Technical Report TR-NCCA-2008-03

An Exact Representation of Polygonal Objects by C^1-continuous Scalar Fields Based on Binary Space Partitioning

Oleg Fryazinov, Alexander Pasko, Valery Adzhiev

The National Centre for Computer Animation
Bournemouth Media School
Bournemouth University
Talbot Campus,
Poole, Dorset BH12 5BB
United Kingdom

2008

1. Introduction

Representations of geometric objects by continuous and discrete (sampled) scalar fields have recently attracted a lot of attention from both research and application points of view. This is due to many useful properties of such objects. They can undergo set operations with controllable blending, offsetting, metamorphosis with arbitrarily changing topology, sweeping, and other operations. Scalar field models are quite suitable, for example, for the reconstruction from large clouds of points [OBA*03] and for the description of internal material distribution [BST04].

There is a large legacy of polygonal objects created in CAD, computer animation, and other applications. The availability of new modelling operations and application areas stimulates the search for methods for the conversion of 2D polygons and 3D polygonal objects to representations by zero-level sets (2D contours and 3D isosurfaces) of scalar fields.

Approximate and exact (up to the finite precision of computing scalar field values) representations have to be distinguished. Several known approximations of polygonal objects by scalar field isosurfaces are suitable for visualization, animation, re-meshing and other purposes. On the other hand, approximation errors can be critical and even fatal in some applications such as computer-aided manufacturing and medical simulations [PTJ00].

An exact representation can be obtained using signed Euclidean distance from a given point to the polygonal mesh [PT92]. The main problem with this solution is that the Euclidean distance has points with the derivatives discontinuity (vanishing gradients) in its domain, which can cause appearance of unexpected artefacts in further operations on the object [BS04] [FPSM06].

A problem with some conversion methods is that they generate not only the desired approximating zero-value isosurface but some additional isosurfaces inside or outside the considered solid object. Such additional internal or external zero-value points can be wrongly classified as object's boundary points and thus damage an application. Also additional zero-value isosurfaces destroy the distance property of the scalar field, which is important in further operations on objects, for example, in blending and material properties modelling.

A 2D polygon can be exactly described by a continuous real function of two variables built using a monotone set-theoretic formula (see details in the next section). This solution produces a function with zero values only at the polygon edges and no additional internal or external zero-value contours are generated. However, the monotone formula has no direct extension to the 3D polygonal object case.

The problem considered here is to find an algorithm for the generation of scalar fields describing 2D and 3D polygonal objects with defining real functions satisfying the following requirements:

- real function of point coordinates takes zero value exactly at the polygonal object boundary and has different signs for internal and external points;
- no extra zero-value isosurfaces should be generated;
- C^1 continuity of the function in the entire domain.

Taking into account existing difficulties with 3D extensions of algorithms specifically designed for 2D polygons, the best solution would be an algorithm with a dimension independent formulation such that it can be directly applied in 2D, 3D and higher dimensional space.

The main contributions of this paper are: 1) a new algorithm for the construction of the set-theoretic expression for the given polygonal object; 2) an algorithm for the procedural scalar field evaluation at the given point and 3) several extensions to the basic algorithms to satisfy the optimization criteria. The proposed algorithms are based on the binary space partitioning (BSP) of the object by the planes passing through the polygonal faces. The constructed BSP-tree structure is used to generate the set-theoretic expression procedurally with one to four set operations assigned to each internal node of the tree, and a halfspace assigned to each tree leaf corresponding to a partitioning plane. The scalar field is generated when we use some type of R-functions in the tree nodes and defining functions of halfspaces in the leaves. Due to the nature of BSP, this algorithm is practically dimension independent after the step of the BSP-tree construction for the given polygonal object of arbitrary dimensionality. The BSP-tree optimization is discussed and some extensions of the basic tree construction algorithm are proposed. We also provide several examples illustrating applications of BSP-fields describing polygonal objects.

2. Previous works

We discuss in this section several classes of methods for the conversion of polygonal objects to scalar field representations: continuous and discrete field approximations, exact representations utilizing distance functions and different versions of set-theoretic expressions. Blobby models [Mur91], radial-basis functions (RBF) [SPOK95] [YT02], and multi-level partition of unity implicits (MPU) [OBA*03] produce a single isosurface which can approximate a given cloud of polygonal mesh vertices. While highly complicated meshes with huge number of vertices are well approximated, simple objects with the small number of vertices have rather big approximation errors when distances to polygonal faces are taken into account. The approximation with compactly supported radial basis functions (CSRBF) [MYR*05] has problems of creating bumpy surfaces and additional unwanted zero-value isosurfaces not passing through given vertices. The polygonal mesh approximation method based on moving least squares (MLS) [SOS04] deals with undesirable oscillations by adding points with normal constraints across the surface of each polygonal face.

The piecewise linear approximation of the signed distance function [WK03] allows for a multiresolution representation of the given mesh with the fast evaluation of the approximate distance. This method involves the binary space partitioning in a way different from our approach. Another approximation method of the signed distance function for a 3D mesh interpolates between distance functions of its planar cross-sections [COSL98]. A pseudo-distance function is used in the HybridTree [AGCA06], which allows for polygonal meshes to act as implicit surface primitives in various free-form modelling operations.

Discrete approximation methods sample signed distance or some other continuous function at the nodes of a regular volumetric grid or an octree grid [FPRJ00] [Ju04]. A physics-based level set method was used in [ZO02] to approximately reconstruct a given polygonal surface with normal constraints by a discrete scalar field sampled initially with signed distance function values.

Continuous and discrete scalar field approximations of polygonal meshes are useful for mesh repair, re-meshing, rendering, object carving, animation, and metamorphosis. However, errors inherent to approximation methods are not acceptable in some critical applications such as computer-aided manufacturing, material distribution modelling [BST04], and medical simulations [PTJ00].

A polygonal mesh can be exactly represented by a zero-level isosurface of the signed distance function from the given point to the mesh polygons, which allows for offsetting, metamorphosis, smoothing, set operations and other object manipulations [PT92]. The points of C^1 distance function discontinuity form curves and surfaces in space that can cause appearance of unexpected edges in further operations such as blending, additional areas of stresses in strength analysis, and other problems.

Another general approach to the exact conversion is to describe a solid object with the given polygonal boundary using set-theoretic (or simply set) operations on the supporting halfspaces bounded by planes (straight lines in 2D) passing through polygonal faces (edges in 2D) and on some additional planar halfspaces in the general case. The theoretical basis for this approach is given by the Beynon theorem [Bey74], which implies that a piecewise linear function defining a polyhedron can be expressed by applying pointwise min and max operations to a finite set of linear functions. When a set-theoretic expression is obtained, one can formally define the scalar field by replacing the halfspaces by their defining linear functions and using min/max (or other R-functions as explained below) for the set-theoretic operations (see [Sha07] for more details).

An object resulting from the set-theoretic operations has the defining function expressed as follows:

$f_3 = f_1 \vee_\alpha f_2$ for the union;
$f_3 = f_1 \wedge_\alpha f_2$ for the intersection, where f_1 and f_2 are defining functions of initial objects and $\vee_\alpha$, $\wedge_\alpha$ are signs of R-functions. One of the classes of R-functions is

$$\begin{aligned} f_1 \vee_1 f_2 &= max(f_1, f_2) \\ f_1 \wedge_1 f_2 &= min(f_1, f_2) \end{aligned} \tag{1}$$

These functions are C^1 discontinuous at all points where $f_1 = f_2$. R-functions of another class:

$$\begin{aligned} f_1 \vee_0 f_2 &= f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \\ f_1 \wedge_0 f_2 &= f_1 + f_2 - \sqrt{f_1^2 + f_2^2} \end{aligned} \tag{2}$$

have C^1 discontinuity only at points where both arguments are equal to zero. A recently proposed class of R-functions called SARDF (Signed Approximate Real Distance Function) operations [FPSM06] provides smooth approximation of the min/max operations and therefore of the signed distance functions for complex objects constructed using set-theoretic operations on primitives defined by distance functions. The distance property of a defining function is important in several applications such as rendering and shape metamorphosis in computer graphics, aesthetic design, modelling material properties of objects in layered manufacturing, formulation of boundary conditions in engineering analysis, modelling offsets in computer-aided design, and others [BS04].

There are several approaches to constructing set-theoretic representations of a given polyhedron. A convex polyhedron is an intersection of all supporting halfspaces. A concave polyhedron has to be represented by set operations on specially selected convex polyhedra or its own supporting halfspaces. The cell partition [SV93] results in the representation of a concave polyhedron as union of its convex parts (cells). These convex cells share common faces inside the initial polyhedron. When R-functions are applied to get the polyhedron's defining function, "internal zeroes" appear at the points of all shared internal faces. Similar effects occur when applying the more general BRep-CSG conversion algorithm to the polygonal mesh [BC03].

In the convex decomposition of 2D polygons [WW82] [TM84], a polygon is represented by its convex hull with some inner regions subtracted. These inner regions are processed recursively in the same manner to generate lower levels of the convex decomposition. The application of min/max or other R-functions to this representation leads to the appearance of "external zeroes" at the edges of the nested convex hulls with the disadvantages discussed earlier.

The optimal set-theoretic expression of a 2D polygon called a monotone formula [Rva74] [Pet84] includes each of the supporting halfplanes only once and does not include any additional halfplane. An efficient algorithm for deriving

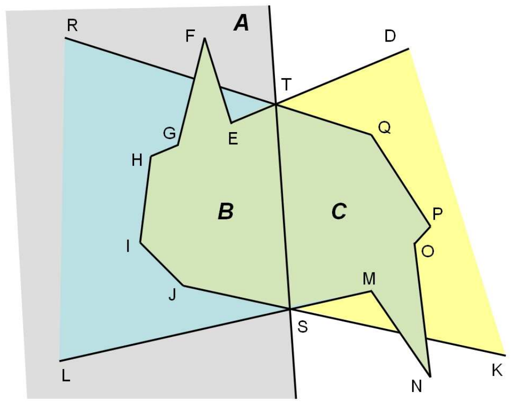

Figure 1: *A simple polygon (green) constructed from three planar halfspaces: A (grey) with the boundary line ST, B (yellow and green left to the line ST), C (blue and green right to the line ST).*

this representation from an arbitrary given polygon was proposed in [DGHS88]. The remarkable property of the monotone formula is that it does not generate any internal or external zeroes when applying R-functions.

It is difficult to extend exact 2D conversion algorithms to 3D polyhedra. Unfortunately, an analogue of the monotone formula for 3D polyhedra is not known. The convex decomposition algorithms based on nested convex hulls can be extended to 3D space, but they do not converge for some types of polyhedra [KW92]. There is a need of a dimension independent conversion algorithm, which can be applied directly to polygons in 2D, polyhedra in 3D and to higher dimensional polytopes.

3. Scalar fields based on BSP-trees

In this section we present our approach to the exact conversion. We suppose that the initial polygonal object is a closed manifold and contains no degenerate boundary elements. If these requirements are not satisfied, the resulting scalar field will not be an exact representation in the general case.

First, we consider a set-theoretic construction of a 2D polygon as a representative of the general case problem. We show in subsection 3.1 that the existing methods are not satisfactory in terms of the above requirements to the scalar field. Then, we propose an original set-theoretic solution to the given 2D problem (section 3.2) and the proposed solution is applied in section 3.3 to devise a basic dimension independent algorithm and its optimizations for the exact conversion.

3.1. Construction of a scalar field for a simple 2D polygon

As an introduction to our approach, let us consider the set-theoretic construction of a simple polygon on a 2D plane from three semi-infinite planar halfspaces as shown in Fig. 1. Three intersecting halfspaces a given as follows: A (shown in grey in Fig. 1) with the boundary straight line ST, B (yellow and green left to the line ST) with the boundary DEFGHIJK, C (blue and green right to the line ST) with the boundary LMNOPQR. The boundaries of B and C intersect in the points S and T. The problem is to construct a defining function $f(x,y)$ for the simple polygon IJSMNOPQTEFGH (shown in green in Fig. 1) such that $f(x,y) = 0$ only at the points of the polygon boundary, $f(x,y) > 0$ inside the polygon, and $f(x,y) < 0$ outside the polygon. The function has to be C^1 continuous everywhere except the polygon boundary, where only C^0 continuity is allowed. Note that no internal or external non-boundary points are allowed to have zero function value or zero function gradient value.

The presented 2D problem can be simply solved by using a monotone formula [Rva74] [Pet84] [DGHS88] mentioned above. However, as it was mentioned an extension of the monotone formula construction algorithm to the case of 3D polyhedrons is problematic. Therefore, we are looking for an alternative dimension independent solution.

Another approach is to apply a kind of cell partitioning [BC03] to the polygon as shown in Fig. 2. The boundaries of two halfspaces ($A \cap B$) (left in Fig. 2) and ($\neg A \cap C$) (middle) share the segment ST. After applying union to these halfspaces we obtain the desired polygon (right in Fig. 2). The defining function constructed using the set-theoretic expression and R-functions is as follows:

$$f_P = (f_A \wedge_\alpha f_B) \vee_\alpha (-f_A \wedge_\alpha f_C) \qquad (3)$$

This defining function for the polygon takes zero values at its boundaries. Additionally, the function takes zero values at the internal segment ST, which becomes a set of points with internal zeroes in respect to the polygon. The convex decomposition approaches will generate external zeroes of the function. Therefore, the cell partitioning and convex decomposition do not satisfy the requirements to the solution.

3.2. Proposed solution

We propose to directly construct a set-theoretic expression for the polygon in such way that the corresponding defining function does not have internal or external zeroes.

The proposed steps of the set-theoretic construction of a polygon with the defining function without non-boundary zeroes are shown in Fig. 3. At each step, the union or intersection operation is applied to planar regions, which do not share boundary elements inside or outside the desired polygon. this guarantees the defining function without internal and external zeros. The defining function constructed

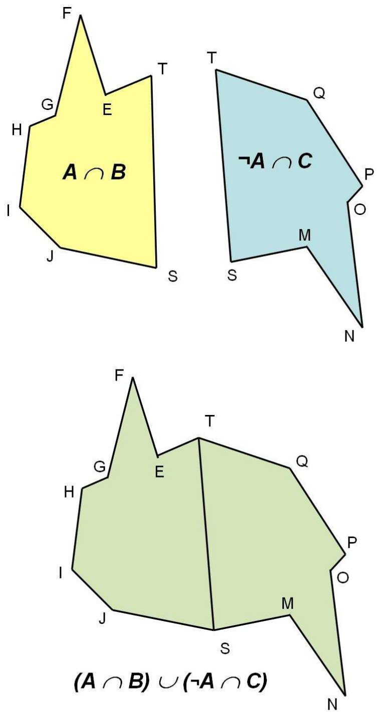

Figure 2: *Cell partition of the polygon with internal zeroes of the defining function at the points of the segment ST*

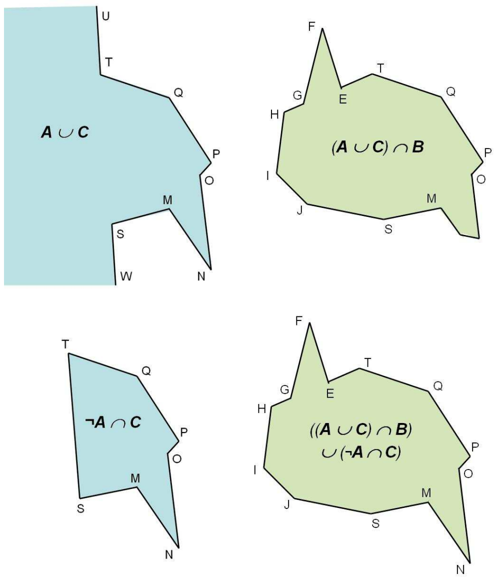

Figure 3: *Set-theoretic construction of the polygon with a defining function without non-boundary zero points.*

using the set-theoretic expression shown in Fig. 3d and R-functions is as follows:

$$f_s = ((f_A \vee_\alpha f_C) \wedge_\alpha f_B) \vee_\alpha (-f_A \wedge_\alpha f_C) \qquad (4)$$

In total, the evaluation of the function f_s requires applying four R-functions. This example was selected as a representative of the general case and we can use the derived defining function for any configuration involving two intersecting halfspaces and a partitioning line (plane in 3D). However, there can be an important particular case, when the number of the required R-functions is reduced from four to two. If the boundary of the halfspace B does not intersect the boundary of C in the area right to the halfplane A (no intersection of the ray JK with the segments MN and NO in the above example), then the polygon can be described by a simple set-theoretic expression: $(A \cup C) \cap B$. Symmetrically, if the boundary of the halfspace C does not intersect the boundary of B in the area of the halfplane A (no intersection of the ray QR with the segments EF and FG), then the polygon can be described by another expression: $(\neg A \cup B) \cap C$.

3.3. BSP-tree construction and function evaluation

In this section we describe our algorithm for the scalar field generation for the given polygonal object. The presented in the previous section set-theoretic description and the scalar field generation for a 2D polygon reduces the problem to operations on two parts of the polygon divided by a partitioning straight line. If we select only supporting straight lines (continuations of polygon edges) for partitioning the polygon and apply the described procedure to the both parts of the polygon recursively, finally we can construct a set-theoretic expression involving only supporting halfplanes. The data structure corresponding to such a recursive procedure is the binary space partitioning tree (BSP-tree) [FKN80]. Note that the same procedure can be applied in 2D and 3D spaces with very small modifications. There are two independent parts in the scalar field generation based on BSP-trees: a preprocessing step of the BSP-tree construction and the function evaluation at the given point utilizing the constructed BSP-tree. In this section we present an algorithm for the construction of BSP-tree in 3D case. The construction of BSP-tree in the 2D case can be easily done in the similar way.

3.3.1. Basic BSP-tree construction

The construction of the BSP-tree is a recursive procedure that contains the following main steps: selection of the base polygon according to some criteria, construction of the partitioning plane containing the base polygon, division of the rest of polygons into the "positive" and "negative" groups based on the side of the partitioning plane, and recursive processing of the "positive" set and the "negative" set. In our approach we use the classic construction procedure of the BSP-tree that has been formalized in [FKN80]. As BSP is a dimension independent structure by its nature, the algorithm is given for the case of a 3D input polyhedron with planar polygons as its boundary faces, but can be directly applied in 2D with reformulation for polygon edges and partitioning straight lines. Each partitioning plane contains at least one polygonal face. The algorithm for the BSP-tree construction is as follows:

```
Build_Tree (polygon_list pl){
  node current_node = Make_Node();
  p0 = Select_Polygon(pl);
  current_node.cut_plane = Make_Plane(p0);
  polygon_list pos_list = 0;
  polygon_list neg_list = 0;
  for each polygon k in pl except p0
    classify p[k] against cut_plane;
    if (pl[k] is on the positive side)
      Add(pl[k], pos_list);
    else if (pl[k] is on the negative side)
      Add(pl[k], neg_list);
    else {
      Split_Polygon(pl[k], cut_plane,
                    pos_parts, neg_parts);
      Add (pos_parts, pos_list);
      Add (neg_parts, neg_list);
    }
  if (pos_list is not empty)
    current_node.pos_tree =
        Build_Tree(pos_list);
  else current_node.pos_tree = 0;
  if (neg_list is not empty)
    current_node.neg_tree =
        Build_Tree(neg_list);
  else current_node.neg_tree = 0;
  return current_node;
}
```

Here "pos" refers to positive and "neg" refers to negative. The BSP-tree building procedure `Build_Tree` processes the list pl of the input polygonal faces. At the first iteration this list is a list of all faces in the input mesh. A polygon that is selected by the `Select_Polygon` function is the base for a partitioning (cutting) plane `cut_plane` passing through it. The selection criteria for the polygon are discussed in subsection 3.3.3. The equation for this plane is calculated by the `Make_Plane` procedure. The rest of the polygons from the input list are classified against the partitioning plane into two groups depending on which side of the plane they are residing. We classify the polygon as positive, if for any vertex $\mathbf{P}_x$ from this polygon the next inequality is satisfied:

$$(\mathbf{P}_x - \mathbf{P}_0) \odot \mathbf{n} \geq 0,$$

where $\mathbf{P}_0$ is a point on the base plane and $\mathbf{n}$ is the normal to the plane. If some polygon intersects the partitioning plane, it is split into two parts by `Split_Polygon`, each of which is added to the respective list. The polygons in `pos_list` and in `neg_list` are processed recursively by the procedure `Build_Tree` to create positive and negative subtrees of the created node. Finally a node of the BSP-tree is created by `Make_Node`, which stores `cut_plane`, `pos_tree`, and `neg_tree`.

3.3.2. Function evaluation procedure

The set-theoretic expression and the corresponding functional expression for a single partitioning plane (straight line in 2D) were given in Section 3.2. In general, one needs to build a corresponding Constructive Solid Geometry (CSG) tree for the given polygonal object and then to apply R-functions in its nodes to evaluate the entire scalar field. In our case, the constructed BSP-tree helps evaluate the scalar field procedurally without building an equivalent CSG-tree. The evaluation procedure for the scalar field at the given point starts from the root of the BSP-tree and applies the following functional expressions at the nodes recursively:

$$f(x) = \begin{cases} f_a, pos\_tree = 0, neg\_tree = 0 \\ f_a \wedge_\alpha f_b, pos\_tree \neq 0, neg\_tree = 0 \\ f_a \vee_\alpha f_c, pos\_tree = 0, neg\_tree \neq 0 \\ ((f_a \vee_\alpha f_c) \wedge_\alpha f_b) \vee_\alpha (-f_a \wedge_\alpha f_c), otherwise \end{cases}$$

where for the given node f_a is a signed distance to the partitioning plane of the current node, f_b is a defining function for the positive subtree of the node, and f_c is a defining function for the negative subtree.

3.3.3. Optimization of BSP-trees

The selection of the base polygon and the partitioning plane is the crucial part of the BSP-tree construction. Depending on criteria for the base polygon selection different BSP-trees can be obtained. In our work we use two different approaches: "naive" selection, where the polygon is randomly selected from the polygon list, and "optimized" selection. The optimization means using selection criteria that allow for obtaining a tree with the following properties:

- Minimization of polygon splitting operations to reduce the total number of nodes and the number of operations in the function evaluation
- Minimization of computational errors during the function evaluation and BSP-tree construction;
- Balancing the BSP tree, i.e., minimization of difference between positive and negative list for the minimization of the depth of the tree.

To provide the generation of BSP-trees with these properties, we propose to use the following criteria:
For the given partitioning plane P and the list of polygons F containing the list of vertices V:

$$K_{split}(P) = N_{split}$$

$$K_{dist}(P) = \min_{v \in V, v \notin P} (distance(v, P))$$

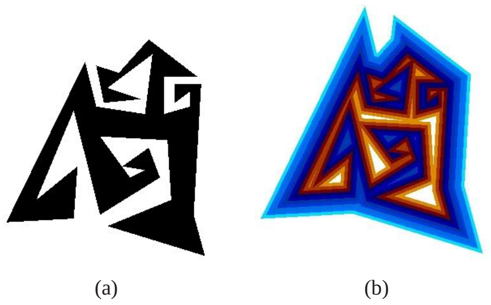
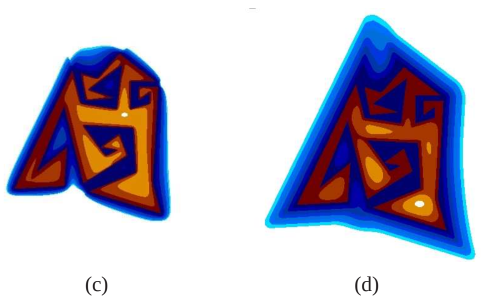

Figure 4: *(a) 2D polygon and colour maps of the scalar fields for its optimized BSP-tree with applied min/max function (b), R-functions of Eq. 2 (c), SARDF operations (d).*

$$K_{angle}(P) = \min_{f \in F, f \notin P} (angle(\mathbf{n}_f, \mathbf{n}))$$

$$K_{tree}(P) = \begin{cases} \frac{N_{front}}{N_{back}}, N_{back} > N_{front} \\ \frac{N_{back}}{N_{front}}, otherwise \end{cases}$$

where N_{front} is a number of faces that lie in the positive half-space of the plane P and N_{back} is a number of faces that lie in the negative half space, N_{split} is a number of faces that are split by the plane P and $\mathbf{n}$ is a normal to P. The meaning of these criteria is the following: maximization of K_{dist} and K_{angle} allows to avoid of degenerate faces after splitting of the polygon, maximization of K_{tree} allows to balance the tree and minimization of K_{split} allows to minimize the polygon splitting operations.

We select the base polygon with the plane P, where one of the following conditions is satisfied:

- $K_{split} = 0$
- $K(P) = K_{dist}(P) * K_{angle}(P) * K_{tree}(P)$ is maximal

Fig. 4 shows examples of BSP-fields generated using an optimized BSP-tree for a 2D polygon. The shown colour maps illustrate the behaviour of different R-functions (from left to right): min/max, R-functions with square roots (Eq. 2), and SARDF operations ("smooth min/max") [FPSM06]. Note that an exact representation of the polygon by a signed scalar field is obtained in all cases. The min/max functions do not provide a satisfactory field as it has lines with vanishing gradients of the defining function in the domain. The R-functions defined by Eq. 2 generate a C^1-continuous field, but it does not well approximate the distance function. The SARDF type of R-functions generate a scalar field, which both is C^1-continuous and provides better approximation of the distance function. On the other hand, min/max operations provide the highest speed of calculations and SARDF operations are the slowest. Therefore, the choice of R-functions entirely depends on the requirements of the particular applications to the scalar field.

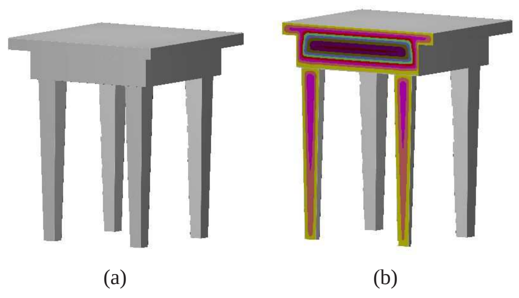

Figure 5: *Table model*

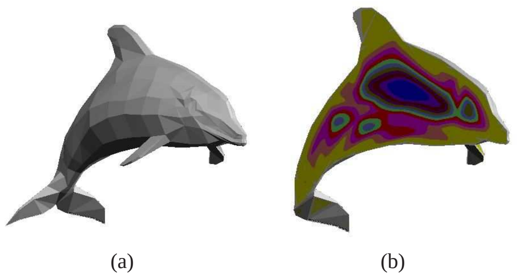

Figure 6: *Dolphin model*

4. Experiments

We first discuss the results of our experiments with the basic and optimized BSP-tree construction algorithms. Then, several operations employing the obtained scalar fields are illustrated.

4.1. Exact conversion of polygonal objects with sharp features and arbitrary topology

We did not make any assumption about objects geometry features or its topology in the formulations of the BSP-tree generation and the function evaluation algorithms. Here we show how our method can be applied to objects that usually cannot be easily converted using existing methods. Figs. 5, 6 and 7 illustrate the conversion of polygonal models with sharp features. These figures include the original mesh and a colour map of a cross-section of the functionally represented

				Non-optimized		Optimized	
Model	Vertices	Faces	Planes	Splittings	Set operations	Splittings	Set operations
Block with hole	72	144	20	118	80	0	31
Table	64	124	59	56	113	0	68
Dolphin	282	562	562	1503	3249	572	1775
Rocker arm (low poly)	470	940	933	3253	6221	976	2828
Chain	768	1536	384	118	80	0	31

Table 1: *Tests of the BSP-tree optimization*

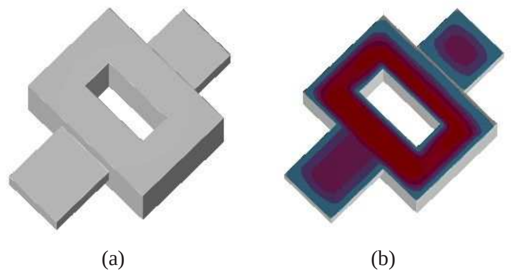

Figure 7: *Block with hole model*

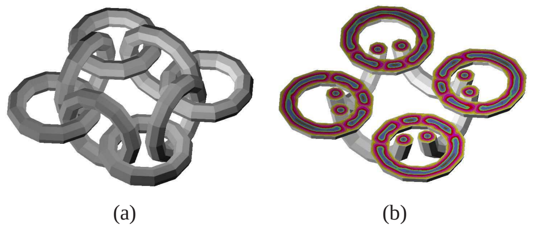

Figure 8: *Chain model*

model that we obtain from the initial model. The colour distribution for each model is set up only to illustrate the behaviour of the defining function, not to compare models with each other. For example, for the "Block with hole" model, which has a quite small number of polygons, the approximation methods based on RBF and MPU [SPOK95] [YT02] [OBA*03] can only produce some oval shapes for the block and for the hole, which is unacceptable in most applications. Figures 7, 8 and 9 illustrate the conversion of polygonal

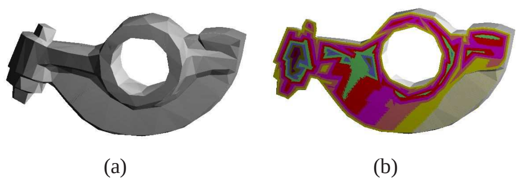

Figure 9: *Rocker arm model*

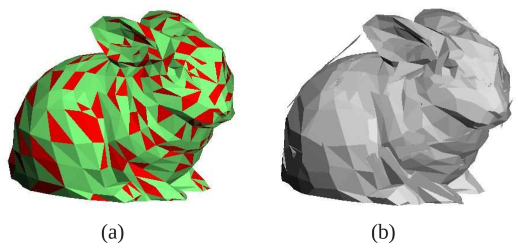

Figure 10: *Model with missing polygons and BSP field*

models with non-zero genus and objects with disjoint components.

The main feature of the monotone formula for a 2D polygon is the minimal number (N-1) of set-theoretic operations on N supporting halfspaces of the polygon. The main purpose of the BSP-tree optimization described above is to minimize the number of nodes in the BSP-tree and thus the total number of set-theoretic operations. Table 1 shows the results of testing the optimization. Here the number of planes means the number of unique supporting halfspaces with planar boundaries (several mesh triangles can belong to one plane). The main result is that we can achieve the drastic reduction of the number of polygon splitting operations. In the case of low number of polygons the optimization leads to the elimination of splitting and to the number of set-theoretic operations very close to the number of planes (see "Block with hole" and "Table"). For more complex models the number of set-theoretic operations remains about three times larger than the number of planes. Further research is necessary to achieve the minimal number provided by the monotone formula in 2D.

4.2. Conversion of incomplete meshes

As mentioned above, the input meshes should be closed manifolds. However, if the mesh is not a closed manifold, the BSP field can be created from the input mesh and in many cases represent the model that topologically and geometrically close to the original model. In figure 10 we show how our method can be applied to a model with missing polygons. From the original mesh (see figure 10a) we remove

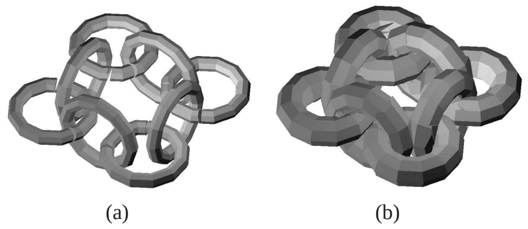

Figure 11: *Offset*

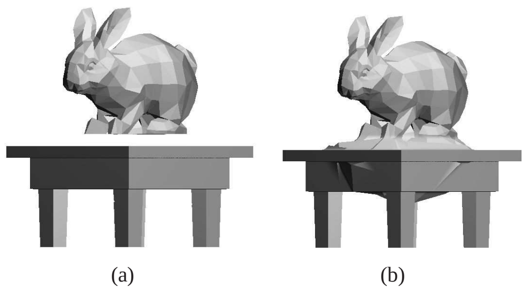

Figure 12: *Blend*

25% of triangles (red colour in the figure) and create BSP field from the rest of triangles (green colour). The resulting functionally represented model (see Fig. 10b) is visually close to the original, however some artefacts appeared. It means that for the objects with incomplete boundaries the algorithm is not guaranteed to produce the intuitively expected exact representation.

4.3. Offsetting and blending of polygonal meshes

For some models the BSP-field is close to the monotonous field. In this case the BSP-field has distance properties and we can use an offset operation with simple modification of the function: $F(x,y,z) - d \geq 0$, where d is an offset value. For example, in chain model (see Figure 8) we have BSP-field with distance properties for positive offset (see 11b), however we do not have distance properties for negative offset (see Fig. 11a). Unfortunately, at present we cannot guarantee the distance properties for the generated BSP-field. This can be the base for the future research on the BSP-tree optimization. If the distance property is provided, we also can apply a blending operation to two converted polygonal objects. Fig. 12 shows a blending union with added material between the Rabbit and the Table models.

4.4. Sweeping

Sweeping by a moving solid is one of the most important operations in CAD. It can be applied in simulation of numerically controlled machining, robot motion planning, and maintainability simulation. In general, the operation is

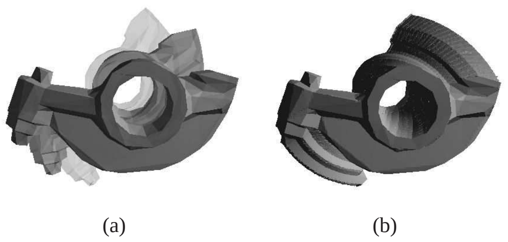

Figure 13: *Phases of motion of the functionally represented polygonal rocker arm along a helical trajectory (a), and the solid sweep (b).*

problematic for solids with complex topology, shape varying sweep generators, and sweeps with self intersections. We have applied the algorithm for sweeping by a moving solid [SP96] devised for function-based solid models. Its advantage is the generality of the approach resolving the above difficulties. Fig. 13 shows a sweep by the Rocker arm modelwith non-zero genus moving along a helical trajectory. The initial polygonal model was converted to a scalar field model and the algorithm [SP96] was applied to obtain the scalar field defining the final sweep.

4.5. Self-replicating

Self-replicating of solid objects represented by polygonal meshes can easily be done if we have function representations of these objects. In this case we can apply any periodic function to the function representation and obtain the needed result. On figure 14 we show how the Rabbit model can be self-replicated by adding only several lines of code to the original function.

5. Conclusions

We proposed, implemented and tested a new dimension independent algorithm for the conversion of a polygonal object to a representation by a signed scalar field without vanishing gradients and extra zero-level isosurfaces. The algorithm provides an exact representation of polygonal objects including those with small number of vertices, sharp features, missing polygons, non-zero genus, and several disjoint components. Under exact representation we mean the theoretical model without taking into account the finite precision of computing scalar field values.

The existing problems of input polygonal meshes such as self-intersections, topological inconsistencies, large number of holes, and triangles with very high aspect ratios influence the quality of the obtained results. A robust conversion procedure requires special mesh pre-processing to provide an input mesh as a closed manifold. The function evaluation procedure is time consuming. For some applications it can be reasonable to switch to continuous approximations of the

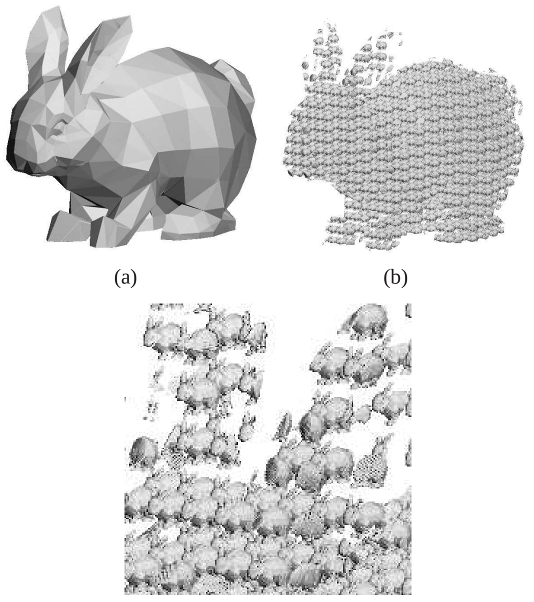
(a) (b) (c)

Figure 14: *Self-replicating of the Rabbit model: a) Original model, b) Self-replicated model, c) Zoom of the self-replicated model in the head area.*

obtained scalar fields using B-splines or wavelets. However, it would mean loosing the exact representation of the initial polygonal object.

The proposed algorithm is in some aspects superior in comparison with the monotone set-theoretic formula available for 2D polygons. For example, the monotone formula is not directly applicable to objects with non-zero genus and with disjoint components. The ultimate goal of this research is to achieve for 3D polyhedra the property of the minimal number of operations of the monotone formula. Although the proposed optimizations of the basic algorithm have significantly reduced the number of operations, the monotone formula property has been achieved only for relatively simple objects with the low number of polygons. Further research will be needed in this direction, for example, on the detection of the special cases described in Section 3.2.

References

[AGCA06] ALLEGRE R., GALIN E., CHAINE R., AKKOUCHE S.: The HybridTree: Mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. *Graphical Models 68*, 1 (January 2006), 42–64.

[BC03] BUCHELE S. F., CRAWFORD R. H.: Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), ACM, pp. 135–144.

[Bey74] BEYNON W. M.: Combinatorial aspects of piecewise-linear maps. *Journal of the London Mathematical Society 2*, 7 (1974), 719–727.

[BS04] BISWAS A., SHAPIRO V.: Approximate distance fields with non-vanishing gradients. *Graph. Models 66*, 3 (2004), 133–159.

[BST04] BISWAS A., SHAPIRO V., TSUKANOV I.: Heterogeneous material modeling with distance fields. *Comput. Aided Geom. Des. 21*, 3 (2004), 215–242.

[COSL98] COHEN-OR D., SOLOMOVIC A., LEVIN D.: Three-dimensional distance field metamorphosis. *ACM Trans. Graph. 17*, 2 (1998), 116–141.

[DGHS88] DOBKIN D., GUIBAS L., HERSHBERGER J., SNOEYINK J.: An efficient algorithm for finding the csg representation of a simple polygon. *SIGGRAPH Comput. Graph. 22*, 4 (1988), 31–40.

[FKN80] FUCHS H., KEDEM Z. M., NAYLOR B. F.: On visible surface generation by a priori tree structures. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques* (1980), ACM, pp. 124–133.

[FPRJ00] FRISKEN S. F., PERRY R. N., ROCKWOOD A. P., JONES T. R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 249–254.

[FPSM06] FAYOLLE P.-A., PASKO A., SCHMITT B., MIRENKOV N.: Constructive heterogeneous object modeling using signed approximate real distance functions. *Journal of Computing and Information Science in Engineering, ASME Transactions 6*, 3 (2006), 221–229.

[Ju04] JU T.: Robust repair of polygonal models. *ACM Trans. Graph. 23*, 3 (2004), 888–895.

[KW92] KIM Y. S., WILDE D. J.: A convex decomposition using convex hulls and local cause of its non-convergence. *ASME Journal of Mechanical Design 114*, 3 (Sept. 1992), 459–467.

[Mur91] MURAKI S.: Volumetric shape description of range data using "blobby model". In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), ACM, pp. 227–235.

[MYR*05] MORSE B. S., YOO T. S., RHEINGANS P., CHEN D. T., SUBRAMANIAN K. R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (2005), ACM, p. 78.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470.

[Pet84] PETERSON D.: *Halfspace representation of extrusions, solids of revolution, and pyramids*. Tech. rep., Sandia National Laboratories, Albuquerque, NM, 1984.

[PT92] PAYNE B. A., TOGA A. W.: Distance field manipulation of surface models. *IEEE Comput. Graph. Appl. 12*, 1 (1992), 65–71.

[PTJ00] PETER J., TORNAI M., JASZCZAK R.: Analytical versus voxelized phantom representation for monte carlo simulation in radiological imaging. *Medical Imaging, IEEE 19*, 5 (2000), 556–564.

[Rva74] RVACHEV V.: Methods of the algebra of logic in mathematical physics. *Ukrainian Mathematical Journal 27*, 4 (1974), 472–474.

[Sha07] SHAPIRO V.: Semi-analytic geometry with r-functions. *Acta Numerica 16* (2007), 239–303.

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 896–904.

[SP96] SOURIN A., PASKO A.: Function representation for sweeping by a moving solid. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (1996), 11–18.

[SPOK95] SAVCHENKO V. V., PASKO A., OKUNEV O. G., KUNII T. L.: Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14* (1995), 181–188.

[SV93] SHAPIRO V., VOSSLER D. L.: Separation for boundary to csg conversion. *ACM Trans. Graph. 12*, 1 (1993), 35–55.

[TM84] TOR S. B., MIDDLEDITCH A. E.: Convex decomposition of simple polygons. *ACM Trans. Graph. 3*, 4 (1984), 244–265.

[WK03] WU J., KOBBELT L.: Piecewise linear approximation of signed distance fields. In *Proceedings of Vision, Modelling and Visualization 03* (2003), pp. 513–520.

[WW82] WOODWARK J. R., WALLIS A. F.: Graphical input to a boolean solid modeller. *CAD 82* (1982), 681–688.

[YT02] YNGVE G., TURK G.: Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics 8*, 4 (2002), 346–359.

[ZO02] ZHAO H., OSHER S.: Visualization, analysis and shape reconstruction of unorganized data sets. *Geometric Level Set Methods in Imaging, Vision and Graphics* (2002).