

# Synthetic carving using implicit surface primitives

A. Pasko<sup>a,\*</sup>, V. Savchenko<sup>a</sup>, A. Sourin<sup>b</sup>

<sup>a</sup>Shape Modeling Laboratory, University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan

<sup>b</sup>School of Applied Science, Nanyang Technological University, Nanyang Avenue, Singapore, Singapore 639 798

## Abstract

Several techniques of computer-aided synthetic carving are presented. We describe both procedural methods for relief carvings and patterned lattices, as well as interactive carving. Different techniques of depth data generation for relief carving are described: polygon-to-function conversion, pattern-dependent interpolation, and ray-casting. All proposed methods are based on using implicit surfaces or, more generally, the function representation of geometric objects. © 2001 Elsevier Science Ltd. All rights reserved.

**Keywords:** Three-dimensional shape modelling; Synthetic carving; Implicit surface; Isosurface; F-rep

## 1. Introduction

Designing complex three-dimensional (3D) shapes is one of the most challenging problems in computer art. Sculpting techniques providing global and local deformations have recently become one of the central themes in computer graphics literature (see, for example, Refs. [2,6,13,15,21]). As was postulated in Ref. [5], “a system which will allow the reshaping of illusory computer material by carving into it, adding material to it or modeling it would open new and exciting creative opportunities”.

Carving is considered as a human activity aimed at producing new shapes by cutting or engraving on initial shapes. Here, we discuss synthetic carving as an application of computer-aided shape modeling using local deformations for removing and/or adding material. The following types of modeling are discussed:

1. Definition of a pattern followed by a projection operation that projects the pattern on to the basic model for carving. A pattern can be defined by discrete depth data or a continuous height function of two variables. Existing 3D painting systems provide means for the depth data preparation [7,22]. Traditionally, depth data has been used to appropriately elevate nodes of a polygonal mesh to model a carved relief [3]. Here, we propose several different ways to generate depth data (polygon-to-function conversion, pattern-dependent interpolation,

and ray-casting a 3D shape) and to use that data to carve on objects constructed of implicit surface primitives.

2. Interactive local modifications of the basic shape using a modeled carving tool (a chisel) and set-theoretic operations. This approach was implemented for the voxel [1,4] and constructive solid geometry (CSG) models [6]. We present interactive carving based on implicit surface primitives.

This paper aims to illustrate our approach to providing shape modeling tools for computer art based in a unified manner on the *function representation* (or *F-rep*) of the underlying model [8]. This representation is a generalization of an implicit surface model and incorporates such different geometric models as skeleton-based implicits, CSG, voxel objects, sweeping, and medial axis. In fact, any object that is defined with an inequality  $f(x, y, z) \geq 0$  can be included in the model. The representation supports different operations such as set-theoretic, offsetting, blending, metamorphosis, and deformations with space and function mappings [17]. The advantage of this model is that the result of any supported operation can be an input for the next operation.

To convert a CSG object to F-rep, one can replace solid primitives (leaves of the CSG-tree) by the corresponding functions, then replace set-theoretic operations by *R-functions* [8,14] and introduce a procedure that evaluates the defining function for the entire object by tracing the obtained F-rep tree. This conversion is illustrated in detail in Section 2.1.1. To convert B-rep to F-rep, we have to convert B-rep to an intermediate voxel or CSG representation. A voxel array can be converted to a continuous function using some interpolation procedure.

\* Corresponding author. Tel.: +81-242-372606; fax: +81-242-372728.

E-mail addresses: pasko@u-aizu.ac.jp (A. Pasko), savchen@u-aizu.ac.jp (V. Savchenko), sourin@computer.org (A. Sourin).

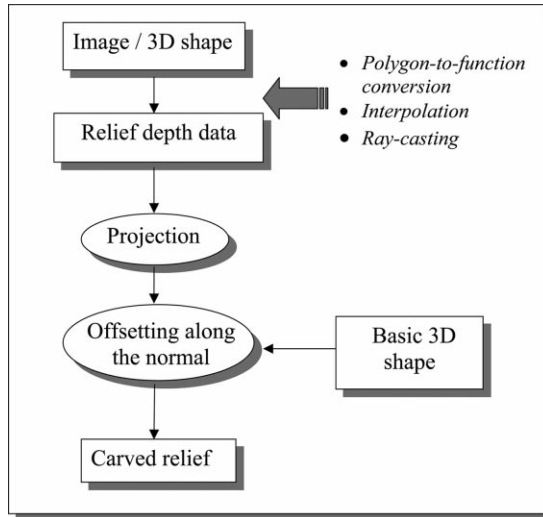


Fig. 1. Relief carving procedure.

In this paper, we describe and illustrate several techniques of F-rep-based computer-aided carving. There are different types of carved shapes: relief carvings, patterned lattices, sculptures and engravings made with chisels. We first propose procedural models for relief carvings and patterned lattices (Section 2), then describe an interactive tool for carving (Section 3), and finally give an example that combines the proposed techniques (Section 4).

## 2. Procedural carving

The function  $f(x, y, z)$  of a 3D shape can be defined with a single equation or with a procedure that evaluates the function at any given point. A procedure for synthetic carving inputs both a defining function of a basic shape and depth data which will become a pattern for carving, and from this it calculates the defining function for the carved shape.

### 2.1. Relief carving

In the case of relief carving (see Fig. 1), we start from a 2D image (line drawing or grayscale raster) or a 3D shape and generate depth data. Different ways of generating such depth data are described in the subsections below. Then, relief carving is modeled with the “offsetting along the normal” operation [8]. The defining function of the result is represented as follows:

$$f' = f(x + dN),$$

where  $f$  is a defining function of the basic 3D shape,  $d$  is an offset distance defined by the generated depth data, and  $N$  is a gradient vector of the function  $f$  in the given point  $x$ . Note that we have reformulated the notion of “offsetting along the normal” by substituting the function gradient for the surface

normal. We describe the following three procedures for generating depth data for carving: polygon-to-function conversion, pattern-dependent interpolation, and ray-casting.

#### 2.1.1. Polygon-to-function conversion

An arbitrary 2D polygon (convex or concave) can be represented by a real function  $f(x, y)$  taking zero value at polygon edges. The polygon-to-function conversion problem is stated as follows. A 2D simple polygon is bounded 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 non-adjacent edges sharing a point, and convex if its interior is a convex set. The polygon-to-function conversion algorithm should satisfy the following requirements:

- it should provide an exact polygon boundary description as the zero set of a real function;
- no points with zero function value should exist inside or outside of the polygon;
- it should allow for the processing of any arbitrary simple polygon without any additional information.

A monotone set-theoretic expression with  $R$ -functions allows such a conversion [10,12,14]. Then, depth data is generated by taking only the positive values of the function (i.e. points inside the polygon). The depth values for points outside the polygon (corresponding to negative function values) are set to zero. Offsetting along the normal modulated by the depth data can be applied to any arbitrary F-rep object to carve the relief.

Now, we describe the monotone formula construction and the defining function evaluation algorithm. Rvachev [14] and Peterson [12] independently proposed representing a concave polygon with a set-theoretic formula where each of the supporting half-planes appears exactly once and no additional half-plane is used. It is illustrated in Fig. 2a and b. A counter-clockwise ordered sequence of coordinates of polygon vertices  $A_1(x_1, y_1), A_2(x_2, y_2), \dots, A_n(x_n, y_n)$  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

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

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  $\Omega_i^+$  and negative in an open region  $\Omega_i^-$  located to the left and to the right of the line, respectively. When the region  $\Omega^+$  is bounded by a convex polygon, it can be given by the logical formula

$$\Omega^+ = \Omega_1^+ \cap \Omega_2^+ \cap \dots \cap \Omega_n^+.$$

If  $\Omega^+$  is an external region of a convex polygon then

$$\Omega^+ = \Omega_1^+ \cup \Omega_2^+ \cup \dots \cup \Omega_n^+.$$

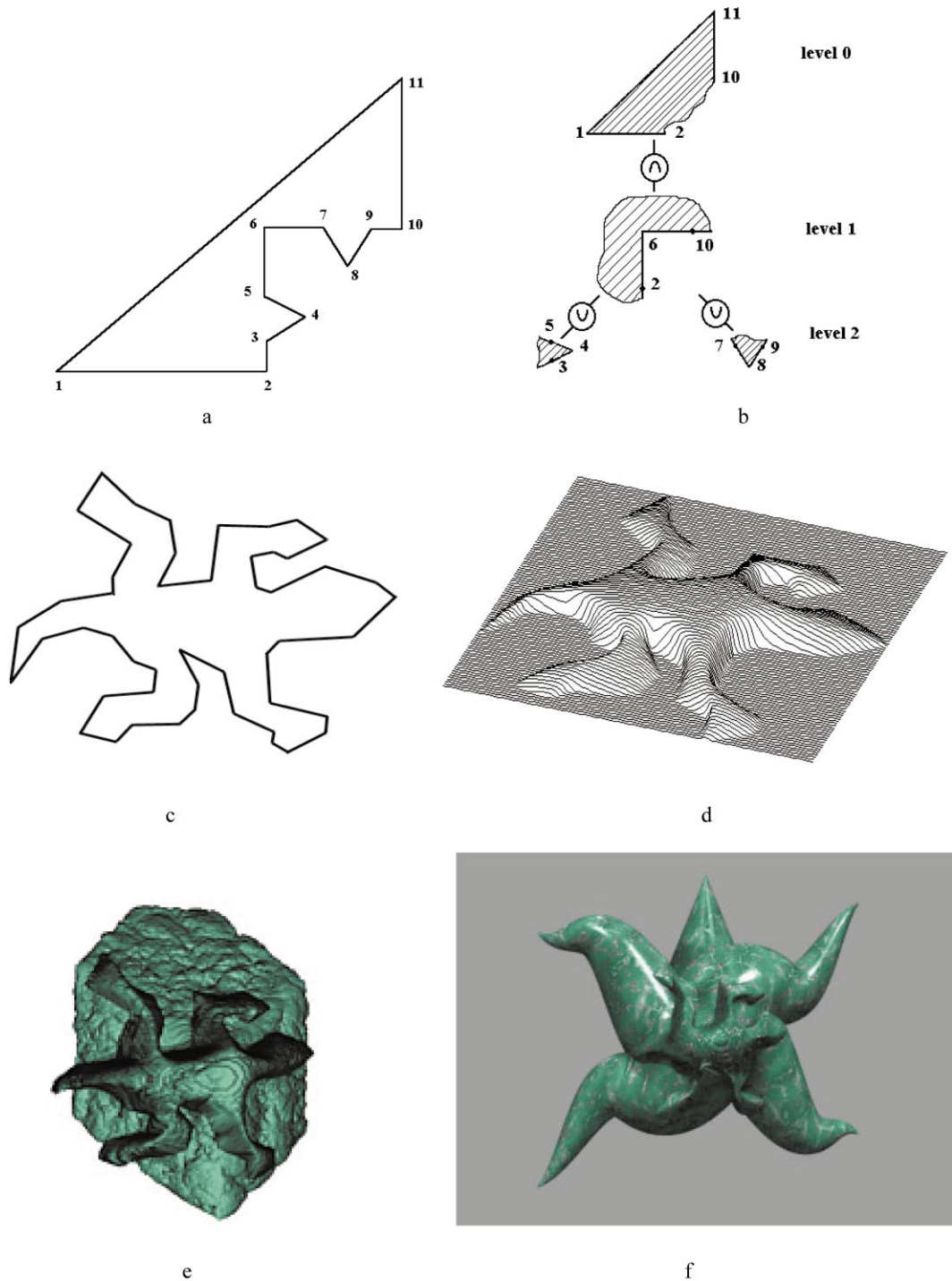


Fig. 2. Relief carving with the polygon-to-function conversion: (a) concave polygon; (b) tree representing monotone set-theoretic formula; (c) initial polygon for relief carving; (d) depth data generated by the polygon-to-function conversion procedure; (e) reptile carved on a stone using offsetting along the normal controlled by depth data; and (f) reptile carved on a medial axis object.

Let us consider the concave polygon shown in Fig. 2a and a tree representing its monotone formula in Fig. 2b. Note that in the tree in Fig. 2b, the convex polygon  $A_1A_2A_{10}A_{11}$  is a root (level 0), the polygon  $A_2A_6A_{10}$  is level 1, and the polygons  $A_3A_4A_5$  and  $A_7A_8A_9$  are level 2. The internal region  $\Omega^+$  of the initial polygon is defined by

the following formula:

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

This formula is especially good in that each region is

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

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

The final formula for the defining function is obtained by replacing the symbols  $\Omega_i^+$  by  $f_i$ , symbol  $\cap$  by  $\wedge$  and symbol  $\cup$  by  $\vee$  in the monotone formula. For our example (Fig. 2a), the defining function for the polygon is

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

where the following  $R$ -functions are used:

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

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

In practice, monotone formula construction results in a tree structure (see Fig. 2b). To evaluate the defining function at a given point, the algorithm traces the tree from the leaves to the root, evaluates the defining functions of half-planes and applies corresponding  $R$ -functions to them.

The steps of the conversion procedure are illustrated in Fig. 2: polygon input (Fig. 2c), depth data generation (Fig. 2d), and relief carving on two different objects (Fig. 2e and f). The basic stone shape in Fig. 2e was modeled by the algebraic sum between the defining functions of a set-theoretic solid and solid noise [9]. The basic shape in Fig. 2f was reconstructed from a medial axis model as described in Ref. [11] and then deformed by a non-linear space mapping. Note that as a result of offsetting along the normal, the reptile's surface follows the features of the basic shapes in both carvings. The final model was rendered with specialized implicit surfaces ray-tracing software [20].

### 2.1.2. Pattern-dependent interpolation

Another way to generate depth data is the pattern-dependent scheme for interpolation of scattered data with the help of the finite element method (FEM). It is used to obtain a smooth approximation of scattered data for the input image. The 2D input data is represented by a set of scattered points obtained from digitizing a hand-made line drawing. These points have arbitrary coordinates  $(x, y)$  and scalar height values assigned to each point based on the gray level, for instance, zero for black and 255 for white pixels. Our experiments with a straightforward approach, such as applying the blurring operation, have shown that it cannot provide smooth carving areas for input line drawings, because of the local character of this algorithm.

We utilize the method of interpolation proposed in Ref. [16] based on the minimum-energy property. The problem of constructing interpolating functions for scattered data is

well known. Here, we do not consider this problem in detail, we only sketch the basic ideas of the variational schema and describe our algorithm.

The problem with constructing interpolating functions by FEM for 2D space  $\Omega$  is stated as follows: given data points  $P_i(x, y)$ ,  $i = 1, 2, \dots, N$ , which are scattered in the sense that there are no assumptions about the disposition of the independent data, and the data set  $r$  is associated with the points, we have to construct a smooth function  $\sigma_h(x, y)$ , which takes on the value  $r_i$  at points  $P_i(x, y) \in \Omega$ , if it is possible, or satisfies the condition:

$$\|A\sigma_h - r\|^2 = \min,$$

where  $A$  is some linear bounded operator. Along with this condition, the function  $\sigma_h(x, y)$  has the minimum energy of all functions that interpolate the values  $r_i$ . This conforms to the following minimum condition, which defines operator  $T$ :

$$\int_{\Omega} [\sigma_{xx}^2 + \sigma_{yy}^2] d\Omega = \min,$$

where the integral is taken over the image space  $\Omega$ .

In our case, we solve the general problem of generating a smooth approximation of the scattered data. For the solution to this problem, two matrices  $T$  and  $A$  have to be constructed. After that, a non-singular linear system of algebraic equations  $(\alpha T + A)\sigma = f$  with the positive symmetric matrix  $(\alpha T + A)$  can be solved with the help of the wide class of iterative or direct methods with  $\alpha$  as a smoothness weight. We use a piecewise linear approximation over the rectangular mesh. The bilinear basis function  $\omega_{ij}(x, y) = \omega_i(x)\omega_j(y)$  corresponds to each node of the rectangular mesh. The solution of the data approximation can be found in the form:

$$\sigma_h(x, y) = \sum_{ij} \sigma_{ij} \omega_{ij}(x, y).$$

Unless we adjust the notation to make vector  $\sigma = \{\sigma_{ij}\}$  one-dimensional, the matrices  $T$  and  $A$  will be four-indexed:  $t_{ijkl} = (T\omega_{ij}, T\omega_{kl})$ ,  $a_{ijkl} = (A\omega_{ij}, A\omega_{kl})$ , or more explicitly

$$t_{ijkl} = \int \int_{\Omega} [\partial\omega_i(x)/\partial x \omega_j(y) \partial\omega_k(x)/\partial x \omega_l(y) \\ + \omega_i(x) \partial\omega_j(y)/\partial y \omega_k(y) \partial\omega_l(y)/\partial y] dx dy,$$

$$a_{ijkl} = \sum_{m=1}^N \omega_i(x_m) \omega_j(y_m) \omega_k(x_m) \omega_l(x_m),$$

where  $P_m$  has coordinates  $(x_m, y_m)$ ,  $m = 1, 2, \dots, N$ . Taking into account that

$$t_{ijkl} = d_{ik} u_{jl} + u_{ik} d_{jk},$$

where  $u_{ps} = \int \omega_p(x) \omega_s(x) dx$ ,  $d_{ps} = \int \omega'_p(x) \omega'_s(x) dx$  and indexes  $p, s \in \{i, j, k, l\}$ , elements of the matrix  $T$  can be calculated.

The reason for the selection of the FEM approximation

of scattered data is that this method is very well formalized and consists of several clearly defined steps. Our pattern-dependent interpolation algorithm consists of the following operations:

1. *Generation of original scattered data.* Pictures are digitized by using a scanner with a resolution of 300 by 300 pixels.
2. *Sorting the data.* At this stage, we put calculated data in a grid of  $100 \times 100$  size according to the chosen network pattern (see Fig. 3a). The reduction of the grid resolution helps to reduce the calculation time and provides additional smoothing of the depth data. Such sorting also allows us to calculate effectively the right part components of the system of linear equations

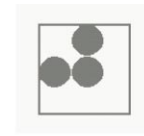
$$f_{ij} = (r, A\omega_j) = \sum_{m=1}^N r_m \omega_i(x_m) \omega_j(y_m).$$

3. *Numerical assembly.* This step calculates the values of all elements in the FEM matrix according to its logical structure defined for the rectangular FEM mesh.
4. *Cholesky factorization.* The problem is reduced to the solution of the system of linear algebraic equations,  $Ax = b$ , where  $A$  is a band matrix of coefficients,  $x$  is a vector of unknown node values and  $b$  is a vector of right parts.
5. *Calculating the right part* of the resulting linear algebraic system. *Solving* lower and upper triangular matrices.
6. *Linear interpolation over the mesh.*

The presented techniques thus far have assumed that the image array being calculated is smaller than the image array being displayed. To this end, the problem is to calculate color/intensity values for each pixel. The measured processing time on an SGI Indigo<sup>2</sup> workstation for the above-mentioned steps is about 1 s. The values obtained in the grid nodes are used as depth data to carve the image on some surface. We must stress that the FEM approximation of scattered image points gives us the option to control the smoothness of the carved shape by using the smoothness weight  $\alpha$ . Fig. 3 shows a relief carved on an elliptic surface with depth data generated from a line drawing using the interpolation scheme presented.

### 2.1.3. Ray-casting

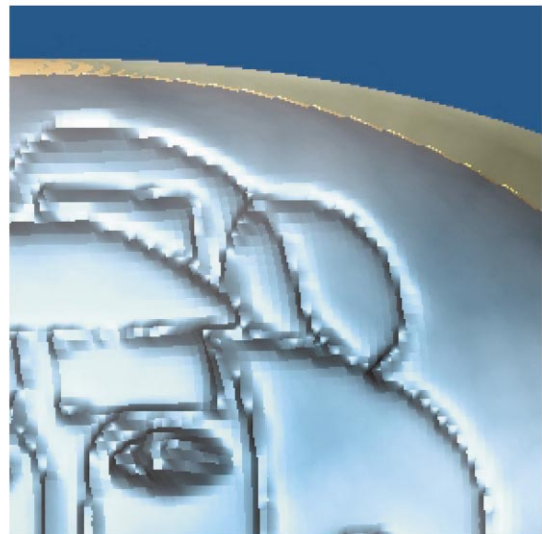
Three-dimensional shapes can also be used to generate depth data for carving. One could propose to simply make a set-theoretic union between the basic shape and the auxiliary 3D shape. Actually, the result can be quite different, because carving with depth data allows the carved shape to follow the features of the basic shape as can be observed in Fig. 2. To generate the depth data, we sample the height of the auxiliary 3D shape with traditional ray-casting at a set of grid points. The distances between the viewing plane and the ray-surface intersection points will provide the desired



a



b



c

Fig. 3. Relief carving with depth data generated by the pattern-dependent interpolation.

depth data. To generate the depth data shown in Fig. 4a, we applied ray-casting to a 3D volumetric head with hair grown and styled on it as described in Refs. [18,19]. The result of carving with this depth data is shown in Fig. 4b. Other representations can also be used to generate depth data using ray-casting or z-buffer algorithms.

### 2.2. Carving patterned lattices

A 2D object defined by a function of two variables can be used not only for relief carving, but also for cutting through a thin 3D shell to produce a patterned lattice. The procedure

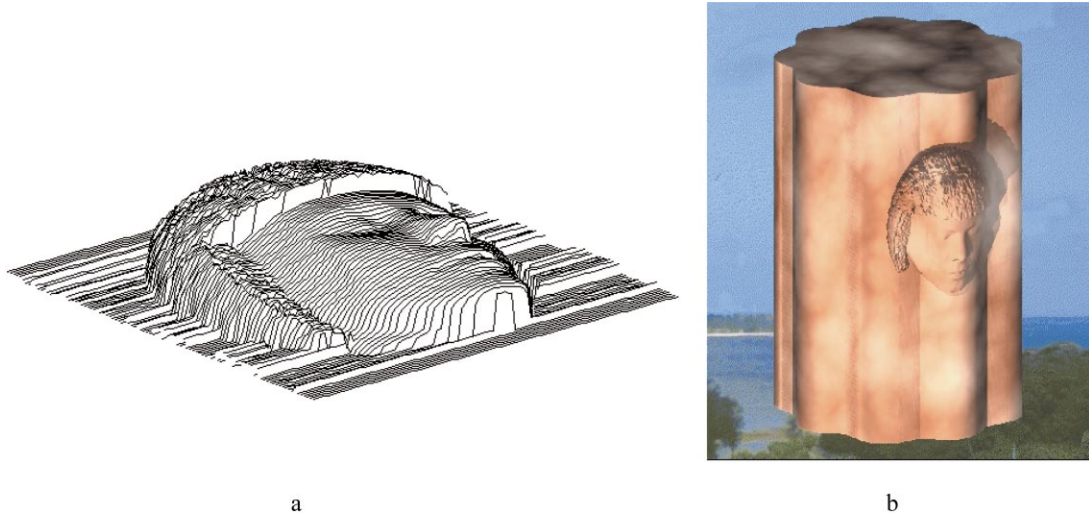


Fig. 4. Relief carving (b) with depth data (a) generated by ray-casting a 3D shape.

of carving a patterned lattice includes the following steps (see Fig. 5):

1. Convert a discrete image pattern into a continuous function of two variables using some interpolation scheme (bilinear or the FEM-based one described in Section 2.1.2).
2. Define a basic 3D shell, which will carry the final lattice.
3. Introduce a projection operation using cylindrical, spherical, or any other coordinate system similar to texture mapping.
4. Apply the projection to the 2D object defined by the continuous function of two variables to generate a 3D swept pattern in space.
5. Intersect the shell with the swept pattern to generate the patterned lattice.

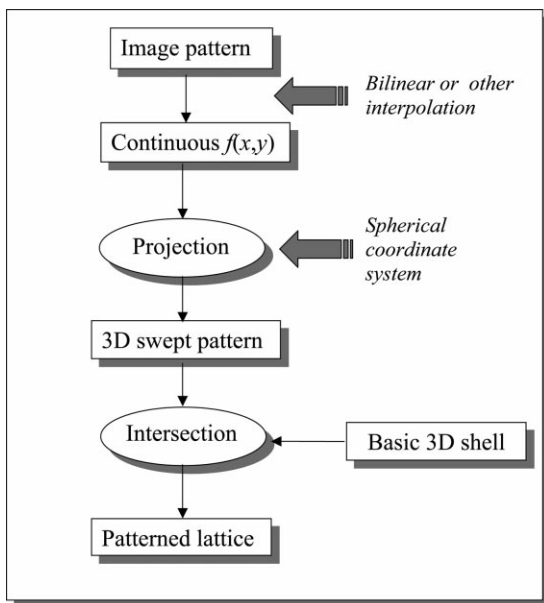


Fig. 5. Procedure of modeling a patterned lattice.

All these steps are expressed in terms of operations on real functions and procedurally defined F-rep models for patterned lattices. Fig. 6b shows an example of a patterned lattice modeled with a simple image pattern (Fig. 6a) multiplied and projected using the spherical coordinate system. The same object is used in the complex example presented in Section 4.

### 3. Interactive carving

In the previous section, we considered synthetic carving where sophisticated procedures create shapes based on parameter values as defined by the user. The user of such procedures may not necessarily be an artist skilled in carving or other artistic techniques. He/she probably should have artistic imagination and must know how to use the modeling software. In this section, we consider another approach to carving using implicit surface primitives where the computer serves only as a virtual tool and neither replaces the artist nor provides any new artistic techniques. The goal of this approach is to allow the artist to work in the virtual environment exactly like, or at least as close to working in real life as possible. The work of art is to be created with virtual models of familiar tools, and familiar real-world techniques are to be used. The advantages of this approach are that the object being created can be made from any virtual material and that there are no problems with “ordering” the many different tools as needed.

Wood carving requires differently shaped cutters (see Fig. 7). Piece by piece, the wood is carved with these cutters. While doing interactive carving, the artist operates virtual cutters chosen from a set of tools with predefined shapes and sizes. If required, the custom-made tools can be redefined by varying the geometric parameters of the cutter models. In synthetic carving, the artist carves in the same way he/she does real carving. The artist chooses the tool,

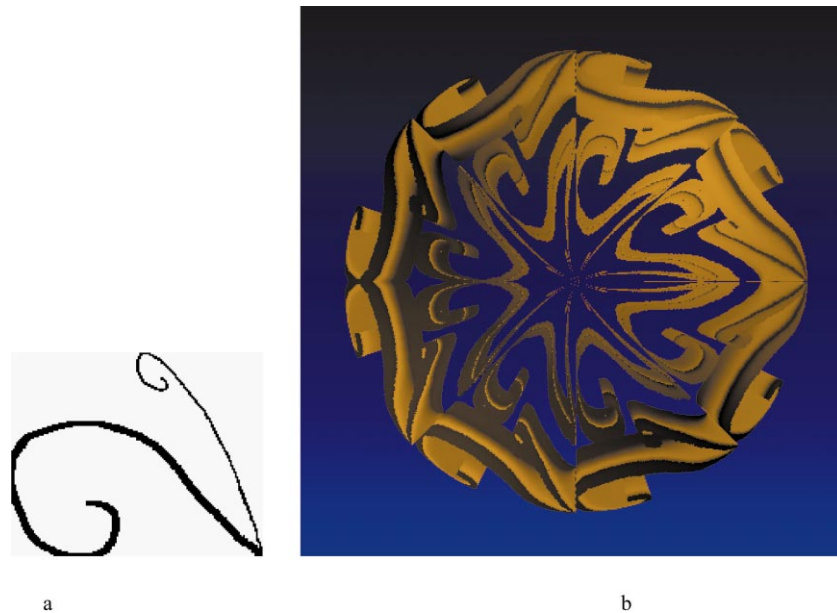


Fig. 6. Carving a patterned lattice: (a) initial image pattern; and (b) 3D patterned lattice.

carves the workpiece, and observes the result. If the result is not acceptable, multilevel undo operations can be used. For better immersion, a graphics tablet can be used. Graphics tablets with pressure-sensitive pens allow the artist to control the depth of carving almost in the same way as in real carving. The program is implemented as a kind of interactive solid modeling tool where the cutter-objects are subsequently subtracted from the workpiece-object. Internally, all the cutters and the workpieces themselves are defined functionally, and the final object is also functionally defined. All operations are implemented as subtractions of solid objects. The final object is represented in the data structure as a binary tree where each node is a set-theoretic operation and the leaves are cutters. For visualization, interactive ray-tracing is used. Since a typical model may contain several thousands of solid objects to be subtracted from the workpiece, ray-tracing may take quite a long time, since it implies many function evaluations for each ray cast. To accelerate this process, only those parts that were affected by the most recent carving are to be redrawn. To estimate the size of the affected area and to detect which cutter instances are involved, the bounding boxes for the

cutters and the spatial organization of the data structure are used. The size of the bounding boxes are about the size of the cutters. This method ensures reasonable redrawing time for the affected areas that does not exceed 0.5 s for complex models, and thus provides both photo-realism and interactivity. The final or interim model of the object can be saved in the de facto standard POV-Ray data format for further high-quality ray-tracing or for later use with other models. We use an extension of POV-Ray that allows us to visualize any object represented with a real function [20]. In Fig. 8a, an example of carving on a lacquered wooden board is presented. In Fig. 8b, we used a voxel head as a workpiece for interactive carving.

The same approach has been extended to virtual making of chased art. When real engraving or chasing on metal is being done, the copper, brass or silver foil is placed on top of a rubber sheet or resin. First, lines and edges are dragged with the stumps (see Fig. 9), which are firmly held in the artist's fist. Next, the foil is turned over and embossed areas are made with other stamps either by rubbing the foil or by hitting it many times. Then, the foil may be turned over again, put on a hard wooden surface, and the background pattern is made. Then, the foil is turned over several times so that the process continues until perfection is achieved. After that, coloring with chemicals remains to be done. Simulating chasing with the computer, we also use functionally based constructive solid geometry where embossing the metal with stamps is simulated by union and subtraction with blending (defined with modified  $R$ -functions [8]). The blending parameters are functions of the geometric size of the stamp and the thickness and stiffness of the metal foil. It allows us to perform a pseudo-physical simulation of the actual process. For example, when doing carving, a graphics tablet should be used for better immersion and a pressure-sensitive pen naturally simulates

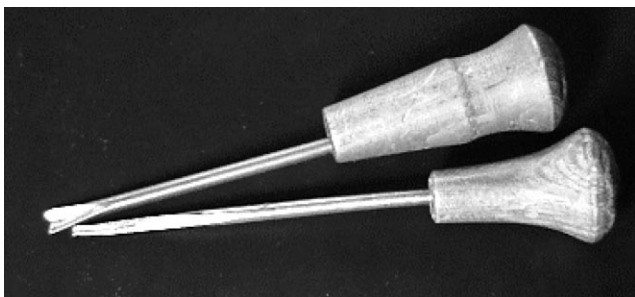


Fig. 7. Cutters used for wood carving.

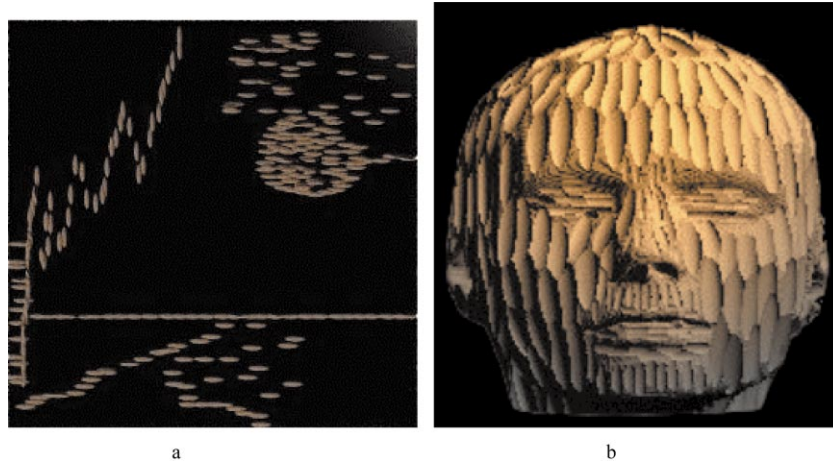


Fig. 8. (a) Interactive carving on a lacquered wooden board; (b) interactive carving on a voxel head.



Fig. 9. Tools used for chasing on metal.

the application of the stamps. Similar to carving, interactive ray-tracing is used only where affected areas are redrawn after each application of the stamps. Here, a more complicated estimation of the size of the affected areas is required when an operation with blending is used, since the result of this operation expands beyond the bounding box of the stamp. In Fig. 10a and b, interactively chased pictures are presented.

#### 4. Application example

In this section we illustrate how the discussed carving techniques can be applied for designing complex 3D scenes in computer art. The image “Geometric mentality” presented in Fig. 11 was created using several F-rep models with synthetic carving and a modification of the POV-Ray ray-tracing program that is able to render implicitly defined surfaces (isosurfaces). Subsequently we explain the variety of models that were used to generate Fig. 11. The central

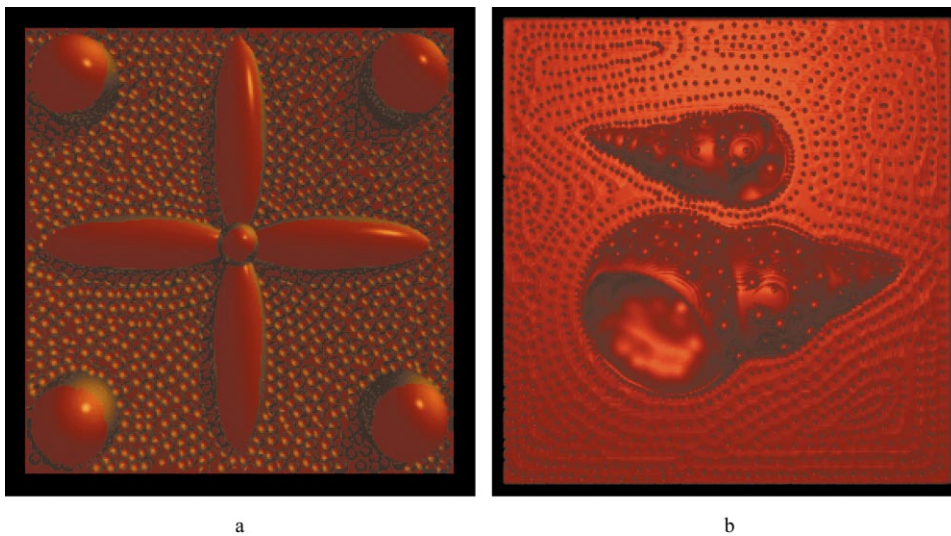


Fig. 10. (a) Chased pattern; and (b) another chased picture “Sea shells”.



Fig. 11. Functionally defined scene “Geometric mentality”.

object of the picture is a head that has a drawer filled with typical solid primitives. It is created by interpolating the head’s voxel data and then applying set-theoretic operations to the thus functionally defined head and solid primitives. Refer to Ref. [19] for the basics of the composition of the voxel data and functionally defined geometric objects. The hair strands are modeled as functionally defined generalized cylinders [18,19]. To make the hairstyle, we apply set-theoretic operations and non-linear transformations to the cylinders. The candle-holder is modeled as a patterned lattice (Fig. 6). The candle is modeled as a cylinder with the top part sculpted by applying the set-theoretic difference with an ellipsoid to remove the undesirable part. The wax “tears” are defined using the 1D medial axis model [11]. The table and the plate are constructive solid objects defined functionally. The “reptiles” sitting on the leg of the table are created with the relief carving (Fig. 2). The background image with the clouds and the stylized comet is created using pattern-dependent interpolation (Section 2.1.2). Because of all the shapes and the scene complexity, the final ray tracing with extended POV-Ray took several hours on a Silicon Graphics Onyx Reality Engine2 workstation.

## 5. Conclusions

We presented new techniques for modeling relief carving and patterned lattices. An interactive software for carving and chasing has been implemented and tested. All described tools for synthetic carving are unified on the foundation of the function representation. The authors believe that this representation and the described carving techniques will allow computer artists to discover a new source of shapes.

The current output of our modeling system is restricted to halftone rendering and polygonization. The final model output using rapid prototyping equipment will be the subject of future work.

## Acknowledgements

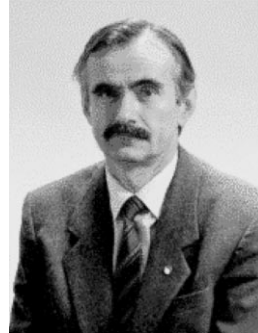
We would like to sincerely thank Eric Fausett for software support and for his help with the stylistic issues of this paper. Comments by Prof. Carlo Séquin and referees helped to extend and improve the paper.

## References

- [1] Avila R, Sobierajski L. A haptic interaction method for volume visualization. IEEE Visualization '96. Silver Spring, MD: IEEE Computer Society, 1996. p. 197–204.
- [2] Bærentzen JA. Octree-based volume sculpting. In: Wittenbrink CM, Varshney A, editors. IEEE Visualization '98, Late Breaking Hot Topics Proceedings, Silver Spring, MD: IEEE Computer Society, 1998. p. 9–12.
- [3] Chua CK, Gay R, Hoheisel W. Computer aided decoration of ceramic tableware. Part I: 3-D decoration. Computers and Graphics 1997;21(5):641–53.
- [4] Galyean T, Hughes J. Sculpting: an interactive volumetric modeling technique. SIGGRAPH'91 Proceedings, 1991. p. 138–48.
- [5] Keskeys DJ. Computer sculpture — new horizons. Computer Graphics 1994;28(4):255–7.
- [6] Mizuno S, Okada M, Toriwaki J. Virtual sculpting and virtual woodcut printing. The Visual Computer 1998;14:39–51.
- [7] van Overveld CWAM. Painting gradients: free-form surface design using shading patterns. Graphics Interface '96. Canadian Human-Computer Communication Society, 1996. p. 151–8.
- [8] Pasko A, Adzhiev V, Sourin A, Savchenko V. Function representation in geometric modeling: concepts, implementation and applications. The Visual Computer 1995;11(8):429–46 (<http://www.u-aizu.ac.jp/public/www/labs/sw-sm/FrepWWW/F-rep.html>).
- [9] Pasko A, Savchenko V. Constructing functionally defined surfaces. Implicit Surfaces'95, Eurographics Workshop, Grenoble, France, INRIA, 1995. p. 97–106.
- [10] Pasko A, Savchenko A, Savchenko V. Polygon-to-function conversion for sweeping. Implicit Surfaces'96, Eurographics/SIGGRAPH Workshop, Eindhoven, The Netherlands, 1996. p. 163–71.
- [11] Pasko A, Savchenko V. Projection operation for multidimensional geometric modeling with real functions. In: Strasser W, Klein R, Rau R, editors. Geometric modeling: theory and practice, Berlin/Heidelberg: Springer, 1997. p. 197–205.
- [12] Peterson D. Halfspace representation of extrusions, solids of revolution, and pyramids. SANDIA Report SAND84-0572. Albuquerque, NM: Sandia National Laboratories, 1984.
- [13] Raviv A, Elber G. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. Technical Report CIS9903. Israel: Technion, 1999.
- [14] Rvachev VL. Methods of logic algebra in mathematical physics. Kiev: Naukova Dumka, 1974.
- [15] Rossignac J, editor. Special issue on interactive sculpting. ACM Transactions on Graphics 1994;13(2).
- [16] Savchenko V. 2D sample based interpolation: visualization of particle clouds and function-to-volume conversion. Proceedings of the IASTED International Conference Computer Systems and Applications (March 30–April 2, 1998, Irbid, Jordan). IASTED, 1998. p. 163–9.
- [17] Savchenko V, Pasko A. Transformation of functionally defined shapes by extended space mappings. The Visual Computer 1998;14:257–70.
- [18] Sourin A, Pasko A, Savchenko V. Using real functions with application to hair modelling. Computers and Graphics 1996;20(1):11–19.
- [19] Savchenko V, Pasko A, Sourin A, Kunii T. Volume modelling: representations and advanced operations. Computer Graphics International '98 (June 22–26 1998, Hannover, Germany). Silver Spring, MD: IEEE Computer Society, 1998. p. 4–13.
- [20] Suzuki R. POV-Ray 3 isosurface patch, <http://www.public.usit.net/rsuzuki/e/povray/iso/>.
- [21] Wang W, Kaufman A. Volume sculpting, Symposium on Interactive 3D Graphics. New York: ACM Press, 1995 (p. 151–6).
- [22] Williams L. 3D paint. Computer Graphics 1990;24(2):225–33.



**Alexander A. Pasko** is an assistant professor at Shape Modeling Laboratory, University of Aizu, Japan. He received his MSc and PhD degrees in computer science from Moscow Engineering Institute (MEPI, Russia) in 1983 and 1988. He was a researcher at MEPI from 1983 to 1992. His research interests include solid modeling, animation, multidimensional visualization, experimental data processing, and computer art. He is a member of ACM SIGGRAPH, IEEE and Eurographics Association.



**Vladimir V. Savchenko** is a professor at Shape Modeling Laboratory, Computer Software Department, University of Aizu, Japan. He took his MSc in applied mechanics at Moscow Aviation Institute in 1971 and PhD in theoretical mechanics from the Institute of Applied Mathematics in Moscow in 1985. He was a head of the Computational Mechanics Department at the Institute of System Analysis, part of the Russian Academy of Sciences, from 1989 to 1992. His research interests include parallel processing, geometric modeling, computer-aided design and artificial life.



**Alexei I. Sourin** received his MS Diploma in Computer Engineering in 1983 from Moscow Engineering Physics Institute (MEPI), Russia. For five years he has been involved in developing computer graphics systems as an engineer and later as a research assistant at the Department of Physics of Superconductivity at MEPI. In 1988 he completed his dissertation on mathematics and software for problems of time-dependent geometry and received his PhD degree in Computer Science. Since 1993, he has been a lecturer and later an associate professor at the Nanyang Technological University, Singapore.