Computer-Aided Design 58 (2015) 43–50

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Reliable detection and separation of components for solid objects defined with scalar fields☆

Oleg Fryazinov*, Alexander Pasko

Bournemouth University, UK

HIGHLIGHTS

- A technique for find the number of disjoint components for a model defined implicitly.
- Various methods for the separation of components for these models.
- An adaptive spatial continuation for the fast and reliable enumeration.

ARTICLE INFO

Keywords:
Component analysis
Scalar fields
Implicit surfaces
Component detection
Component separation

ABSTRACT

The detection of the number of disjoint components is a well-known procedure for surface objects. However, this problem has not been solved for solid models defined with scalar fields in the so-called implicit form. In this paper, we present a technique which allows for detection of the number of disjoint components with a predefined tolerance for an object defined with a single scalar function. The core of the technique is a reliable continuation of the spatial enumeration based on the interval methods. We also present several methods for separation of components using set-theoretic operations for further handling these components individually in a solid modelling system dealing with objects defined with scalar fields.

1. Introduction

Modern development of CAD/CAM shows the shift from the representation of the objects by its boundary to volumetric representations. This allows users to consider internal structure of the object as well as to define volumetric attributes and properties of the objects in the way closer to the real-life heterogeneous object representation. One of the useful ways to represent the real-life volume objects is using scalar fields. This means that for any point in space a predicate (function) is defined allowing to distinguish points inside the object, outside the object and on the surface of the object. In additional, it gives a measure of some algebraic distance from the given point to the object surface. Such a scalar field is usually considered as a definition of the object geometry in the implicit form.

Scalar fields allow for performing operations on the objects that are very hard to achieve using traditional methods operating with surfaces (Boundary Representation or BRep), such as blending within a certain area or shape metamorphosis with arbitrary changes in topology. On a contrary, some problems that have been already solved for BRep models are yet to be solved for the geometry defined with scalar fields. One of these questions is topological analysis, i.e., detection of the holes, disjoint components and other features for the geometry defined with a scalar field. In this paper, we focus on the analysis of disjoint components, as it is an open issue in modelling with scalar fields. This question is becoming more important with the rapid development of digital fabrication and 3D printing hardware. It is clear that in case of wrongly modelled object the model can break into pieces during the fabrication process and in some extreme cases even break the 3D printing hardware itself.

Traditionally models defined in the implicit form were analysed only if the defining function (scalar field) was simple enough and easy to analyse. In this work, however, we are not restricting the defining function and only assume that the model is bounded and we know the box which encloses the point set belonging to the interior and the surface of the object. In practice, where we are taking into account practical modelling systems that deal with the implicit form, such as BlobTree [1] or HyperFun [2], it can be

☆ This paper has been recommended for acceptance by Vadim Shapiro.

* Corresponding author. Tel.: +44 1202 961880.
E-mail address: ofryazinov@bournemouth.ac.uk (O. Fryazinov).

http://dx.doi.org/10.1016/j.cad.2014.08.019
0010-4485/

seen that the defining function can be very complex and far from polynomial.

To date, an analytical solution for topological analysis for the general case has not been found. However, a numerical solution can be found, but it should be reliable, meaning the result should be exact within the given precision. In this paper, we present a technique allowing for detecting the number of disjoint components in the model represented in the implicit form and methods to separate the detected disjoint components. The detection and separation operations result in a continuous and smooth scalar field for each component. These operations are designed such that they can be directly used in a modelling system dealing with the objects represented in the implicit form.

The main contributions of this paper are the following:

1. For a solid model defined with a scalar field, we propose a technique for identifying the number of disjoint components with the pre-defined tolerance without setting any severe restrictions to the tolerance.
2. Various methods for the separation of components are presented.
3. An adaptive spatial continuation is presented for the fast and yet efficient reliable enumeration.

2. Related work

The problem of detection of disjoint components was usually considered in the scope of general topological analysis or as an applied problem. Thus, for the purposes of collision detection, a two dimensional point set was analysed in [3]. For polygonal meshes the analysis of disjoint components was mostly the analysis of the number of shells (structure of connected triangles) within the polygon soup. The question of the number of disjoint components in the polygonal manifold mesh was discussed in [4], where the *corner table* data structure was used to separate disjoint shells.

The problem of detection of the number of disjoint components is related to the null-object detection, for example, when intersecting two objects for collision detection. The collision detection algorithms for BRep typically inspect the boundary components of the intersection to confirm that the boundary is empty [5]. The null object detection was addressed in the Constructive Solid Geometry (CSG) with the primitive redundancy principle allowing for reduction of the CSG-tree to the null tree in the case of the null-object by removing redundant primitives and simplifying the tree [6]. In the case of voxelized objects, the collision detection is addressed by adding a reference to each object to the voxels touched by the object bounding box [7] followed by checking the case when several objects share the same non-empty voxel.

One of the ways to analyse the topology of the model defined implicitly is applying the Morse theory [8]. Thus, in [9] it was used to analyse the topology of the implicit surfaces for the polygonization purposes. However, the Morse theory is hardly applicable for general purposes because of the requirements of C^2-continuity of the defining function and the necessity to derive and to analyse the expressions for the first derivative. Another exact analysis of the topology and geometry of the models defined in the implicit form was done in [10], where the class of models was limited to those defined by the primitive polynomials. Also the analysis of the discrete scalar field by using Reeb graphs along with the Morse theory was presented in [11] Other works on topological analysis of isosurfaces of scalar fields are discussed in [12].

The problem of separation of different components in the object was explicitly set and analysed for two-dimensional geometry defined with BRep in [13] as well as for separating set of connected points by finding the suitable bounding volume for each component in the set [14].

3. Background

3.1. Affine arithmetic and revised affine arithmetic

Affine Arithmetic was introduced in the early 1990s as an extension to the Interval Arithmetic and is a technique allowing to perform computations over uncertain values [15]. Comparing with classic Interval Arithmetic, Affine Arithmetic is generally considered as the technique that provides tighter bounds for computer quantities.

Uncertain values in Affine Arithmetic are represented by the affine forms, i.e., polynomials as follows:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \tag{1}$$

where x_i are known real coefficients and ε_i are noise symbols, i.e., symbolic variables with the values assumed to lie in the interval $\varepsilon_i \in [-1, 1]$.

In Affine Arithmetic, the formula evaluation is performed by replacing operations on real quantities by their affine forms. Similarly to Interval Arithmetic, the inclusion property is applied to Affine Arithmetic, i.e., for any operation $\otimes$:

$$A \otimes B \supset \{a \otimes b, a \in A, b \in B\}$$

where a and b are real values and A and B are uncertain values in the affine form.

All the operations on the affine forms can be divided into affine (exact) and non-affine (approximate) operations. An affine operation is a function that can be represented by the linear combination of the noise symbols of its arguments. Non-affine operations cannot be performed over the linear combination of the noise symbols. In this case, an approximate affine function is used and a new noise symbol is added to the affine form to represent the difference between the non-affine function and its approximation. Additional details regarding the construction of both affine and non-affine operations can be found in the literature related to Affine Arithmetic [15,16].

One of the obvious drawbacks of the Affine Arithmetic is that each non-affine operation introduces one extra noise symbol and therefore for the functions that contain large number of non-linear operators including multiplication the amount of the data for the affine forms can be unbearable. In [17] it was shown that Revised Affine Arithmetic is more suitable for the interval computations for large models defined in the implicit form, as it keeps the number of noise symbols constant still providing tight bounds for the interval of the function.

The revised affine form for the purposes of space partitioning is presented as the following

$$\hat{x} = x_0 + \sum_{i=1}^{3} x_i\varepsilon_i + e_x[-1, 1], \quad e_x \geq 0. \tag{2}$$

Here we have three independent uncertain values, one for each coordinate.

3.2. Affine arithmetic-driven space partitioning

The inclusion property of the Interval Arithmetic as well as its successors, including Affine Arithmetic, allows for partitioning the space into cells that are inside, outside and potentially intersect the surface of the object. The main idea behind the subdivision is to test each cell for inclusion of the zero set of the function, to reject those cells that do not contain zero-value points and subdivide the cell into eight otherwise.

The inclusion property for Affine Arithmetic means that, if the affine form for the expression does not contain the zero value, then the actual function range for the given input interval does