Review of Computer Engineering Research
2017 Vol.4, No.1, pp.30-37
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76.2017.41.30.37

PERTURBATION FUNCTIONS FOR COMPACT DATABASE

Sergey I. Vyatkin[1]

[1]Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences, Academician Koptug ave. 1, Novosibirsk, Russia

Article History
Received: 8 December 2016
Revised: 3 January 2017
Accepted: 16 February 2017
Published: 19 April 2017

Keywords
Perturbation functions
Geometric objects
Geometric operations
Relations
Interactive object modeling
Visualization.

ABSTRACT

3D Objects based on perturbation functions are considered in this paper. For shape creating a set of algorithms and software based on function-defined surfaces that perform an interactive rate and enable intuitive operations was proposed. Interactive modification of the 3D objects allows us to provide high level of detail leading to a photo-realistic appearance of the resulting shapes.

Contribution/ Originality: This study uses a new technique for free-form representation created by mean of the analytical functions which have the following advantages: fewer data for mapping curvilinear surfaces (short database description), fewer geometric operations, simple animation and deformation of surfaces.

1. INTRODUCTION

The idea of representation of an entire complex object by a single real function was applied effectively in skeletal implicit [1] and min/max operations for CSG proposed in Ricci [2]. But it have not found wide acceptance. Because C^1 discontinuity was as a reason. CSG includes algebraic primitives; algebraic patches in Constructive Shells, CSG-sweep models, non-uniform rational B-splines (NURBS) primitives of CSG trees, etc. Implicit models include algebraic surfaces and skeletal objects. B-Rep includes algebraic patches and polygonization algorithms. Sweeping generalized cylinders with skeletal curves. Many to B-Rep conversion are CSG-voxels, CSG-ray-rep, CSG-B-Rep, etc. 3Data (voxels) representation have a polygonization algorithms. Theoretical possibility to derive an implicit description of a surface swept by a mowing solid was proposed in Wang and Wang [3]. Symbolic computations required to produce a formula for the implicit form. Attention is paid in CAGD to functionally defined surfaces because of their closure under some operations such as offsetting and blending [4]. Generalization of the functionally defined objects by introducing a deformation technique using a matrix of free vibrations was proposed in Sclaroff and Pentland [5]. Research on collision detection for implicit surfaces proposed in Gascuel [6]. Similar

+ Corresponding author

polygonization algorithms stress the common nature of implicit and voxel models. Time-dependent transformation of skeletal implicit such as metamorphosis and scheduled Fourier volume morphing are presented in [7, 8]. So, representations by real functions are used in modeling and visualization. These models are not connected to each other. These models are not connected to such well-known representations as CSG, B-rep, sweeping, and spatial partitioning such as voxel models. This obviously retards further research. A uniform functionally description is needed to fill these gaps. It has to: unify all functionally based approaches; be convertible from other representations; be dimension independent; have as reach as possible a set of operations and relations; the research directions needed to introduce the function representation; elaboration of a set of operations closed on function-based multidimensional space-time objects; specification, implementation, and application of an interactive modeling system based on functionally defined representation; analysis and experiments on different computer architectures. Investigation of field of application function representation is especially effective. There are requirements to a modeling environment: provision of multiple representations with converting all other representations into the function representation; uniform description of objects of different dimensions; simple definition, including a symbolic one, of a new primitives and operations by the end user; ability to create complex objects from simple ones keeping the type of the representation, closure property; access to simple components of a complicated model with a possibility to modify them and relationships between them; static and time-dependent geometric modeling; efficiency of rendering algorithms.

Conclusions from the above overview are: more deep connection with CSG is possible; more deep connection with sweeping is possible; more deep connection with voxel models is possible.

New method description of functionally objects without their approximation by polygons or spline-patches is considered.

2. GEOMETRIC OBJECTS

Describing complex geometric objects by specifying the function of deviation (second-order function) from the base surfaces was proposed in Vyatkin [9]. The object is a composition of the base surface and the perturbation functions

$$F'(x,y,z) = F(x,y,z) + \sum_{i=1}^{N} R_i(x,y,z)$$

where the perturbation function $R(x, y, z)$ is found as follows

$$R_i(x,y,z) = \begin{cases} Q_i^3(x,y,z), if\ Q_i(x,y,z) \geq 0 \\ 0, if\ Q_i(x,y,z) < 0 \end{cases}$$

Herein, $Q(x, y, z)$ is the perturbing quadric.

The obtained surfaces are smooth (see Fig. 1).

The figure 2 shows a scene, whose description required 4Kbyte information, which is 500 times less than the polygonal description that would take 2Mbyte information.

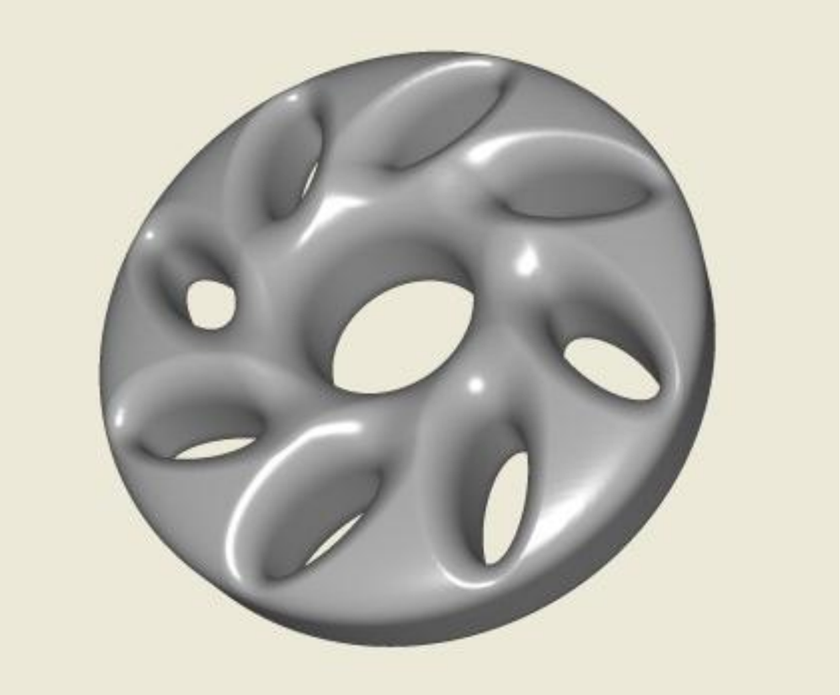

Figure-1. Function-based object with perturbation functions.

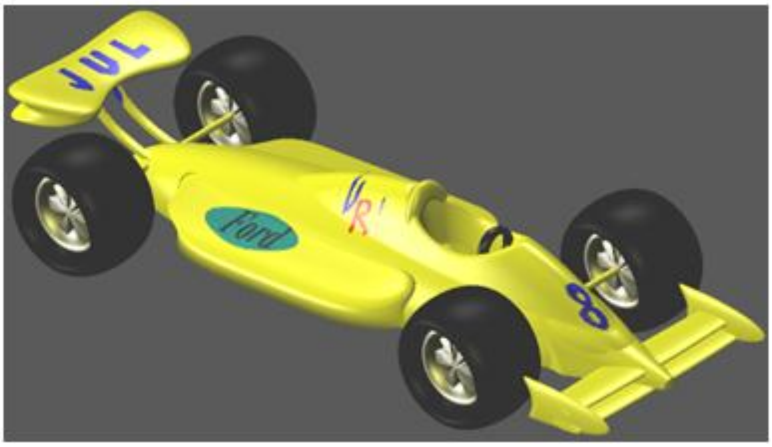

Figure-2. Complex function-based geometric object.

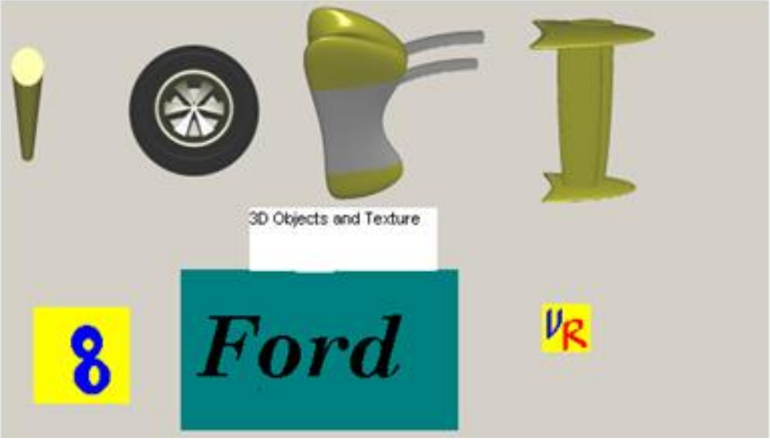

Figure-3. Simple geometric objects.

3. GEOMETRIC OPERATIONS

Two forms of elements of the set of objects are simple objects and complex objects (see Figs. 2 and 3). A complex object is a result of operations on simple objects [9].

The set of geometric operations Φ is:

$$\Phi_j: M^1+M^2+\ldots+M^n \rightarrow M,$$

where n is the number of operation operand.

Let the object G_1 be represented as $f_1(\boldsymbol{X}) \geq 0$. The unary operation ($n = 1$) of the object G_1 means operation G2=$\Phi_j(G_1)$ with the definition

$$f_2=\psi(f_1(\boldsymbol{X})) \geq 0,$$

where ψ is a continuous real function of one variable.

3.1. Projections

Let the source object $G_1 \subset E_n$ be defined by the function

$$f_1(x_1,x_2,\ldots,x_i,\ldots,x_n) \geq 0$$

and its projection $G_2 \subset E_{n-1}$ be defined by the function

$$f_2(x_1,x_2,\ldots,x_{i-1},x_{i+1},\ldots,x_n) \geq 0$$

The object G_2 can be described as a union of sections of the object G_1 by the hyper plane $x_i=C_j$, where $C_{j+1} = C_j+\Delta x_i$, $j=1, N$ and $C_1=x_{imin.}$ Let be the function for the section. As a result, the function for the projection at $\Delta x_i \rightarrow 0$ is a union of all functions f_1j:

$$f_2=f_{11} \vee f_{12} \vee \ldots \vee f_1j \vee \ldots \vee f_{1N}.$$

For the whole scene we fixed one coordinate, depending on what projection had to be obtained.

3.2. Offsetting

Let the source object be described by the function $f(X)>0$, the obtained object will be defined by the function $F=f(X)$ + C, where C<0 determines the negative offsetting (compression) and C>0 determines the positive offsetting. Adding together the positive or negative constant yields extension or compression of the object.

3.3. Set-Theoretic Operations

Let the 3D objects G_1 and G_2 be described as $f_1(\boldsymbol{X}) \geq 0$ and $f_2(\boldsymbol{X}) \geq 0$. The of the 3D objects G_1 and G_2 means operation $G_3=\Phi_j(G_1,G_2)$ with the definition

$$f_3=\psi(f_1(\boldsymbol{X}),f_2(\boldsymbol{X})) \geq 0,$$

where ψ is the continuous real function of two variables.

3.4. 3D Metamorphosis

Let F_1 and F_2 be values of the functions of the first and second 3D objects, respectively. Then the resulting perturbation function is calculated as follow:

$$F = \beta F_1 + (1 - \beta)\, F_2,$$

where β is the positive continuous function.

For 3D objects with the use of perturbation functions, one can perform 3D morphing of nongomeomorphic objects.

3.5. Twisting

Twisting is a 3D deformation being a particular case of bijective mapping which serves for defining deformations of source objects. For twisting of the source 3D object we found and transformed its coordinates *x, y, z*.

3.6. Global and Local Deformation

If we want to propagate the deformation it should be somehow added to all object that it affects. Actually the scene is organized so that it is no possibility to add object only by referencing i.e. without copying.

3.7. Sweeping

Swept volume as a projection of a moving object from the $4D(x,y,z,t)$ space to the $3D(x,y,z)$ space is considered. Then we draw the solid each time new coordinates that were changed by the proper law. In so doing, the previous images are preserved in the memory and used to obtain the result of swept volume. The newly formed figure is a union of images of the swept solid for different positions.

3.8. Relations

A binary relation is a subset of the set $M^2=MXM$. It can be described as

$$S_i : MXM \rightarrow I$$

The examples of binary relations are inclusion, point membership, interference or collision.

3.8.1. Collision Detection

We propose a method of collision detection without using any bounding volumes around each object and pre-processing stage [10]. The object collision is detected in a constant time, and the detection of events is absolutely ensured. Method is based on the relation of object intersections, and on the recursive object space subdivision for search the contact point of the objects.

4. INTERACTIVE GEOMETRIC MODELLING BASED ON THE PERTURBATION FUNCTIONS

Creating models one of the main problems of modern computer graphics. Commercial systems of geometric modelling require experienced users significant time, expertise and artistic talent to create 3-D forms. For interactive mode, many systems require triangulation model. For visualization used special fast interactive ray tracing and interactive polygonization algorithms. This leads to a loss of accuracy and complexity of the calculations. Method of surface deformation without pre-triangulation was proposed in Vyatkin [11]. Have been developed an algorithms and the set of C++ classes: adaptive ray casting for scenes containing functionally-based objects (including OpenGL color/depth buffers compatibility); C++ classes for functionally-based objects representation; C++ classes for rendering of functionally-based objects; C++ classes for proposed interface. The system can be expanded to include new algorithms and features.

These class hierarchies make up the core package which can be expanded to add functionality or to change some underlying mechanisms. Other tasks of the work including automatic (optimal) segmentation, manipulation of free forms, and seamless rendering (triangulation) are developed using these classes.

5. VISUALIZATION

The main point at this stage is the efficient finding of the ray intersection. This task is similar to visualization in volumetric tomography, where the density function is defined by discrete data. In our case, we use an analytically defined density function. The scene is in a unit three-dimensional cube. We omit the initial transformations and pay more attention to the main part of the method. The observer looks along the Z axis (Fig. 4).

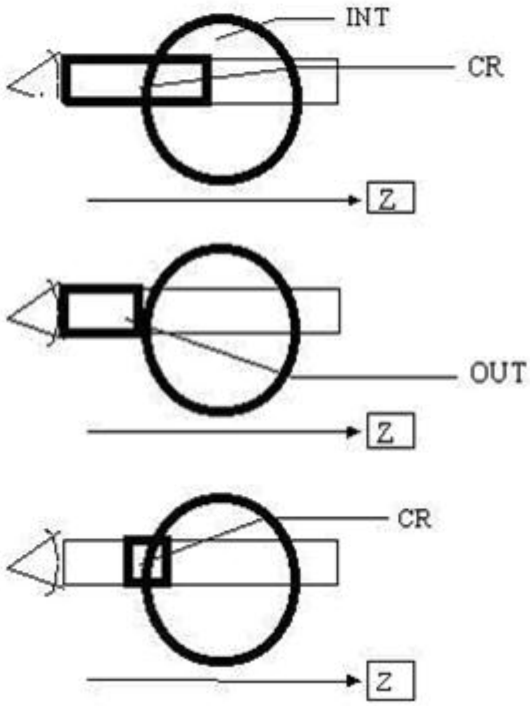

Figure-4. Ray casting

Rays pass through the image plane, and each of them corresponds to a pixel on the image. In the search for the points of ray intersection, each ray is divided along the Z axis to form a set of voxels. Thus, we obtain a density function along the ray which depends on one variable. The task is to find the point at which the function vanishes.

The visualization time is reduced by using the computational resources of a GPU with CUDA (NVIDIA). The result of running the programs on different processing units is the same even if they may have a different number of multiprocessors. Among the functions of the GPU was to calculate the coordinates of points of the surfaces, normals, and illumination. Geometric transformations were performed by the CPU. The DirectX was used for visualization. Testing was performed on Intel Core2 CPU E8400 3.0 GHz, GPU and 9800 GT 470 GTX processors [12].

6. POLYGONIZATION METHOD

Many geometric calculations are performed on polygonal models. Navigable nature of polygons (list of vertices,

edges, faces) supports non-graphical operations such as placing an object on a surface. Systems of rapid prototyping operate with polygonal models. Octree subdivision mesh extraction method based on recursion is proposed [13]. The model is subdivided into smaller cells. The process repeats, until the minimum cell size is reached. All the leaf cells containing the surface are polygonized. Unlike the predictor-corrector methods, this algorithm does not need an initial point to begin the polygonization with. It detects the surface automatically without holes. Proposed algorithm of surface points building works faster than step-by-step algorithm. Step-by-step calculations look every cell of space to define whether object is inside or outside the cell, but proposed algorithm can skip big amount of empty cells at once. During the surface triangulation, a mesh is calculated from a multitude of points.
For analyzing how close the function-based surface the polygonal mesh were used two error metrics.

7. CREATING SURFACES FROM POLYGONAL MESHES

In work [14] authors have applied their method to a variety of polygonal meshes. However, their method requires enormous computing expenses (running time of implicitization on a 195MHz R10k up to 24 hours) and keeping of big amount information - N^3.

*We developed necessary algorithms f*unctionally-based surface creating *that is most important*: automatic segmentation and connectivity – compactness/connectivity/smoothness, hierarchical triangle meshes, choice of basic primitives-quadrics, surface fitting (using extreme-stationary points and point/normal analysis), and generation of functions [15]. In this method we directly generated perturbation functions from the triangular mesh.

8. CONCLUSION

The main merits of our approach are the following: reduced database for describing curvilinear objects (representation of objects by free-form surfaces reduces 100 times and more the database description compared with their representation by polygons); efficiency of the masking rendering technique; reduction of the load on the geometry processor and decrease of data flow from it to the raster subsystem; simple animation and morphing of scenes. The proposed visualization algorithm along with the possibility to visualize arbitrary surfaces of free-forms and inhomogeneous volume spaces offers a wide scope of application. The free-form representation has a wide field of applications (graphics systems for visualizing functionally defined objects, CAD 3-D simulation systems, 3-D web visualization, prototyping, etc.). More effective tools for designing, manipulating, and deforming free-form 3-D shapes are needed in CAD, animation, and virtual reality applications.

Funding: This study received no specific financial support.
Competing Interests: The author declares that there are no conflicts of interests regarding the publication of this paper.

REFERENCES

[1] J. Bloomental, *Introduction to implicit surfaces/ (Eds.)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1997.

[2] A. Ricci, "A constructive geometry for computer graphics," *Computer Journal*, vol. 16, pp. 157-160, 1973. *View at Google Scholar* | *View at Publisher*

[3] W. P. Wang and K. K. Wang, "Geometric modeling for swept volume of moving solids," *IEEE Computer Graphics and Applications*, vol. 6, pp. 8–17, 1986. *View at Google Scholar* | *View at Publisher*

[4] C. M. Hoffman, " Implicit curves and surfaces in CAGD," *IEEE Computer Graphics Application*, vol. 13, pp. 79-88, 1993. *View at Google Scholar* | *View at Publisher*

[5] S. Sclaroff and A. Pentland, "Generalized implicit functions for computer graphics," *Computer Graphics*, vol. 25, pp. 247-250, 1991. *View at Google Scholar* | *View at Publisher*

[6] M. P. Gascuel, "An implicit formulation for precise contact modeling between flexible solids," in *Siggraph'93, Computer Graphics Proceedings*, 1993, pp. 313-320.

[7] B. Wyvill, C. McPheeters, and G. Wyvil, "Animating soft objects," *Visual Comput.*, vol. 2, pp. 235-242, 1986. *View at Google Scholar*

[8] J. F. Hughes, "Scheduled fourier volume morphing," *Computer Graphics*, vol. 26, pp. 43-46, 1992. *View at Google Scholar* | *View at Publisher*

[9] S. I. Vyatkin, "Complex surface modeling using perturbation functions," *Optoelectronics, Instrumentation and Data Processing*, vol. 43, pp. 226-231, 2007. *View at Google Scholar* | *View at Publisher*

[10] S. I. Vyatkin and B. S. Dolgovesov, "Collision detection of functionally defined objects for constant time," in *Proc. of 15-th International Conference on Computer Graphics GraphiCon '2005 (Novosibirsk, June 20-24, 2005)*, 2005, pp. 164-169.

[11] S. I. Vyatkin, "An interactive system for modeling, animating and rendering of functionally defined objects," *American Journal of Computer Science and Engineering Survey*, vol. 2, p. 102–108, 2014. *View at Google Scholar*

[12] S. I. Vyatkin, "Method of binary search for image elements of functionally defined objects using graphics processing units," *Optoelectronics, Instrumentation and Data Processing*, vol. 50, pp. 606–612, 2014. *View at Google Scholar* | *View at Publisher*

[13] S. I. Vyatkin, "Polygonization method for functionally defined objects," *International Journal of Automation, Control and Intelligent Systems*, vol. 1, p. 1–8, 2015. *View at Google Scholar*

[14] Y. Gary and T. Grey, "Robust creation of implicit surfaces from polygonal meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 346-359, 2002. *View at Google Scholar* | *View at Publisher*

[15] S. I. Vyatkin, "Converting polygonal models to function-based objects," *International Scientific-Technical Magazine Measuring and Computing Devices in Technological Processes*, vol. 1, p. 146–150, 2008.

Views and opinions expressed in this article are the views and opinions of the author(s), Review of Computer Engineering Research shall not be responsible or answerable for any loss, damage or liability etc. caused in relation to/arising out of the use of the content.