Technical Report 96-1-004, University of Aizu, Japan, 1996.

Implicit Curved Polygons

ALEXANDER A. PASKO
Department of Computer Software, The Univ. of Aizu, Aizu-Wakamatsu City, Fukushima, Japan

ANDREI V. SAVCHENKO
Fraunhofer Center for Research in Computer Graphics, Inc., Providence, RI, USA

VLADIMIR V. SAVCHENKO
Department of Computer Software, The Univ. of Aizu, Aizu-Wakamatsu City, Fukushima, Japan

ABSTRACT

This paper presents an approach to the representation of polygons with straight and curved edges by real functions. A "carrier polygon" is an arbitrary simple polygon with straight edges. Application of our method results in a C^k continuous function $F(x,y)$ with zero value at polygon edges. We combine an efficient algorithm employing a monotone function of a boolean operation, with R-functions. The extension of the algorithm to the case of curved edges provides the possibility of designing a great variety of shapes. We discuss and illustrate the following applications: implicitization of parametric curves, implicit free-form design, shape transformation and reconstruction from cross-sections, depth data generation and synthetic relief carving.

1. INTRODUCTION

Recently the representation of geometric objects by real continuous functions of several variables (or so-called implicit representation) has attracted a great deal of attention in many technical applications and animation. The exact conversion from the boundary representation to a real function is an open problem. This paper deals with such conversion of a two-dimensional polygon. First, we propose an algorithm for a simple polygon bounded by straight line segments. We combine an efficient algorithm employing a monotone function of a boolean operation, with R-functions. Although these techniques are known in computational geometry and geometric modeling, the combination of them gives a basis for new applications. We extend the algorithm to the case of curved polygon edges. Thus, the final algorithm allows us to generate implicit curved polygons. The main stress in this paper is on the geometric applications of the proposed approach. Several application examples show its usefulness: implicitization of parametric curves, interactive design of implicit free-form curves, shape transformation (metamorphosis) and reconstruction from cross-sections, depth data generation and synthetic relief carving.

The problem is stated as follows. A two-dimensional simple polygon is defined by a finite set of segments. The segments are the edges and their extremes are the vertices of the polygon. A polygon is simple if there is no pair of nonadjacent edges sharing a point and convex if its interior is a convex set. We consider the problem of representation of polygons by continuous real functions $F(x,y)$ taking zero values at polygon edges. The algorithm should satisfy the following requirements:

- It should provide exact polygon description as a zero set of a real function;
- No points with zero function value should be inside or outside a polygon;
- It should allow processing a simple arbitrary polygon without any additional information.

We give here a brief overview of the works dealing with this problem.

1.1 Distance based methods

Singh and Parent [16] generalize the blobby model [2, 19] for objects with explicitly defined boundaries such as polyhedrons and other B-rep objects. A skeleton point is placed inside the polygon to define a field function which has values 1 at the point and 0 at the object boundary. The function value at a given point is calculated using a distance ratio. The distance ratio is computed by taking the ratio of the shortest Euclidean distance from the given point to the skeleton point and to the boundary. It requires the polygon to be star-shaped with any boundary point "visible" from the skeleton point. Otherwise, several skeleton points have to be placed. A similar approach is discussed in [3] with any topological object defined as a skeleton by the user and placed inside the polyhedron. The drawback of both these methods is that the automatic skeleton generation is not provided.

An interesting approach to the polygon implicitization has been proposed in [7]. The authors use the distance field function defined as the distance from the given point to the closest point on the contour. Calculation of the distance for an arbitrary point leads to an extremely time consuming algorithm. That is why the authors have used a discrete point grid to approximately represent the polygon.

1.2 Function fitting

A polygon can be approximately represented by fitting a real function to the vertices of the polygon. Although existing methods have been developed for surface fitting to 3D points, they are directly applicable to the 2D case. To fit a function to given points Bajaj et al. [1] solve a constrained minimization problem for some distance criteria. The authors use an algebraic surface defined implicitly by a single polynomial equation. Muraki [9] proposed to apply the ‘blobby’ model to fit scattered points. The number and location of the primitives are found by minimization of an 'energy function'. This method is extremely time consuming. Lim et al. [8] describe a technique for a blended union of spherical primitives. Fitting an implicit solid model to a data set is formulated as a sequence of non linear

optimization problems of an increasing number of variables. They show the capability of the proposed procedure through sliced data sets of a femur. This algorithm is also extremely time consuming (from 8 to 35 hours on an HP-700 series workstation).

1.3 Set-theoretic representations

A convex polygon can be represented as the intersection of half-planes. Then, to convert this set-theoretic representation to a single real function one can use min/max [12] or more complex R-functions [13, 14]. A concave polygon needs to be represented by set-theoretic operations on convex polygons or half-planes. There are three basic approaches to do this: cell partition, convex decomposition, and monotone boolean formula. The cell partition [15, 6] results in a concave polygon represented by the set-theoretic union of its convex parts (see Fig. 1a). Note that convex polygons in this representation share common edges (or their segments) belonging to the interior of the initial polygon (segment S_{25} in Fig. 1a). If Constructive Solid Geometry (CSG) is applied, these edges are eliminated by the regularized set-theoretic operations. If one applies min/max or other R-functions to construct a single function, such partition of the polygon results in "internal zeroes". The function will have zero value at all internal edges. Fig. 1b presents the function constructed for the separation example (Fig. 1a) with function values shown as heights. Fig. 1c shows a contour map of the non-negative function values. The following R-functions are used here to construct the single defining function of the polygon:

$$\text{Intersection} \quad (\wedge) \qquad f_1 \wedge f_2 = f_1 + f_2 - \sqrt{(f_1^2 + f_2^2)},$$

$$\text{Union} \quad (\vee) \qquad f_1 \vee f_2 = f_1 + f_2 + \sqrt{(f_1^2 + f_2^2)}. \tag{1}$$

These functions have C^1 discontinuity when both arguments are equal to zero. If C^k continuity is necessary, the following R-functions can be applied:

$$f_1 \wedge f_2 = (f_1 + f_2 - \sqrt{(f_1^2 + f_2^2)})(f_1^2 + f_2^2)^{3/2},$$

$$f_1 \vee f_2 = f_1 + f_2 + \sqrt{(f_1^2 + f_2^2)})(f_1^2 + f_2^2)^{3/2}. \tag{2}$$

One can observe the crease in the function surface and corresponding behavior of the contour lines caused by the internal zero in the segment S_{25}. Internal zeroes are not acceptable for several reasons:

- Points of internal zeroes can be incorrectly classified as boundary points;
- Some operations on implicit objects (blending, offsetting, metamorphosis) suppose distance-like behavior of the defining function and internal zeroes can cause incorrect results of these operations;
- The surface defined by the resulting function can be used in some applications including aesthetic design but a creased surface like in Fig. 1b is not satisfactory.

The convex decomposition of a concave polygon results in the representation by set-theoretic differences between convex polygons [18, 17]. First, a convex hull of the whole given polygon has to be found. Then, the polygon can be represented as the convex hull with one or several inner regions subtracted. These inner regions can be processed in the same way with generation of lower levels of decomposition. The result is the representation by a CSG-like tree with convex polygons as leaves. Note that direct application of min/max or other R-functions to this representation leads to the appearance of "external zeroes" similar to "internal zeroes" with the disadvantages discussed earlier.
The monotone formula approach discussed in the next section provides the defining function without internal and external zeroes. The contour map of the function that results from this method is shown in Fig. 1d.

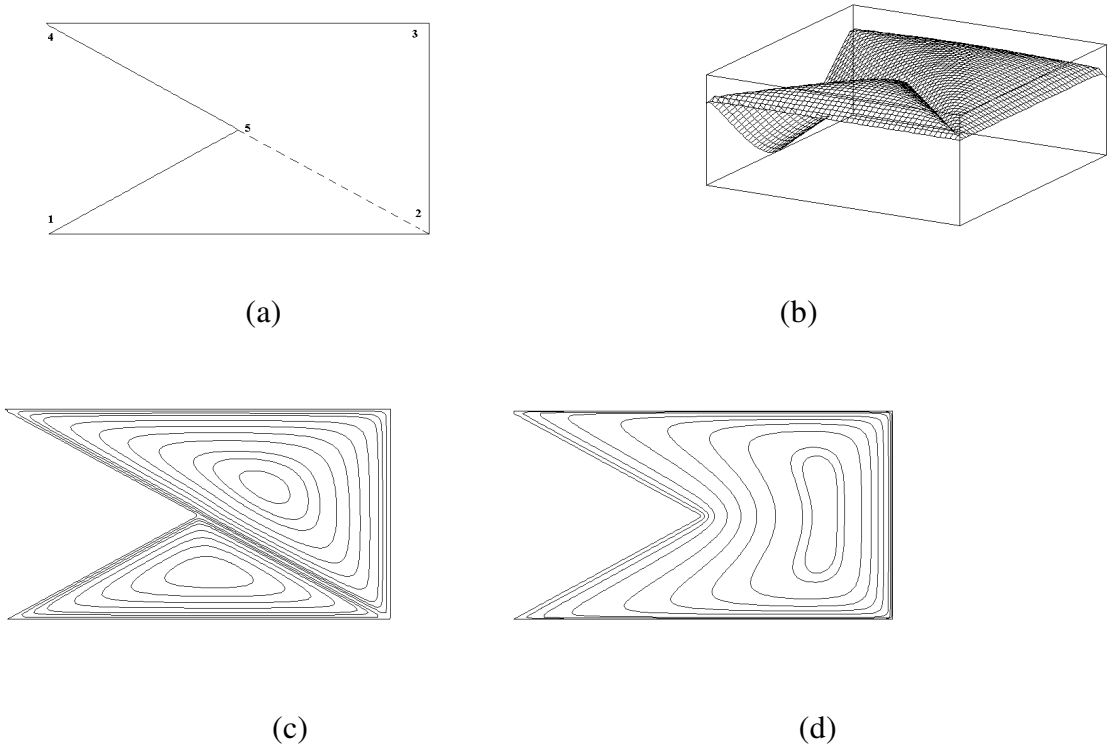


Figure 1. (a) A concave polygon with the segment S_{25} used for the cell partition; (b) the surface of the defining function constructed for the separation example (a); (c) a contour map of the surface (b) for the non-negative function values with "internal zeroes"; (d) a contour map of a function without "internal zeroes" generated using a monotone formula.

2. MONOTONE FORMULA AND DEFINING FUNCTION

In this section we describe the monotone formula construction, defining function evaluation algorithm, and its extension to the case of curved polygon edges.

2.1 Monotone formula

Rvachev [13] and Peterson [10] independently proposed to represent a concave polygon by a set-theoretic formula where each of the supporting half-planes appears exactly once and no additional half-plane is used. Dobkin et al. [4] presented an efficient algorithm for deriving this representation from a given polygon. This approach does not generate any internal or external zeroes when applying R-functions. It is illustrated in Fig. 2. A counter-clockwise ordered sequence of coordinates of polygon vertices $A_1(x_1, y_1)$, $A_2(x_2, y_2)$,..., $A_n(x_n, y_n)$ serves as the input. Also coordinates are assigned to a point $A_{n+1}(x_{n+1},y_{n+1})$ coincident with $A_1(x_1,y_1)$. It is obvious that the equation

$$f_i \equiv -x(y_{i+1} - y_i) + y(x_{i+1} - x_i) - x_{i+1}y_i + x_i y_{i+1} = 0 \qquad (3)$$

defines a line passing through points $A_i(x_i,y_i)$ and $A_{i+1}(x_{i+1}, y_{i+1})$; f_i is a function positive in an open region Ω_i^+ and negative in an open region Ω_i^- located respectively to the left and to the right of the line. When the region Ω^+ is bounded by a convex polygon, it can be given by the logical formula

$$\Omega^+ = \Omega_1^+ \cap \Omega_2^+ \cap ... \cap \Omega_n^+ . \quad (4)$$

If Ω^+ is an external region of a convex polygon then

$$\Omega^+ = \Omega_1^+ \cup \Omega_2^+ \cup ... \cup \Omega_n^+ . \quad (5)$$

Let us consider a concave polygon shown in Fig. 2a and a tree representing its monotone formula in Fig. 2b. Actually, the approach to the monotone formula construction is similar to the convex decomposition discussed in the previous section. Note that the tree in Fig.2b has the convex polygon $A_1A_2A_{10}A_{11}$ as a root (level 0), the polygon $A_2A_6A_{10}$ at the level 1, and the polygons $A_3A_4A_5$ and $A_7A_8A_9$ at the level 2. The internal region Ω^+ of the initial polygon is defined by the following formula:

$$\Omega^+ = \Omega_1^+ \cap \left(\Omega_2^+ \cup \left(\Omega_3^+ \cap \Omega_4^+\right) \cup \Omega_5^+ \cup \Omega_6^+ \cup \left(\Omega_7^+ \cap \Omega_8^+\right) \cup \Omega_9^+\right) \cap \Omega_{10}^+ \cap \Omega_{11}^+ .$$

This formula is especially nice in that each region is presented in it only once. It is worth emphasizing that the set-theoretic operation applied to a region is determined by the tree level which this region belongs to. Because $\Omega_2^+ \equiv \Omega_5^+$ and $\Omega_6^+ \equiv \Omega_9^+$ (see Fig.2a), it can be simplified as follows:

$$\Omega^+ = \Omega_1^+ \cap \left(\Omega_2^+ \cup \left(\Omega_3^+ \cap \Omega_4^+\right) \cup \Omega_6^+ \cup \left(\Omega_7^+ \cap \Omega_8^+\right)\right) \cap \Omega_{10}^+ \cap \Omega_{11}^+ . \quad (6)$$

We implemented the monotone formula construction in about 17K of code in ANSI C including necessary memory management functions. Our data structure is the segmented tree introduced by Bettley (see [11]). We tested the computational cost of the algorithm with the worst case example of "sawteeth" presented in [17]. Polygons with the number 100 to 1000 vertices were processed. The number of half-plane tests obtained from the experiment is perfectly described by the formula $H = 1.24\,N^2 - 11.72\,N + 41.8$, where N is a number of vertices. Calculations took 0.89 seconds on Silicon Graphics Indigo[2] for the "sawteeth" polygon with 500 vertices.

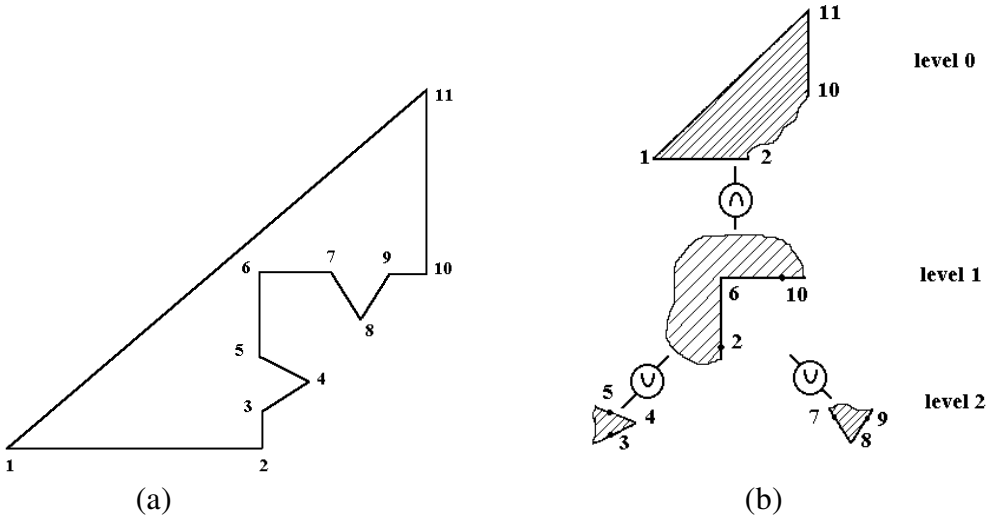


Figure 2. A concave polygon (a) and a tree (b) representing its monotone boolean formula.

2.2 Real function calculation

The final formula for the defining function is obtained by replacing the symbols Ω_i^+ by f_i, symbol $|\,|$ by $\wedge$ and symbol $\cup$ by $\vee$ in the monotone formula. For our example, applying formula (6) results in

$$F = f_1 \wedge (f_2 \vee (f_3 \wedge f_4) \vee f_6 \vee (f_7 \wedge f_8)) \wedge f_{10} \wedge f_{11}. \qquad (7)$$

In practice, the monotone formula construction results in a tree structure (see Fig. 2b). To evaluate the defining function value at the given point, the algorithm traces the tree from the leaves to the root, evaluates defining functions of half-planes and applies corresponding R-function (1) or (2) to them. Fig. 1d shows an example of a contour map of the function generated for the polygon in Fig. 1a using the presented approach. A more complex concave polygon defined by a real function *F(x,y)* is shown in Fig. 3a (black pixels are drawn where $F(x,y) \geq 0$).

2.3 Algorithm extension for curved polygons

Let us now extend the algorithms designed for straight-edged polygons to polygons with curved edges. We define curved edges by the displacement of the straight edges of the carrier polygon. First, we introduce the following function:

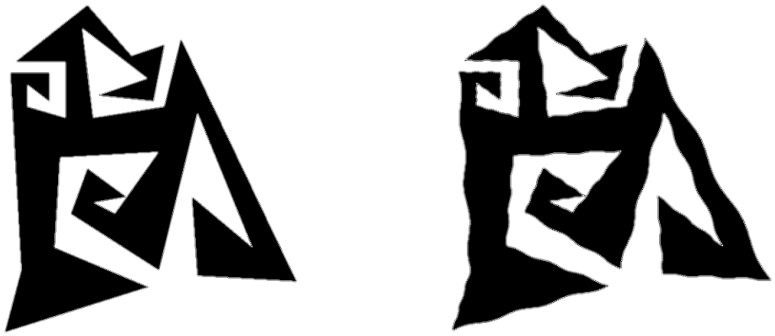
$$\begin{aligned} &u = d(v), \text{ if } \ 0 \leq v \leq 1 \text{ and} \\ &u = 0, \quad \text{if } \ v < 0 \text{ or } \ v > 1, \end{aligned} \qquad (8)$$

where $d(v)$ is a continuous real function defined on the unit segment with $d(0) = 0$ and $d(1) = 0$. Let functions $u = Tu_i\,(x,y)$ and $v = Tv_i\,(x,y)$ define the mapping of the straight edge A_iA_{i+1} onto the unit segment on the v-axis. We describe the half-space going through the curved edge A_iA_{i+1} as follows:

$$\tilde{f}_i\,(x,y) = d(\,Tv_i\,(x,y)\,) - Tu_i\,(x,y). \qquad (9)$$

Actually, the function $d(v)$ can be thought of as the displacement of the current edge along the normal. Note that the carrier polygon and the resulting curved polygon have to be simple, i.e., without intersections of non-adjacent edges. With this definition of supporting half-planes for curved edges one can construct a monotone formula for the carrier polygon and then evaluate the defining function with the use of R-functions applied to functions (9). An example of a curved polygon with the edges displaced by the function $d(v) = \sin(kv)$ is shown in Fig. 3b.

The equation (8) allows the function $d(v)$ to change its sign in the unit segment. It means that the curved edge can intersect the carrier straight edge. The only requirement is a finite number of intersection points. This definition differs from that of the splinegon [5] where the region bounded by the straight and curved edges must be convex. It gives us more freedom for designing implicit curved polygons. On the other hand, thus defined, curved polygons can be converted to the splinegon by including all of the above mentioned intersection points in the carrier polygon. Because the monotone formula is constructed for the carrier polygon and then applied to the curved polygon, intersections of supporting straight lines with other curved edges can occur. It requires these intersection points to be included in the carrier polygon as additional vertices with the reconstruction of the monotone formula. Application of the algorithm extension to curved polygons is discussed in the next section.



(a) (b)

Figure 3. a) Concave polygon defined by a real function - black pixels are drawn where $F(x,y) \geq 0$;b) curved polygon with the edges displaced by the sin(kv) function.

3. APPLICATIONS

3.1 Implicitization of parametric curves

A straightforward approach to approximate implicitization is to polygonize a parametric curve first and then to apply the discussed algorithm for straight-edged polygons. However, it can require a large number of polygon vertices to achieve acceptable accuracy and therefore time consuming function evaluation at a given point. One can apply curved polygons as an alternative.

Let us illustrate this by an example of the implicitization of a parametric cubic spline (Fig. 4). First, the carrier polygon has to be defined. In the case of the parametric cubic spline, control points of the spline (markers in Fig. 4a) are good candidates for polygon vertices. We construct a tentative carrier polygon by listing control points in the same order as for the spline. To construct a correct curved polygon, we need the spline segments to be one-to-one projected onto corresponding straight edges. In other words, if the i-th spline segment is defined by the functions $sx_i(t)$ and $sy_i(t)$, from (8) we can see that the curve

$$\begin{aligned} u &= Tu_i (sx_i(t), sy_i(t)) \\ v &= Tv_i (sx_i(t), sy_i(t)) \end{aligned} \qquad (10)$$

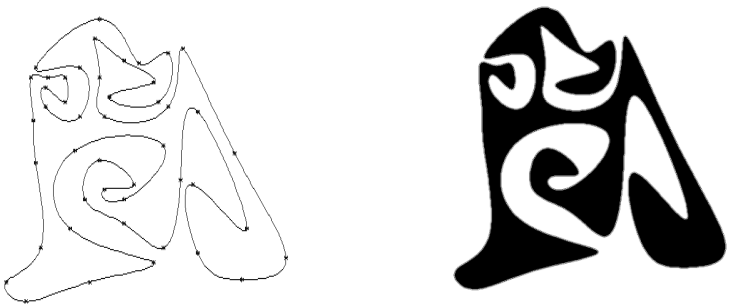
in the uv-plane has to be a graph of a single valued function $u = d(v)$. We check this by finding points of this curve where a tangent line is parallel to the u-axis. If such a point is not an end point of the segment and not an inflection point, it must be included in the final carrier polygon. The second stage mentioned above is to find intersections of straight lines supporting polygon edges with other curved edges. These intersection points have to be included in the carrier polygon as additional vertices with the following reconstruction of the monotone formula. This procedure adds more vertices than necessary. An algorithm for the optimal carrier polygon construction is a subject of further research.

Once the carrier polygon and the monotone formula are constructed, a problem arises during the function evaluation in the given point (x,y). Since the parametric curve (10) defines the displacement function (8) for the spline segment, we have to solve the equation $Tv_i (x,y) = Tv_i (sx_i(t), sy_i(t))$ in terms of t. We use the bisection algorithm to find the root t_0. Then the displacement function value $d(v) = Tu_i (sx_i(t_0), sy_i(t_0))$ can be calculated. Finally, we get the implicit form for the spline segment:

$$\tilde{f}_i (x,y) = Tu_i (sx_i(t_0), sy_i(t_0)) - Tu_i (x,y). \qquad (11)$$

Generalization of this approach to arbitrary parametric curves needs an algorithm for the tentative carrier

polygon construction that lies out side the scope of this paper.



(a) (b)

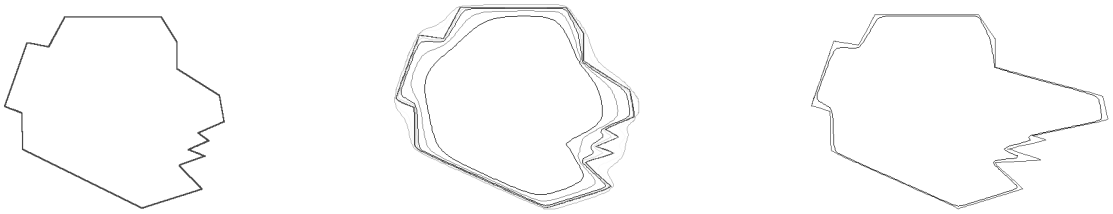
Figure 4. a) Parametric cubic spline; b) Area bounded by a corresponding implicit curve - black pixels are drawn where $F(x,y) \geq 0$;

3.2 Interactive design of implicit free-form curves

One of the well-known advantages of parametric representation is the possibility to design free-form shapes in the interactive manner. Here, we apply our approach to support interactive design of implicit curves illustrated by Fig. 5. First, the user introduces an arbitrary simple polygon (Fig. 5a). Then there are two ways to approach shape modification:

- Changing the offset parameter (Fig. 5b). A smooth curve can be defined by the function $F_c(x,y) = F(x,y) + c$, where $F(x,y)$ is a defining function of the control polygon, $c \neq 0$ is an offset parameter. When the R-functions (7) are applied, $F_c(x,y)$ is C^k continuous everywhere. Fig. 5b shows several offset curves with the different values of the offset parameter.
- Editing the control polygon (Fig. 5c). The vertices of the control polygon can be moved and new vertices can be inserted. These actions require the monotone formula to be reconstructed.

The chosen offset curve follows the changes of the control polygon. It takes 0.6 sec on a Silicon Graphics Indigo^2 workstation to recalculate the curve for the present example.



(a) (b) (c)

Figure 5. Implicit free-form design: a) control polygon; b) choosing the offset parameter; c) editing the control polygon.

3.3 Two-dimensional shape transformation

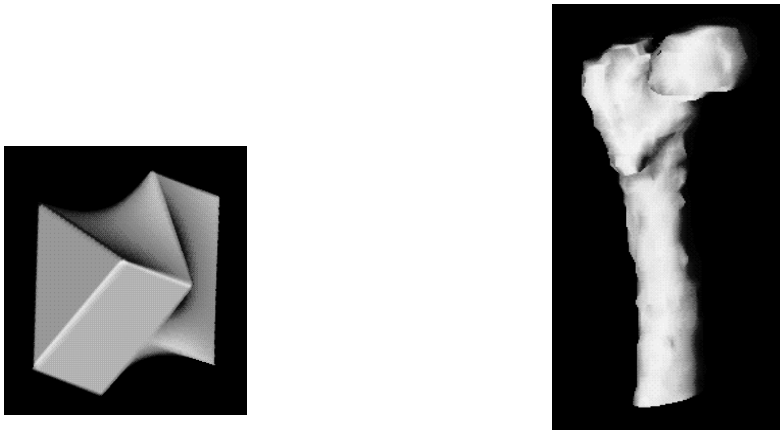
Metamorphosis is a transformation of one shape into another. Given two polygonal models, the transformation determines the in-between shapes. It is illustrated in Fig. 6, where the Escher's bird is transformed into the quadrangle. Here, both polygons are represented by real functions. The simple linear interpolation between two functions determines the in-between shapes. Note that interior color or image can be interpolated together with the shape. It is interesting to note good resemblance between the generated shapes in Fig. 6a and the original shapes in Fig. 6b.

(a) (b)

Figure 6. a) Polygonal shape transformation using interpolation between defining functions b) Original transformation (fragment of M. Escher's "Day and night")

3.4 Three-dimensional reconstruction from cross-sections

As an illustration of the capability of our approach Fig. 7 represents a 3D object reconstructed from a series of parallel cross-sections. Fig. 7a illustrates reconstruction of a solid from three cross-sections. Two triangles and one square contours have been used. We apply linear interpolation between real functions generated for cross-sections by our algorithm. Fig. 7b presents the reconstruction of a femur from a set of 53 parallel cross-sections obtained from medical images produced by computer tomography. The proposed algorithm can generate highly concave and branching objects.



(a) (b)

Figure 7. a) Reconstruction of a solid from three parallel cross-sections; b) Femur reconstructed from 53 polygonal slices.

3.5 Generation of depth data and synthetic relief carving

Depth (height) data are extensively used in cutting simulation, digital terrain modeling, prototyping and computer vision. We apply our approach here to generate depth data from a polygonal contour. Then we apply these data to model the process of carving an arbitrary solid.

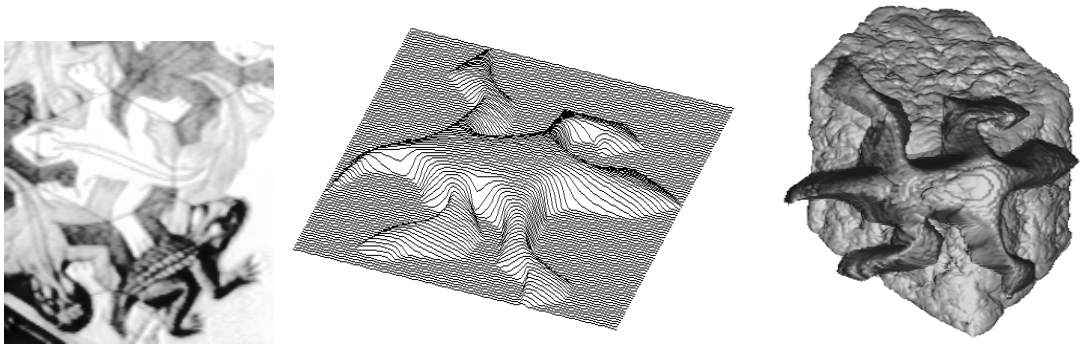
To define depth data in the functional form $z = h(u,v)$ we construct a monotone formula for the given polygonal contour. The resulting defining function $F(u,v)$ (see 2.2) has positive values inside the contour, zero values on the contour and negative values outside the contour. So, we can model depth data by assigning

$$h(u,v) = 0, \text{ if } F(u,v) < 0 \text{ and}$$
$$h(u,v) = k\,F(u,v), \text{ if } F(u,v) \geq 0, \qquad (12)$$

where $k > 0$ is a scaling coefficient. Values of the function $h(u,v)$ can be calculated at grid points and stored in a two-dimensional array. Fig. 8 shows the original image (Fig. 8a) the polygonal contour was taken from, the depth data generated from the contour (Fig. 8b) and its application in carving (Fig. 8c). We model relief carving with the "offsetting along the normal" operation defined as follows:

$$G'(X) = G(X + hN), \qquad (13)$$

where $X = (x,y,z)$, $G(X)$ is a defining function of an arbitrary three-dimensional solid, N is a gradient vector of the function G in X, $h = h(T(X))$ defined by (12), and $T(X)$ is a mapping of X onto the uv-plane. Fig. 8c shows the result of this carving operation with the depth data in Fig. 8b.



(a) (b) (c)

Figure 8. Generation of depth data from a polygon and its application in synthetic carving:
a) original image (a fragment of “Reptiles” by M. Escher);
b) depth data generated from a reptile contour;
c) reptile carved on a stone.

We also use implicit curved polygons to generate functions *G(X),* i.e., to describe objects for applying the carving operation to them. Fig. 9 illustrates carving with modeled depth data on a pillar defined by the extrusion of a curved polygon.

Figure 9. Synthetic carving on a pillar modeled by the extrusion of a curved polygon.

5. CONCLUSION

This paper has presented a new approach to defining curved polygons by real functions. It combines the results of computational geometry and the theory of R-functions. The applications discussed and illustrated in the paper show practical usefulness of the proposed algorithms. We see the main advantage of our approach in generating implicit models of complex objects with further modification of them by different operations. For example, our algorithm along with the offset-operation results in the technique of synthetic carving that can be used in aesthetic design.

For visualization of three-dimensional objects we use ray tracing adapted to general implicits and polygonization. When applying our approach to a large polygonal data set, it may be time-consuming and even impractical for fast visualization. For example, polygonization of the reconstructed femur (Fig. 7b) takes 111 seconds for calculating 17312 polygons on a Silicon Graphics Indigo^2 workstation. On the other hand, parallel implementation of the visualization algorithms on the network of workstations has shown practically linear speed-up with an increasing number of processors.

An optimal algorithm for the parametric cubic spline implicitization and its generalization to arbitrary parametric curves will be a subject of future research. Also, we suppose that our 3D reconstruction scheme where C^k continuity is provided will benefit the simulation of dynamic interaction of moving rigid bodies. We would like to use such functional descriptions of reconstructed solids for the segmentation of 3D volumetric objects as well.

REFERENCES

[1] Bajaj C., Ihm I., Warren J. Higher order interpolation and least-squares approximation using implicit algebraic surfaces, *ACM Transactions on Graphics*, Vol. 11, No. 4, 1992, pp. 327-347.
[2] Blinn J.F. A generalization of algebraic surface drawing, *ACM Transactions on Graphics*, Vol. 1, No. 3, 1982, pp.235-256.
[3] Dischler J.-M., Ghazanfarpour D. A geometrical based method for highly complex structured textures generation, *Computer Graphics Forum*, Vol. 14, No. 4, 1995, pp.203-215.
[4] Dobkin D., Guibas L., Hershberger J., Snoeyink J. An efficient algorithm for finding the CSG representation of a simple polygon, *SIGGRAPH'88, Computer Graphics*, Vol. 22, No. 4, 1988, pp.31-40.
[5] Dobkin D.P., Souvaine D.L. Computational geometry in a curved world, *Algorithmica*, vol.5, 1990, pp.421-457.
[6] Hoffmann C. On the separability problem of real functions and its significance in solid modeling, *Computational Algebra, Lecture Notes in Pure and Applied Mathematics*, vol.151, Marcel Dekker, 1993, pp.191-204.
[7] Jones M.W., Chen M. A new approach to the construction of surfaces from contour data, *EUROGRAPHICS'94, Computer Graphics Forum*, Vol. 13, No. 3, 1994, pp.75-84.
[8] Lim C., Turkiyyah G., Ganter M., Storti D. Implicit reconstruction of solids from cloud point sets, *Third Symposium on Solid Modeling and Applications*, C.Hoffmann and J.Rossignac (Eds.), 1995, pp. 393-402.
[9] Muraki S. Volumetric shape description of range data using 'blobby' model, *SIGGRAPH'91*, *Computer Graphics*, Vol. 25, No. 4, 1991, pp. 227-235.
[10] Peterson D. Halfspace representation of extrusions, solids of revolution, and pyramids, *SANDIA Report SAND84-0572*, Sandia National Laboratories, Albuquerque, NM, 1984.
[11] Preparata F.P., Shamos M. I. Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.
[12] Ricci A. A constructive geometry for computer graphics, *The Computer Journal*, Vol. 16, No. 2, 1973, pp. 157-160.
[13] Rvachev V.L. Methods of Logic Algebra in Mathematical Physics, Naukova Dumka, Kiev, 1974.
[14] V. Shapiro. Real functions for representation of rigid solids, *Computer Aided Geometric Design*, Vol. 11, No. 2, 1994, pp. 151-160.
[15] Shapiro V., Vossler D.L. Separation for boundary to CSG conversion, *ACM Transactions on Graphics*, Vol. 12, No. 1, 1993, pp.35-55.
[16] Singh K., Parent R. Polyhedral shapes as general implicit surface primitives, *Technical Report OSU-CISRC-5/94-TR24*, Ohio State University, 1994.
[17] Tor S.B., Middleditch A.E. Convex decomposition of simple polygons, *ACM Transactions on Graphics*, Vol. 3, No. 4, 1984, pp.244-265.
[18] Woodwark J.R., Wallis A.F. Graphical input to a Boolean solid modeller, *CAD 82*, A.Pipes (Ed.), Brighton, U.K., 1982, pp.681-688.
[19] Wyvill G., McPheeters C., Wyvill B. Data structure for soft objects, *The Visual Computer*, Vol. 2, No. 4, 1986, pp.227-234.