d** the boundary of a translational sweep based on envelope approximation; **e** sequential steps in envelope approximation (*gray area* is the k-th line sweep $L_{j,k}$)

short line segments in the middle of the interpolation. Thus, when the object does not shrink an edge quite dramatically, the linear interpolation does not approximate the motion of a line segment appropriately. Thus we approximate the line sweep as a rectangle rather than two triangles.

Ahn et al. [3] approximate each line sweep by the convex hull of two line segments, whereas our method allows nonconvex line sweeps. Figure 7c shows an example where each line sweep is a quadrangle; thus there is no difference between the two methods in this case. Figure 8c shows an example where each line sweep is composed of two triangles. Note that, in the case of Fig. 8, our approach of using nonconvex line sweeps produces a better approximation than the convex-hull-based approach of Ahn et al. [3].

Let $L_{j,k}$ denote a line sweep from $l_{j,k}$ to $l_{j+1,k}$, and let $\mathcal{L}_j$ denote the union of all these line sweeps:

$$\mathcal{L}_j = \bigcup_{1 \le j \le n} L_{j,k}, \tag{3}$$

where n is the number of line segments in the polygon A^p. We define an *approximate sweep operation*

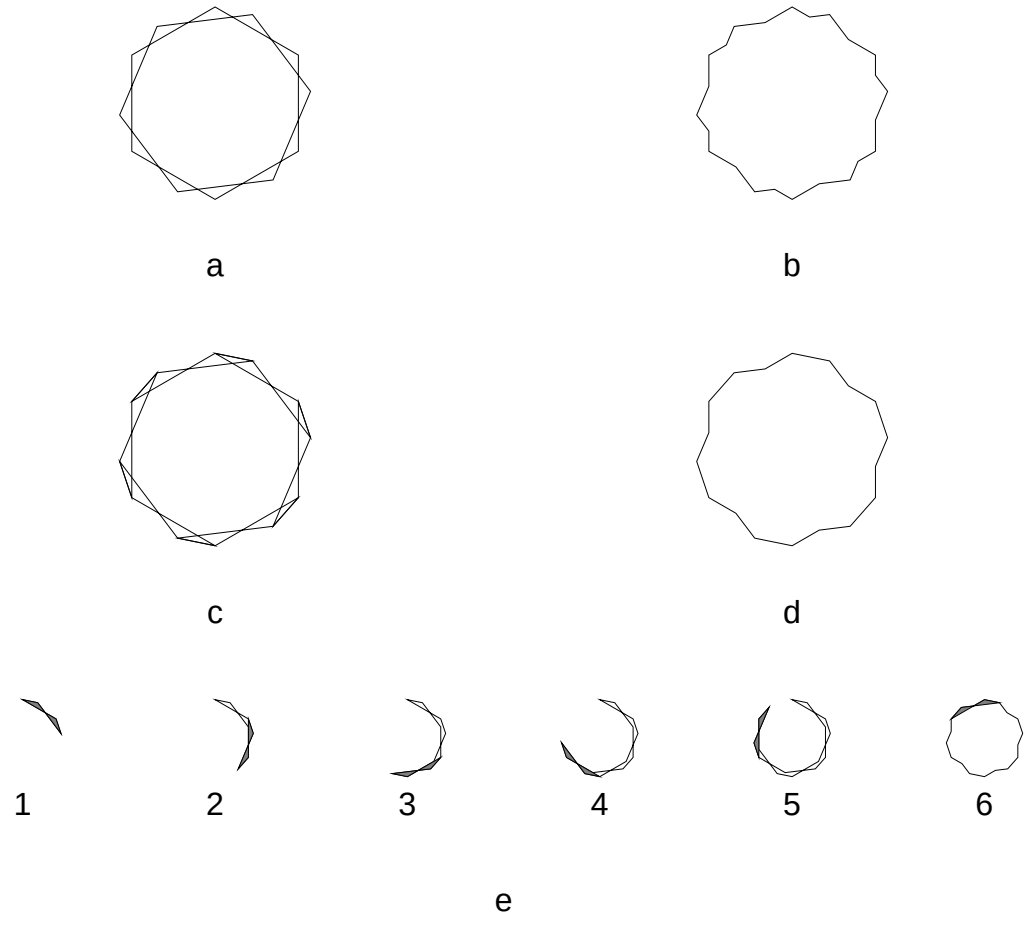


Fig. 8a–e. Line sweeps where $l_{j,k}$ and $l_{j+1,k}$ intersect: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** each $L_{j,k}$ is a pair of triangles; **d** the boundary of a rotational sweep based on envelope approximation; **e** sequential steps in envelope approximation (*gray area* is the *k*-th line sweep $L_{j,k}$)

$\vee$ on A_j^p and A_{j+1}^p as

$$A_j^p \vee A_{j+1}^p = A_j^p \cup A_{j+1}^p \cup \mathcal{L}_j. \tag{4}$$

Let $\vee^\partial$ denote the boundary of the $\vee$ operation. Figures 9 and 10 illustrate how our approximation method works. Note that, in these examples, $A_j^p \cup \mathcal{L}_j$ is the same as $A_j^p \vee A_{j+1}^p$ (i.e., there is no need to make A_{j+1}^p a union.)

When a moving object is convex, as is the case of Fig. 10, many line sweeps make no contribution to the final sweep envelope. For example, (a) $L_{j,1}$ is redundant since it is included in the interior of A_{j+1}^p, and (b) $L_{j,8}$ is redundant since $L_{j,9}$ and $L_{j,10}$ are closer to the final envelope boundary. After eliminating all redundant line sweeps, Fig. 11 shows that two line sweeps $\{L_{j,10}, L_{j,22}\}$ are sufficient for generating the same result as that of Fig. 10.

Consider a convex moving object A under a translation. In Fig. 12a, the first instance A_j translates to the second instance A_{j+1} by a displacement **v**. The swept area is the same as the convex hull of A_j and A_{j+1} (see Fig. 12b). Its boundary is the union of envelope lines e_u and e_l and some boundary curves of

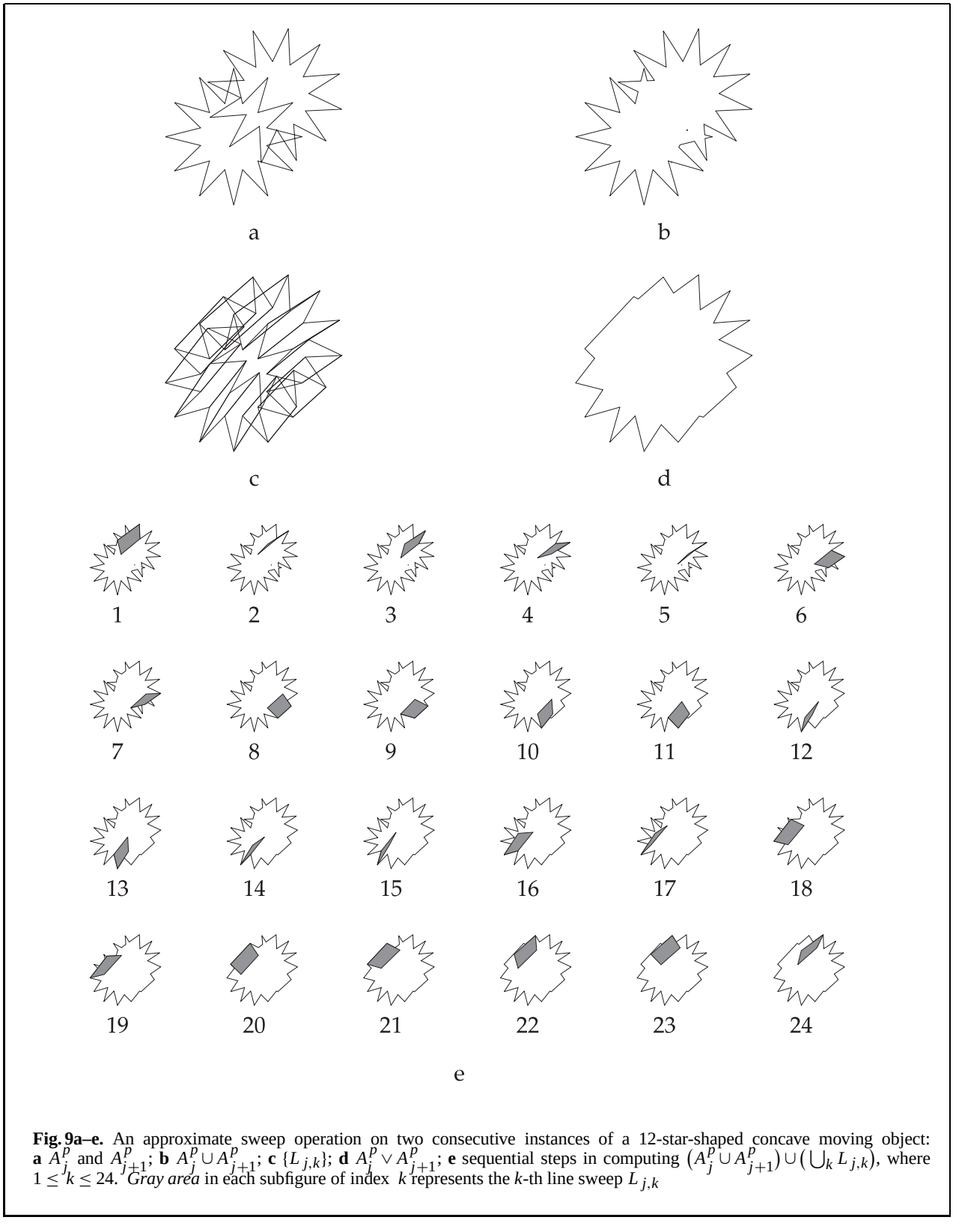


Fig. 9a–e. An approximate sweep operation on two consecutive instances of a 12-star-shaped concave moving object: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_k L_{j,k}\right)$, where $1 \le k \le 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

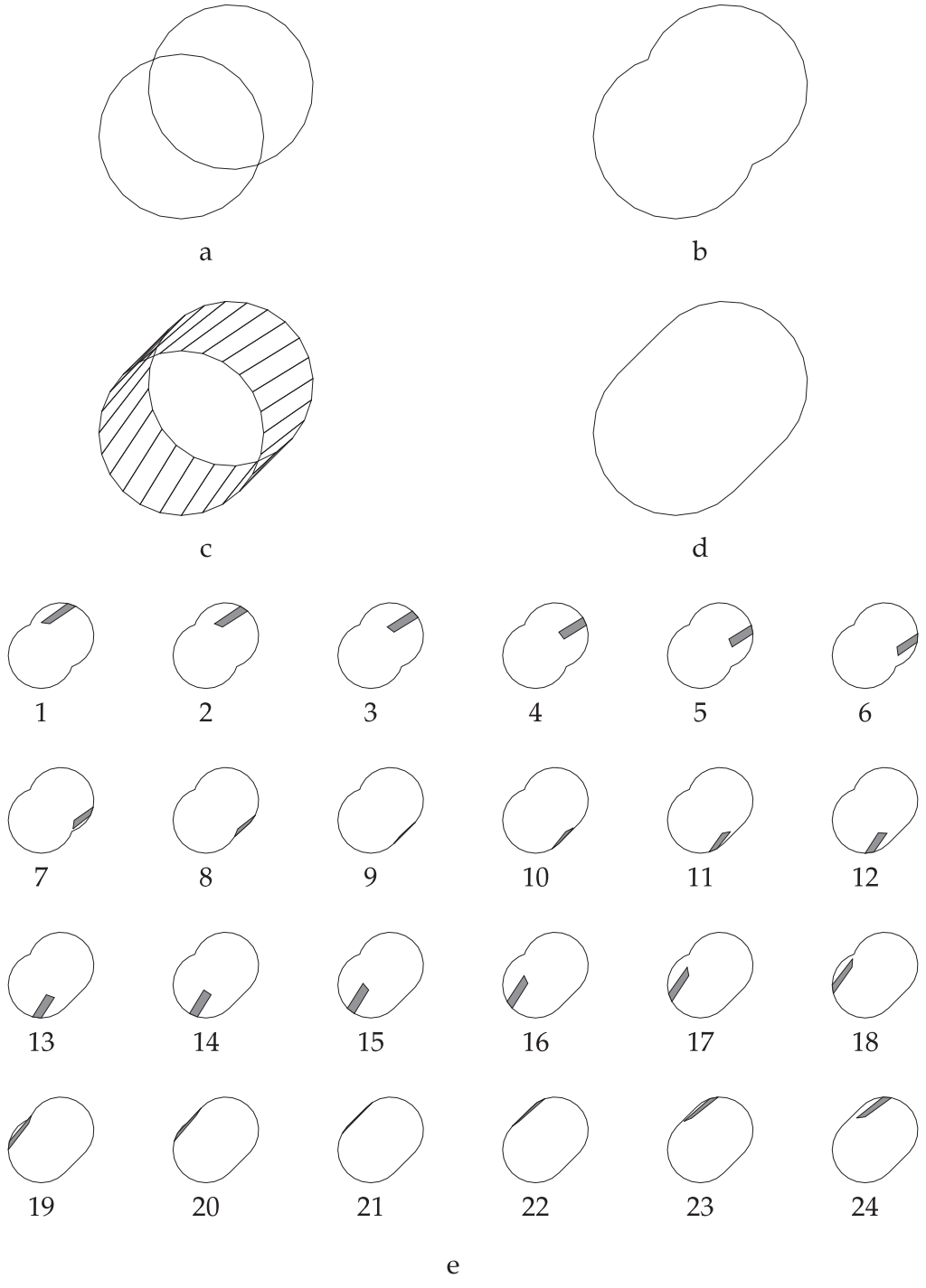


Fig. 10a–e. An approximate sweep operation on two consecutive instances of a convex moving object: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_k L_{j,k}\right)$, where $1 \le k \le 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

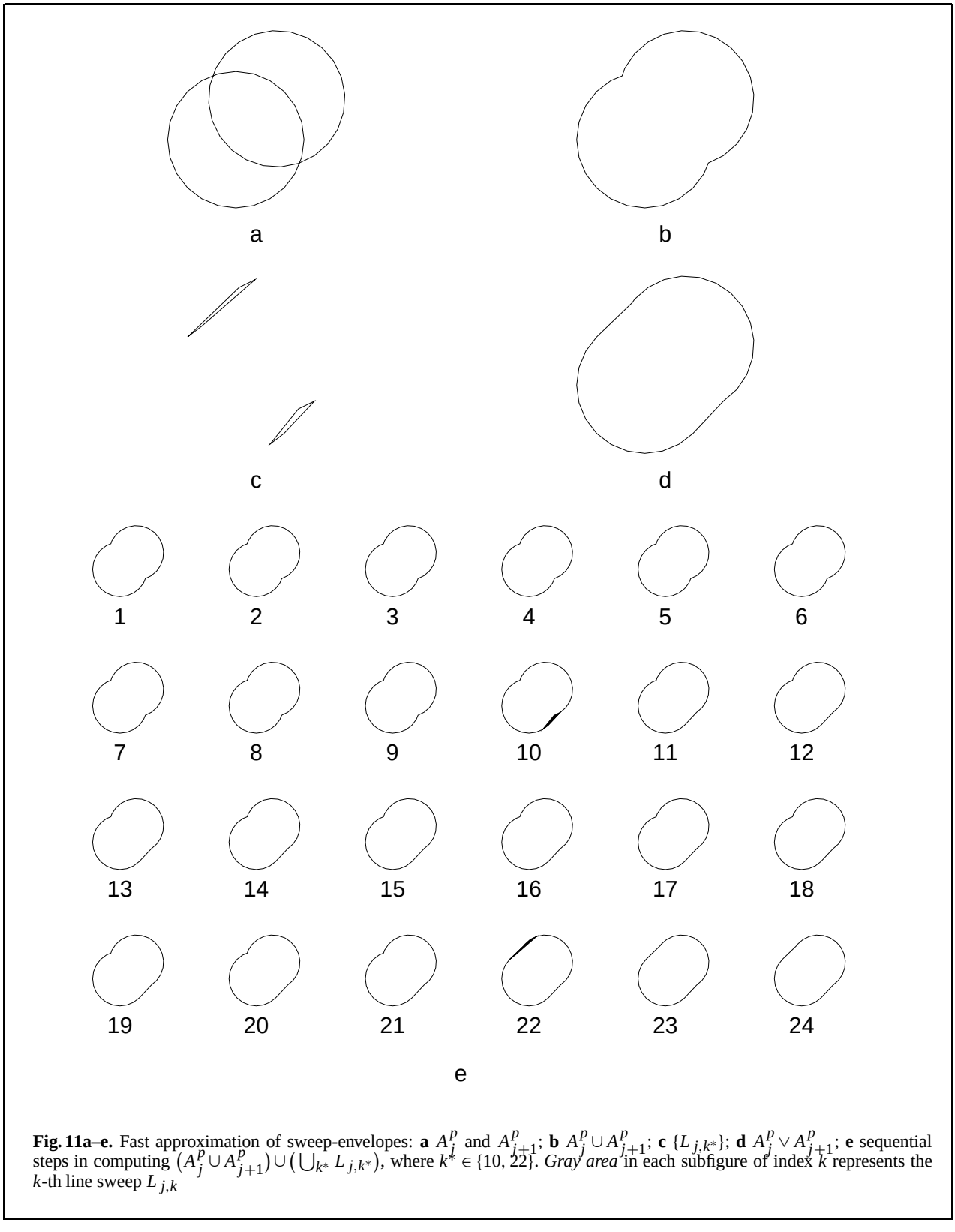


Fig. 11a–e. Fast approximation of sweep-envelopes: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k^*}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_{k^*} L_{j,k^*}\right)$, where $k^* \in \{10, 22\}$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

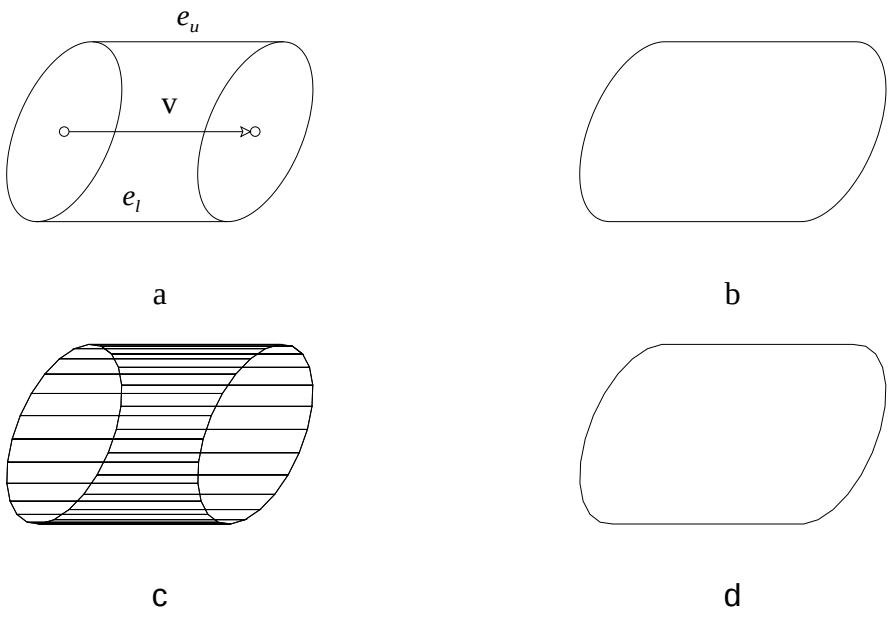


Fig. 12a–d. A translational sweep of a convex moving object along a fixed direction **v**: **a** two instances of the moving object and envelope lines e_u and e_l; **b** the exact boundary of a sweep; **c** line sweeps; **d** an approximate boundary of the sweep

A_j and A_{j+1}. Note that e_u and e_l are line segments parallel to **v**. Moreover, e_u and e_l are defined by the extreme points of A_j and A_{j+1} in the directions orthogonal to that of **v**. In Fig. 12c, we approximate the envelope edges $e_{j,k}$, applying this technique to polygonal approximations A_j^p and A_{j+1}^p.

When an object moves with rotation and shape change, the selection of appropriate envelope edges $e_{j,k}$ becomes more difficult (see Figs. 13 and 14). In Fig. 13, we form a union of all line sweeps to generate the swept area, whereas Fig. 14 shows that two line sweeps $L_{j,10}$ and $L_{j,22}$ are sufficient for generating the same result as that of Fig. 13. Thus, we do not need to consider the set $\mathcal{L}_j$ of all line sweeps. A reduced set $\mathcal{L}_j^*$ of line sweeps is sufficient for the construction of the sweep envelope:

$$\mathcal{L}_j^* = \bigcup_{k^*} L_{j,k^*},$$

$$A_j^p \vee A_{j+1}^p = A_j^p \cup A_{j+1}^p \cup \mathcal{L}_j^*,$$

where the k^*th line sweeps are those that may potentially contribute to the sweep envelope. In other words, $\mathcal{L}_j^*$ does not include redundant line sweeps that can easily be excluded, considering the local configuration of adjacent line sweeps.

In Figs. 13 and 15, most line sweeps belong to the case of Fig. 6a. This is also the case in most practical examples (though the pure rotation of Fig. 8 is an exception). Assume that three consecutive line sweeps $L_{j,k-1}$, $L_{j,k}$, and $L_{j,k+1}$ are of the type of Fig. 6a. Then $L_{j,k-1}$ and $L_{j,k}$ share an edge $(p_{j,k}, p_{j+1,k})$. Moreover, this edge $(p_{j,k}, p_{j+1,k})$ is redundant if each side of $(p_{j,k}, p_{j+1,k})$ belongs to either $L_{j,k-1}$ or $L_{j,k}$. (That is to say, the edge $(p_{j,k}, p_{j+1,k})$ is redundant if it is oriented oppositely in $L_{j,k-1}$ and $L_{j,k}$; see Ahn et al. [3].) A similar argument applies to the line sweeps $L_{j,k}$ and $L_{j,k+1}$ and their shared edge $(p_{j,k+1}, p_{j+1,k+1})$. This technique applies to nonconvex as well as convex moving objects. The detection of other redundancies is more complex, and we skip the details here.

Using a much reduced set improves the efficiency of our algorithm significantly. Given two convex objects with the same number of vertices, the set $\mathcal{L}_j^*$ usually contains two line sweeps. As shown in Figs. 15 and 16, there may be more than two line sweeps contributing to the sweep envelope, in particular, when the sampling is taken relatively sparsely

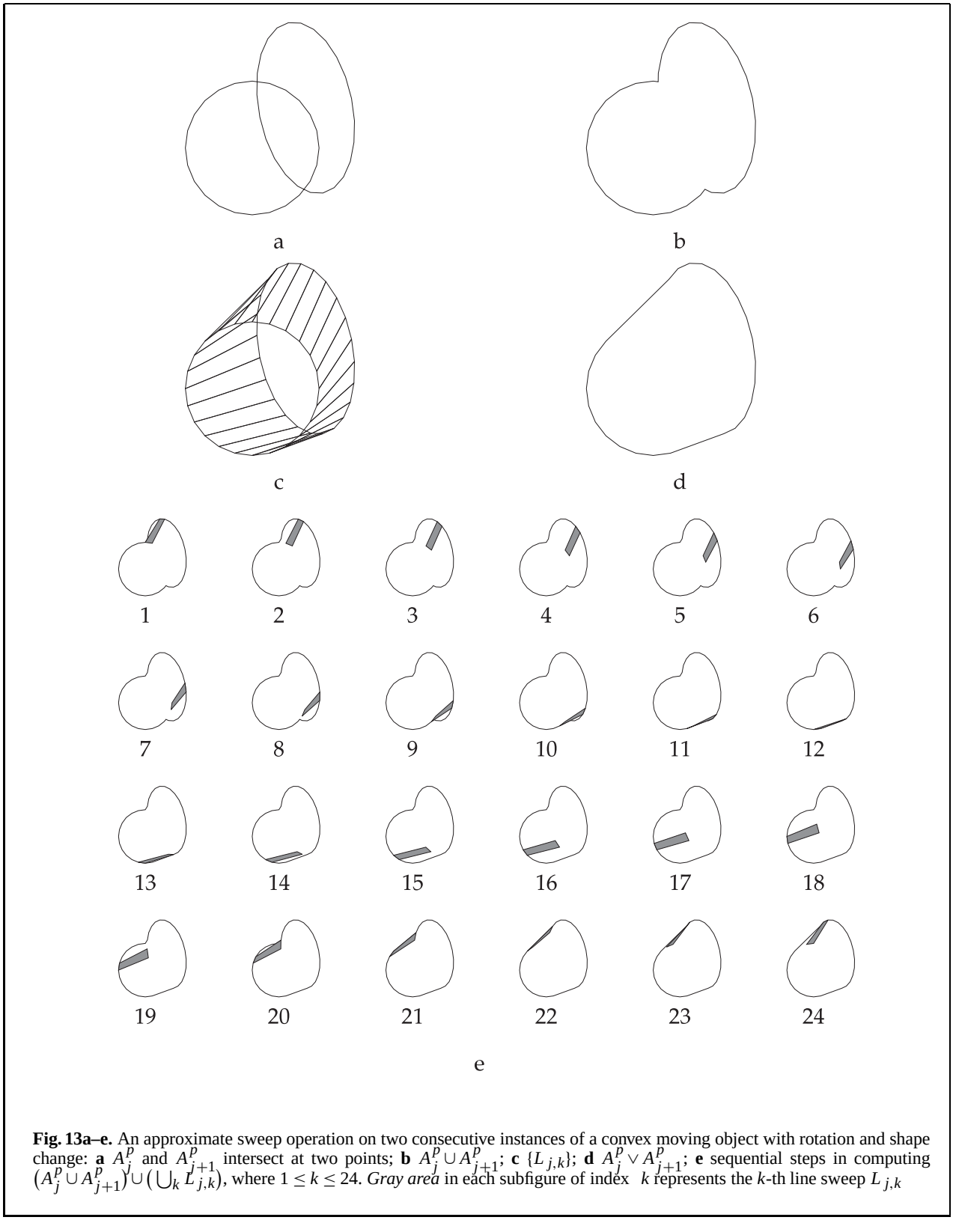


Fig. 13a–e. An approximate sweep operation on two consecutive instances of a convex moving object with rotation and shape change: **a** A_j^p and A_{j+1}^p intersect at two points; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_k L_{j,k}\right)$, where $1 \le k \le 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

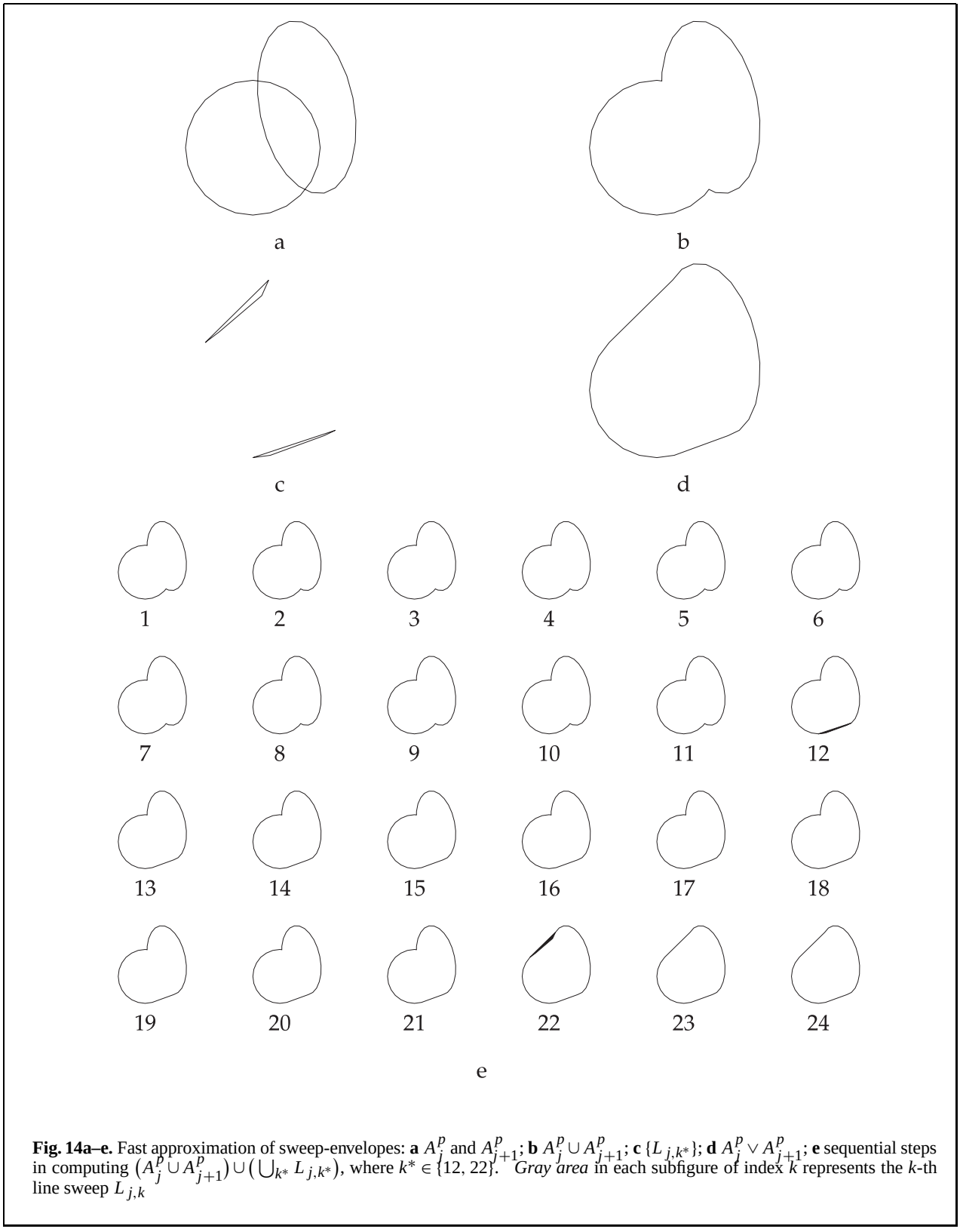


Fig. 14a–e. Fast approximation of sweep-envelopes: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k^*}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_{k^*} L_{j,k^*}\right)$, where $k^* \in \{12, 22\}$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

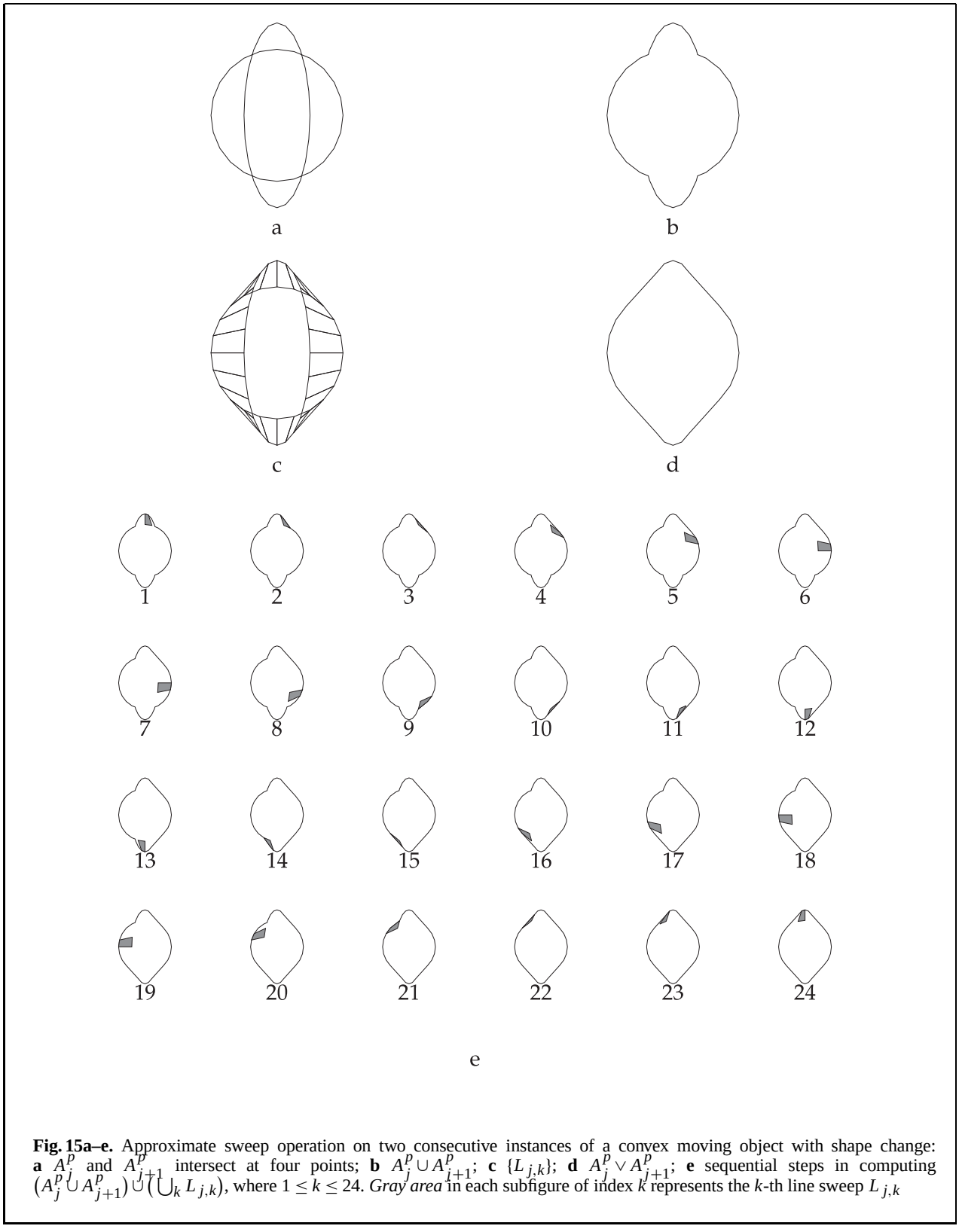


Fig. 15a–e. Approximate sweep operation on two consecutive instances of a convex moving object with shape change: **a** A_j^p and A_{j+1}^p intersect at four points; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_k L_{j,k}\right)$, where $1 \le k \le 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

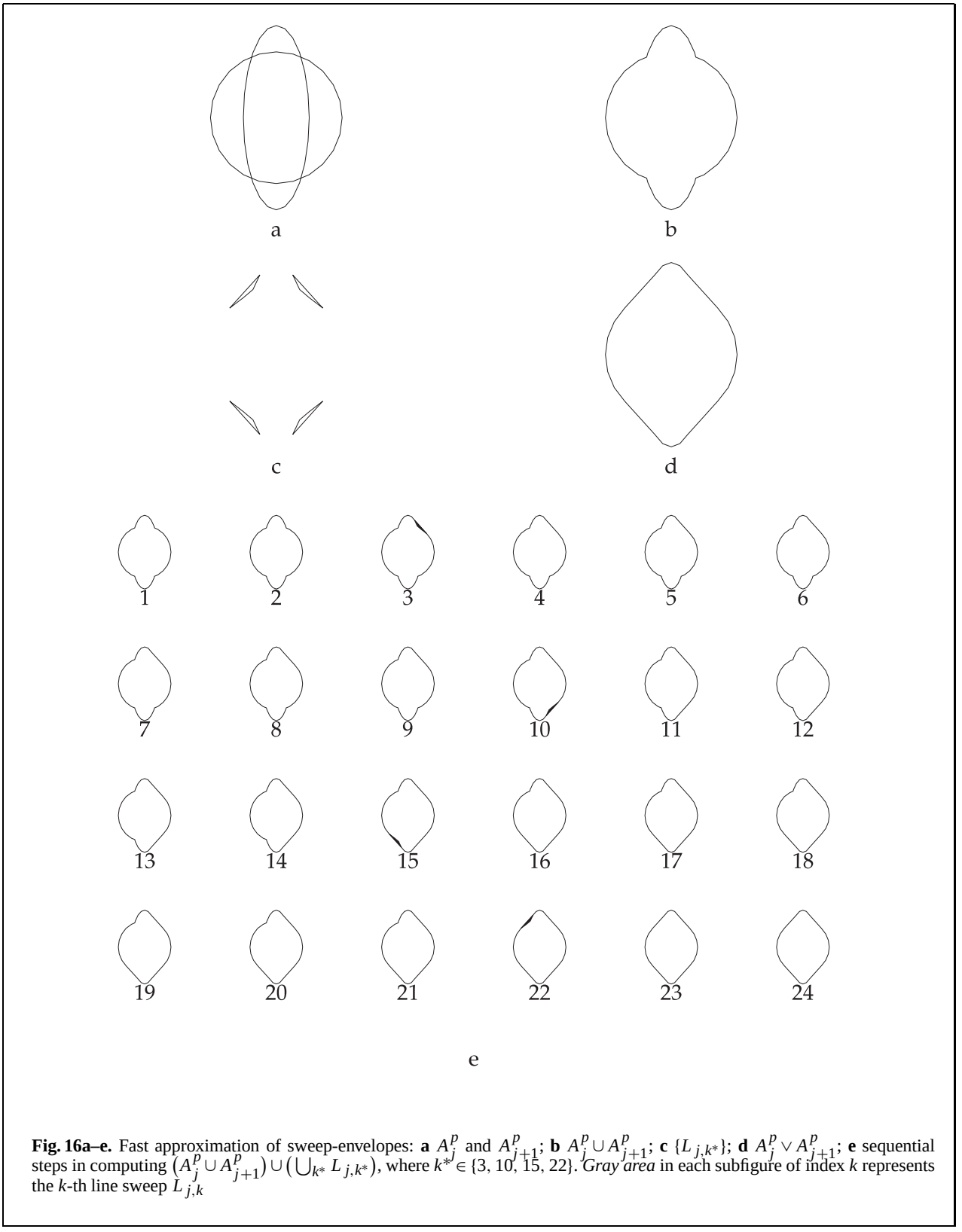


Fig. 16a–e. Fast approximation of sweep-envelopes: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cup A_{j+1}^p$; **c** $\{L_{j,k^*}\}$; **d** $A_j^p \vee A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cup A_{j+1}^p\right) \cup \left(\bigcup_{k^*} L_{j,k^*}\right)$, where $k^* \in \{3, 10, 15, 22\}$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

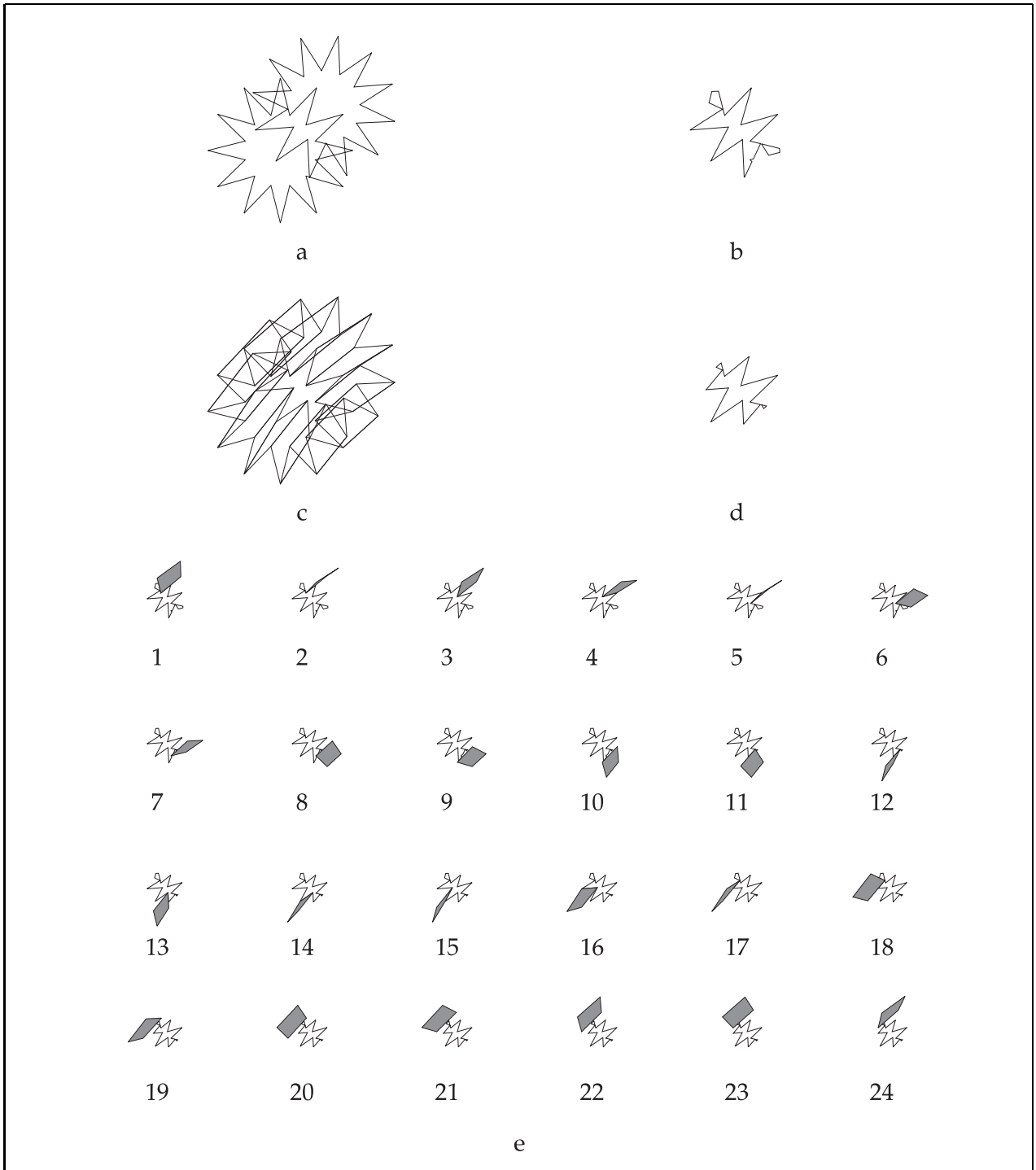


Fig. 17a–e. An example of approximate unsweep operation on two instances of a 12-star-shaped concave object: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cap A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \wedge A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cap A_{j+1}^p\right) \cap \left(\bigcap_k \left(L_{j,k}\right)^c\right)$, where $1 \leq k \leq 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k}$

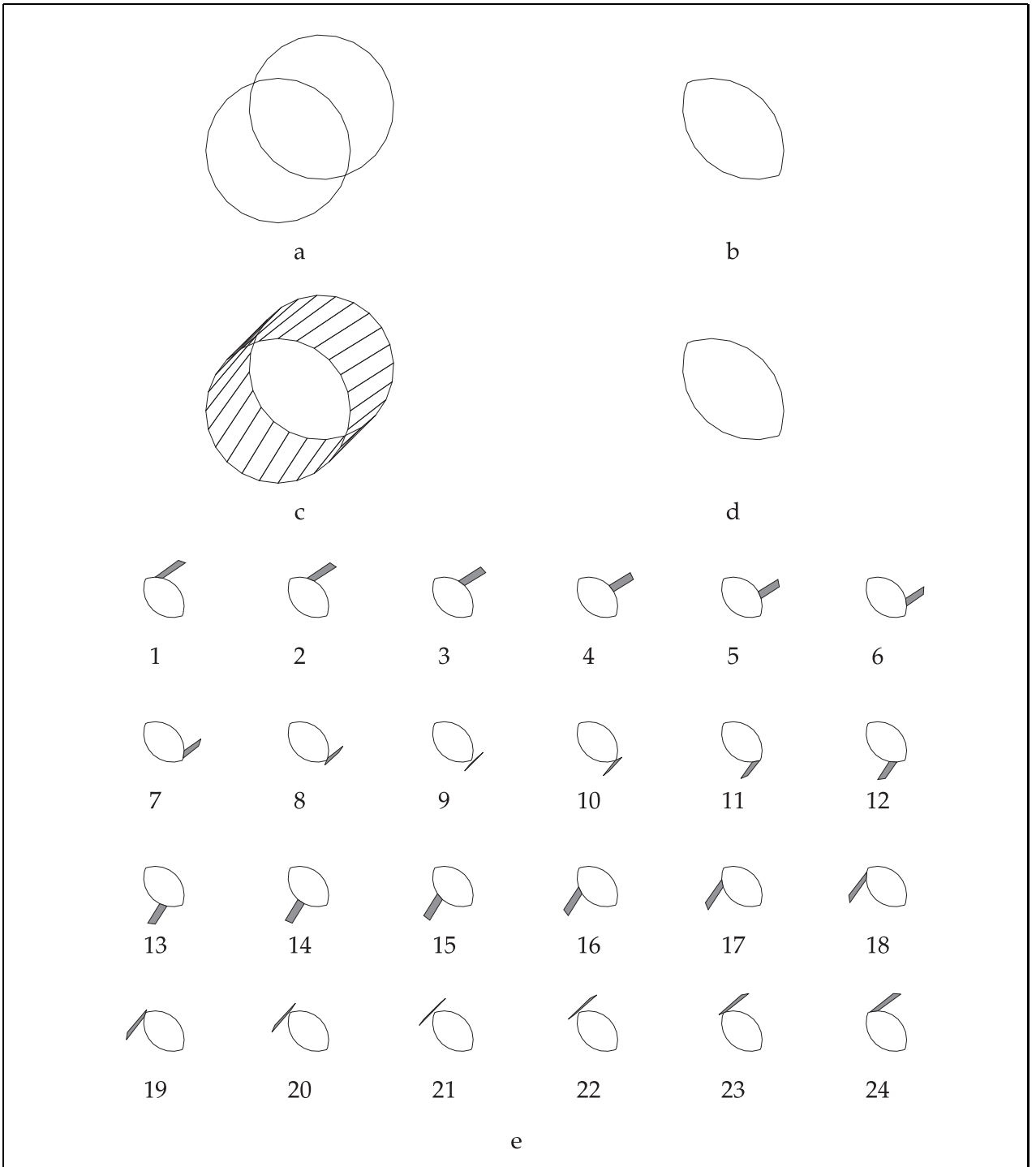


Fig. 18a–e. An example of approximate unsweep operation on two instances of a convex object: **a** A_j^p and A_{j+1}^p; **b** $A_j^p \cap A_{j+1}^p$; **c** $\{L_{j,k}\}$; **d** $A_j^p \wedge A_{j+1}^p$; **e** sequential steps in computing $\left(A_j^p \cap A_{j+1}^p\right) \cap \left(\bigcap_k \left(L_{j,k}\right)^c\right)$, where $1 \leq k \leq 24$. *Gray area* in each subfigure of index k represents the k-th line sweep $L_{j,k} L_{j,k}$

or the moving object changes its shape quite dramatically. Even in this case, our algorithm detects contributing line sweeps correctly. Figure 16e shows that $\mathcal{L}_j^* = \{L_{j,3}, L_{j,10}, L_{j,15}, L_{j,22}\}$.
Now, we define an *approximate unsweep operation* $\wedge$ on A_j^p and A_{j+1}^p as

$$A_j^p \wedge A_{j+1}^p = A_j^p \cap A_{j+1}^p \cap (\mathcal{L}_j)^c = \left(A_j^p \cap A_{j+1}^p\right) - \mathcal{L}_j. \quad (5)$$

Let $\wedge^\partial$ denote the boundary of the $\wedge$ operation. Figures 17 and 18 show two unsweeps: one for a nonconvex object and the other for a convex object. In particular, Fig. 17b and d are the results of $A_j^p \cap A_{j+1}^p$ and $A_j^p \wedge A_{j+1}^p$, respectively. Note that Fig. 17b and d are different since $A_j^p \wedge A_{j+1}^p$ is obtained by subtracting some line sweeps $L_{j,k}$ from $A_j^p \cap A_{j+1}^p$.

4 General sweep boundary

In this section, we define the general sweep as an infinite union, and we introduce an algorithm that approximates the boundary of a general sweep.
A planar rigid motion $M(t)$ at time t is represented as a 3×3 homogeneous matrix

$$M(t) = \begin{bmatrix} R(t) & \mathbf{v}(t) \\ \mathbf{0} & 1 \end{bmatrix},$$

where $R(t)$ is a 2×2 rotation matrix and $\mathbf{v}(t)$ is a translation vector $[v_x(t) \;\; v_y(t)]^T$. In a general sweep, arbitrary shape transformation can be applied to the moving object. As in Fig. 19, the moving object can change its shape dynamically with possible topological changes as well (i.e., while creating and destroying holes). Therefore, the shape transformation $F(t)$ is nonlinear in general.
Given a rigid motion $M(t)$ and a shape transformation $F(t)$, the object A at time t is represented as

$$A(t) = M(t) \cdot F(t) \cdot A. \quad (6)$$

Moreover, $\partial A(t)$ represents the boundary of A at time t. A general sweep produces a new object:

$$S = S(A, M, F) = \bigcup_t M(t) \cdot F(t) \cdot A = \bigcup_t A(t), \quad (7)$$

where $\bigcup_t$ means an infinite union of the moving object. $S(t_1) = \bigcup_{t_0 \le t \le t_1} A(t)$ represents a partially swept object from an initial time t_0 up to the current time t_1. By the definition of boolean boundary operations given in (1), the boundary of a general sweep is defined as

$$\partial S = \partial S(A, M, F) = \partial \left(\bigcup_t A(t) \right) = \bigcup_t^\partial A(t). \quad (8)$$

In theory, S and ∂S are the exact swept object and its boundary, respectively. However, in practice, it is nontrivial to compute the geometric objects defined by (6)–(8). As explained in the introduction, the sweep-envelope curves are high-degree

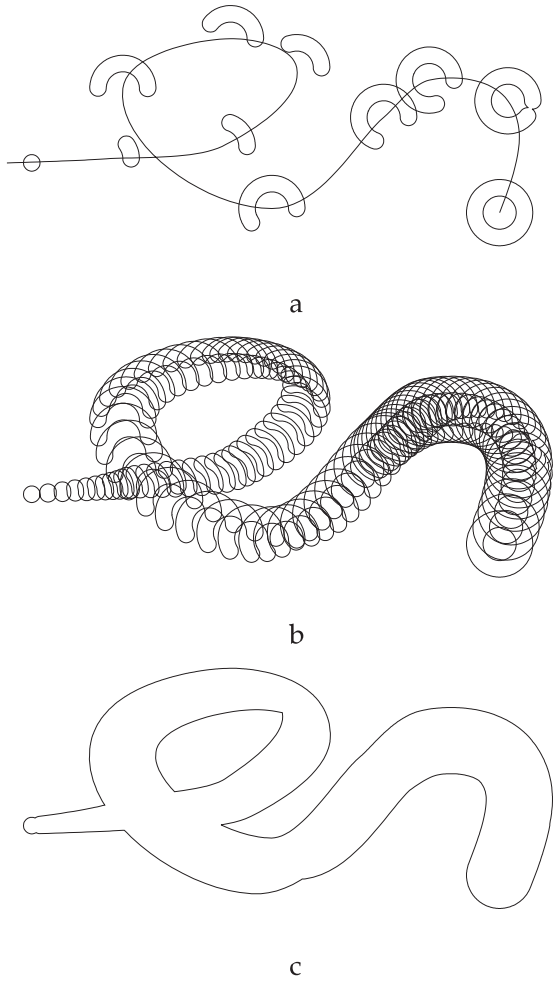


Fig. 19a–c. General sweep: **a** the moving object changes its shape from a circle to a torus; **b** sampled instances of the moving object; **c** the boundary curve of the general sweep

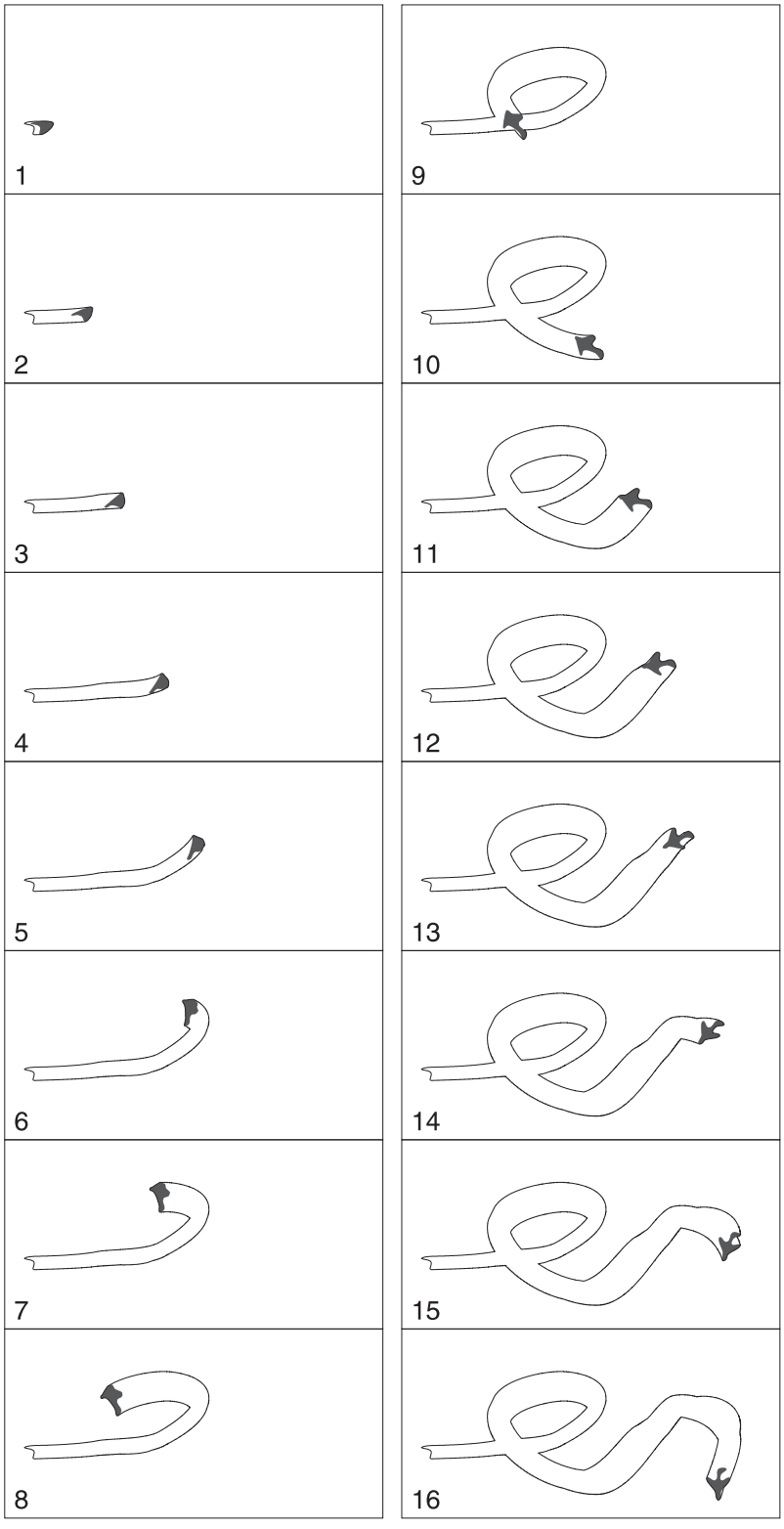


Fig. 20. Incrementally generating the boundary of a general sweep

algebraic curves, and they may have complex self-intersections. Thus we need approximation; in particular, polygonal approximation makes the boundary construction considerably easier.

Given an initial time t_0, a final time t_e, and the number of discrete time intervals n, a discrete time domain is defined as $D = \{t_j | t_j = t_0 + j\Delta t\}$, where $0 \le j \le n$ and $\Delta t = (t_e - t_0)/n$. From this point of view, (7) and (8) can be discretized, and we can compute an approximate polygonal sweep S^p and its polygonal boundary ∂S^p as follows:

$$S^p = S^p(A^p, M, F) = \bigcup_{t\in D} A^p(t) = \bigcup_{0\le j\le n} A^p_j, \quad (9)$$

$$\partial S^p = \partial S^p(A^p, M, F) = \bigcup_{t\in D}^{\partial} A^p(t) = \bigcup_{0\le j\le n}^{\partial} A^p_j,$$

where A^p_j is a polygonal approximation of $A_j = A(t_j)$.

Equation (9) generates a jagged swept area and its shape is dependent on the sampling density (see Figs. 4 and 5). To generate a smoother boundary, we need to use an envelope approximation method as described in Sect. 3. Using the line sweep $\mathcal{L}_j$ of (3) and the approximate sweep operation of (4) for two consecutive instances A_j and A_{j+1}, we can approximate the general sweep as

$$S^p(t_0) = S^p_0 = A^p_0, \quad (10)$$

$$S^p(t_{j+1}) = S^p_{j+1} = S^p_j \cup \left(A^p_j \vee A^p_{j+1}\right) \quad (11)$$
$$= S^p_j \cup A^p_{j+1} \cup \mathcal{L}_j.$$

Equation (10) is an initialization step and (11) describes an induction step. Thus, our algorithm has a sequential nature. Equation (11) implies that the current sweep S^p_j is incremented to S^p_{j+1} by adding the union $A^p_{j+1} \cup \mathcal{L}_j$ to S^p_j. Figure 20 shows the procedure of incrementally generating the boundary of a general sweep. The algorithm does not compute the envelope curve algebraically. Consecutive instances of a moving object are connected with straight line segments (instead of smooth envelope curve segments).

5 Sweep-related problems

In this section, we present algorithms that can solve problems related to the sweep: unsweeps, Minkowski sums and differences, and constant radius offsets.

5.1 Unsweep

The unsweep is a geometric operation that defines an area belonging to "every" instance of a moving object. In contrast, a sweep operation defines an area belonging to at least one instance of the moving object (see (7)). Figures 21 and 22 illustrate the difference between sweep and unsweep. Sweep generates a area larger than that of the moving object, whereas unsweep generates a smaller area. The unsweep operation may result in an empty area as shown in Fig. 22d. Ilies and Shapiro [13] introduce the unsweep operation. They consider moving objects with a fixed shape. As a set-theoretic dual of the sweep, the unsweep plays an important role in the design of mechanical parts.

Given a rigid motion $M(t)$ and a shape transformation $F(t)$, the unsweep of an object A is defined as follows:

$$U = U(A, M, F) = \bigcap_t M(t) \cdot F(t) \cdot A$$
$$= \bigcap_t A(t). \quad (12)$$

Similarly to the sweep operation, we can construct an unsweep using an envelope approximation of (5). Approximation of an unsweep boundary is given by the following induction step:

$$U^p(t_0) = U^p_0 = A^p_0, \quad (13)$$

$$U^p(t_{j+1}) = U^p_{j+1} = U^p_j \cap \left(A^p_j \wedge A^p_{j+1}\right)$$
$$= U^p_j \cap A^p_{j+1} \cap \left(\mathcal{L}_j\right)^c.$$

5.2 The Minkowski sum

We describe an algorithm for computing the boundary of a Minkowski sum based on the relationship between the sweep and the Minkowski sum. The Minkowski sum of two objects A and B is defined as

$$A \oplus B = \{a + b \mid a \in A,\ b \in B\}.$$

It is easy to check that the Minkowski sum is commutative: $A \oplus B = B \oplus A$.

The boundary of a Minkowski sum is closely related to the translational sweep of an object with a fixed

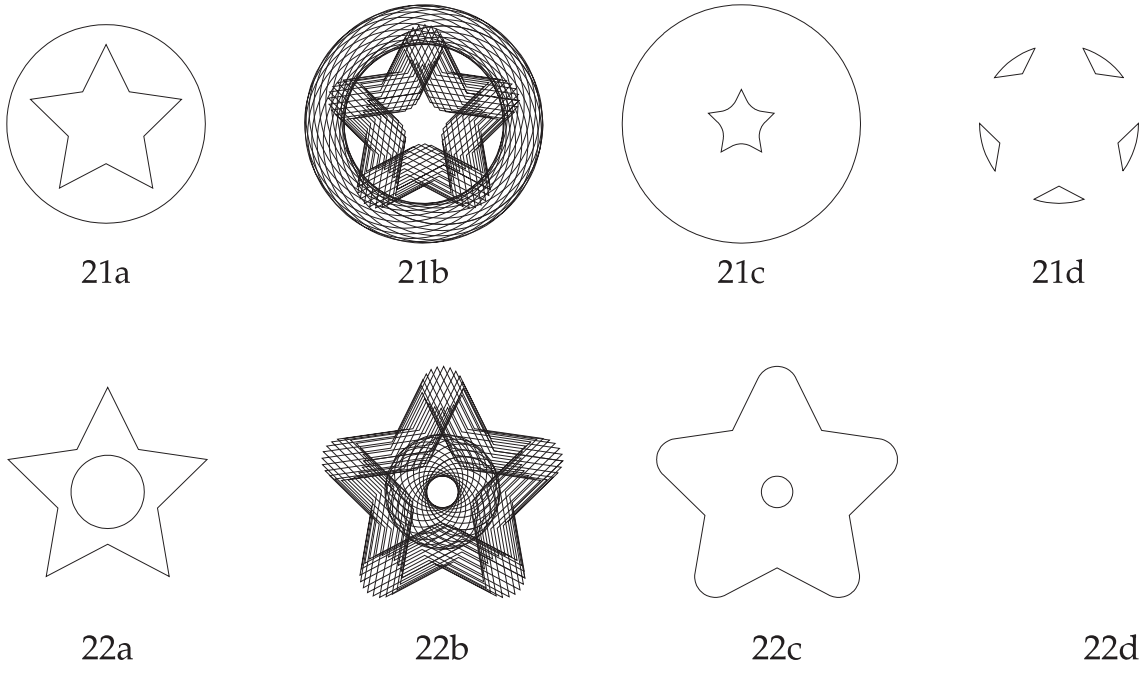


Fig. 21a–d. Sweep and unsweep: **a** a moving object; **b** the object moves along a circle; **c** the sweep; **d** the unsweep
Fig. 22a–d. Sweep and unsweep: **a** a moving object; **b** the object moves along a circle; **c** the sweep; **d** an empty unsweep

shape and orientation. Assume that A is fixed, and B moves around the boundary of A, while the origin O_B of B traces along the boundary of A. (We may assume that the origin O_B is located in the interior of the object B.) Let S_B denote the swept area of B generated by this motion. Then the area S_B includes the difference of $A \oplus B$ and A:

$$S_B \supset ((A \oplus B) - A)\,.$$

Moreover, we can represent the Minkowski sum by a union of two sets:

$$A \oplus B = A \cup S_B. \tag{14}$$

Let $C_A = \bigcup_{i \geq 0} C_{A,i}$ be the boundary curves of an object A, and S_B be the swept area of B tracing along C_A (see (2)). The translational sweep S_B is computed as

$$\begin{aligned} S_{B,i} &= S(B, M_{A,i}, I) = \bigcup_t M_{A,i}(t) \cdot I \cdot B \\ &= \bigcup_t M_{A,i}(t) \cdot B, \\ S_B &= \bigcup_{i \geq 0} S_{B,i}, \end{aligned} \tag{15}$$

where the identity map I means that there is no shape change, and each $M_{A,i}(t)$ is a homogeneous motion matrix representing the translation along each boundary curve $C_{A,i}(t)$:

$$M_{A,i}(t) = \begin{bmatrix} I & C_{A,i}(t) \\ \mathbf{0} & 1 \end{bmatrix}.$$

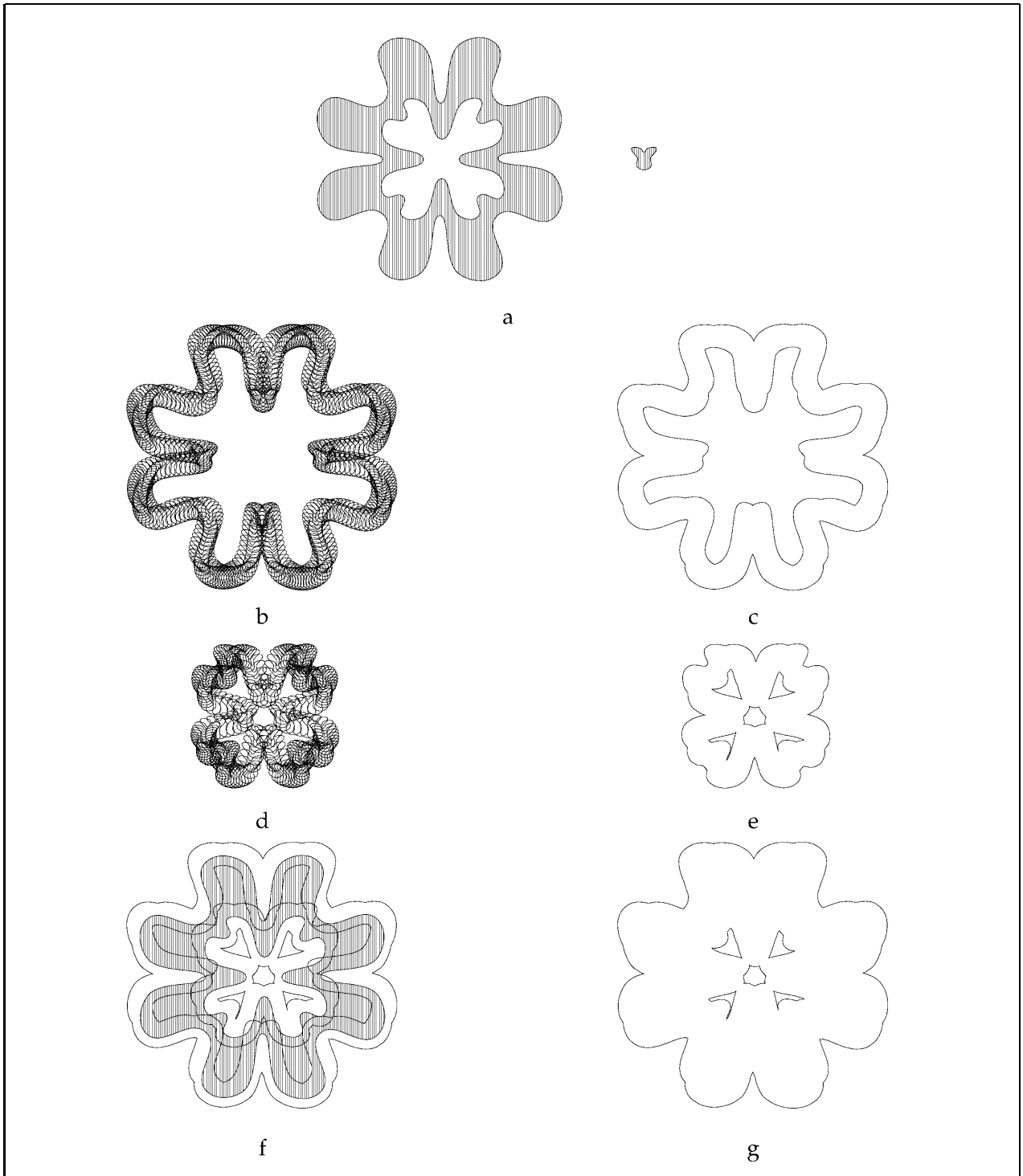
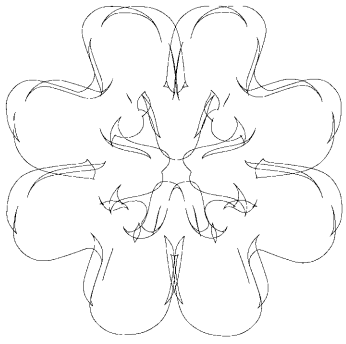
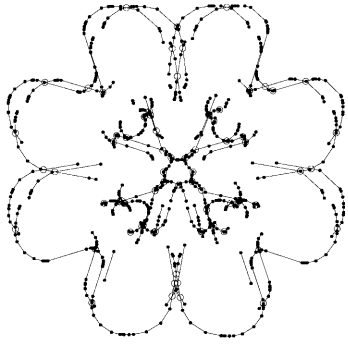


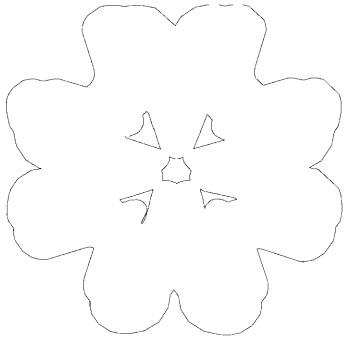
Fig. 23a–g. Construction steps for the boundary of a Minkowski sum: **a** two objects A and B; **b** B moves along the exterior boundary $C_{A,0}$ of A; **c** $\partial S_{B,0}$; **d** B moves along the hole boundary $C_{A,1}$ of A; **e** $\partial S_{B,1}$; **f** $A \cup^{\partial} S_{B,0} \cup^{\partial} S_{B,1}$; and **g** $\partial(A \oplus B)$



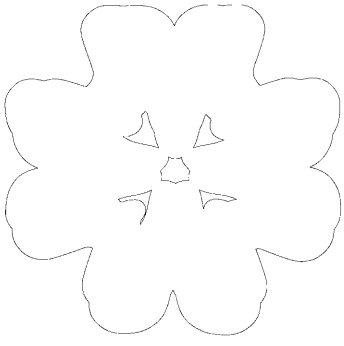
24a



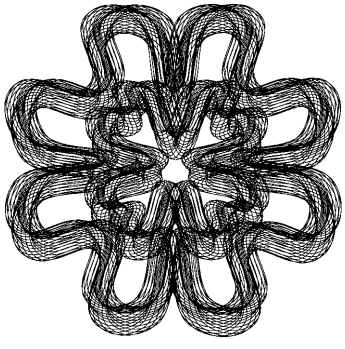
24b



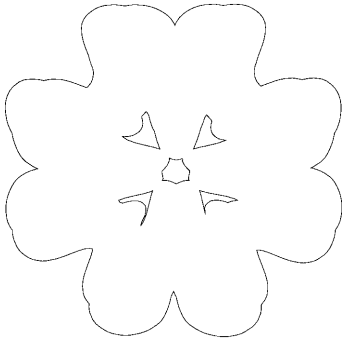
24c



24d



25a



25b

Fig. 24a–d. Computing the boundary of a Minkowski sum based on a convolution curve approximation: **a** convolution curves of two objects of Fig. 23a; **b** polygonal approximations of some convolution curves except some obviously redundant segments; **c** a polygonal approximation of the boundary of a Minkowski sum; **d** a refinement of the result (**c**) using spline curves

Fig. 25a,b. The Minkowski sum $B \oplus A$ for the objects of Fig. 23a: **a** sweeping A around the boundary of B; **b** $\partial(B \oplus A)$

The Minkowski sum of (14) can be reformulated with (15):

$$A \oplus B = A \cup S_B = A \cup \left(\bigcup_{i \geq 0} S_{B,i} \right). \tag{16}$$

Now, we can represent the boundary of a Minkowski sum as

$$\partial (A \oplus B) = A \overset{\partial}{\cup} S_B = A \overset{\partial}{\cup} \left(\bigcup_{i \geq 0}^{\partial} S_{B,i} \right). \tag{17}$$

Thus, we can approximate the boundary of a Minkowski sum following a construction scheme similar to that of (7)–(11):

$$\partial(A \oplus B) \simeq \partial \left(A^p \oplus B^p \right) = A^p \overset{\partial}{\cup} S_B^p, \tag{18}$$

where S_B^p is a polygonal approximation of S_B. Figure 23a shows the construction steps for the boundary of a Minkowski sum. Figure 23 a shows two input objects: a flower-shaped object A and a Y-shaped object B. For boundary curves $C_{A,0}$ and $C_{A,1}$ of the object A, Fig. 23c and e show the corresponding sweeps $S_{B,0}$ and $S_{B,1}$, respectively. The boundary $\partial(A \oplus B)$ of Fig. 23g is constructed by computing $A \cup^{\partial} S_{B,0} \cup^{\partial} S_{B,1}$ (see Fig. 23f).

In contrast to our algorithm, Lee et al. [16] compute the boundary of a Minkowski sum based on a convolution curve approximation, where redundant convolution curves are later eliminated and the remaining curves generate the boundary of a Minkowski sum. For the objects shown in Fig. 23a, Fig. 24 illustrates how the algorithm of Lee et al. [16] works. In particular, Fig. 24b and c show how the redundant loops are eliminated from the convolution curves of Fig. 24a.

By the commutativity of the Minkowski sum, we can sweep the object A around the object B:

$$B \oplus A = B \cup S_A = B \cup \left(\bigcup_{j \geq 0} S_{A,j} \right).$$

In this case, the exact and approximate boundaries of a Minkowski sum are given as

$$\partial (B \oplus A) = B \overset{\partial}{\cup} S_A, \tag{19}$$

$$\partial \left(B^p \oplus A^p \right) = B^p \overset{\partial}{\cup} S_A^p. \tag{20}$$

Note that (17) and (19) represent the same boundaries of a Minkowski sum:

$$\partial(A \oplus B) = A \overset{\partial}{\cup} S_B = B \overset{\partial}{\cup} S_A = \partial(B \oplus A).$$

In (18) and (20), however, the corresponding approximate boundaries of a Minkowski sum are not exactly the same; nevertheless, they have similar shapes when reasonable samplings and polygonizations are used:

$$\begin{aligned} \partial(A^p \oplus B^p) &= A^p \overset{\partial}{\cup} S_B^p \simeq B^p \overset{\partial}{\cup} S_A^p \\ &= \partial(B^p \oplus A^p). \end{aligned} \tag{21}$$

For the objects of Fig. 23a, Fig. 25a shows the result of sweeping A around the boundary of B. By the commutativity of the Minkowski sum, Figs. 23g and 25b must be the same boundary curves.

When we make a union of two alternative approximations of a Minkowski sum, we get a smoother boundary of the Minkowski sum. Figures 26 and 27 illustrate how this approach works. The Minkowski sums of Fig. 26c and e look similar, though they result from the two different sweep sequences of Fig. 26b and d, respectively. When we overlap Fig. 26c and e, their difference can be observed in Fig. 26f. Figure 26g is the union of Fig. 26c and e, and it is less jagged than Fig. 26c and e. Figure 27 is generated with the same objects as Fig. 26, but with sparser samplings. Figure 27f shows that the two boundaries of Fig. 27c and e are very different. However, the boundary of Fig. 27g is significantly less jagged than those of Fig. 27c and e. In Figs. 26 and 27, we make a union of each instance of a moving object without envelope approximation so as to show the effectiveness of our approach. This technique can be applied to other boundary construction problems as well (e.g., for interactive sweeping and offsetting).

5.3 Minkowski difference

The Minkowski difference is closely related to the unsweep with no shape change. As the inverse of the Minkowski sum, the Minkowski difference is defined as follows [24]:

$$A \ominus B = \left(A^c \oplus B \right)^c.$$

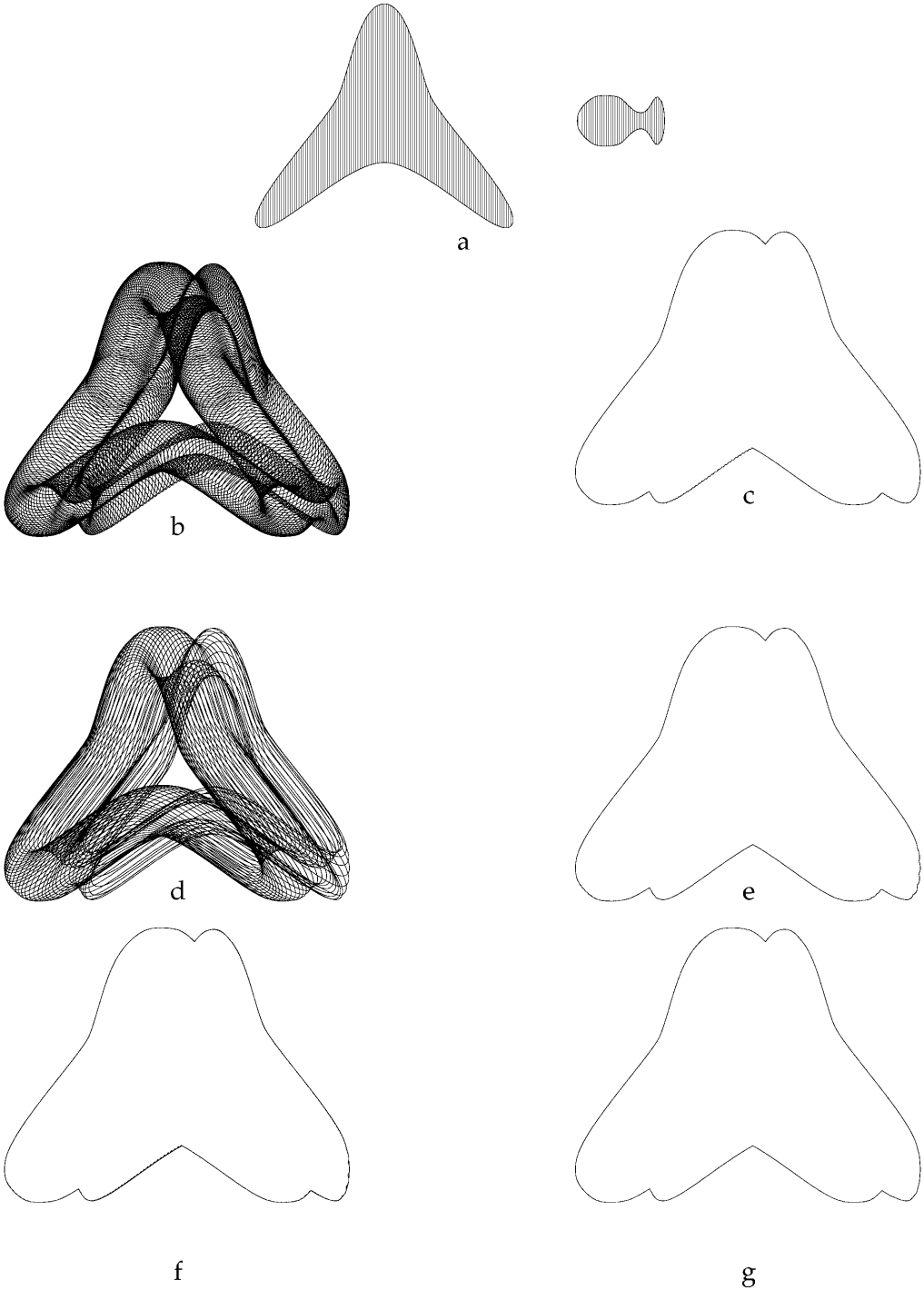


Fig. 26a–g. Constructing the boundary of a Minkowski sum using a blending method for a dense sampling: **a** two objects A and B; **b** B sweeps along the boundary of A; **c** a rough polygonal approximation $\partial S_B^p \simeq \partial(A \oplus B)$; **d** A sweeps along the boundary of B; **e** a rough polygonal approximation $\partial S_A^p \simeq \partial(A \oplus B)$; **f** ∂S_B^p and ∂S_A^p; **g** a blending ∂S_B^p and ∂S_A^p as an approximation of $\partial(A \oplus B)$

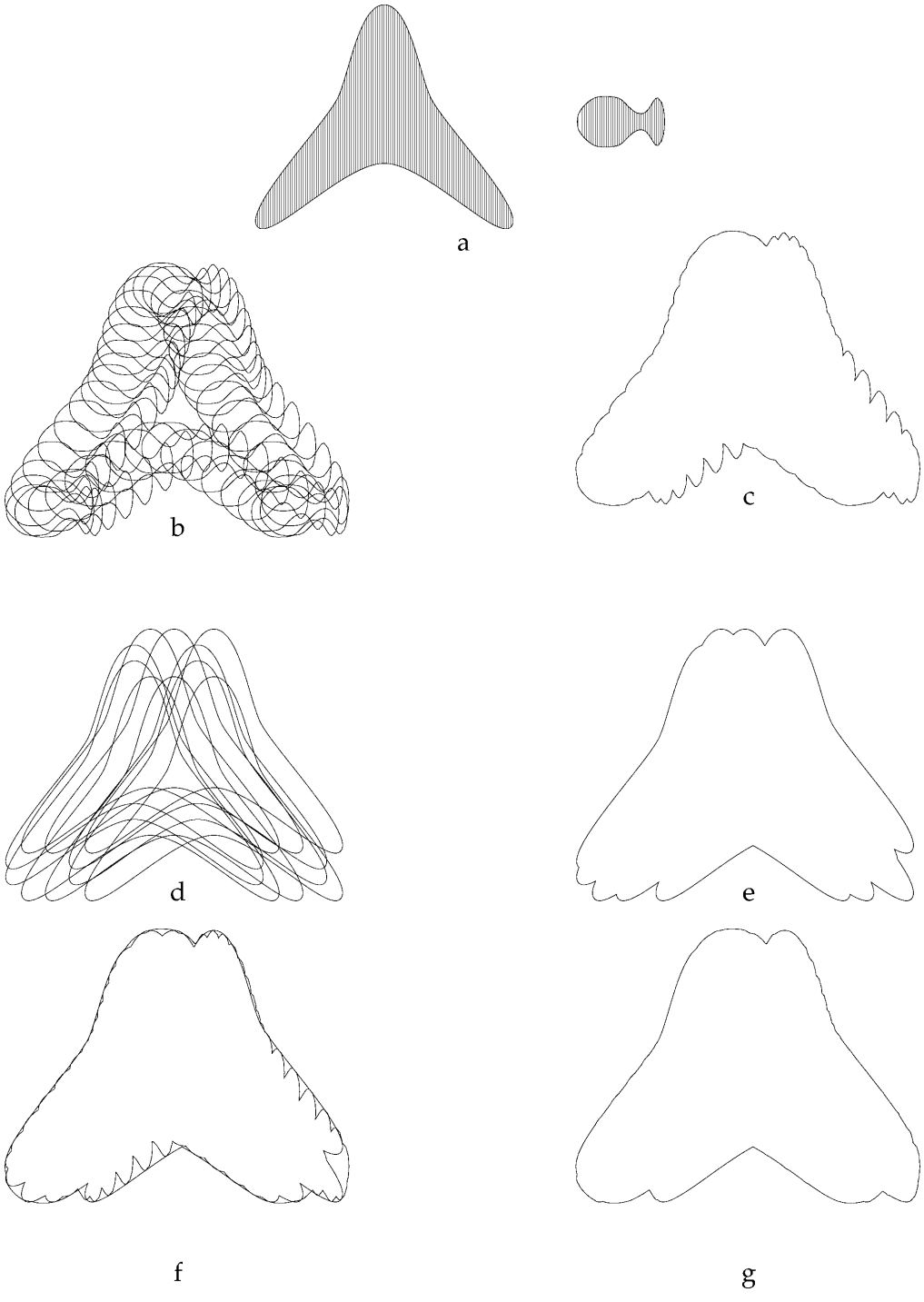


Fig. 27a–g. Constructing the boundary of a Minkowski sum using a blending method for a sparse sampling: **a** two objects A and B; **b** B sweeps along the boundary of A; **c** a rough polygonal approximation $\partial S_B^p \simeq \partial(A \oplus B)$; **d** A sweeps along the boundary of B; **e** a rough polygonal approximation $\partial S_A^p \simeq \partial(A \oplus B)$; **f** ∂S_B^p and ∂S_A^p; **g** the union $\partial S_B^p \cup \partial S_A^p$ as an approximation of $\partial(A \oplus B)$

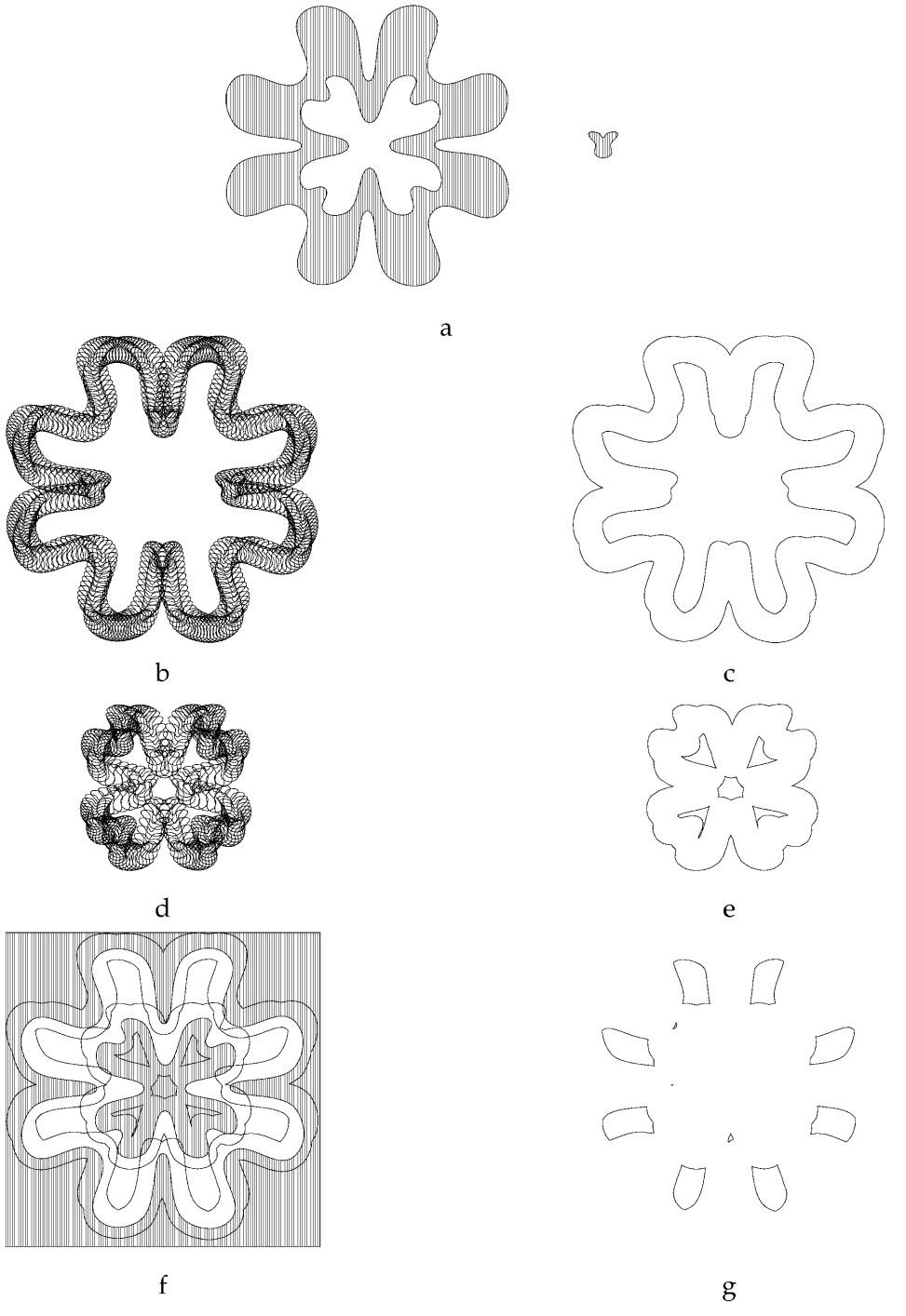


Fig. 28a–g. The boundary of a Minkowski difference: **a** objects A and B; **b** B sweeps along the exterior boundary $C_{A,0}$; **c** $\partial S_{B,0}$ is the sweep boundary of **b**; **d** B sweeps along the hole boundary $C_{A,1}$; **e** $\partial S_{B,1}$ is the sweep boundary of **d**; **f** $\left(A^c \cup^{\partial} S_{B,0} \cup^{\partial} S_{B,1}\right)$; **g** $\partial(A \ominus B)$

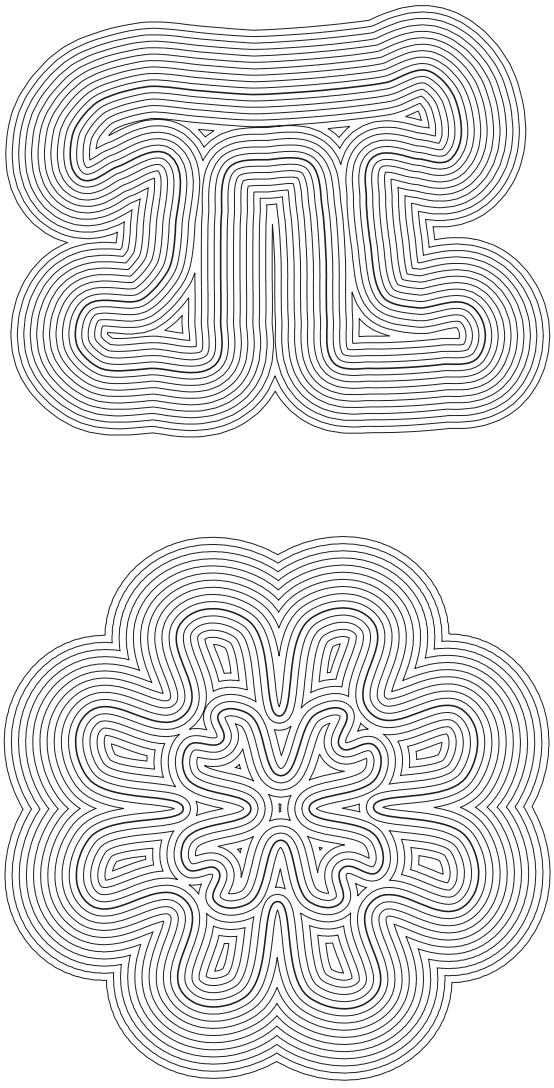


Fig. 29. Examples of offset curves

Let S_B denote the sweep of B around the boundary of A. By complementing both sides of (15), we obtain the unsweep of B^c around the boundary of A:

$$U_{B^c} = (S_B)^c = \left(\bigcup_{i \geq 0} S_{B,i} \right)^c . \tag{22}$$

From (16) and (22), the Minkowski difference can be reformulated as

$$\begin{aligned} A \ominus B &= \left(A^c \cup S_B\right)^c \\ &= A \cap (S_B)^c \\ &= A \cap U_{B^c} \\ &= A \cap \left(\bigcap_{i \geq 0} U_{B^c,i} \right), \end{aligned} \tag{23}$$

where $U_{B^c,i}$ represents the unsweep of B^c around the ith boundary curve of A. The boundary of a Minkowski difference is given as

$$\partial(A \ominus B) = A \overset{\partial}{\cap} U_{B^c} = A \overset{\partial}{\cap} \left(\bigcap_{i \geq 0} U_{B^c,i} \right). \tag{24}$$

We approximate the boundary of a Minkowski difference and those of (7)–(11) similarly.

$$\partial(A \ominus B) \simeq \partial\left(A^p \ominus B^p\right) = A^p \overset{\partial}{\cap} U^p_{B^c},$$

where $U^p_{B^c}$ is a polygonal approximation of U_{B^c}. Figure 28 shows the steps for constructing the boundary of a Minkowski difference. In Fig. 28f, the gray area represents the complement of the object A of Fig. 28a. Note that the Minkowski difference is not commutative. Hence, in contrast to the case of constructing a Minkowski sum, we cannot use the technique of taking the union of two alternative representations to improve the quality of approximation.

5.4 Constant radius offset

The constant radius offset is a special case of the general sweep and the Minkowski sum. Given a curve C, we offset the curve C by sweeping a circular disc D_r of a fixed radius r along the trajectory curve C. The offset boundary of the trajectory curve C is represented as

$$\partial S = \partial S(D_r, M_C, I), \tag{25}$$

where M_C is the translation along the trajectory curve C, and the identity map I means no shape change.

When the disc D_r moves around the boundary of an object A, the offset boundary is given as

$$\partial S = \partial S(D_r, M_{\partial A}, I), \tag{26}$$

where $M_{\partial A}$ is the translation along the boundary of A. Figure 29 shows the examples of offset curves generated for various radii r.

6 Conclusion

In this paper, we presented an algorithm that computes an approximate polygonal boundary of the general sweep for an arbitrary 2D curved object with holes. We define a general sweep as an infinite union of the moving object and generate the sweep boundary in an incremental manner. Compared with previous algorithms, our approximation algorithm has many advantages.

- It is easy to implement the algorithm robustly since it is based on boolean operations on simple polygons.
- Envelope approximation is easy and relatively precise since it is based on line sweeps (which may be nonconvex).
- Incremental sweep construction makes the algorithm useful for interactive shape design, collision detection, and mechanical part design.
- The unsweep can be implemented in the same way as the sweep operation by replacing unions with intersections.
- Utilizing various computational shortcuts, the implemented algorithm demonstrates real-time performance for moving objects with reasonable shape complexity and topological changes as well.

The 3D extension of our algorithm requires a robust implementation of boolean operations on simple polyhedra. Since we start with simple polyhedra, their union and difference are much easier to implement than the construction of the boundary of a 3D swept volume from a set of sweep-envelope surfaces (approximated by complex polyhedra) which may have complex self-intersections. The 3D extension is an immediate goal of our future research.

Acknowledgements. The authors wish to acknowledge the financial support of the Korea Research Foundation in the program year of 1998.

References

1. Abdel-Malek K, Yeh H-J (1997) Geometric representation of the swept volume using Jacobian rank-deficiency conditions. Comput Aided Des 29:457–468
2. Agin G, Binford T (1976) Representation and description of curved objects. IEEE Trans Comput 25:439–449
3. Ahn J-W, Kim M-S, Lim S-B (1993) Approximate general sweep boundary of a 2D curved object. CVGIP: Graph Models Image Process 55:98–128
4. Bajaj C, Kim M-S (1989) Generation of configuration space obstacles: the case of moving algebraic curves. Algorithmica 4:157–172
5. Blackmore D, Leu MC, Shin F (1994) Analysis and modeling of deformed swept volumes. Comput Aided Des 26:315–326
6. Blackmore D, Leu MC, Wang LP (1997) The sweep-envelope differential equation algorithm and its application to NC machining verification. Comput Aided Des 29:629–637
7. Chang T-I, Lee J-H, Kim M-S, Hong SJ (1998) Direct manipulation of generalized cylinders based on B-spline motion. Visual Comput 14:228–239
8. Farin G (1990) Curves and surfaces for computer aided geometric design: a practical guide, 4th edn. Academic Press, San Diego, Calif
9. Foley J, Dam A van, Feiner S, Hughes J (1990) Computer graphics: principles and practice, 2nd edn. Addison-Wesley, Reading, Mass
10. Ghosh PK (1988) A mathematical model for shape description using Minkowski operators. Comput Vision Graph Image Process 44:239–269
11. Greiner G, Hormann K (1998) Efficient clipping of arbitrary polygons. ACM Trans Graph 17:71–83
12. Guibas L, Ramshaw L, Stolfi J (1983) A kinetic framework for computer geometry. In 24th Annual Symposium on Foundations of Computer Science, IEEE, Tuscon, Arizona, pp 100–111
13. Ilies HT, Shapiro V The dual of sweep. Comput Aided Des 31:185–201
14. Jüttler B (1995) Spatial rational motions and their application in computer aided geometric design. In: Dæhlen M, Lyche T, Schumaker LL (eds) Mathematical methods for curves and surfaces. Vanderbilt University Press, La Vergne, Tenn., pp 271–280
15. Kim M-S, Park E-J, Lim S-B (1993) Approximation of variable-radius offset curves and its application to Bézier brush-stroke design. Comput Aided Des 25:684–234
16. Lee I-K, Kim M-S, Elber G (1998) Polynomial/rational approximation of Minkowski sum boundary curves. Graph Models Image Process 60:136–165
17. Lozano-Pérez T (1983) Spatial planning: a configuration space approach. IEEE Trans Comput 32:108–120
18. Margalit A, Knott GD (1989) An algorithm for computing the union, intersection or difference of two polygons. Comput Graph 13:167–183
19. Martin RR, Stephenson PC (1990) Sweeping of three-dimensional objects. Comput Aided Des 22:223–234
20. Parida L, Mudur SP (1994) Computational methods for evaluating swept object boundaries. Visual Comput 10:266–276
21. Pottmann H (1997) General offset surfaces. Neural Parallel Sci Comput 5:55–80
22. Requicha AAG, Voelcker HB (1985) Boolean operations in solid modeling: boundary evaluation and merging algorithms. Proc IEEE 73:30–44

23. Rossignac J, Requicha AAG (1986) Offsetting operations in solid modeling. Comput Aided Geom Des 3:129–148
24. Serra J (1986) Introduction to mathematical morphology. Comput Vision Graph Image Process 35:283–305
25. Wang WP, Wang KK (1986) Geometric modeling for swept volume of moving solids. IEEE Comput Graph Appl 6:8–17
26. Weld J, Leu M (1990) Geometric representation of swept volumes with application to polyhedral objects. Int J Robotics Res 9:105–117
27. Lee JH (1999) General sweep boundary construction based on envelope and set operations. PhD Thesis, CSE dept, POSTECH

JOO-HAENG LEE is a Senior Researcher at the Electronics and Telecommunications Research Institute (ETRI), Korea. He received his BS, MS, and PhD in Computer Science from POSTECH, Korea, in 1994, 1996, and 1999, respectively. His research interests include geometric modeling, computer graphics and animation.

SUNG JE HONG is a Professor in Computer Science at POSTECH, Korea. He received a BS in Electronics Engineering from Seoul National University, Korea, in 1973. He also received an MS in Computer Science from Iowa State University in 1979, and a PhD in Computer Science from The University of Illinois at Urbana-Champaign in 1983. From 1983 to 1989, he was a research staff member at Corporate Research and Development, General Electric. His research interests include VLSI architectures, parallel processing, and computer graphics.

MYUNG-SOO KIM is an Associate Professor in Computer Engineering at Seoul National University, Korea. He received a BS and MS in Mathematics from Seoul National University, Korea, in 1980 and 1982, respectively. He also received MS degrees in Applied Mathematics (1985) and Computer Science (1987) from Purdue University, where he completed his PhD in Computer Science in 1988. Since then, until recently, he was associated with Computer Science, POSTECH. His research interests include geometric modeling and computer graphics