Polygon-to-Function Conversion for Sweeping

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 describes an approach to the representation of polygons by real functions and its application to sweeping. We combine an algorithm employing a monotone function of a boolean operation, with R-functions. Application of this method results in a continuous function *F(x,y)* with zero value at polygon edges. We discuss and illustrate different sweeping techniques with the use of functionally defined generators.

1. INTRODUCTION

Recently the representation of geometric objects by real continuous functions of several variables (so-called *implicit representation* or *F-rep*) 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. We discuss an algorithm for a simple polygon bounded by straight line segments. We combine an 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 discuss and illustrate in this paper the application of the conversion for sweeping by a planar contour. The resulting swept solid is represented by a single real function of three variables which is defined analytically or procedurally. This functionally defined solid can be a subject for further operations such as set-theoretic, blending, generalized offsetting, metamorphosis and others [10].

The polygon-to-function conversion 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 [19] generalize the skeleton based functions [2, 24] 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 Approximation with wavelets

An approximate method of boundary-to-function conversion was proposed in [22]. The method uses the binary raster/voxel representation as an inermediate one between the boundary and the real function. First, the boundary is rasterized. Then, the area enclosed by the boundary is fill by one of the raster filling algorithms. This conversion step generates a characteristic function taking value 1 at inside/boundary points and value 0 outside the area. This function is discontinuous at the boundary points. The next step of the conversion consists in smoothing the characteristic function using wavelets. The authors pay specific attention to do not displace the boundary points while constructing the smooth real function.

1.3 Function fitting

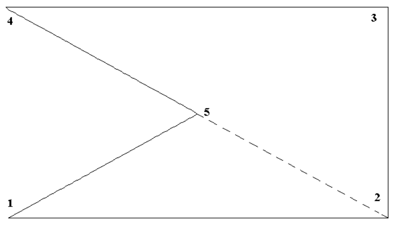
A polygon can be approximately represented by fitting a real function to the vertices of the polygon. Although existing methods have been developed for surface fitting to 3D points, they are directly applicable to the 2D case. To fit a function to given points Bajaj et al. [1] solve a constrained minimization 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).

1.4 Set-theoretic representations

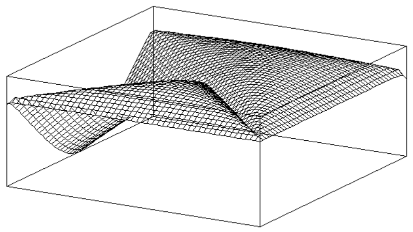
A convex polygon can be represented as the intersection of half-planes. Then, to convert this set-theoretic representation to a single real function one can use min/max [13] or more complex R-functions [14, 17]. A concave polygon needs to be represented by set-theoretic operations on convex polygons or half-planes. There are three basic approaches to do this: cell partition, convex decomposition, and monotone boolean formula. The cell partition [18, 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:

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

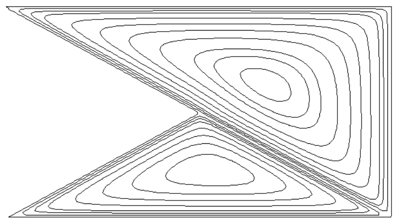
Union $(\vee)$ $\quad f_1 \vee f_2 = f_1 + f_2 + \sqrt{(f_1^2 + f_2^2)}.$ (1)



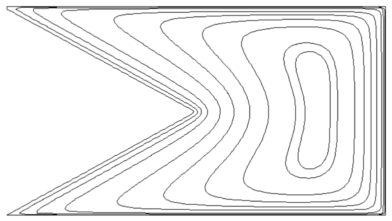
(a)



(b)



(c)



(d)

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.

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:

$$f_1 \wedge f_2 = (f_1 + f_2 - \sqrt{f_1^2 + f_2^2})(f_1^2 + f_2^2)^{k/2},$$
$$f_1 \vee f_2 = (f_1 + f_2 + \sqrt{f_1^2 + f_2^2})(f_1^2 + f_2^2)^{k/2}. \quad (2)$$

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 F-rep 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 [23, 21]. 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.

2. MONOTONE FORMULA AND DEFINING FUNCTION

In this section we describe the monotone formula construction and the defining function evaluation algorithm. Rvachev [14] and Peterson [11] 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

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

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

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

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

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

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 $A_1A_2\,A_{10}A_{11}$ as a root (level 0), the polygon $A_2\,A_6A_{10}$ 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:

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

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 [12]). We tested the computational cost of the algorithm with the worst case example of "sawteeth" presented in [21]. Polygons with the number 100 to 1000 vertices were processed. The number of half-plane tests obtained from the experiment is well approximated 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.

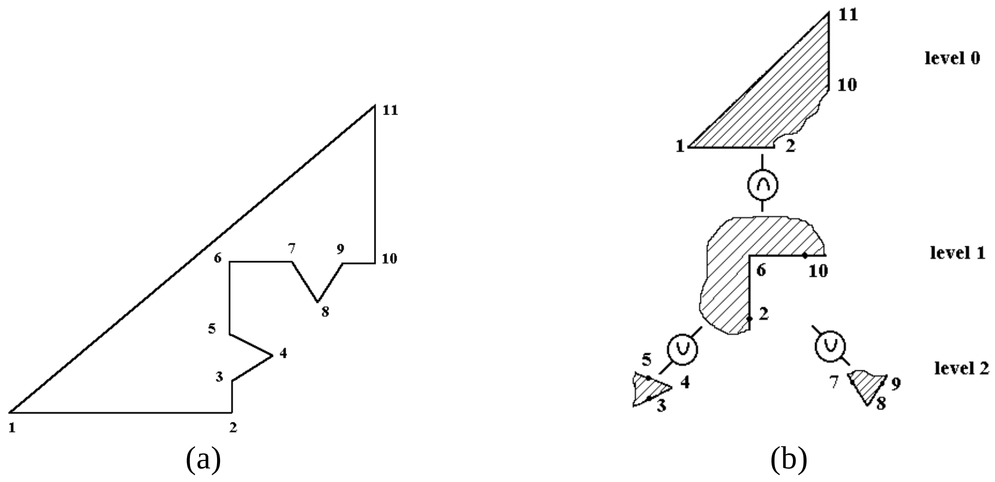


Figure 2. A concave polygon (a) and a tree (b) representing its monotone boolean formula.

The final formula for the defining function is obtained by replacing the symbols Ω_i^+ by f_i, symbol $\cap$ 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 $F(x,y) \geq 0$).

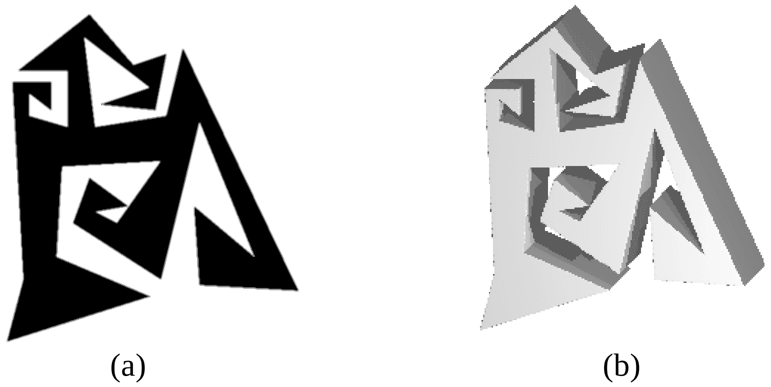


Figure 3. a) Concave polygon defined by a real function - black pixels are drawn where $F(x,y) \geq 0$;
b) translational sweeping (extrusion) with the functionally defined generator.

3. SWEEPING WITH CONVERTED POLYGONS

Sweeping is one of the most powerful operations in geometric modeling. There are two basic types: sweeping by a planar generator and sweeping by a moving solid. We investigated sweeping by a moving solid in [20]. In this paper we deal with sweeping by a planar generator, which is quite different in its mathematical formulation and modeling algorithms. The result of this type of sweeping is defined as a 3D solid swept by a planar surface patch moving along some trajectory. Usually, two basic cases are distinguished. They are translational sweep (extrusion), i.e. a sweep along a straight line; and rotational sweep (revolution), i.e. a sweep about an axis.

The basic idea is to use a polygon converted to a real function as a generator and to construct a real function defining the swept solid. In this case sweeping operation becomes closed on F-rep. Let the function $f_1(x,y)$ defines some polygon on the *xy*-plane. The function $f_2(z) = z \wedge (a-z)$ defines the segment [0,a] on the *z*-axis. Let us consider the following definition of a 3D object:

$$f_3(x,y,z) = f_1(x,y) \wedge f_2(z) \qquad (8)$$

This object is defined as an intersection of the infinite cylinder with the boundary $f_2(x,y) = 0$ and two halfspaces $z \geq 0$ and $a \geq z$. This definition corresponds to the translational sweeping of the polygon along the segment. Applying (8) with R-functions (1) or (2) we obtain a real continuous function defining the swept object. Fig. 3b shows a thus defined 3D object swept by the concave polygon from Fig.3a. Note that *z*-coordinate serves here as a parameter of the sweeping trajectory. Therefore, more general sweeping can be modelled by making the polygon generator dependent on *z*. For example, sweeping by the polygon moving in the direction orthogonal to *z*-

axis can be described as

$$f_3(x,y,z) = f_1(x - \phi_1(z), y - \phi_2(z)) \wedge f_2(z) \tag{9}$$

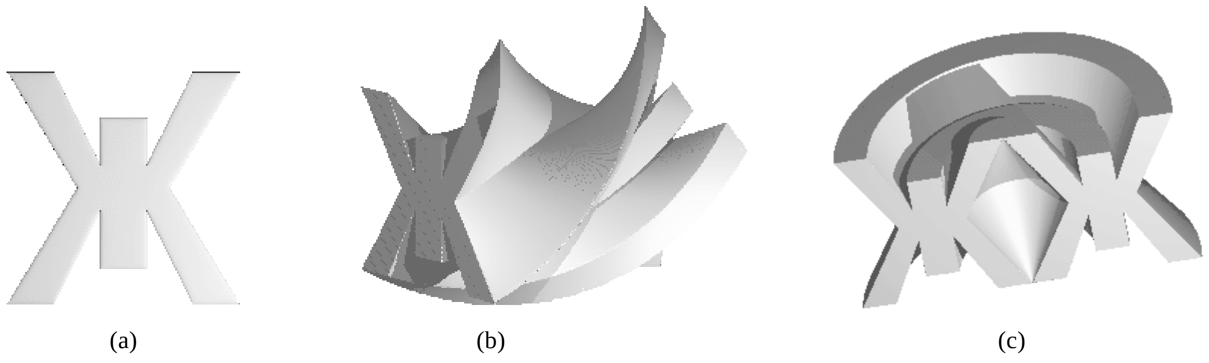

(a) (b) (c)

Figure 4. a) Polygonal boundary of a Cyrillic letter converted to a real function;
b) translational sweeping of the letter changing its orientation;
c) rotational sweeping;

More general type of sweeping with the polygon changing its position, orientation and shape (sometimes called a *generalized cylinder*) can be described with the coordinates transformation:

$$x' = \phi_1(x,y,z),$$
$$y' = \phi_2(x,y,z), \tag{10}$$

where ϕ_1 and ϕ_2 are arbitrary continuous functions. Substituting (10) in (8) we get the following expression for the general type sweeping:

$$f_3(x,y,z) = f_1(\phi_1(x,y,z), \phi_2(x,y,z)) \wedge f_2(z) \tag{11}$$

Fig.4 shows two examples of the general type sweeping by a polygon converted to a real function (Fig.4a). Fig.4b illustrates the case of the polygon changing its orientation during motion. Here, the coordinate transformation (10) is rotation about *z*-axis with the rotation angle linearly dependent on *z*:

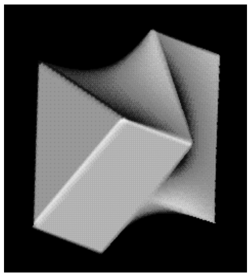
$$\theta = kz, c = cos(\theta), s = sin(\theta),$$
$$x' = xc + ys$$
$$y' = -xs + yc$$

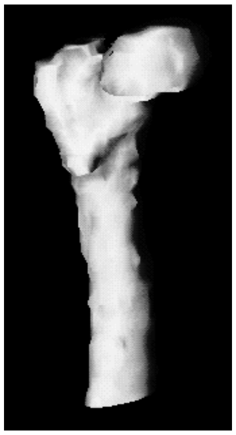
Fig.4c illustrates the rotational sweeping. It is defined by means of the translational sweeping (9) and the coordinate transformation (10) which is the mapping from the Cartesian coordinate system to the cylindrical one:

$$x' = \sqrt{x^2 + y^2}$$
$$y' = arctan(y / x)$$

In other words, this transformation means that *x*-coordinate of the translational sweep is interpreted as the angle, and its y-coordinate as the radius of the cylindrical coordinate system.

As an illustration of the capability of our approach Fig. 5 represents 3D objects reconstructed from parallel cross-sections. This reconstruction can be thought as extrusion with variable-shape generator. Fig. 5a illustrates reconstruction of a solid from three cross-sections. Two triangles and one square contours have been used. We apply the linear interpolation between real functions generated for cross-sections by our algorithm. Fig. 5b 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. In the case of two or more branches the cross-section is represented as a union of several polygonal regions separately converted to their defining functions [16].





(a) (b)

Figure 5. a) Reconstruction of a solid from three parallel cross-sections; b) femur reconstructed from 53 polygonal slices.

4. CONCLUSION

An approach presented in this paper combines the results of computational geometry and the theory of R-functions for the exact conversion from a simple polygon to a real function of two variables. The discussed and illustrated application for sweeping helps to extend a set of geometric objects available in shape modeling and computer graphics with real functions. We see the main advantage of this approach in generating “implicit” models of complex objects with further modification of them by different operations.

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. 5b) 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.

We suppose that our 3D reconstruction scheme with C^k continuity provided will benefit the collision detection and simulation of dynamic interaction of moving rigid bodies [15]. 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] Pasko A., Adzhiev V., Sourin A., Savchenko V. Function representation in geometric modeling: concepts, implementation and applications, *The Visual Computer*, vol.11, No.8, 1995, pp.429-446. See also http://www.u-aizu.ac.jp/public/www/labs/sw-sm/FrepWWW/F-rep.html

[11] Peterson D. Halfspace representation of extrusions, solids of revolution, and pyramids, *SANDIA Report SAND84-0572*, Sandia National Laboratories, Albuquerque, NM, 1984.

[12] Preparata F.P., Shamos M. I. Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

[13] Ricci A. A constructive geometry for computer graphics, *The Computer Journal*, Vol. 16, No. 2, 1973, pp. 157-160.

[14] Rvachev V.L. Methods of Logic Algebra in Mathematical Physics, Naukova Dumka, Kiev, 1974.

[15] Savchenko V.V., Pasko A.A. Collision detection for functionally defined deformable objects, *Implicit Surfaces'95, Eurographics Workshop Proceedings*, B.Wyvill and M.-P. Gascuel (Eds.), INRIA, 1995, pp. 217-221.

[16] Savchenko V., Pasko A. Reconstruction from contour data and sculpting 3D objects, *Journal of Computer Aided Surgery*, Vol. 1, Suppl., Proceedings of Second International Symposium on Computer Aided Surgery ISCAS'95, 1995, pp.56-57.

[17] V. Shapiro. Real functions for representation of rigid solids, *Computer Aided Geometric Design*, Vol. 11, No. 2, 1994, pp. 151-160.

[18] Shapiro V., Vossler D.L. Separation for boundary to CSG conversion, *ACM Transactions on Graphics*, Vol. 12, No. 1, 1993, pp.35-55.

[19] Singh K., Parent R. Polyhedral shapes as general implicit surface primitives, *Technical Report OSU-CISRC-5/94-TR24*, Ohio State University, 1994.

[20] Sourin A.I., Pasko A.A. Function representation for sweeping by a moving solid, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, No.1, 1996, pp.11-18.

[21] Tor S.B., Middleditch A.E. Convex decomposition of simple polygons, *ACM Transactions on Graphics*, Vol. 3, No. 4, 1984, pp.244-265.

[22] Velho L., Terzopoulos D., Gomes J. Constructing implicit shape models from boundary data, *Graphical Models and Image Processing*, Vol. 57, No. 3, 1995, pp.220-234.

[23] 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.

[24] Wyvill G., McPheeters C., Wyvill B. Data structure for soft objects, *The Visual Computer*, Vol. 2, No. 4, 1986, pp.227-234.