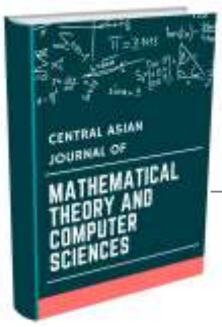


CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES

http://centralasianstudies.org/index.php/CAJMTCS

Volume: 02 Issue: 05 | May 2021 ISSN: 2660-5309

A brief introduction to Implicit surfaces

Abdullayev Alisher Sa'dullayevich[1]

Mamataliyev O'tkir Ashurovich[2]

Qurbanov Axmad Pardayevich[3]

[1,2,3]*TFTU TF named after I.Karimov*

Annotation.

We know that there will be some difficulty in calculating some surfaces. the reason for this is that the calculation of surfaces becomes a problem as a result of the uneven aggregation of the surfaces by some other surface below or above. in this article you will learn some similar, mathematical equations of surfaces that can occur in nature. this article briefly provides information on several surfaces. we will provide more interesting and complete information in the next issues of the magazine.

ARTICLEINFO

Article history:

Received 29 April 2021

Revised form 05 May 2021

Accepted 15 May 2021

Ключевые слова: zero, implicit function ,implicit primitives, smooth surfaces, conventional mathematical functions , boundary representation

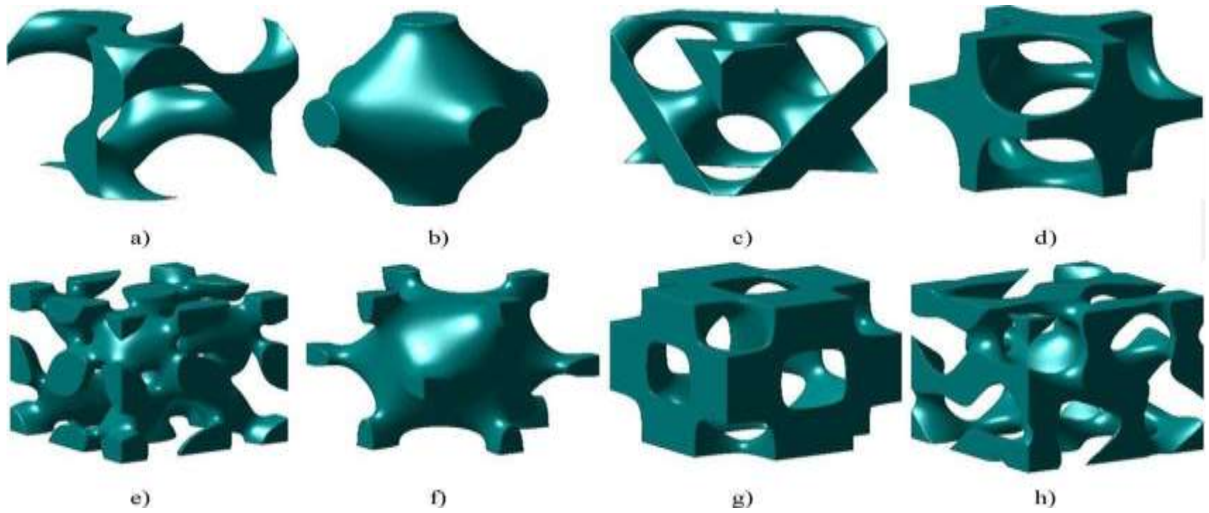
---***---

Do not worry about your difficulties in mathematics;

I can assure you that mine are still greater.

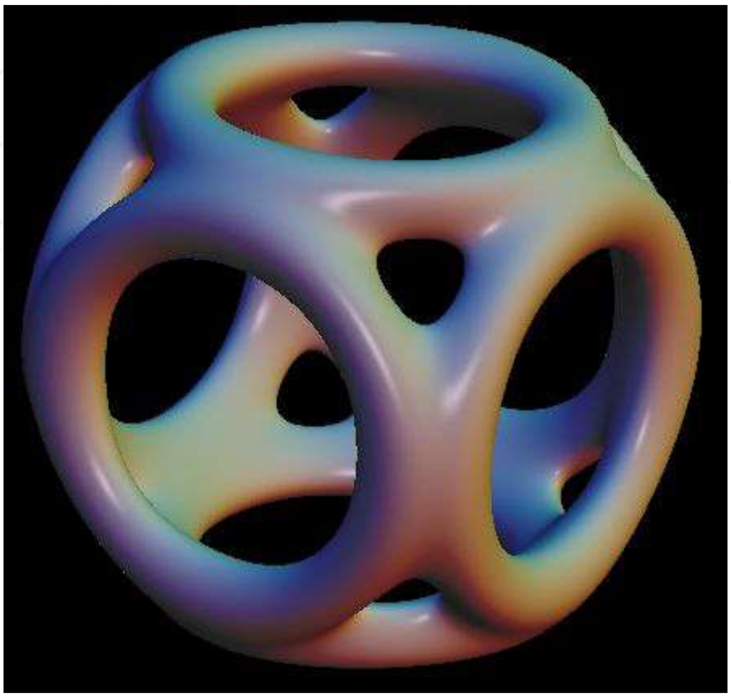
(Albert Einstein) An implicit surface is a set of points $\boldsymbol{p}$ such that $f(\boldsymbol{p}) = 0$, where f is a trivariate function (*i.e.*, $\boldsymbol{p} \in \boldsymbol{R}^3$). The surface is also known as the *zero set* of f and may be written f-*1*(0) or $Z(f)$. According to the implicit surface theorem, if zero is a regular value of f, then the zero set is a two-dimensional manifold. An *iso-surface* is a

similar set of points for which $f(\boldsymbol{p}) = c$, where c is the *iso-contour value* of the surface. The function f is sometimes called the *implicit function*, although we prefer *implicit surface function*.



In many cases, f partitions space into an inside and an outside. By convention, f is usually written such that $f(\boldsymbol{p}) < 0$ describes a volume of points enclosed by the surface, $f(\boldsymbol{p}) = 0$. This ability to enclose volumes and the ability to represent blends of volumes endow implicit surfaces with inherent advantages in geometric surface design. This is particularly so for skeletal design, as the relation between skeleton and surface is, generally, volumetric. Inherent in this relation is the use of a distance metric between the point $\boldsymbol{p}$ and the skeleton. We examine these metrics according to their blend properties and ease of implementation.

We suggested it is easier for the designer of natural forms to work with skeletal geometry than with the usually more complex geometry of the corresponding surface. The volumetric relation described by $f(\boldsymbol{p}) < 0$ underlies the correspondence between skeleton and surface. Thus, the combination of skeletal and implicit techniques is particularly appropriate for many natural forms. The smoothness sometimes associated with natural forms may be obtained as a blend of component volumes, which we call implicit *primitives*. The skillful combination of primitives is an important task for designers who wish to define implicitly an interesting or useful shape. Primitives may also be combined by set operations and functional composition (such as deformations), but blends are the most important combination concerning the design of smooth surfaces.



Traditionally, computer graphics has favored the parametric surface over the implicit because the parametric is easier to render and is more convenient for certain geometric operations, such as the computation of curvature or the control of position and tangency. Specifically, ‘‘parametric surfaces are generally easier [than implicit surfaces] to draw, tessellate, subdivide, and bound, or to perform any operation on that requires a knowledge of 'where' on the surface'' [Rockwood 1989]. Parametric and implicit surface representations are also distinguished by the compactness of their mathematical expression [Ricci 1973]. This seems particularly true for definitions that involve distance. For example, given a sphere centered at ***c***, with radius *r*, the parametric definition is:

(1,1) $(p_x, p_y, p_z) = (c_x + r\cos\theta\cos\tau, c_y + r\sin\theta, c_z + r\cos\theta\sin\tau)$,
$\theta \in (0,\pi),\ \tau \in (0,2\pi)$.

The implicit definition is considerably more compact:

(1.2)$(p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 - r^2 = 0$

Implicit surface definitions are very general; they can represent discrete pointsets, algebraic surfaces, and procedurally defined implicit surfaces. A discrete pointset can be represented by a function that returns 0 for ***p*** a member of the set, 1 otherwise. Generally, this is not useful because the function is discontinuous; pointsets can, however, be adapted for use in surface fitting [Hoppe *et al*. 1992]. Algebraic surfaces are commonly found in computer graphics, and include quadrics [Foley *et al*. 1990] and superquadrics [Barr 1981]. There has been a recent renewal of interest in general algebraic surfaces [Sederberg 1985], [Sederberg 1987], [Bajaj 1992]. They may be ray-traced [Blinn 1982], or, in the case of quadric surfaces, they are suitable for incremental scan-line techniques [Mathematical Applications Group 1968].

Although simple distance constraints can be expressed analytically, the defining function need not be analytic, but, as observed in [Ricci 1973], may be *procedural*. That is, a designer is free to specify any arbitrary process that, given a point in space, computes a real value. The procedure may employ conventional mathematical functions, conditionals, tables, and so on. A procedurally evaluated implicit surface function is not the same as a 'procedure model' [Newell 1975]. Both involve procedures, but the former is utilized to evaluate a point in space and the latter is utilized to construct a parametric surface. An example procedural implicit surface, discussed in section 5.6.14.1, performs several geometric operations to yield a value for *f* that would be difficult to express analytically.

In the following sections we consider several aspects of an implicit surface, including its relation to solid modeling, its application to skeletons, its visualization and polygonization, and its refinement by added surface detail. In 1973, a 'constructive geometry' was introduced for the purpose of defining complex shapes derived from operations (such as union, intersection, and blend) upon primitives [Ricci 1973]. The surface was defined as the boundary between the half-spaces *f*(***p***) < 1 and *f*(***p***) > 1; the former was considered the 'inside,' or solid portion, of an object. From this initial approach to *solid modeling* evolved *constructive solid geometry*, or *CSG*. With CSG, an object is evaluated 'bottom-up' according to a binary tree. The leaf nodes are usually restricted to low degreepolynomial primitives, such as spheres, cylinders, ellipsoids, and tori. The internal nodes represent Boolean set operations. The difference between solid modeling and implicit modeling is somewhat subtle. The surface of a solid model must enclose a finite volume. Consequently, the surface is everywhere equivalent to a two-dimensional disk and is, therefore, a two-dimensional *manifold* [Mäntylä 1988]. Implicit surfaces can enclose finite volumes as well; for example, $f(x, y, z) = x^2 + y^2 + z^2 - 1$ represents the unit sphere.But implicit surfaces can also represent unbounded surfaces; for example, $f(x, y, z)= z$ represents the *xy*-plane. Solid modeling is not limited to constructive models but may include, among others, *decomposition* models and *boundary* models [*ibid*.]. CSG is, however, the dominant form of solid modeling. The literature of constructive solid geometry emphasizes the robust representation of all intermediate results within the treestructured evaluation. Usually this intermediate representation is the *boundary representation*, or *BRep*. It is a versatile representation from which several geometric properties, such as volume and center of gravity, are readily computed.1 It is generally accepted in solid modeling that boundary representations must be closed under all Boolean operations. The requirement to maintain intermediate boundary representations places an

extraordinary demand on the process of CSG evaluation. These concerns are expressed in [*ibid*.]: Unfortunately, Boolean set operations algorithms for boundary representations are in general plagued by two kinds of problems: First, to be effective, a set operations algorithm must be able to treat all possible kinds of geometric intersections . . . [which] easily leads to a very [complex] case analysis. Second, the very case analysis must be based on various tests for

overlap, coplanarity, and intersection which are difficult to implementrobustly in the presence of numerical errors. This explains the preoccupation in CSG literature with the robustness of edgeedge

and edge-surface intersections. In comparison, concrete representations for implicit surfaces are formed without intermediate evaluations, greatly reducing the affects of numerical instability. Additionally, implicit surfaces need not be defined according to a binary tree or any other graph. Early development of geometric modeling, which embraces both surface and solid modeling, was motivated by engineering applications in the automotive, aerospace, aviation, and shipping industries and by training applications such as real-time, interactive flight simulators. This development involved an interplay of visualization and geometric modeling techniques that inextricably linked computer graphics and geometric modeling. For example, as graphics systems became faster and more flexible, designers were encouraged to develop ever more sophisticated models, many of which required new techniques in geometric modeling. Indeed, much of the development in surface and solid modeling is reported in the literature of computer graphics.

Skeletal Design

Having discussed the value of skeletally defined implicit surfaces, we now consider specific methods for their evaluation and definition. As described in section 2.4, the skeleton is a collection of elements, each of which generates a volume. Within an implicit context, we call such a volume a *skeletal primitive*, which we denote by

Pi(***p***), for skeletal element *i*. Thus, *P* is a function from R^3 (or R^2 for illustrative purposes) to R, and, usually, is C^1 continuous. The implicit surface function may be a blend of these primitives, *i.e*., *f*(***p***) = $g(\boldsymbol{p}, P_1 + P_2 + \cdots + P_n)$ = 0, and the implicit surface is the covering, or manifold, of the skeleton. Ideally, we wish our implementation of implicit surface algorithms to be indifferent to skeletal

complexity. This leads to two principal methods to evaluate the implicit surface:

(3) *fskeleton* = *frroot-c*, where *frlimb* = *max* (*flimb*, S*frchildren*) , or

(4) *fskeleton* = S*flimb-c*

In both (3) and (4), *flimb* refers to the implicit primitive defining the volume surrounding the particular skeletal element. In (3), *fr* is a recursive function equal to the implicit primitive of a limb or the sum of *fr* applied to each of the child limbs, whichever is greater. Contrary to solid modeling convention, we

assume *flimb* increases with decreasing distance to the limb; thus, *max*, rather than *min*, is appropriate. *fskeleton*(***p***) is the recursive function applied to the root limb of the skeleton; in (4), *fskeleton*(***p***) is simply the summation of all primitives. The recursive function yields a more smooth transition in limb radii at branch points. The computational load for (3) and (4) can be reduced by providing an axisaligned bounding box around each skeletal element. For ***p*** outside the bounding box of *limbi*, the influence of *limbi* is presumed non-existent; that is, *flimbi*(***p***) = 0. It is simple to test for ***p*** within an axis-aligned box. The interpolation of two implicit surfaces can be accomplished in several ways. The most accessible method is to interpolate the individual functions that define the surfaces. For example, in the figure below, we interpolate a torus, $(x^2 + y^2 + z^2 + {r_{major}}^2 - {r_{minor}}^2)^2$- $4{r_{major}}^2$ $(x^2 + y^2)$, and a sphere, $x^2 + y^2 + z^2 + {r_{major}}^2$.

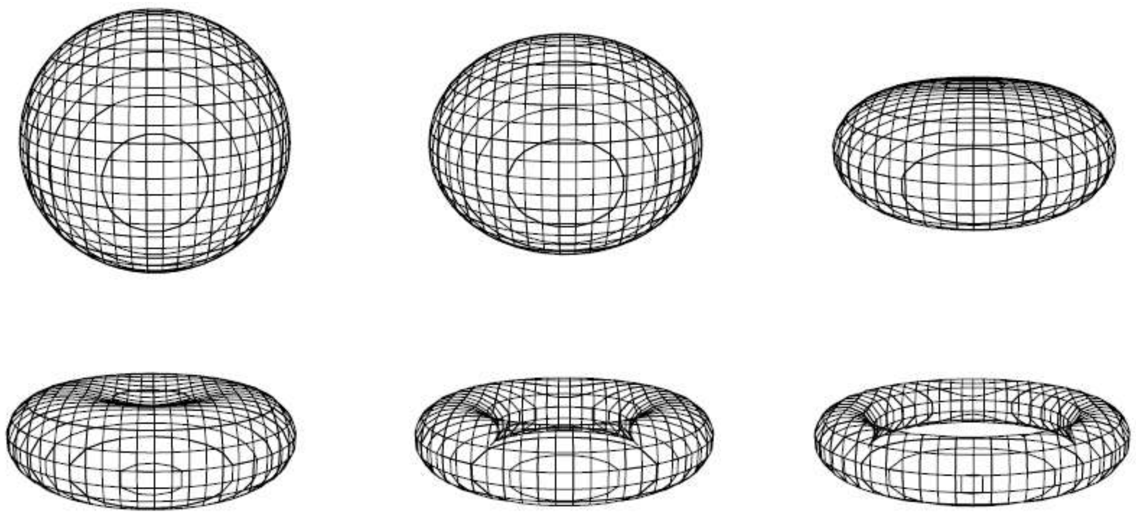


Figure 1. Interpolation of Sphere and Torus Functions

Interpolation of two functions, however, is not appropriate for skeletal forms because rigid body transformations can be lost. For example, in the following illustration, generated with a two-dimensional implicit contour follower, we employ three different interpolations.

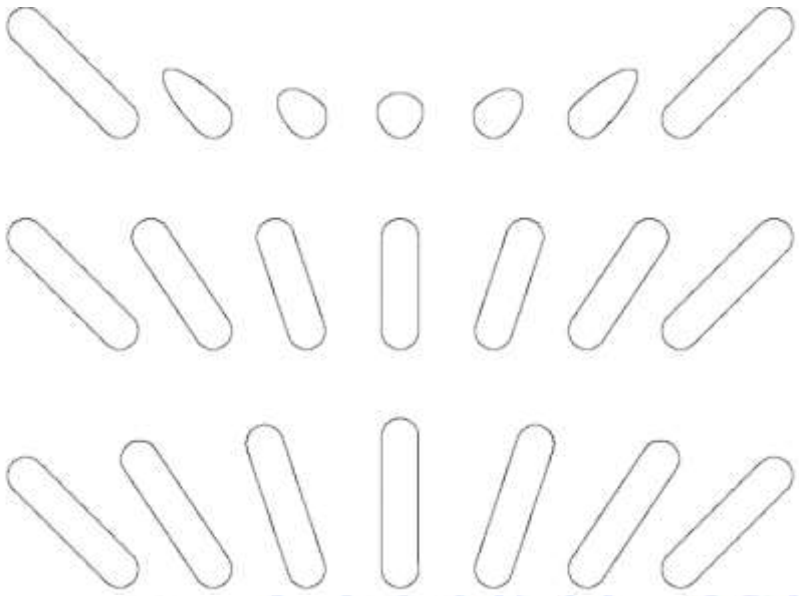


Figure 2. Implicit Interpolations

*top: interpolation of implicit contour functions*3 *middle: interpolation of segment endpoints bottom: interpolation of segment angle*

Visualization

Because an implicit formulation does not produce surface points by substitution, root-finding must be employed to visualize an implicit surface.4 This can be performed by ray-tracing, polygon scan conversion, or contour tracing. We briefly consider each of these methods. Implicit surfaces may be rendered directly by ray tracing, assuming a ray-surface intersection procedure is provided for a given surface. Often the intersection calculation can be accelerated by culling those pieces of the surface bounded by axis-aligned boxes not intersected by the ray. This process is known as *spatial subdivision* [Glassner 1984], [Samet 1990] of the implicit volume, and was applied to a procedural implicit surface [Bloomenthal 1989] to produce the image below. Other methods to accelerate the ray-tracing of implicit surfaces include symbolic algebra for surfaces defined by polynomials [Hanrahan 1983], the Lipschitz condition for surfaces with bounded gradient derivatives [Kalra and Barr 1989], and sphere-tracing for surfaces with bounded derivatives [Hart 1993]. In addition to shaded images, it is possible to create contour-line (or *section*-line) drawings of implicit surfaces by intersecting the surface with a series of planes, each perpendicular to the line of sight and receding from the viewpoint [Ricci 1973]. For each plane, the zero-set contour is drawn, excepting those parts obscured by previously drawn contours. In [Bloomenthal 1989] the implicit surface was spatially partitioned by an octree [Meagher 1982] to produce the following image. It is simple to compute the intersection of octree and plane; each intersected terminal node of the octree produces a section of the contour. Contour line drawings are particularly useful for engineering applications [Forrest 1979]. Nonetheless, many geometric shapes are best abstracted into Boolean set theoretic operations.

References

1.Blinn, J. F. 1982. A Generalization of Algebraic Surface Drawing. ACM Trans. Graph. 1, 3, 235--256.

2.Bouthors, A. and Nesme, M. 2007. Twinned meshes for dynamic triangulation of implicit surfaces. In Proceedings of the Graphics Interface Conference. 3--9.

3.Brochu, T. and Bridson, R. 2009. Robust topological operations for dynamic explicit surfaces. SIAM J. Sci. Comput. 31, 4, 2472—2493

4. P. Barla, J. Tollot, L. Markosyan X-toon: extended toon shader Non-Photoreal Animation and Show 4 International Symposium Proceedings (NPAR '06) (2006)

5. E. Vital Brazil, I. Macêdo, MC Sousa, LH de Figueiredo, L. Velho Drawing Vermation Hermite-RBF implications Sketch-based interfaces and modeling materials

6. X. Vendlend Minimal radius functions with minimal precision and compact support with piece-by-piece polynomial Development of Computational Mathematics, 4 (1) (1995)