BSP-fields: An Exact Representation of Polygonal Objects by Differentiable Scalar Fields Based on Binary Space Partitioning

Oleg Fryazinov, Alexander Pasko, Valery Adzhiev

Bournemouth University, UK

Abstract

The problem considered in this work is to find a dimension independent algorithm for the generation of signed scalar fields exactly representing polygonal objects and satisfying the following requirements: the defining real function takes zero value exactly at the polygonal object boundary; no extra zero-value isosurfaces should be generated; C^1 continuity of the function in the entire domain. The proposed algorithms are based on the binary space partitioning (BSP) of the object by the planes passing through the polygonal faces and are independent of the object genus, the number of disjoint components, and holes in the initial polygonal mesh. Several extensions to the basic algorithm are proposed to satisfy the selected optimization criteria. The generated BSP-fields allow for applying techniques of the function-based modeling to already existing legacy objects from CAD and computer animation areas, which is illustrated by several examples.

Key words: Implicit surfaces, Boundary representation, Function representation, Binary Space Partitioning, BSP-field, Exact Conversion

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. Several modeling operations have been formulated specifically for objects defined by scalar fields, for example, controllable blending operations, offsetting, metamorphosis with arbitrarily changing topology, sweeping, and

others [6][20][30][22]. Scalar field models are quite suitable, for example, for the reconstruction from large clouds of points [31][19] and for the description of internal material distribution [5][12].

There is a large legacy of polygonal objects created in CAD, computer animation, and other areas. The availability of new above-mentioned modeling 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. For example, two polygonal meshes converted to scalar fields can be blended together with a smooth transition between two surfaces. Some promising application areas for converted meshes are function-based volume modeling and rendering, especially for objects with multiple materials in CAD as well as in rapid prototyping and fabrication; simulation of different physical properties of objects in medicine and geology [13].

Approximate and exact (up to the finite precision of computing scalar field values) representations have to be distinguished. Exact representations allow for the derivation of an analytical expression for the function, which then can be evaluated with the finite machine precision. Approximate representations allow for a built-in error in the numerical function evaluation within the given maximal approximation error. 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 [25].

An exact representation can be obtained using signed Euclidean distance from a given point to the polygonal mesh [24]. The main problem with this solution is that the Euclidean distance is not C^1-continuous and has points with the derivatives discontinuity (vanishing gradients) in its domain, which can cause appearance of unexpected artefacts in results of further operations on the object [4][12]. When applying blending to such objects, unexpected creases can appear on the smooth transition surface. In material design, areas with vanishing gradients can create additional unwanted stresses inside fabricated objects.

A problem with some conversion methods [7][33][35][37] 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 [27] (see Fig. 1).

Another concern is providing the distance property of the defining scalar field, which has to change its sign at the object boundary and its absolute values have to increase with the growing distance from the object surface. Additional zero-value isosurfaces destroy the distance property of the scalar field, which is important in further operations on objects, for example, in offsetting, blending, and material properties modeling.

In contrast to the existing conversion methods, the problem considered in this paper is to find an algorithm for generation of scalar fields describing 2D and 3D polygonal objects with defining real functions satisfying the following requirements:

- the function takes zero value exactly at the points of 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.

The solution to this problem exists for polygons in two-dimensional space [23][26]. A 2D polygon can be exactly described by a continuous real function of two variables built using a monotone set-theoretic formula (see Fig. 1e and details in the next section), which guarantees a well-formed set representation of the polygon [27]. This solution produces a function with zero values only at the polygon edges and no additional internal or external zero-value contours are generated. For 3D space it is a long-standing unsolved problem. The monotone formula has no direct extension to the 3D polygonal object case. Therefore, the best solution would be to devise an algorithm with a dimension independent formulation such that it can directly be applied in 2D, 3D and higher dimensional space.

The main contributions of this paper are: 1) a new algorithm for the construction of the well-formed set-theoretic representation 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

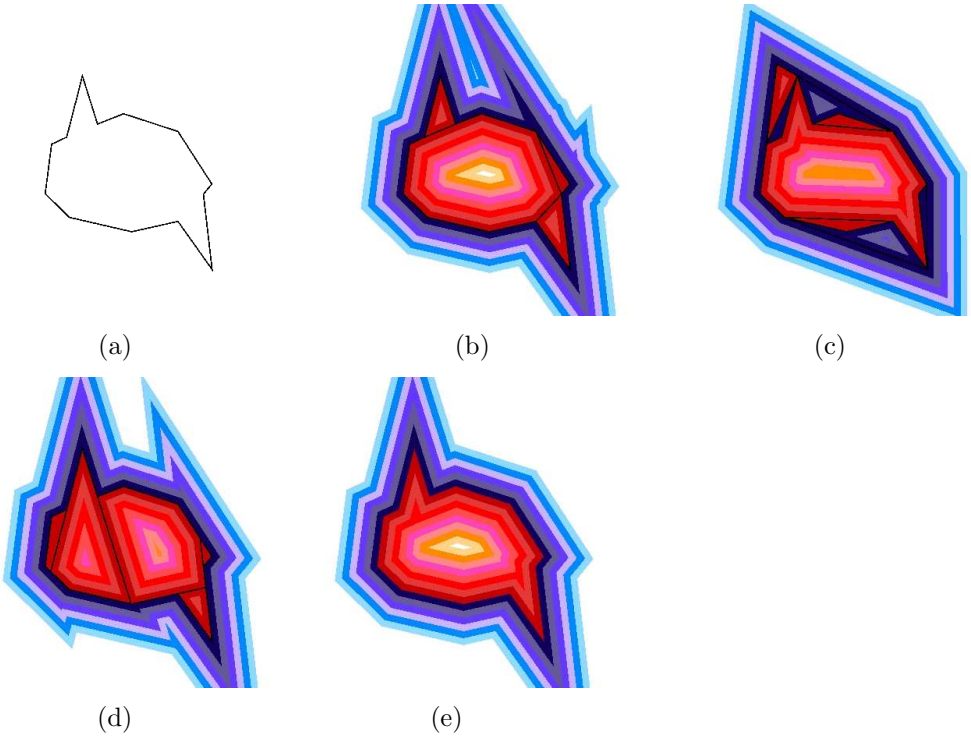


Figure 1: Existing methods to construct the scalar field for the given polygon in two-dimensional space: a) initial polygon; b) cell decomposition; c) convex decomposition; d) BSP CSG; e) monotone formula. Zero sets are indicated by black colour. The colour scheme is described of Fig. 6.

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 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 that illustrate 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. They are both continuous and discrete field approximations as well as exact representations utilizing distance functions and different versions of set-theoretic expressions.

Approximation methods based on blobby models [16], radial-basis functions (RBF) [31][40], and multi-level partition of unity implicits (MPU) [19] 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 the compactly supported radial basis functions (CSRBF) [17] has problems of creating bumpy surfaces and additional unwanted zero-value isosurfaces not passing through the given vertices. The polygonal mesh approximation method based on moving least squares (MLS) [29] deals with undesirable oscillations by adding points with normal constraints across the surface of each polygonal face. This method allows for obtaining an exact mesh respresentation, however, either the result is a function with discontinuities or the numerical calculations can become unstable in the neighborhood of the polygon as pointed out by the authors.

The piecewise linear approximation of the signed distance function [36] 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 [8]. A pseudo-distance function is used in the HybridTree [1], which allows for polygonal meshes to act as implicit surface primitives in various free-form modeling operations.

Discrete approximation methods sample signed distance or some other continuous function at the nodes of a regular volumetric grid or an octree grid [11][14]. A physics-based level set method was used in [41] 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 modeling [5], and medical simulations [25].

A polygonal mesh can be exactly represented by a zero-level isosurface of the signed distance function of the given point and the mesh polygons. This allows for offsetting, metamorphosis, smoothing, set operations and other object manipulations [24] to be applied to polygonal meshes. The points of C^1 distance function discontinuity form curves and surfaces in space that can cause appearance of unexpected edges in results of further operations such as blending, additional areas of stresses in strength analysis, and other problems. The approach introduced in [4] allows to define the polygonal mesh as a zero-set of a continuous function, but it solves only a part of the problem stated above, because the obtained function has the same sign inside and outside the polygonal object. Therefore, the entire polygonal object is not represented as a solid with its inside and outside distinguished by the function.

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). In the general case these operations can be applied to additional planar halfspaces. The theoretical basis for this approach is given by the Beynon theorem [3], 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 functions (or other R-functions as explained below) for the set-theoretic operations. An R-function is a real function of several variables with its sign depending only on the signs of its arguments, not their values (see [28] 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

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

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

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

They 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 [12] provides smooth approximation of the min/max operations and therefore of the signed distance functions for complex objects constructed using set-theoretic operations. The distance property of a defining function is important in several applications such as rendering and shape metamorphosis in computer graphics, aesthetic design, modeling material properties of objects in layered manufacturing, formulation of boundary conditions in engineering analysis, modeling offsets in computer-aided design, and others [4].

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 [33] 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 (see Fig. 1b). Similar effects occur when applying the more general BRep-CSG conversion algorithm to the polygonal mesh [7] (see Fig. 1d).

In the convex decomposition of 2D polygons [37][35], 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 (see Fig. 1c).

The optimal set-theoretic expression of a 2D polygon called a monotone formula [26][23] includes each of the supporting halfplanes only once and does not include any additional halfplane. An efficient algorithm for deriving this representation from an arbitrary given polygon was proposed in [9]. The remarkable property of the monotone formula is that it does not generate any internal or external zeroes when applying R-functions (see Fig. 1e).

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. A representation of a 3D concave polyhedron by a series of convex components with alternating signs (for union and difference operations) [38] allows for obtaining a set-theoretic expression for the given polyhedron, however in case of non-convergence [39][15] this method is not applicable directly. 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. In this paper, we propose a novel approach to the exact conversion of polygonal objects to corresponding scalar fields.

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 oriented 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 original algebraic and set-theoretic solutions to the given 2D problem (sections 3.2 and 3.3) and the proposed solution is applied in section 3.4 to devise a basic dimension independent algorithm and its optimizations for the exact conversion.

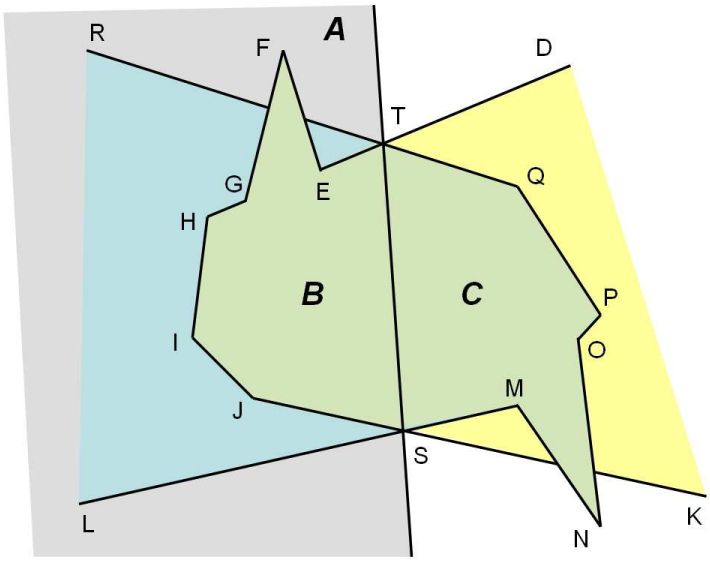


Figure 2: A simple polygon (green) constructed from three planar halfspaces: A (grey) with the boundary line ST, B (yellow and green left to the line ST), C (blue and green right to the line ST).

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. 2. Three intersecting halfspaces A, B, C are given as follows: A (shown in grey in Fig. 2) with the boundary straight line ST; B (green area left to the line ST and yellow area) with the boundary DEFGHIJK; C (blue area and green area 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. 2) 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 [26][23][9] mentioned above. However, as it was stated an extension of the monotone formula construction algorithm to the case of 3D polyhe-

drons is problematic. Therefore, we are looking for an alternative dimension independent solution.

3.2. Algebraic approach

We can follow the idea of [32] that, if one can independently construct two models of the object, one with external zeroes and another with internal zeroes, then an algebraic sum of two corresponding defining functions does not have non-boundary (internal or external) zero points.

This approach is illustrated by Figs. 3 and 4 for our example. The construction of a polygon with external zeroes of the defining function is shown in Fig. 3. Two halfspaces ($\neg A \cup B$) (Fig. 3a) and ($A \cup C$) (Fig. 3b) have common parts of their boundaries along rays TU and SW. When these two halfspaces intersect, they form the desired polygon with two mentioned rays attached to its boundaries (Fig. 3c). The defining function constructed using the obtained set-theoretic expression and R-functions is as follows:

$$f_{p\_ext} = (-f_A \vee_\alpha f_B) \wedge_\alpha (f_A \vee_\alpha f_C) \tag{3}$$

where f_A, f_B, f_C are defining functions of the initial halfspaces A, B, and C, and $\wedge_\alpha$, $\vee_\alpha$ are symbols of R-functions for union and intersection correspondingly. This function takes zero values at the points of the rays TU and SW, thus creating external zeroes of the polygon.

The construction of a polygon with internal zeroes of the defining function (Fig. 4) is based on a kind of cell partitioning [7] (see Fig. 1d). The boundaries of two halfspaces ($A \cap B$) (Fig. 4a) and ($\neg A \cap C$) (Fig. 4b) share the segment ST. After applying union operation to these halfspaces we obtain the desired polygon (Fig. 4c). The defining function constructed using the set-theoretic expression and R-functions is as follows:

$$f_{p\_int} = (f_A \wedge_\alpha f_B) \vee_\alpha (-f_A \wedge_\alpha f_C) \tag{4}$$

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 polygon defining function without external and internal zeroes can be constructed from the functions in Eq. 3 and Eq. 4 using algebraic summation:

$$f_p = f_{p\_ext} + f_{p\_int} \tag{5}$$

In total, the evaluation of the function f_p requires applying six R-functions and one algebraic summation.

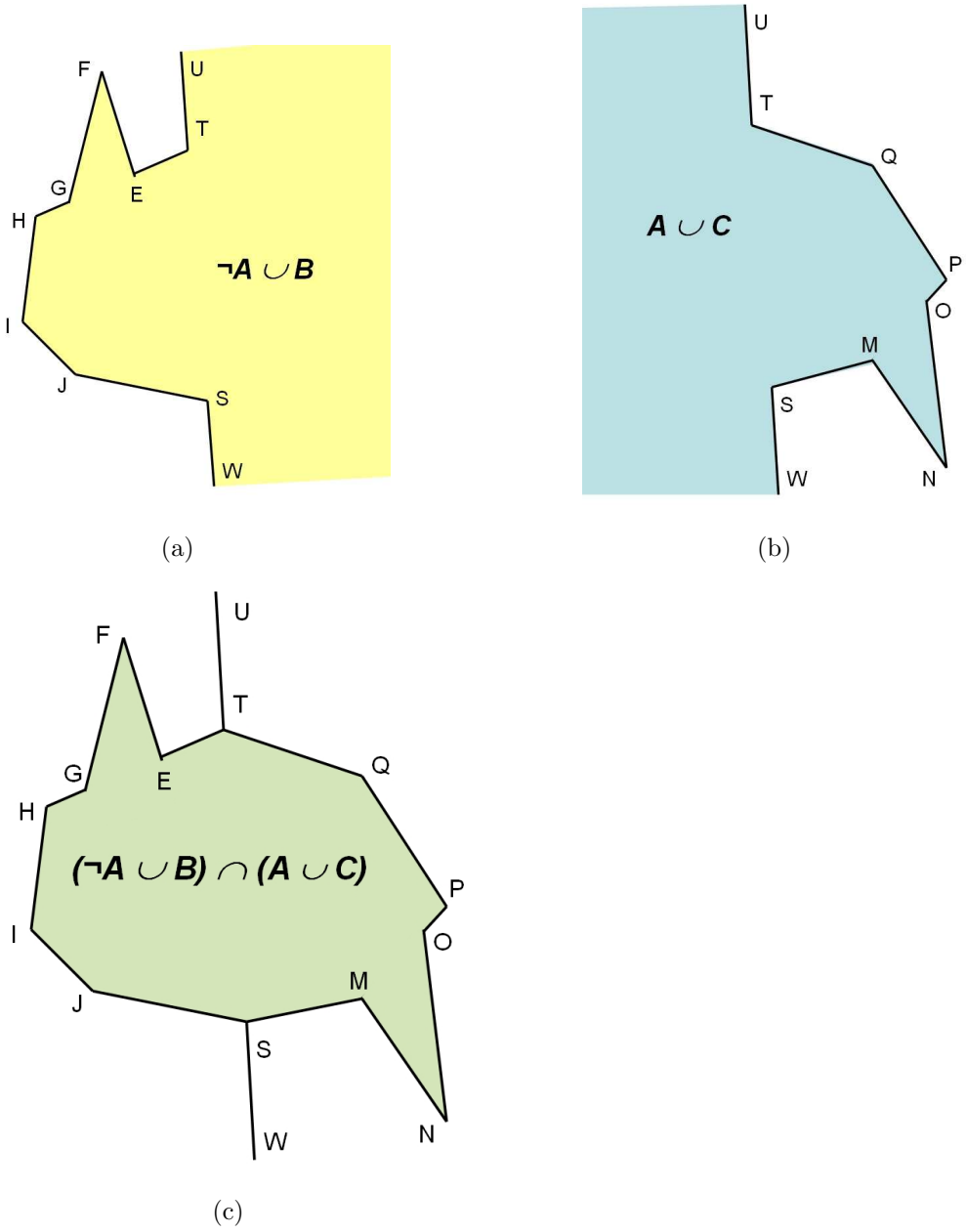
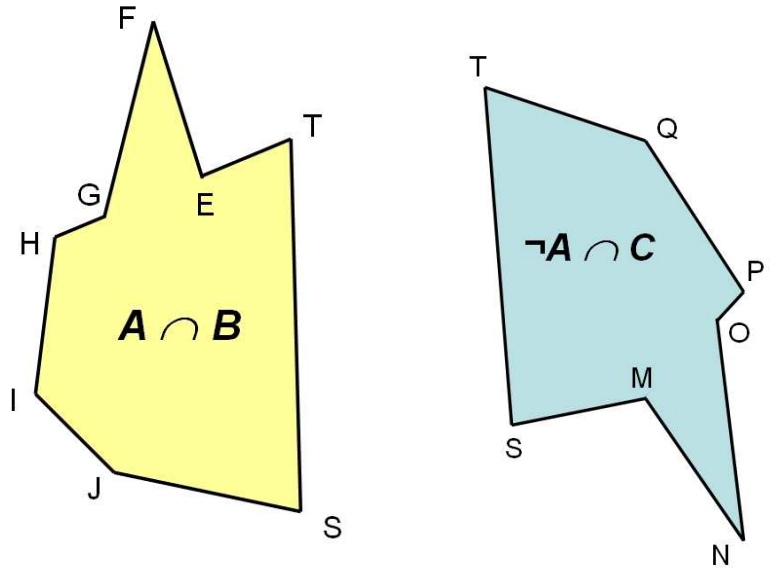
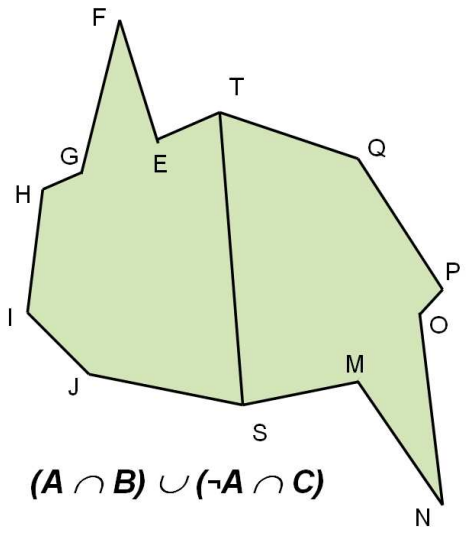


Figure 3: Construction of a polygon with external zeroes of the defining function



(a)

(b)



(c)

Figure 4: Cell partition of the polygon with internal zeroes of the defining function at the points of the segment ST

3.3. Proposed set-theoretic approach

We propose to directly construct a set-theoretic expression for the polygon in such a way that the corresponding defining function does not have internal or external zeroes. To prevent extra zero sets, every point in the interior of the resulting polygon has to be in the interior of some area (argument of a set operation) such that the union and intersection operations ensure that the gradient at the point will be well defined.

The proposed steps of the set-theoretic construction of a polygon with the defining function without non-boundary zeroes are shown in Fig. 5. At each step, either the union or the 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 using the set-theoretic expression shown in Fig. 5d and R-functions is as follows:

$$f_s = ((f_A \vee_\alpha f_C) \wedge_\alpha f_B) \vee_\alpha (-f_A \wedge_\alpha f_C) \tag{6}$$

In total, the evaluation of the function f_s requires applying four R-functions. Another advantage of using only set-theoretic operations to construct the polygon, if compared with the algebraic approach, is that instead of standard R-functions with square roots we can apply SARDF-operations [12] to get smoothly approximated distance field for the polygon.

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, for an arbitrary polygon 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 of Fig. 2), 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: $(\neg A \cup B) \cap C$. Such special cases with lower number of operations require additional treatment for detecting and handling.

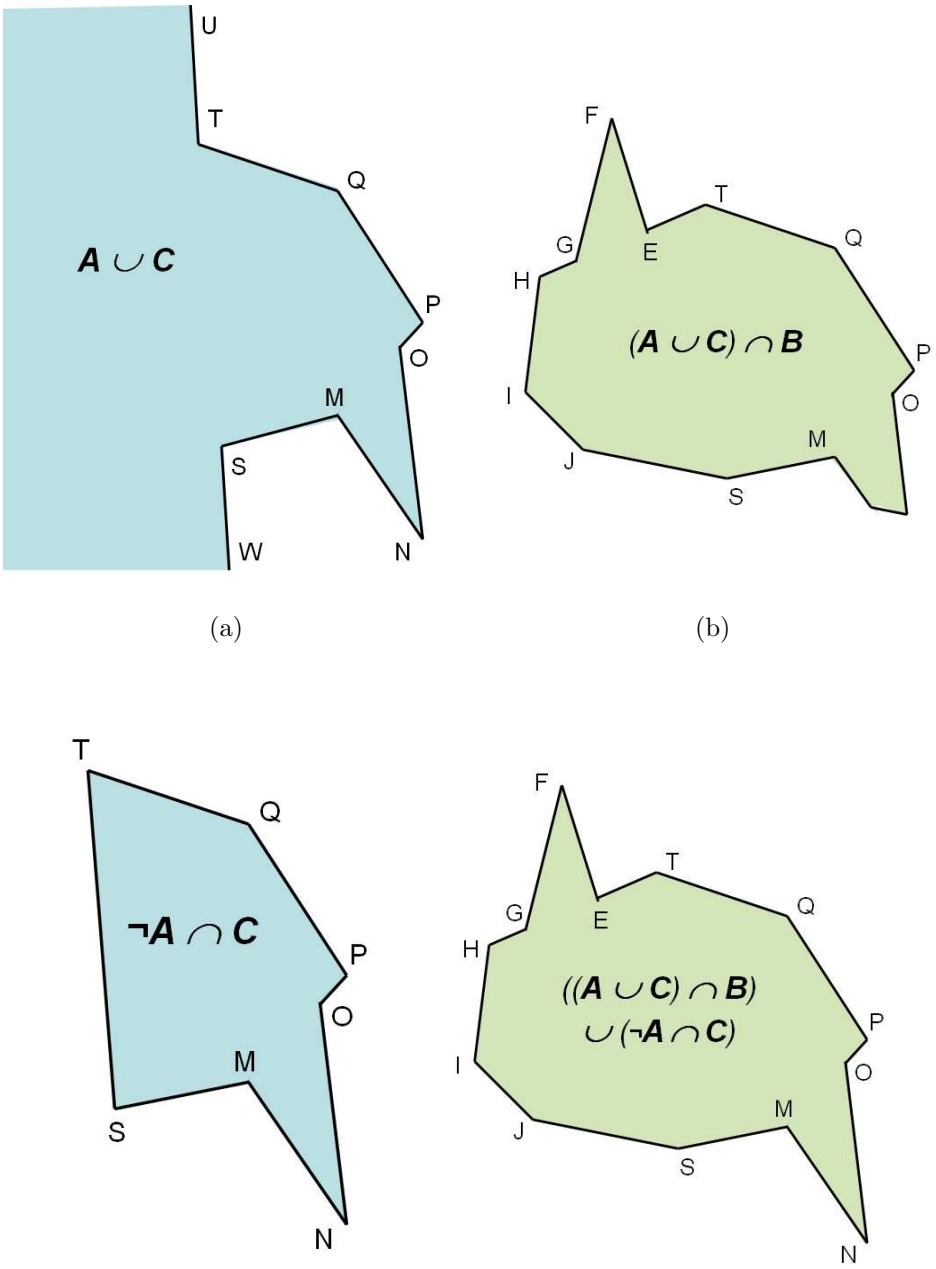


Figure 5: Set-theoretic construction of the polygon with a defining function without non-boundary zero points.

3.4. BSP-tree construction and function evaluation

In this section we describe our algorithm for the scalar field generation for the given polygonal object. The set-theoretic description and the scalar field generation for a 2D polygon, presented in the previous section, reduce 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)) [10]. By definition, a BSP-tree is a data structure that represents the recursive subdivision of N-dimensional space into two half-spaces by hyperplanes. Each space partitioning procedure separates N-dimensional cells that are in the interior and in the exterior of the half-space. The hyperplane does not have to be explicitly defined, as it can be obtained from the cells. The method of space partitioning works equally for closed and non-closed sets and does not depend on topology of the original set. Examples of the BSP-tree construction can be found in many papers related to BSP-trees such as [18].

As the BSP-tree algorithm treats the sets of different dimensions identically, 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 pre-processing 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 the BSP-tree in the 3D case. The construction of BSP-tree in the 2D case can easily be done in a similar way.

3.4.1. Basic BSP-tree construction

The construction of the BSP-tree is a recursive procedure that contains the following main steps:

1. selection of the base polygon according to some criteria;
2. construction of the partitioning hyperplane containing the base polygon;
3. division of the rest of polygons into the "positive" and "negative" depending on whether the polygons lie in the interior or the exterior after partitioning by the hyperplane;

4. 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 [10]. 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 BSP-tree building procedure processes the list of the input polygonal faces. At the first iteration this list is a list of all faces in the input mesh. We select a polygon that is the base for a partitioning (cutting) plane passing through it. The selection criteria for the polygon are discussed in subsection 3.4.3. 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 following 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, $\mathbf{n}$ is the normal to the plane, and $\odot$ is the symbol of the dot product. If some polygon intersects the partitioning plane, it is split into two parts, each of which is added to the respective list. Note that unlike BSP-tree construction for computer graphics purposes the polygon can not just be added to the both "positive" and "negative" lists without splitting. In our method we obtain the correct tree only if the polygon that intersects partitioning plane is split. The polygons in "positive" and "negative" lists are processed recursively to create positive and negative subtrees of the created node.

3.4.2. 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:

$$f(x) = \begin{cases} f_a, & \text{positive and negative subtrees are empty} \\ f_a \wedge_\alpha f_b, & \text{negative subtree is empty and positive is not} \\ f_a \vee_\alpha f_c, & \text{positive subtree is empty and negative is not} \\ ((f_a \vee_\alpha f_c) \wedge_\alpha f_b) \vee_\alpha (-f_a \wedge_\alpha f_c), & \text{otherwise} \end{cases} \quad (7)$$

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.

3.4.3. 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 the criteria for the base polygon selection we can use different approaches. In our work we use two approaches: "naive" selection, where the base polygon is randomly selected from the polygon list, and "optimized" selection. In this section we present our techniques to optimize BSP-tree for our method. Unlike the methods for BSP-tree optimization for rendering purposes, such as in [18], our motivation for BSP-tree optimization is reduction of numerical errors while applying the BSP-tree construction and the function evaluation procedure for the constructed tree. Moreover, the split operation for the polygons that intersect partitioning plane should be applied and each split operation can add numerical errors. Therefore, for the optimization we use the criteria that allow for obtaining a tree with the following properties:

- Minimal number of polygon splitting operations reducing the total number of nodes and the number of operations in the function evaluation
- Minimal computational errors during the function evaluation and the BSP-tree construction;

To provide the generation of BSP-trees with these properties, we propose several optimization criteria. Given the list of the polygons M containing the list of vertices V, we select the base polygon with the partitioning plane P, if one of the following conditions is satisfied:

- $K_{split} = 0$

- $K(P) = K_{dist}(P) * K_{angle}(P)$ is maximal and $K_{split} \neq 0$ for each polygon in M

Here
$K_{split}(P) = N_{split}$
$K_{dist}(P) = \min\limits_{v \in V, v \notin P}(distance(v, P))$
$K_{angle}(P) = \min\limits_{m \in M, m \notin P}(angle(\mathbf{n}_m, \mathbf{n}))$
where 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 appearing of degenerate faces after splitting of the polygon and minimization of K_{split} allows to minimize the number of the polygon splitting operations. Another way to minimize the accumulation of computational errors is to replace polygon splitting operation (with calculations of new vertices) by the evaluation of predicates on the base of plane equations, however it lies outside the scope of this paper.

Fig. 6 shows examples of BSP-fields generated using an optimized BSP-tree for a 2D polygon. The shown colour maps correspond to different R-functions: min/max, R-functions with square roots (Eq. 2), and SARDF operations ("smooth min/max") [12]. 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 areas 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 ones. Therefore, the choice of R-functions entirely depends on the requirements of particular applications to the scalar field.

4. Applications

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.

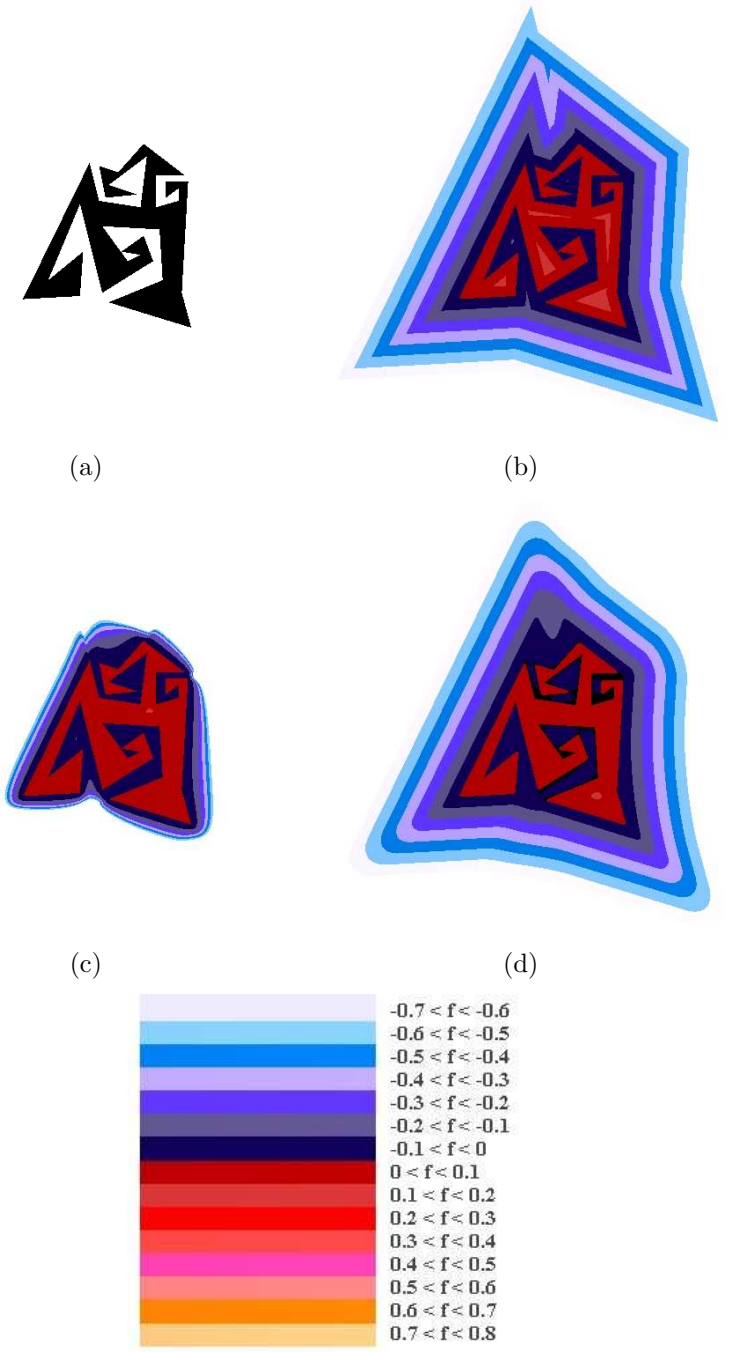


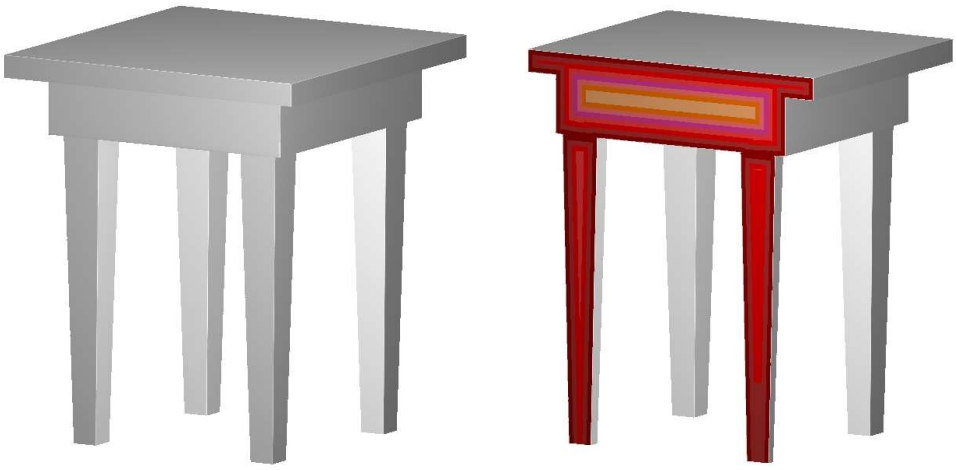
Figure 6: Scalar fields for an optimized BSP-tree built for the given polygon (a) with applied min/max functions (b), R-functions (c), and SARDF operations (d); (e) Colour distribution.

				Non-optimized		Optimized	
Model	Vertices	Faces	Planes	Splittings	Set operations	Splittings	Set operations
Table	64	124	59	56	113	0	68
Block with hole	72	144	20	118	80	0	31
Dolphin	282	562	562	1503	3249	572	1775
Rocker arm (low poly)	470	940	933	3253	6221	976	2828
Chain	768	1536	384	2961	3656	1219	2260

Table 1: Tests of the BSP-tree optimization

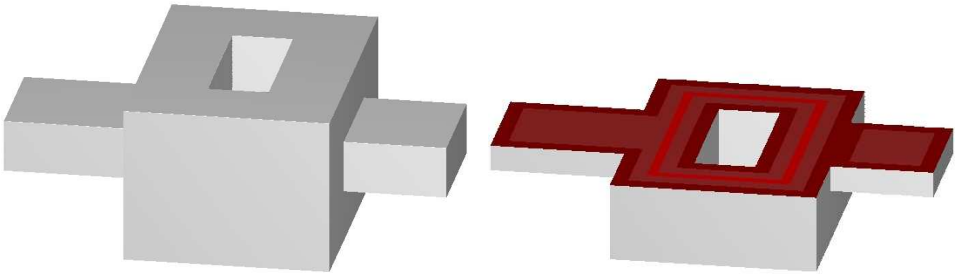
		Non-optimized		Optimized	
Model	Polygons	BSP create	query	BSP create	query
Table	124	~ 0	~ 0	~ 0	~ 0
Block with hole	144	~ 0	~ 0	~ 0	~ 0
Dolphin	562	0.441	0.16	3.385	0.1
Rocker arm (low poly)	940	0.411	0.33	23.273	0.16
Chain	1536	0.531	0.18	20.74	0.12
Fauset	2143	1.933	0.691	73.977	0.3
Triceratops	5660	3.835	0.962	297.027	0.621
Fandisk	12946	35.091	1.022	1265.05	0.421
Rocker arm (high poly)	20088	58.825	8.982	19267.4	2.884
Buddha	35800	183.263	14.13	14744.8	8.211
Turtle	102998	269.858	18.436	n/a	n/a

Table 2: Timing of the BSP-tree optimization. "BSP create" denotes time (in seconds) to construct BSP-tree, "query" denotes the function evaluation time for 1000 queries (in seconds) at random points for the constructed field. Timings are absent for the optimized BSP-tree for the last model due to very long time for the creation of the tree.



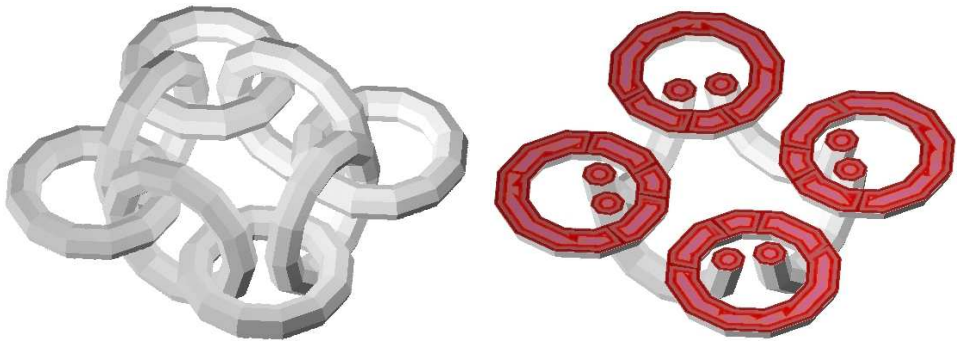
(a) (b)

Figure 7: Table model



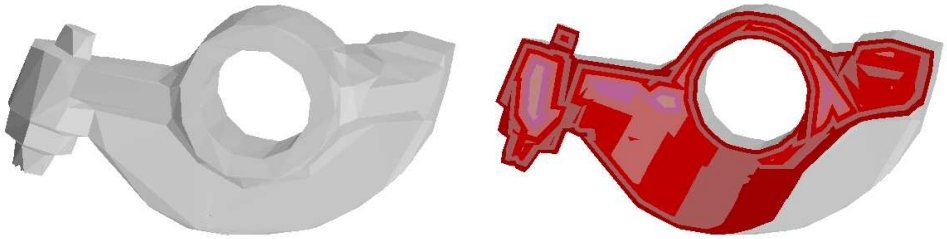
(a) (b)

Figure 8: Block with hole model



(a) (b)

Figure 9: Chain model



(a) (b)

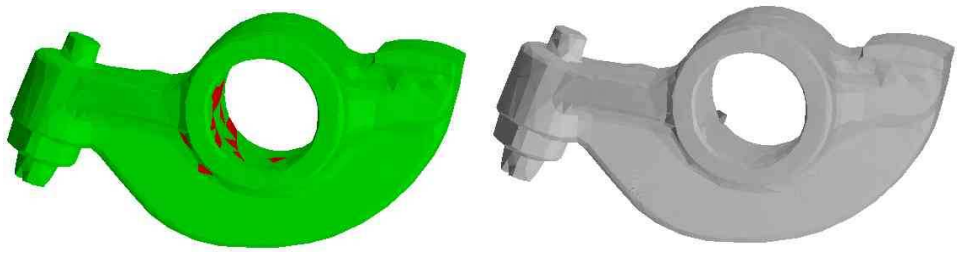
Figure 10: Rocker arm model

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

We did not make any assumption about objects geometry features or their 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 easily be converted using existing methods. Figs. 7 and 8 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 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 quite a small number of polygons, the approximation methods based on RBF and MPU [31][40][19] can only produce some oval shapes for the block and for the hole, which is unacceptable in most applications. Figures 8, 9 and 10 illustrate the conversion of polygonal 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. Timings for optimized and non-optimized trees are shown in Table 2. Several models were selected with different number of polygons yet not all of them are illustrated by figures. In our implementation, the construction of optimized BSP-trees for large meshes can be relatively slow (for models with more than 100000 polygons it becomes impractical), however, processing time for the point query for optimized trees is significantly lower. It can be seen that the optimization can decrease the function evaluation time up to the factor of



(a) (b)

Figure 11: A model with missing polygons and a zero-isosurface of the constructed BSP-field.

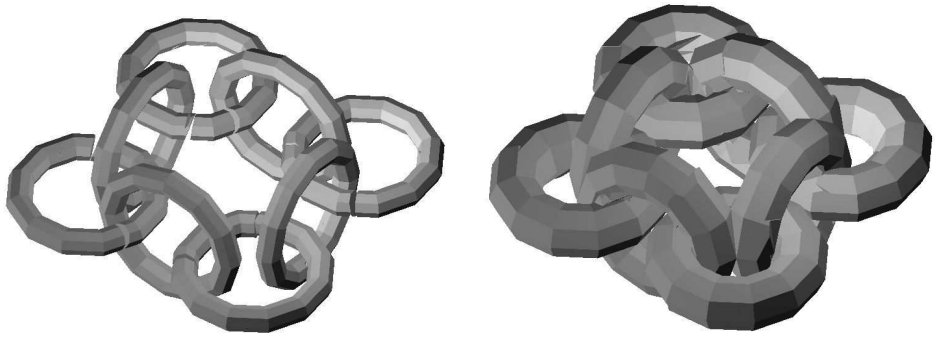
three. Also, one can see that the speed of the BSP-tree construction and optimization does not depend directly on the number of faces, and it can take more time to construct the tree for the mesh with a lower number of polygons than for the mesh with a larger number of polygons.

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 represents the model that is topologically and geometrically close to the original model. In Fig. 11 we show how our method can be applied to a model with missing polygons. From the original mesh (see Fig. 11a) we removed several triangles (red colour in the figure) and created a BSP field from the rest of triangles (green colour). The resulting functionally represented model (see Fig. 11b) is visually close to the original, however some artefacts did appear, because we can not guarantee the recovery of the unknown boundary of the surface. In general, this conversion is applicable to incomplete meshes with cracks and other small defects.

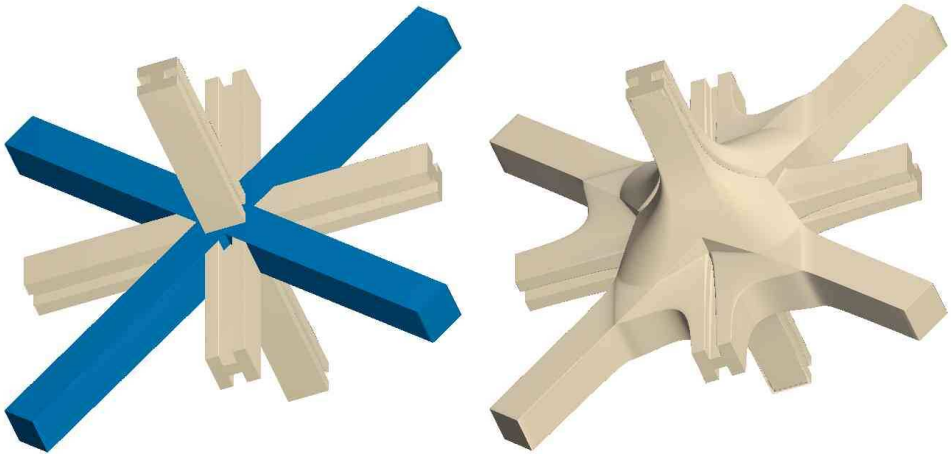
4.3. Offsetting and blending of polygonal meshes

Offsetting operation in solid modelling creates a contracted or expanded version of the given solid. In the case of the function-based model, we can implement a simple type of an offset operation with the following modification of the defining function: $F(x, y, z) - d \geq 0$, where d is an offset value. In general, offsetting operation applied to BSP-fields does not guarantee constant distance offset. For example, for the Chain model (see Fig. 9)



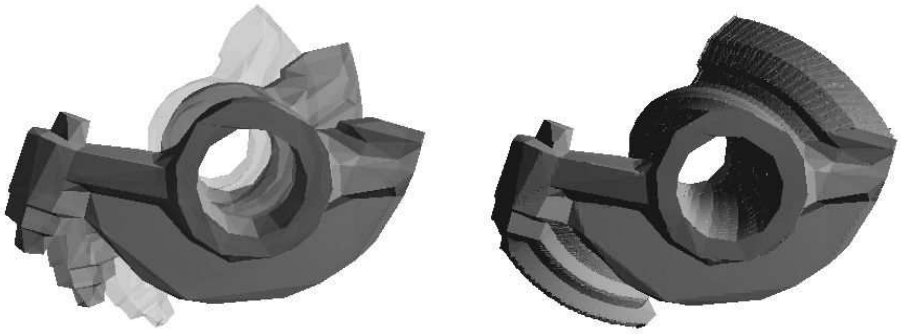
(a) (b)

Figure 12: Offsetting of the Chain model by changing the function value for the isosurface



(a) (b)

Figure 13: Blending union with added material between two polygonal models: a) Initial two polygonal objects, b) Blending union



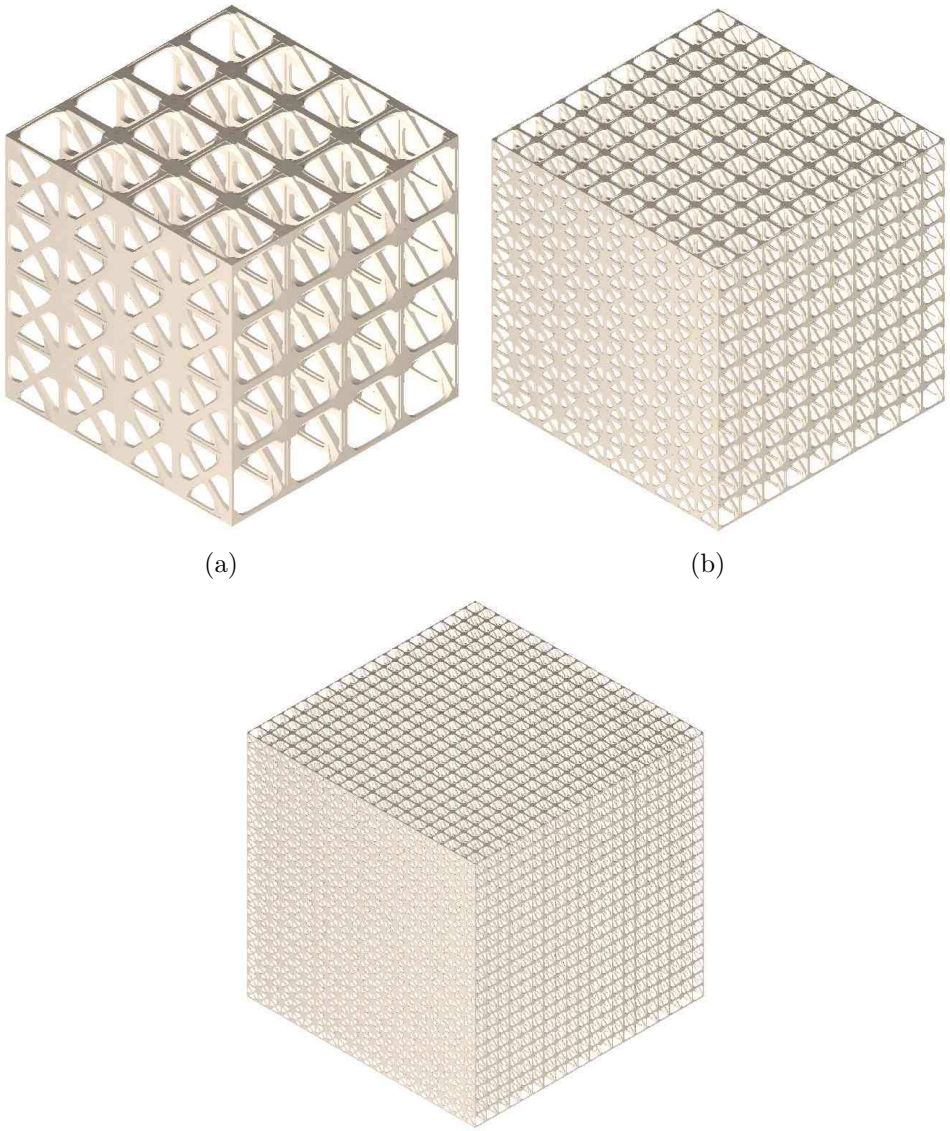
(a) (b)

Figure 14: Phases of motion of the functionally represented polygonal rocker arm along a helical trajectory (a), and the solid sweep (b).

we show negative offset in Fig. 12a and positive offset in Fig. 12b. It can be seen that the defining function has distance properties outside the object, however there is no distance property inside the object that results in artefacts of the negative offset. This can be the basis for the future research. If the distance property is provided, we can also apply a blending operation to two converted polygonal objects. Fig. 13 shows a bounded blending union operation [22] with added material between two polygonal objects converted to scalar fields: a cross made of H-channel beams (see Fig. 13a, light colour) and a cross made of square beams (see Fig. 13a, dark colour).

4.4. Sweeping

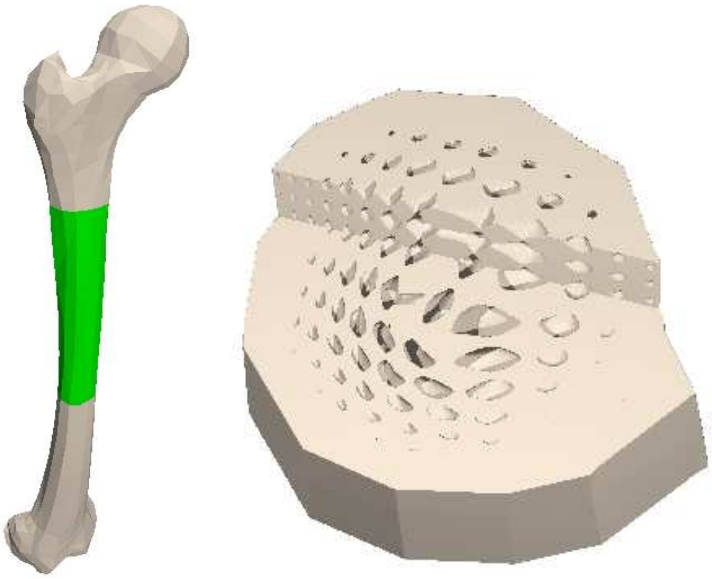
Sweeping by a moving solid is one of the most important operations in CAD. It can be applied in simulation of numerically controlled machining, robot motion planning, and maintainability simulation. In general, the operation is problematic for solids with complex topology, shape varying sweep generators, and sweeps with self intersections. We have applied the algorithm for sweeping by a moving solid [30] devised for function-based solid models. Its advantage is the generality of the approach resolving the above difficulties. Fig. 14 shows a sweep by a mechanical part with non-zero genus moving along a helical trajectory. The initial polygonal model was converted to a scalar field model, and the algorithm [30] was applied to obtain the scalar field defining the final sweep. Note that the artefacts in the sweep surface were caused by the given numerical threshold for the algorithm.



(a) (b)

(c)

Figure 15: Replicating of the converted polygonal model inside a cubic volume with three different values of lattice density (required memory and rendering time are the same for all three lattices).



(a) (b)

Figure 16: a) A polygonal bone model, green colour denotes the extracted segment of interest; (b) Cut-away section of the the segment of interest showing the generated porous microstructure.

4.5. Modeling microstructures

Recent developments in heterogeneous objects modeling include internal structures of objects with size of details orders of magnitude smaller than the overall size of the object. The use of scalar fields allows for procedural modeling of microstructures such as lattices and porous media in a compact, precise and arbitrarily parametrized way [21].

Lattices are spatial structures consisting of some initial model periodically replicated inside the given volume. In Fig. 15 we take the initial model, the blending union between two polygonal objects (see Fig. 13b), and replicate it using a periodic space mapping. In this example we use a box as a bounding volume for microstructures, however other bounding volumes can be applied. Note that the rendering time (we use ray-casting here) is almost the same for these three lattices as we use the same function with different parameter values for the lattice density.

Modeling porous media, for example, bone microstructures, can be done in a similar way. The representation of the bone geometry by a scalar field helps to create procedural function-based models of regular and pseudo-random pores distribution inside the bone volume (see Fig. 16). In this example we extracted the segment of interest from the initial mesh and created the porous structure for this segment. Note that the pore sizes decrease close to the bone surface. This is provided by the porous microstructure generation procedure based on the generated scalar field for the polygonal geometry of the bone segment. The parameterized model of pores distribution can be applied to study the development of bone diseases such as osteoporosis.

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-value isosurfaces. The algorithm provides an exact representation of polygonal objects including those with small number of vertices (less than a hundred for instance), 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 obtained scalar fields using B-splines or wavelets. However, it would mean loosing the exact representation of the initial polygonal object.

The proposed algorithm generates a well-formed set-theoretic representation for a polygonal object (see [27]), which is in some aspects superior in comparison with the 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 a low number of polygons. Further research will be needed in the direction of monotone formula construction for 3D case, for example, on the detection of the special cases when the number of required R-functions is reduced from four to two as described in 3.3.

It is obvious that polygonal splitting and calculation of new triangle vertices lead to numerical error accumulation. This can be avoided by switching to pure predicate evaluation based on initial plane equations as discussed in [34][2]. This is another area of our future research.

Acknowledgements

The polygonal models are from the INRIA Gamma team research database and the shape repository of the AIM@SHAPE project.

References

[1] Allegre R., Galin E., Chaine R., Akkouche S.: The Hybrid Tree: Mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. *Graphical Models 68*, 1 (January 2006), 42–64.

[2] Bernstein G., Fussell D.: Fast, Exact, Linear Booleans. *Computer Graphics Forum 28*, 5 (2009), 1269–1278.

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

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

[5] Biswas A., Shapiro V., Tsukanov I.: Heterogeneous material modeling with distance fields. *Comput. Aided Geom. Des. 21*, 3 (2004), 215–242.

[6] Bloomental J. et al: Introduction to Implicit Surfaces. *Morgan Kaufmann Publishers Inc.* (1997)

[7] 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.

[8] Cohen-Or D., Solomovic A., Levin D.: Three-dimensional distance field metamorphosis. *ACM Trans. Graph. 17*, 2 (1998), 116–141.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Heterogeneous Objects Modelling and Applications *Lecture Notes in Computer Science, vol. 4889, Eds. Pasko A., Adzhiev V., Comninos P., Springer, (2008), 285 p.*

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

[15] 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.

[16] 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, 227–235.

[17] 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.

[18] Naylor B.F.: Constructing Good Partitioning Trees. *Proc. Graphics Interface '93*, (1993), 181–191.

[19] 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.

[20] Pasko A., Adzhiev V., Sourin A., Savchenko V.: Function representation in geometric modeling: concepts, implementation and applications. *Vis. Comp. 11*, 8 (1995), 429–446.

[21] Pasko A., Vilbrandt T., Fryazinov O., Adzhiev V.: Procedural Function-based Spatial Microstructures *Technical Report TR-NCCA-2009-02, ISBN 1-85899-123-4, The National Centre for Computer Animation, Bournemouth University, UK* (2009), 15 p.

[22] Pasko G., Pasko A., Kunii T.: Bounded blending for function-based shape modeling. *IEEE Comput. Graph. Appl. 25*, 2 (2005), 36–45.

[23] Peterson D.: *Halfspace representation of extrusions, solids of revolution, and pyramids.* Tech. rep., Sandia National Laboratories, Albuquerque, NM, (1984).

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

[25] 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.

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

[27] Shapiro V.: Well-Formed Set Representations of Solids. *International Journal of Computational Geometry and Applications 9*, 2 (1999), 125–150.

[28] Shapiro V.: Semi-analytic geometry with R-functions. *Acta Numerica 16* (2007), 239–303.

[29] 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.

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

[31] Savchenko V. V., Pasko E. A., Okunev O. G., Kunii T. L.: Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 4 (1995), 181–188.

[32] Shapiro V.: Real functions for representation of rigid solids. *Computer-Aided Geometric Design 11*, 2 (1994), 153–175.

[33] Shapiro V., Vossler D. L.: Separation for boundary to CSG conversion. *ACM Trans. Graph. 12*, 1 (1993), 35–55.

[34] Sugihara, K., Iri, M.: A solid modelling system free from topological inconsistency. *J. Inf. Process. 12*, 4 (1989), 380–393.

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

[36] Wu J., Kobbelt L.: Piecewise linear approximation of signed distance fields. In *Proceedings of Vision, modeling and Visualization 03* (2003), pp. 513–520.

[37] Woodwark J. R., Wallis A. F.: Graphical input to a Boolean solid modeller. In *Proc. CAD'82*, (1982), 681–688.

[38] Tang K., Woo T.: Algorithmic aspects of alternating sum of volumes part 1: data structure and difference operation. In *Computer-Aided Design 23*, 5 (1991), 357–366.

[39] Tang K., Woo T.: Algorithmic aspects of alternating sum of volumes. Part 2: Nonvergence and its remedy. In *Computer-Aided Design 23*, 6 (1991), 435–443.

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

[41] Zhao H., Osher S.: Visualization, analysis and shape reconstruction of unorganized data sets. *In Geometric Level Set Methods in Imaging, Vision and Graphics*, Springer, (2002), 681–688.