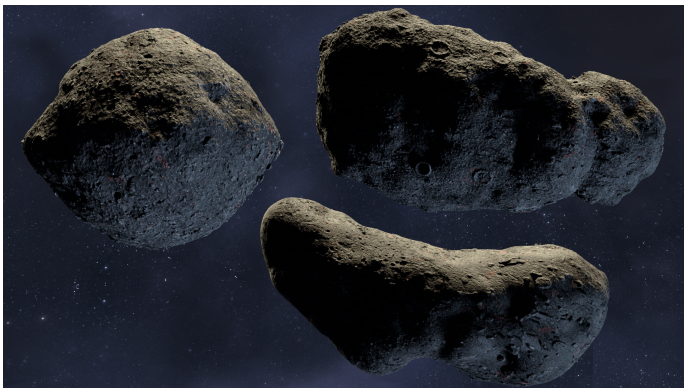
PROCEDURAL 3D ASTEROID MODEL SYNTHESIS



A general approach to automatically generate arbitrary 3D asteroid model

Dem Fachbereich Informatik
der Universität Bremen
eingereichte

Dissertation

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
von

Doktor-Ingenieur (Dr.-Ing.) Xizhi Li

Referenten der Arbeit:	Prof. Dr. Gabriel Zachmann
	Prof. Dr. Udo Frese
Tag der Einreichung:	20. July. 2020
Tag des Kolloqiums:	29. September. 2020

ABSTRACT

In computer graphics there exist mainly two ways to represent 3D shapes: implicit methods and explicit methods. The explicit method is quite obvious which uses triangle meshes to represent any shape precisely, while the distinct disadvantage is hard to compute interactions between triangle meshes which is very important in lots of graphic applications. By contrast, the implicit method is able to estimate the interaction easily but difficult to represent arbitrary shape (especially complex surfaces considered) accurately.

In the planning stage of space missions, there is an increasing demand for diverse surface details of small celestial body to be applied in virtual testbed based simulation systems. Implicit surface is one of the most promising solutions to this problem. They are powerful both for the modeling of 3D asteroid models and animating movement of rovers on the virtual testbed. The construction of 3D models comes from basic geometric primitives (i. e., sphere, cone, ...) and incrementally sums up their corresponding scalar fields into more complex shapes which represent shapes easily and compactly. Moreover, this compact representation makes it convenient to compute arbitrary patches of virtual testbed on demand, and meanwhile enables those patches contain a dynamically changing topology.

However, one conspicuous weakness in implicit modeling is relying on the manual trial and error method to obtain corresponding parameters of implicit functions, and this work is usually tedious and inefficient. In addition, the implicit modeling system can only generate smooth surface, nevertheless, in most practical applications the surface details are the dominant elements. For instance, in space the terrain-based navigation system and optic-based ground guiding system rely on the terrain features of the celestial surface. Therefore, adding realistic surface details on the implicit surface is another key challenge in the generation of celestial (i. e., asteroid) 3D models. What's more, as the fast iteration of graphics hardware, the demand for high-quality 3D objects in nearly all graphic applications (i. e., AAA games, movies) grows exponentially. The traditional way to create 3D models by artists becomes not only more expensive but also hardly satisfy enormous requirements. Even hiring enough artists to help building the scene, the expense is not sustainable for the 3D industry.

In this thesis we propose new methods to automatically generate an implicit representation of 3D asteroid models, inspired not only by sphere packing but also from noise models. They enables:

- a novel invariant shape descriptor to be evaluated on GPU side with CUDA; the statistical histogram of the shape descriptor is used to represent the highly detailed 3D asteroid model,

- an automatic method (AstroGen) to approximate the given constraint shape with sphere packing based metaballs,

- an optimization method which use the distance between different asteroids' histogram as target function and *particle swarm optimization* (PSO) algorithm to optimize the parameters of each asteroid's implicit representation (makes the implicit modeling into a machine learning task),

- a new procedural noise model to generate the surface details on the implicit surface, the details behave in a coherent way with the underlying surface.

Ever since the arise of general GPU, the computation speed of computers has increased notably faster than its memory bandwidth. The direct consequence of this trend is that compute-intensive algorithms (especially parallelizable algorithms) become increasingly attractive. This is the main reason to explain the recent popularity of procedural methods. We believe that the latest tendency in hardware (i. e., GPU, Cloud Computing) justify the necessity to take a reconsideration of procedural methods. Our procedural algorithm fits this trends quite well and has great potential in nearly all areas of computer graphics.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Prof.Dr.Gabriel Zachmann. He was inspiring, enthusiastic about expanding the boundary of computer graphic and virtual reality research in our group, and allowed me fantastic research environment as well. Without him, this dissertation would not possible exist.

I also would like to express my gratitude to Prof.Dr.Udo Frese for accepting the co-advisorship.

I am grateful to Dr.René.Weller, my co-supervisor. He shared the burden of my research, guided me from the very beginning paper submission to reviewing this dissertation and exhibited excellent insights and leadership throughout my study.

Nothing in this dissertation would have been carried out without my co-authors and collegues. Thanks to all of you, it was a great pleasure to work with you all: Dr.Patrick Draheim, Abhishek Srinivas, Jörn Teuber, Christoph Schröder, Philipp Dittmann, Maximilian Kaluschke, Janis Roßkamp, Toni Tan, Roland Fischer, Hermann Meißenhelter, Andre Mühlenbrock. Many thanks also to Helga Reinermann, Sabine Dolhs, Tanja Rethemeyer and other technicians and trainees for your precious supports and helps in the device, laboratory and daily life. I thank all the students and staffs at CGVR. All of you make the life in this group colorful and unforgettable. I would like to thank the China Scholarship Council (CSC) for funding my PhD project. I have learned and benefited so much from all of you both professionally and personally.

I would like to dedicate my dissertation to my parents for their unreserved love and support. They gave me the confidence to change my career entirely, to move to a new country, and to start everything on my own. I would like to express my deepest love to my wife and daughter Ida, who gave me lots of sweet troubles but firmly my belief to overcome the difficulties. I am infinitely grateful to both of you.

CONTENTS

Part I

THAT WAS THEN, THIS IS NOW

1

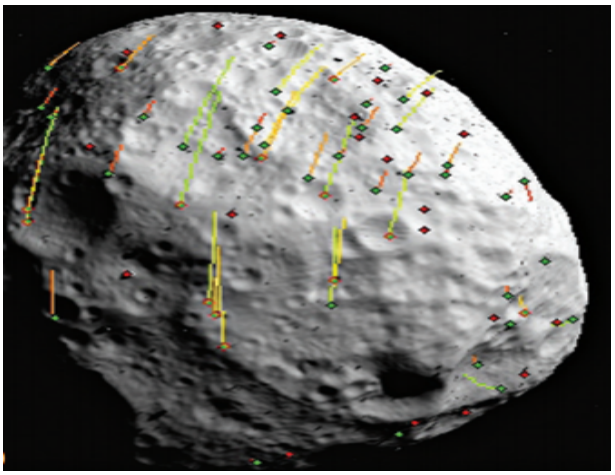
INTRODUCTION

The choice of 3D representation of real world objects plays a crucial role in various fields and applications. The classical geometric mesh representation is the most popular method and adopted by the majority of commercial applications. They are more accurate since they are able to capture the inherent geometric structures defined by vertices' connectivity. Meanwhile, a geometric mesh is also able to represent the non-manifold surfaces that necessary for many applications, such as character's hair or cloth simulation. Point clouds, however, lack numerical stability on one hand, and absent inherent 3D spatial structure on the other - which is essential for efficient 3D object deformation and animation. Another common representation of "organic" shapes uses a combination of continuous potential fields instead of the discrete connected points, known as implicit surface.

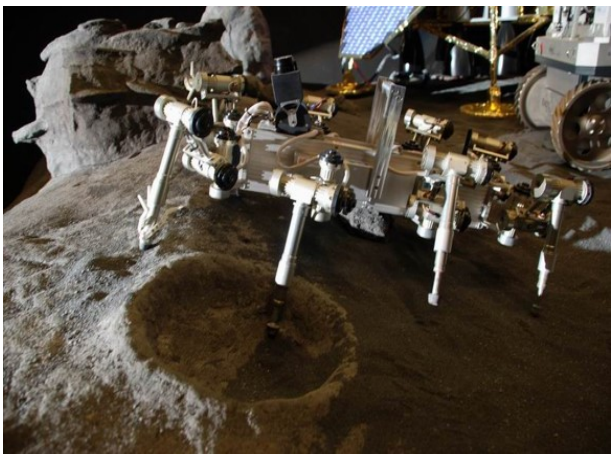
However, recent development in 3D production pipeline requires large numbers of high quality meshes at low cost. Obviously, traditional explicit mesh representation needs tremendous efforts to achieve high quality for individual pieces of meshes and unable to satisfy the 3D industry. Moreover, the intrinsic low efficiency of explicit meshes makes it difficult to evaluate interactions between meshes. However, in the physically-based [66] simulation systems and many animation systems, the collision detections algorithm plays a crucial part. The trend of high quality 3D meshes resulting further worse situations where more vertices must be considered. Consequently, the explicit representation could be a limiting factor for the future of 3D industry while the implicit representation would be more beneficial.

Our work begins from the requirement of real space mission. Before we start the mission, usually we will build a virtual environment to validate our design. For instance, we need a virtual testbed with surface details to help testing the terrain-based navigation system (see Fig. 1.1a); and also the optic-based tracking and landing system (see Fig. 1.1b). Moreover, we can print out the virual testbed and create a physical mock-up to test our devices in physical world.

Generally, our knowledge about space origins from the earth based observation system such as lightcurve inversion or bi-static radar. Since then, we have accumulated lots of low-poly real shapes of small celestial bodies but without surface details. In recent decades, several space missions (i. e., "Rosetta" mission, "Dawn" mission) have sent back a bunch of images about the real asteroid's surface. In Figure 1.2 we can notice the uneven distribution of craters, and also the distribution of other terrain primitives (some places with dense rocks, and others are flat) follows the rule of spatial heterogeneity. Additionally, high-resolution models are important for space simulation especially when the spacecraft moving towards the asteroid and we need different resolution of asteroid models to generate a level-of-details image



(a)



(b)

Figure 1.1: (a) Terrain based asteroid navigation system [92]. (b) Virtual testbed for rover validation [DFKI GmbH].

for the recognition algorithm. Modeling 3D asteroid with implicit modeling system is an attractive alternative to explicit method which usually involves lots of attempts to approximate a particular shape: this paradigm can hardly adapt the future trend. We need a new method which enables us to easily model arbitrary geometry with diverse topologies.

Introduced in computer graphics in the 70's, implicit surfaces represent primitive shapes with compact field functions $f : R^3 \rightarrow R$ and blending (i. e., summation) those scalar fields into more complex shapes. This property making it powerful both for the modeling of 3D objects and their corresponding animation. Accordingly, implicit surfaces are well known for their ability to ease collision detection and simulate contact between solid volumetric bodies. A first insight of implicit surface is that collisions can be evaluated in linear time $O(n)$: Let f_1, f_2 be two field functions and S_1, S_2 represent the associated 0-isosurfaces, the inner part of the solid volume being defined by $V_{1,2} = \{p \in R^3 \mid f_{1,2}(p) < 0)\}$. After that, for any given point p on S_1, estimating the sign of $f_2(p)$ provides a simple collision test with the solid volume, simultaneously the absolute value of $f_2(p)$ gives

Figure 1.2: The real image of the asteroid Itokawa. The image comes from the "Rosetta" mission.

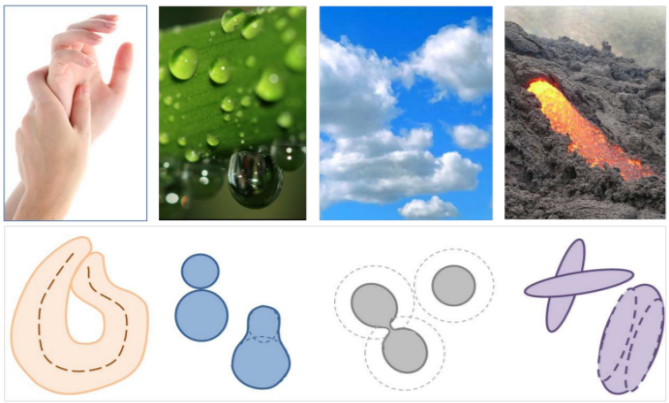
further information about the penetration depth. Another insight of implicit surfaces is their ability to compose new shapes, using a variety of combination operators between potential scalar fields. The major bottleneck of implicit surface is their low accuracy in representation arbitrary 3D objects and the blending operator between implicit primitive is usually limited to model "soft" objects without crisp boundaries. Moreover, the standard blend operator like summation makes the quantity of blending among primitives difficult to predict, i. e., whether topology transformations will occur or not.

Formable implicit surfaces are dramatically because they can seamlessly compose shape with complex and dynamic topologies. Implicit surfaces, more specifically convolution surfaces, are therefore particularly well adapted to combine with sphere-packing algorithm [159] to incrementally construct complex shapes. However, sphere-packing based modeling system present a number of drawbacks which make them difficult to use in practice. More specifically, different geometric primitives (spheres in our case) tend to act differently when their distance varies. Merging their potential field when they get close and separating when they move separate, thus, becomes hardly to predict. For instance, implicit drops of water [66] tend to deform and merge before they collide, the arm of an implicit character [19] may merge with its body if they come too close. With no control around the global topology, implicit blending is unable to acquire expected combination effects.

In the real world, we can observe mainly four types of blending: skeletal-blend, contact-blend, distance-blend and context-dependent blend [168] (see Fig. 1.3b). Blending is generally expressed as a binary operation, except for the n-ary operators [168]; *max*, which result in the effect of two operators *union* and *sum*, and able to deliver smooth blending effects between the input shape. The binary operators [8] prevent blending effects at a distance, and provide more precise control to insure the model always maintain the union topology effects between the input surface. Improving blending operators has always been a major topic of implicit surface research [56], targeting at solving

the previous four major problems in constructive implicit modeling: suppressing bulges when two shapes merge, unwanted blending at a distance, the resulting shape should keeps the topology of the union, and the sharp details to be added without being blown up [56]. The key remark to tackle those problems is to utilize more information - not only their scalar values but neighborhood information as well (i. e., gradient).

(a)



(b)

Figure 1.3: (a) Animations can be launched through a set of implcit surface [19]. (b) Taxonomy of blending behaviors, with real example on top and schematic illustration at the bottom. From left to right: Skeletal-blend, contact-blend, distance-blend and context-dependent-blend [56].

How can an implicit function accurately and compactly represent arbitrary complex shape as well as their inner structures? This thesis attempt to integrate previous research on implicit modeling and introduce a novel automatic implicit solution for 3D asteroid modeling. Our shape representation is based on a skeleton metaballs, origin from the sphere packing [159] algorithm. And a blending operator controls the topology changes between the connected metaballs. The final inflated geometry is controlled by radius, position, and smoothing parameters assigned to each metaball. These shape and topology parameters become variables in the optimization process and can be applied constraints to automatically generate desired result. Then, an

overall similarity measurement is applied as the target function to help iterate the solution space. To reduce our search space and boost the rate of convergence, we use *partical swarm optimization* (PSO) [38] with a histogram-based similarity algorithm to "learn" the parameter space to generate an implicit model from the given mesh. In order to solve this optimization problem we need: 1. suitable constraints which minimize the solution space; 2. an efficient data structure to accelerate the evaluation of the implicit model; 3. an accurate overall similarity measurement to distinguish the generate model with the given mesh.

Our work differs from traditional implicit modeling approaches in several important ways: 1. we have turned the tedious 3D modeling process into an automatic way by minimizing a "fitness" function; 2. we use the PSO to facilitate the neighbor information of inner metaballs to speed up the optimization process; and 3. we proposed a new noise model to synthesis the surface details on the implicit smooth surface. Our research makes the implicit method as accurate as the explicit method by facilitating PSO and available for a series of graphic applications.

1.1 CONTRIBUTION

Our work begins from the requirement of real space mission. Before we start the mission, usually we will built a virtual environment to validate our design. For instance, we need a virtual testbed with surface details to help testing the terrain-based navigation system; and also the optic-based tracking and landing system. We need an efficient algorithm to generate diverse surface details on the irregular shape, and the manipulation must be intuitive.

We present a fully implicit representation of 3D asteroid model with a compact formula. By computing the formula for each point in 3D space we can get the isovalue and then assembled into the implicit surface of the asteroid. Our pipeline (see Fig. 1.4) begins from the constraint shape and in the first step we can use a group of metaballs to represent the smooth global shape of asteroid. Then in the second step we use a noise model to overlap the volumetric terrain on the global smooth shape. Our noise model can generate micro and macro structures on the surface. Especially boulders and craters.

In Chapter 3, we introduce a novel statistically invariant shape descriptor for large-scale high resolution 3D model with local dissimilarities. We combine four initial features that describe the global shape with two novel features, representing the local curvature and the normal perturbation, respectively. Moreover, our shape descriptor is robust to noise and invariant to translation, rotation and scale of the object. The shape descriptor can be evaluated in parallel and therefore we are able to deal with massive data from high-resolution 3D shapes effectively.

In Chapter 4, we contribute the first automatic implicit modeling system - AstroGen - for small celestial bodies, that yields an approximation of any irregular constraint shape of arbitrary resolution. Our

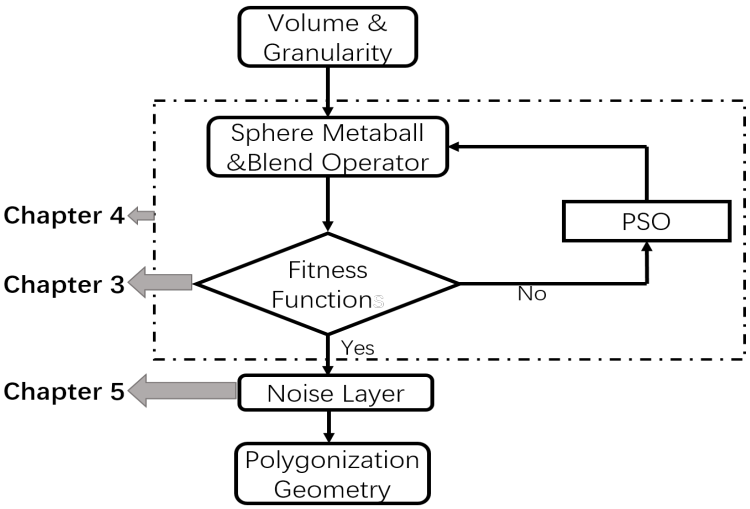


Figure 1.4: The workflow of our method in this thesis.

method begins from a constraint surface (low-poly mesh), then we use a sphere packing algorithm to fill the low-poly mesh with spheres, then we can compute the potential field inside the asteroid and finally a partical swarm optimization algorithm is applied to help us get the optimized smooth global shape.

In Chapter 5, we present a new noise model to generate macro and micro structures on the previous smooth global shape. In order to generate macro structures, we use locally controlled spot noise which can transfer the kernel shape into the textures and is suitable for generating macro structures. For the micro structure, we use the gabor noise by example algorithms.

In short, we have improved the traditional 2D procedural texture into 3D implicit terrain synthesis and achieved good results. We also proposed a new noise model to generate macro and micro terrain structures. Our workflow makes it possible to represent complex 3D asteroid implicitly and each point can be evaluated on demand in parallel. Non-professionals able to manipulate a few intuitive parameters to generate diverse realistic 3D asteroid models.

2

IMPLICIT SURFACE : STATE OF THE ART

The research of using equations and rules to define geometric objects in arbitrary dimensional space is an active area of mathematics and computer science in the past decades. In compute graphics, this research begins from implicit surface modeling and evolve into rendering and animation. In the 3D case, implicit surfaces S_i are usually defined as a set of discrete points $p \in R^3$ in the Euclidean Space where their scalar fields $f_i : R^3 \to R$ equals to the given iso-value $c_i \in R$:

$$S_i = \{P \in R^3 \mid f_i(p) = c_i\} \tag{2.1}$$

Follow the similar definition method, the implicit solid (or volume) v_i can be defined as :

$$v_i = \{P \in R^3 \mid f_i(p) \geq c_i\} \tag{2.2}$$

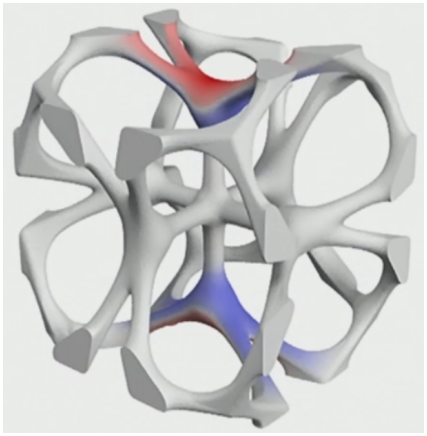
This compact definition empower implicit surfaces two major advantages over explicit surface:

1. The compact definition of the implicit surface makes it well suited to ease the requirement of accurate collision detection algorithms in various applications (i. e., simulate the solid body's contact or visualization). For instance, from the Equation 2.1 and 2.2 we can define two implicit surfaces S_1, S_2. Let point p on the surface S_1, directly evaluating the sign (compare with the iso-value c) of $f_2(p)$ provides a simple collision test with solid v_2, meanwhile the absolute value of $|f_2(p)|$ afford the further information about the penetration depth.

2. Another ideal property is their capacity to be seamlessly combined in a geometrically correct and concise ways to generate new complex implicit shapes, at the same time consume a relatively small memory footprint. More specifically, the composition is consist by a given operator $g : R^n \to R$ which yields new implicit surface S_j at the c-isosurface of the resulting field $f_j = g(f_0, ..., f_n)$. Each primitives f_i are summed up by simply exerting the operator g to their respective scalar field, regardless of their relative positions.

Since then, implicit surface has became a powerful 3D object representation method in computer graphics, i. e. modeling and animating objects with arbitrary topology. In short, complicate implicit surface with changing topologies are handled by a few simple combining algorithms, however, this representation must guarantee a continuous, hole-free manifold. In general, we use at least C^1-continuous (or even C^2-continuous where smooth curvature are needed) scalar field, leading to a continuously varying normal vector field which is a crucial

(a) Implicit representation of garments and character, yielding plausible animations [19].



(b) This skeleton-based implicit surface was evolved by a stress concentration minimization method to generate microstructures [105].

Figure 2.1: Examples of recent development of implicit surface for modeling, animation and 3D printing.

factor for shading the implicit surface. The unit normal vector $N(p)$ for the implicit surface is computed as the normalized gradient of the field function at that point (∇ means the gradient of f):

$$N(p) = \frac{\nabla f(p)}{\|\nabla f(p)\|} \tag{2.3}$$

There also exist some other terminologies to define the implicit surface, i. e. potential fields in the case of skeleton-based implicit surfaces or implicit function, however, they are not mathematically correct since the function is explicitly defined [8]. In this thesis we use field function f to express the fact that the implicit surface is an iso-surface of the scalar field they defined. In order to make field function f more intuitive to manipulate as a modeling tool, lots of different implicit surfaces have been proposed, i. e. level-sets, functionally-based representation (such as f-reps [109], soft objects [96], convolution surfaces [134], and Radial Basis Function [129]), grid-based representations (voxel), skeleton-based implicit surfaces.

In the remainder of this chapter, we will first review the development of mainstream models used in implicit surface modeling (see Sec. 2.1). Then we will present another major research area in implicit modeling: how to improve the level of control when combining different parts of implicit surfaces (see Sec. 2.2). In the third part (see Sec. 2.3), we will present several methods to add details on the implicit surface. In Section 2.4, we will highlight two ways to visualize the implicit surfaces (i. e., polygonization and ray-tracing). Lastly, we will present a series of applications of implicit modeling, as shown in Section 2.5.

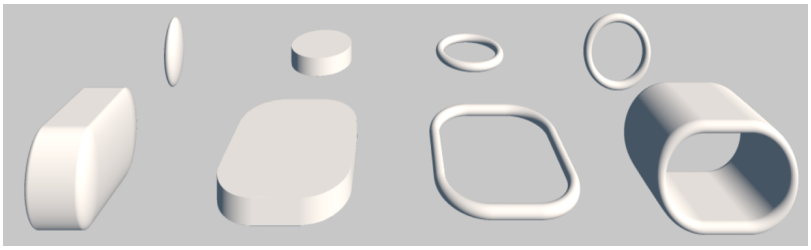


Figure 2.2: Some algebraic basic primitives, which can be combined into more complex shapes using some blending operators [35].

2.1 MODELING IMPLICIT SURFACES

Implicit modeling plays an unalternative role in computer graphics due to their intuitive and elegant representation of complex shape in a mathematical way, rather than explicit representation. Algebraic surfaces are the most basic implicit surfaces which precisely defines the distance field of the primitive geometries (see Fig. 2.2). They rely on a equation defined on the Euclidean coordinates that generally produce a continuous surface. However, these formulas are not practical for artists who design the 3D object, and consequently more intuitive modeling tools are needed. Indeed, it is unpractical to design an elaborate equation to represent arbitrary complex shape but those simple algebraic surfaces are still powerful when they are assembled together. In the following part we will introduce the development of implicit modeling based on the algebraic surface.

2.1.1 *Skeleton-based Implicit Surfaces*

Algebraic surface abstracts the basic geometry primitives (i. e., line, sphere, box) quite well but difficult to predict the result when altered the parameter of the equation. As a result, many complex and deformable objects are either arduous or inefficient to be reconstructed with such unpredictable building blocks. An increasingly popular approach is to use the skeleton-based field function to model the 3D object. Evaluating the isosurfaces along the predefined skeleton field we can achieve a compact and intuitive representation of the complex shape. Field function f is defined on the distance to the skeleton which gives us more control by deform the skeleton or the distance attenuation, meanwhile skeleton surface is a meaningful abstraction of the overall shape of the target object. Generally, skeleton-based field functions can be defined on various skeletons, such as a set of discrete points (as in the spherical distance, see Fig. 2.3), line-segments, curves, triangles or even cylinders, cones, torus, cubes (see Fig. 2.4), as soon as the query point's minimum Euclidian distance to the skeleton can be computed. A naive evaluating the distance $dist$ to the skeleton S to construct the skeleton-based distance field function f brings about the following formula: $f(p) = R - dist(p, S)$, where R is the distance from the geometric skeleton S to the desired surface, and $f(p)$ is also

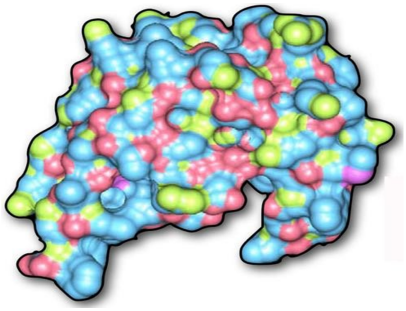


Figure 2.3: The definition of the electron density map underlying the blobby model [110].

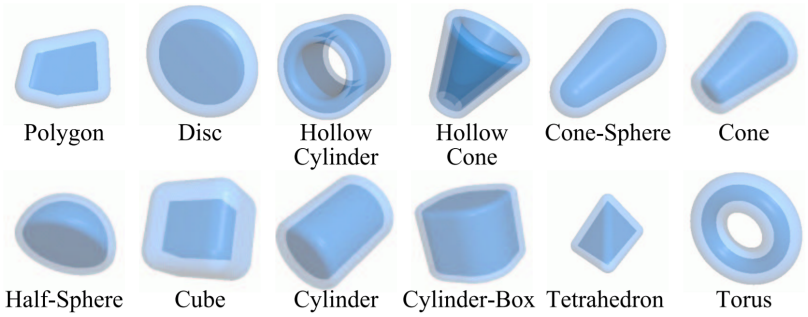


Figure 2.4: Some skeletal primitives implemented by skeleton-based implicit modeling system [4].

called distance surface. This definition actually prescribe the implicit surface twine the skeleton, as illustrated in Figure 2.4.

In order to generate sophisticate structures of the complex skeletons we need more compact support and more flexible local restrictions to the field generated by the skeleton distance surface $f(p)$. Blinn [11] introduced the first kernel-based implicit surface, then the skeleton distance-based field functions are extended by modulate a kernel function K as:

$$f(p) = K(R - dist(p, S)) \tag{2.4}$$

The first kernel-based definition of an skeleton-based implicit surface was introduced to visualize the "blobby" molecules (see Fig. 2.3). Inspired by the electromagnetic properties of atoms, Blinn proposed the *Gaussian* kernel to further adjust the scalar field. The resulting field function $f(p)$ (see Eq. 2.4) is a kind of globally supported field function: field value varies all over the space but is positive inside and outside negative. Subsequently, several alternative kernels have been proposed in the form of piecewise polynomial functions, which are faster to evaluate, in order to define more flexible locally finite support field functions.

Here we list the formula of five mainstream kernel functions which satisfies all requirements of the implicit modeling: smooth, monotonic and bounded (with a represent the scaling factor and b controls the deviation of the primitive):

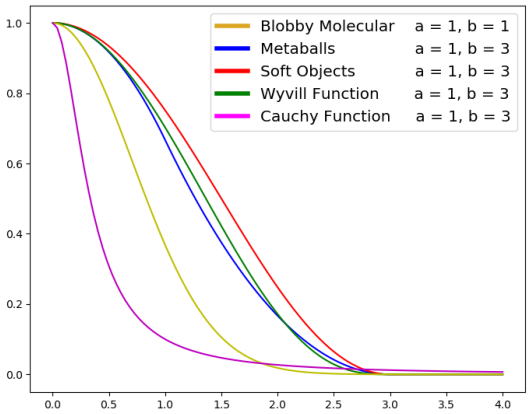


Figure 2.5: The compare of five distance-based field functions.

1. Blobby Molecules (Gaussian Kernel) [11]:

$$K(d) = a \cdot \exp(-(\frac{d}{b})^2) \tag{2.5}$$

2. Metaballs [102]:

$$K(d) = \begin{cases} a(1-3(\frac{d}{b})^2) & if\ 0 \leq d \leq \frac{b}{3}, \\ \frac{3a}{2}(1-\frac{d}{b})^2 & if\ \frac{b}{3} \leq d \leq b, \\ 0 & otherwise \end{cases} \tag{2.6}$$

3. Soft Objects [96]:

$$K(d) = \begin{cases} a(1-\frac{4}{9}(\frac{d}{b})^6 + \frac{17}{9}(\frac{d}{b})^4 - \frac{22}{9}(\frac{d}{b})^2) & if\ d \leq b, \\ 0 & otherwise \end{cases} \tag{2.7}$$

4. Wyvill function [13]:

$$K(d) = \begin{cases} a(1-(\frac{d}{b})^2)^3 & if\ d \leq b, \\ 0 & otherwise \end{cases} \tag{2.8}$$

5. Cauchy function [94]:

$$K(d) = \begin{cases} \frac{a}{1+b^2d^2} & if\ d > 0, \\ 0 & otherwise \end{cases} \tag{2.9}$$

Andrei [94] compare all kernel functions we mentioned above and prove the polynomial function works best in the computation criteria. What's more, each kernel has its own unique "signature" curve which shows noticeable various attenuation rate (see Fig. 2.5).

In brief, there exists two types of method to generate a variety of skeleton-based implicit surfaces: the first type convolve an anisotropic metric around the skeletons (i. e., apply a mixture of kernel functions to compute the distance field along the skeleton); in the second type, the distance metrics which evaluated on the Euclidian space can be replaced by quadric metric or some other metrics in order to bring more variants. The C^1-continuous property of skeleton-based surface made it well suited to construct organic or liquid looking shapes and animate object's topology change over time (i. e., the deformation of human body's hand or leg). However, the advantage of continuity also limit the possibility of skeleton surface to precisely model some sharp shapes (i. e., hair, sharp rocks), and the total computation is too expensive for real-time displaying applications. These drawbacks are reviewed in more details in Sections 2.2 and 2.4.

Another major drawback of kernel-based distance field function is their low degree of continuity (i. e., metaballs, soft objects) compared with point-based distance field function (see Eq. 2.5). Indeed, the continuity of point skeleton field function is C^∞ everywhere inside its support domain. As a contrary, in more complex skeletons this is not the case anymore: the field function is spherical at points proximity, cylindrical at segments proximity and planar at faces proximity. Hence, the overall gradient of the complex skeleton field is continuous but not differentiable, and then the resulting implicit surface is only C^1 continuous.

2.1.2 *Convolution Surfaces*

A logic spread of the skeleton-based implicit surface modeling aims at adding the domain's degree of continuity (avoid unwanted bulges) and fairness regardless of the skeleton type (i. e., point, segment, planar) by replacing the convolution of a kernel function K with a geometry function G. More specifically, different from modulating the distance $dist(p, S)$ with the kernel K, convolution surface's kernel function convolved directly on the predefined skeleton. Thus the field function $f(p)$ defined as the integral (infinite summation) of the contributions of all points p on the skeleton:

$$f(p) = G(p) \otimes K(p) = \int_{R^3} G(r)K(p-r)dr \tag{2.10}$$

where $\otimes$ means convolution, $G(r)$ describes the shape of the skeleton and $K(p)$ is the kernel function to produce the potential value corresponding to each skeleton points. If the skeleton is 0-dimension point-skeleton, in substitute, the formula (see Eq. 2.10) becomes the classical "blob" (see Eq. 2.5). Ascribe to the cumulative property of integration, convolution-based implicit surfaces are isolated of the skeleton segmentation as illustrated in Figure 2.6. We can note that the

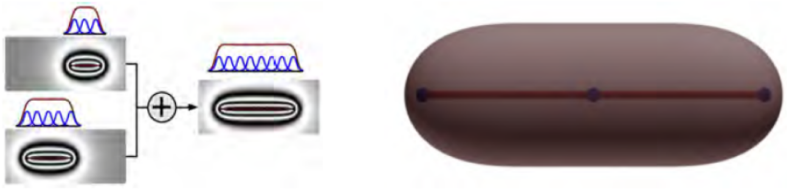


Figure 2.6: Summation of individual part of the skeleton give the same scalar field as integrate the skeleton as a whole [165].

infinite summation along the skeleton make the convolution surface bulge-free over the skeleton-based implicit surface, and the benefit comes from the integration on the null intersection parts. This property then helps the design of arbitrarily complex skeletons defined on a set of primitive geometries, such as points, line-segments, curves, or warping of these primitives [134, 167, 169]. Due to the integral operator (see Eq. 2.10), the additive of different partitions of the skeleton are seamlessly consisted and leave the skeleton field continuous and smooth (see Fig. 2.6). Thus the convolution surfaces becomes an ideal modeling tool to define levels of details (LOD) through the recursive subdivision of their skeleton implicit surface. The property of independent from subdivision enables us to construct diverse networks with curves and surfaces as skeletons, and further more we can evaluate the convolution surface by partition the skeleton into adequate sets of line-segments or triangles.

The first kernel function applied to the convolution surface is *Gaussian kernels*, and subsequently [94, 134] proposed several more efficient closed-form alternative formulas which suitable for convolution along line segment as well as triangle skeletons. In summary, the kernel functions can be classified into three families [167] as:

1. Inverse of order i: $K(d) = \frac{1}{(\frac{d}{\sigma})^i}$
2. Cauchy of order i: $K(d) = \frac{1}{(1+\frac{d}{\sigma})^{\frac{i}{2}}}$
3. Compact polynomial of order i:$K(d) = \begin{cases} (1-(\frac{d}{\sigma})^2)^{\frac{i}{2}} & if\ d < \sigma, \\ 0 & otherwise \end{cases}$

where σ is a resizing constant, *Inverse and Cauchy of order i* functions are C^∞-continuous and the continuity of *Compact polynomial* kernel is given by the order of i. However, polynomial kernel has the advantage of providing a local support compared with the global support of Inverse and Cauchy which eases the requirement of local shape control and enables efficient field queries.

The application of convolution surfaces on volume skeletons (cubes) is performed in Figure 2.10, where the terrain of arbitrary topology (i. e., caves, overhangs) is modeled [117] which is impossible with traditional heightmap based 2.5D terrain modeling system. Afterwards, [53] define a set of terrain primitives by a geometric skeleton (point, segment, curve or contour) to generate mountain, hill, river and road. The elevation of the terrain primitive is able to obtained by

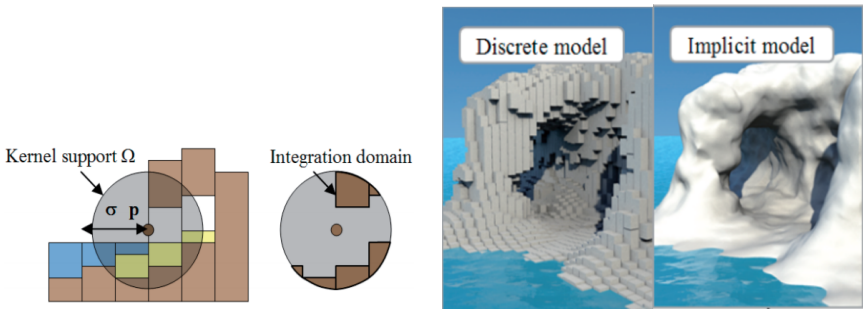


Figure 2.7: (a) Cube skeletons and convolution on volume; (b) Terrain was modeled from the cube skeleton [117].

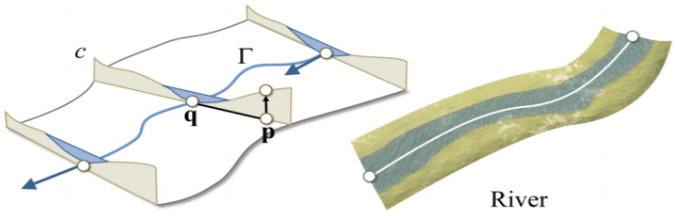


Figure 2.8: River or road can easily generate from the convolution surface with primitives of lines or curves [53].

a weight function depending on the distance from the skeleton (see Fig. 2.8).

WEIGHTED SKELETONS As illustrated in Figure 2.8, for the sake of more precise manipulation of the generated terrain it is crucial to varies the radius of primitives along the skeleton. This lead to an extension of constant weight convolution surface into non-constant weighted skeleton. Their mainly exists three models to generate weighted skeletons (see Fig. 2.9):

1. The standard formulation was developed by Jin [67] to scale the origin convolution field function by $\tau(s)$ as:

$$f(p) = \int_{s\in\Omega} \tau(s) K(\left\| p - \Gamma(s) \right\|) ds \tag{2.11}$$

2. Later on, Hornus [65] propose to scale the distance in the origin convolution field function by $\tau(s)$ as:

$$f(p) = \int_{s\in\Omega} K(\frac{\|p - \Gamma(s)\|}{\tau(s)}) ds \tag{2.12}$$

3. More recently, Zanni [167] showed that their SCALIS method to scale both the distance and the original convolution filed function can effectively avoid blurring and vanishing details as:

$$f(p) = \int_{s\in\Omega} \frac{1}{\tau(s)} K(\frac{\|p - \Gamma(s)\|}{\tau(s)}) ds \tag{2.13}$$

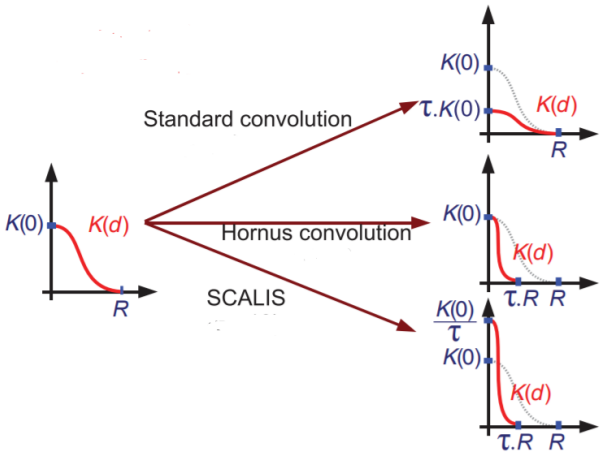
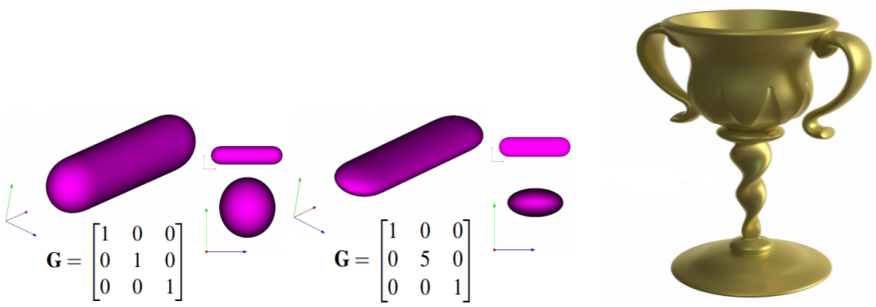


Figure 2.9: Three different models to generate weighted skeletons [165]. While Equation 2.11 scales the height of the kernel, keeping the width of the support unchanged, Equation 2.12 generates a kernel of constant height, but of varying support-size. SCALIS [167] method (Eq. 2.13) scale both.

As Figure 2.9 depicted the three different policies we proposed above to generate weighted skeletons. The standard convolution formula (see Eq. 2.11) can be reused to form a closed-form solution (i. e., convolution triangle [175]). More specifically, $\tau(s)$ can be used to change the height (convolution radius) of the kernel $K(d)$ while keeping the width of the supported area unchanged. In this case, if we set the height $\tau(s)$ to a small value and sharp details can be obtained. The trick used in Equation 2.12 generate a kernel $K(d)$ of constant height, but the size of support area changes. Subsequently, Zanni [167] proposed a novel idea (see Eq. 2.13) by varying both the kernel $K(d)$ height and the size of support area while keeping the area below the curve fixed and it brings the property of scale invariance. Mathematically speaking, in the Equation 2.13 the division by the weight within the kernel $K(d)$ function is no longer convolution anymore but still belongs to the integral surface.

More recently, Alvaro [140] proposed an *aniostropic convolution surface* which is an extension of convolution surface and greatly enhance the diversity of the shape can be generate from 1D skeleton convolution surface. The aniostropy was achieved both in the normal section and the tangential direction, which in particular allows sharper and steeper radius variations (see Fig. 2.10) either along the skeleton or at the endpoint. As a result, the more flexible control of the thickness on the convolution surface is achieved. This property overcoming the shortcoming of traditional convolution surfaces which are not suitable for the design of non-organic shapes.

RADIAL BASIS FUNCTION Another type of method to represent the implicit surface is to model the complex shape directly from the constraint surface points. The previous implicit methods represent hinge parts of the object with a blending function. However, the surface can also be defined from the scalar field of a dense point set, and the



(a) (b)

Figure 2.10: (a) The left image is the isotropic convolution surface and the right side anisotropic convolution surface. (b) Model a cup with twisted surface through the anisotropic surface [140].

free-form objects can be approximated by fitting those scalar fields regardless of their topology.

Such representation is widely used in the digitization of historic heritage (i. e., 3D scan point clouds) for the augmented or virtual reality purposes. Therefore, the resulting field function $f : R^3 \rightarrow R$ of the target is expected to look like $f(p_i) = C$ for all input surface point p_i, and the additional constraint the field function similar like a distance field function. In order to interpolate the surface going through all input points, we must find the coefficient to solve the above linear euqation set ($f(p_i) = C$). This will leads to decompose a huge linear equation set (matrix) with a complexity of O(N^3). A set of compactly supported basis functions have been introduced in order to make the system sparse, faster to solve, and meanwhile reduce the number of input points needed for evaluation [21].

A first approach, called Radial Basis Functions (RBF) [23], is to discretize the field function f with a linear combination of basis functions ϕ centered at the input surface point p_i as [21]:

$$\begin{cases} field\ function & f(p) = \sum_i^N \alpha_i \phi(\|p - p_i\|) \\ subject\ to & f(p_i) = 0 \end{cases} \tag{2.14}$$

where $\|\cdot\|$ is the Euclidean norm on R^3, we can solve a series of equations under the constraint and the correspond real scalar coefficients α_i are then founded. While, a naive solution of $\alpha_i = 0$ will lead to the zero constant function $f = 0$, and thus impossible to obtain the coefficient to represent the surface of the target. To avoid this drawback, more constraints are required and the gradient constraints are promising to avoid the hassle by adding more value constraints with offset points. This result in a simpler and more robust reconstruction technique called Hermite RBF (HRBF) [90]. Adding the gradient of the generated field make it fits with the normal n_i at point p_i (Hermite data $\{(p_i, n_i)\}_{i=0}^N$, a pair of points and normals), leading to the following problem reformulation:

$$\begin{cases} field\ function & f(p) = \sum_i^N \alpha_i \phi(\|p - p_i\|) + \beta_i \nabla \phi_i(\|p - p_i\|) \\ subject\ to & \nabla f(p_i) = n_i\ and\ f(p_i) = 0 \end{cases} \tag{2.15}$$

With N (each $i = 1, 2, ..., N$) the number of samples, and this dense linear matrix can be robustly settled from a standard lower–upper (LU) decomposition algorithm. Naturally, we can obtain the corresponding interpolation coefficients: a scalar value α_i and a vector β_i to reconstruct the geometric surface. The coefficient (in a subspace of the linear space R^n) is a continuously differentiable scalar-valued function in R^n and can be evaluated from the first-order Hermite interpolation - a particular situation of Hermite-Birkhoff interpolation theory [160]. Fortunately, once the coefficients are picked up they can be stored in the memory and works for the particular shape on demand, thus calculating the linear matrix does not affect the performance of evaluation the implicit surface.

The primary function ϕ in RBF must be a real valued function on the definition of $[0, \infty)$ and mostly are of non-compact support. And the choices of basis function ϕ can be made depending on the desired interpolation properties and the dimension of the ambient space [23]:

1. Thin plates spline: $\phi(r) = r^2 log(r)$
2. Polyharmonic splines: $\begin{cases} \phi(r) = r^k, & k = 1, 3, 5... \\ \phi(r) = r^k ln(r), & k = 2, 4, 6... \end{cases}$
3. Gaussian: $\phi(r) = e^{-cr^2}$
4. Multiquadric and triharmonic splines:
   $\phi(r) = \sqrt{1 + (cr)^2}$ *and* $\phi(d) = r^3$

The biggest advantage of RBF method is to represent an entire complex target shape with a compact, continuous and differentiable function. This style of representation surpassing the traditional piecewise parametric surfaces and convolution patches in several aspects. Firstly, the RBF function can be evaluated in the whole 3D space on demand to produce particular meshes, i. e. in simulation system the object can be computed at the desired resolution when and where needed. Secondly, the complex shape representation is able to switch into a more straightforward surface parameterization and optimization problem with a group of discrete, non-uniformly sampled surface points. To sum up the RBF function enlarged the type of shape can be represented by implicit method which previously has been limited to mere "organic" objects such as molecules, into the feasible real-world object acquired by 3D laser or camera scanners (i. e., kinect).

2.2 COMBINING IMPLICIT SURFACES

Functionally based implicit surfaces (so-called "procedural") are well known for their capacity to compose the associate scalar fields of primitive geometries into more complex shapes. The greatest strength during this process is their smooth blending property which leads to the first implicit modeling system called Constructive Solid Geometry [13] (CSG). The main attractive feature of CSG modeling system, compared to parametric surfaces or meshes, is their nice blending

property. Implicit primitives can be used in a constructive tree structure where smooth shapes of cooresponding arbitrary topological genus are progressively blended into more complex ones by simply summing (or other operator) their field functions.

Ricci [123] was one of the first to introduce several basic boolean operations: the union ($\cup$), the intersection ($\cap$), the difference ($\setminus$) to produce a smooth transition between the scalar field, and these operators can be performed hierarchically owing to the Construction Tree. Also those simple operators are called R-function, and can be defined as follows [109]:

1. Union: $(f_1 \cup f_2)(p) = \frac{1}{1+\alpha}(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2})$
2. Intersection: $(f_1 \cap f_2)(p) = \frac{1}{1+\alpha}(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2})$
3. Difference: $(f_1 \setminus f_2)(p) = f_1 \cap (-f_2)$

where $\alpha = \alpha(f_1, f_2)$ is an arbitrary continuous function controls the amount of blend during the composition and must satisfy the following constrains:

1. $-1 < \alpha(f_1, f_2) \leq 1$,
2. $\alpha(f_1, f_2) = \alpha(f_2, f_1) = \alpha(-f_1, f_2) = \alpha(f_1, -f_2)$

If we set $\alpha = 1$ then the operator becomes the famous: $max(f_1, f_2)$, $min(f_1, f_2)$. This type of simplification make it very convenient for calculation but will bring about C^1 discontinuity when $f_1 = f_2$. This discontinuity between scalar fields will lead to the creation of artifacts in the subsequent blending operator. If we set $\alpha = 0$, the union and intersection becomes the summation corresponding to the popular *sum* operator used by Blinn [12] to display molecules and defined as:

1. Union: $(f_1 \cup f_2)(p) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2}$
2. Intersection: $(f_1 \cap f_2)(p) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$

The new formula has C^1 discontinuity only in points where both arguments are equal to zero. Nevertheless you can achieve C^m continuity only by multiply the union and intersection equation (when $\alpha = 0$) with $((f_1)^2 + (f_2)^2)^{\frac{m}{2}}$. When α increases we can notice a decreasing blending effect, as illustrated in Figure 2.11, down to a sharp union when $\alpha \to \infty$.

We will first introduce the extension of CSG named BlobTree [163] which is an autonomous way to consists shapes into complex objects. Then we go through the recent process of operators in solving the four major problems in the previous basic boolean operators.

2.2.1 *The BlobTree*

Implicit surfaces were introduced in geometric modeling for their capability of being robustly combined into a CSG tree. Even with sharp

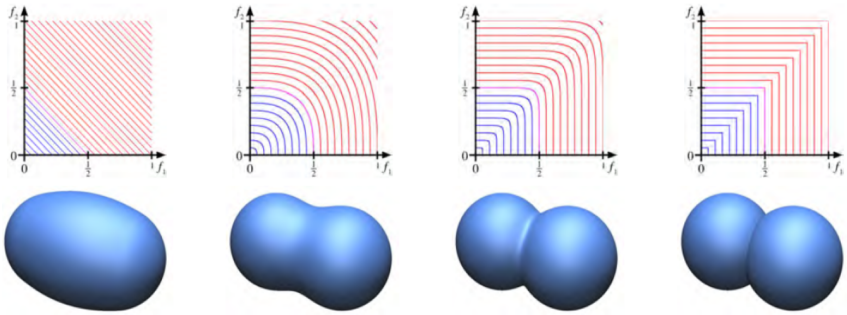


Figure 2.11: Four different values for the parameter α, from left to right, $\alpha = 1$, $\alpha = 3$, $\alpha = 10$, $\alpha \to \infty$ [21].

or smooth transitions at the vicinity intersection area of the combined surface. The classic CSG tree is usually implemented using a dynamic tree structure, whose inner nodes are composition operators and leaves are prime implicit primitives. This process usually involves design an assembly scheme which allows iterative and intuitive combinations of prime geometry primitives sequentially through simple operators to design the complex object. The CSG system successfully describe the relations between Boolean operators but leaving the connection of blending, warping (i. e., deformations) unconsidered.

The BlobTree [163] modeling system is also defined by the expression which combine implicit primitives, however, operators are extended from simple Booleans (i. e. *union* ($\cup$), *intersection* ($\cap$), *difference* ($\setminus$)) to *blending* (i. e. *blend* ($+$), *superelliptic blend* ($\diamond$)), and *warping* (ω). The Boolean and *blending* operators are binary operators and the *warping* is a unary operator. Here, we introduce the *blending* and *warping* operators in detail.

BLENDING OPERATOR: Generally blending two primitives are the same with *union* operator, but *super-elliptic blending* ($\diamond$) achieve larger range of blending and defined as:

$$f_{1\diamond 2}(p) = ((f_1)^n + (f_2)^n)^{\frac{1}{n}}$$

where n varies from 1 to ∞ and the blending operator will change from binary operator to n-ary operator which creates a set of blends between *sum blending* and *union*.

WARPING OPERATOR: Another useful operator in order to create free-form deformable object is to distort the shape of a surface by warping the space in its immediate vicinity, called *warping*. A warp is a continuous function $w(p)$ that maps R into R and fully characterized by the distance to its skeleton $d_i(p)$, potential function $g_i(r)$ which eventually denoted as $\{g_i(r), d_i(p), w_i(p)\}$:

$$f_i(p) = g_i(r) \cdot d_i(p) \cdot w_i(p)$$

The BlobTree import extra operators (*blending* and *warping*) and a hierarchical modeling framework which gives more freedom to the designer. The multiple operators are managed by a tree where each node

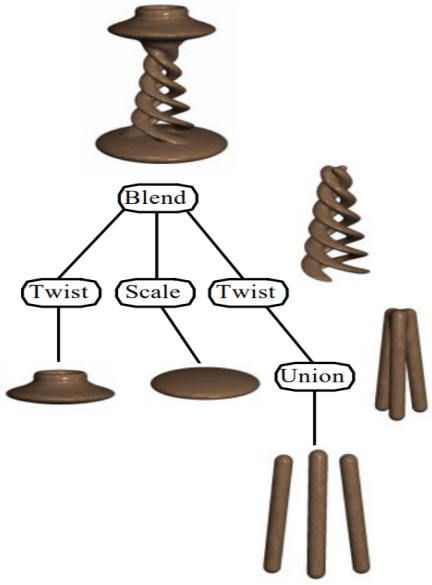


Figure 2.12: The workflow of BlobTree to synthesis the candlestick [163].

represent either composition or warping operators, the leaves of the tree are the implicit primitives and all those elements are recursively traversed to model complex shapes (see Fig. 2.12). Whereas, designing complex objects from scratch can be rather cumbersome even with the BlobTree. The designer would need to think about the order of each operators and how to balance it to make the evaluation more efficient. Consequently, optimization methods were also introduced to accelerate the BlobTree. For instance, [41] introduced the reduction and pruning operator on the BlobTree to limit the cost of the scalar field query. More specifically, the reduction is a simplified version of the *affine transform* operator by pushing them down in the tree. The latter operator - the pruning is to construct a sub tree from a small subset of the implicit primitive in a given region of space. This improvement significantly accelerate the evaluation of the BlobTree.

2.2.2 *Blending*

In fact, soft blending, R-function, Ricci's super-ellipsoid blends and Perlin's set operations are widely applied to smoothly blend soft objects because they have lower computing complexity. However, these operations still face four major difficulties as follows [56] (see Fig. 2.13):

1. Bulging effet: The bulging effect usually observed when two continuous skeleton surfaces (or convolution surface) are blended with a simple sum operator. The main reason is that the classical implicit surfaces are not independent to subdivision, and although the integral of the convolution surface able to ease this problem but it still arises when more than two parts linked together. Therefore, the resulting implicit surface from the blend operator will produce a bulge on the intersection area of the

input surfaces, as depicted in Figure 2.13 (a). This is due to the fact that [167]:

$$k(\min_{g \in S_1 \bigcup S_2} \|g - p\|) \neq k(\min_{g \in S_1} \|g - p\|) + k(\min_{g \in S_2} \|g - p\|)$$

This effect is really annoying when artists wants to create the tiny details (i. e., hair, fur) on a complex object by attaching several primitives. In particular, blending several different primitives will introduce discontinuities of the gradient and issue in more obvious bulge. For instance, modeling the tail of an animal would require consisting of several segments so that the tail can be animated from straight to curved lines, and therefore subject to bulges [167].

2. Locality problem: The implicit primitives blend at a distance means the neighbor area (distance) is the only factor that influence how the implicit surfaces should react (see Fig. 2.13c). This makes assemble implicit primitives difficult because designers usually unclear at which distance the implicit primitives should place to achieve targeting effects. In briefly, it is necessary to accurately prevent the unwanted parts in 3D modeling applications. This unpredictable issue is also serious in animation applications, where pieces of different material in a 3D model should be allowed to deform differently at different distance. For instance, if the arm of a character's move towards its head during the animation, those primitives tend to blend together, which is what we want to eliminate.

3. Absorption problem: Sharp and tiny details are usually smoothed and inflated (merged) when they are blended into larger implicit primitives, since the sharp primitive influence are are totally enclosed into the support region of the larger primitive. This problem occurs when the gap between the support radius of primitives to be blended is too large and it will prevents the creation of thin details such as hairs or tips. That's the reason implicit surface only well-known to generate blobby, organic shapes (see Fig. 2.13b).

4. Topology problem: The composition capacity of implicit primitives provides an intuitive way to reconstruct objects with arbitrary topology. However, the lack of control on the blending operator issue in the topology of the resulting implicit surface unpredictable. For instance, in Figure 2.13 (d) the center of the upper circle with a hole must be reserved, however, the intuitive blending operator will cause unpredictable and uncontrollable filling of the hole and impossible to maintain the topology as we expected. This poor manipulation of topology bring problems where respect parts of topology should be altered by the assembly defined in the tree structure, not of the blend operator.

In general, improving the variety and effectiveness of implicit modeling is equal to the design of new composition operator. To address

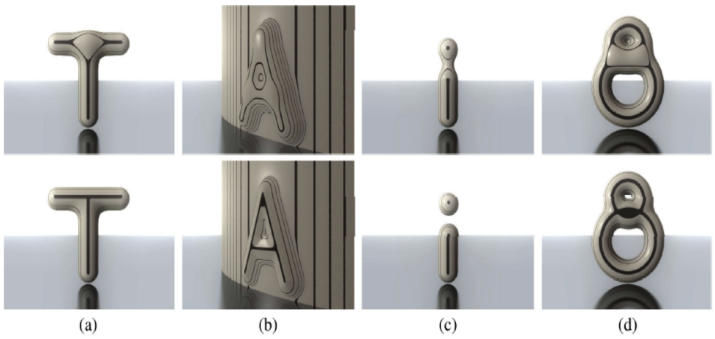


Figure 2.13: (a) Bulging problem; (b) Absorption problem; (c) Locality problem; (d) Topology problem.

the blending problem researchers have presented more user controls to intuitively manipulate the transition shape between the combined object parts. In essence, the blending behavior rely on the properties of each implicit primitives' scalar fields. One promising approach is to use the binary operator, which combine two implicit primitives and then design the transition shape in the support area through a useful representation: the operator space (see Fig. 2.14). As a result, the control of the blending is aim at building the correspondence between the operator space and the object's scalar field (potential field). This abstraction intuitively visualize the effect of a binary operator and provides an instinct tool to control and devise their behavior. Since then the devise of appropriate equation for the blending in operator space is of great interest for researchers who wants to provide better control of the transition shape at the primitive intersection areas. In the following, we will introduce several most typical blending operators to address the blending limitations (previous four major limitations) by defining control over the transition shape between the scalar field of implicit primitives.

A first attempt was made to avoid the drawback of bulge by taking the advantage of the operator space: the clean union. In Ricci's [123] *union* operator, the blend is regulated by a unique parameter α that provides a simple parameter on the smoothness of the transition shape between input primitives but failed to consider the size of the implicit primitives nor allowing elaborate modification of the shape transitions between primitives. A similar but more general operator to Ricci's *union* has been developed by Pasko [109] for globally supported field functions, where the two integer parts refer to the so-called "clean union" operator, and the fractional part relates to the amount of effects applied to induce the blend.

In Rockwood's [124] seminal paper the elimination of the bulging effect can be tackled by a new operator called: the super-elliptic blend. In this operator the range area of the blend is controlled by the cosine value of the angle between field gradients [124]. Even though this operator successfully restrains the unwanted bulges, however, it brings about the locality and topology problems and more importantly the resulting surface is only C^0-continuous. Afterwards, Gourmel [56] proposed the gradient-based blend operator based on the super-elliptic

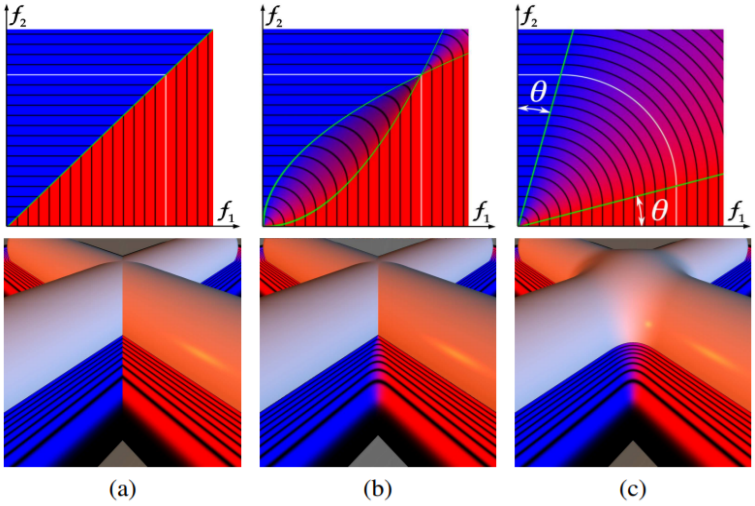


Figure 2.14: Three compositions applied to a pair of cylindrical implicit primitives forming a cross: (a) standard max union; (b) clean union; (c) Barthe's blending operator parameterized by an opening angle θ [56].

operator, which smooth the angle between field gradients. This new gradient-based blending operator is more general, scalable and is capable to limit the amount of blending at the intersect area within the input shape.

Another problem of blending is the composition operator will produce unpredictable artifacts. This is mainly due to the fact that the summation of negative field with positive field will produce unwanted inversion at particular intersecting area. Canezin [22] proposed a solution to this problem based on an extra constraint. Their insight is the initial scalar field should be bounded to a particular range, and the bending operator is defined in order to maintain this property [22]. This inner bounded operator greatly enhanced the intuitiveness of the composition behavior when the multitude successive composition are required for modeling complex objects [22].

The reminder of this subsection explained the previous method in more detail: *clean union*, *gradient-based blending* and *adequate inner bound blending* as follows:

- Clean union:
  - The clean union (see Fig. 2.14) comes from the Pasko's operator. Before we introduce clean union here we show the operator of Pasko's at first as:

$$g(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + (\frac{f_1}{a_1})^2 + (\frac{f_2}{a_2})^2} \quad (2.16)$$

    where a_0 controls the amount of global blending between input implicit primitives (f_1,f_2), and a_1 and a_2 manipulate the amount of asymmetry in the blend operator.

    The clean union operator is mixture operator: the union operator on the combined surfaces and the blending operator at their field functions. This improvement enables

smooth field variants around the surface, thus making the field behave more similar to a distance field. It also prevents the gradient incontinuity in the generated field which may cause the discontinuity in the visualization of reflections on the surface or after further blending compositions. This allows an object built from such an operator to be later merged with another implicit primitive no longer introducing unwanted sharp edges into the blend area.

$$g(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} \tag{2.17}$$

- Gradient-based blending [56]:
  - This operator provides an automatic method to adjust the smoothness and sharpness of the transition shape (in the operator space) between arbitrary functional implicit primitives. Contrary to previous methods, this new operator not only a function of the field function but also consider their gradient. The gradient parameter control the curve (so-called "controller") in operator space to define the blending effect region where a particular pattern is applied to describe the shape of the conjunction area. For instance, when modeling "organic" shapes only small contribution from the blend is required if the gradient is aligned. In contrast, if the blend is the dominant factor and the gradient deviate to some particular angles where blend at maximal to smoothly consist different parts of the "organic" shape. Blend operator is then degrade to the clean union before gradients are opposite in order to design folding effects [21]. Another extra benefit is the gradient can also help prevent the small details vanishing from blurring or inflating when blended with much larger size of primitives.

- Adequate Inner bound for Compact field functions [22]:
  - The development of implicit modeling operator able to offer an accurate control on the shape transition within blending areas of the input primitives. However, these effects rely on the field function where the implicit primitives are defined. As a result, we can partition the whole field space with outside part where functions defined and inside part. Naturally, most efforts have been made on the outer part where the field function defined and little efforts have been made on the inner parts. Whereas, the inner fields are equally important to the outer part when *difference* or *intersection* operators are considered. For instance, when composition operators, such as Boolean *difference*, is used and artifacts may arise in the inner part. This artifact is mainly due to the fact that the term $(1 - f)$ used to perform the required inversion of the field function can become negative [22].

Canezin [165] proposed a modified version of the gradient-based blending operator which meets the new constraint by introducing a new family of boundary curves in the 2D representation of blending operator [165] (operator space).

They also proposed a new set of asymmetric operators tailored for the modeling of line (or segment) details while preserving the integrability of the resulting fields [165]. More specifically, this modified blending improves the field quality when boolean operators are imposed to add small details on a large primitive. This can be achieved from modifying the gradient-based operator by progressively ignoring the field of the details when it becomes larger than the iso-value. The aim is to remove the ghost shape of the details that remains after the operation.

In total, this new bounded operator enables a precise control on the shape of the inner and outer field boundaries and make the differences and intersections seamlessly applied without bring about the discontinuities or field distortions.

2.3 DETAILS OF IMPLICIT SURFACE

In order to enhance the visual realistic or fidelity of implicit surfaces, several ways are proposed to add surface details. In general, there are mainly two directions to represent surface details on the smooth surface (implicit surface) - the image texture and the actual geometric details. Textures are usually related to visual or tactile properties and composed of repeating patterns which are able to mapping onto the smooth surface, such as grass, rock, and meanwhile widely used to preserve various surface's physical properties (i. e., color, reflection, transparency or displacement). Due to this universality the second way - the actual geometric details - can also synthesized from image textures through displacement-mapping or bump-mapping to modify the normal or the actual vertice's position on the 3D mesh. More recently, the virtual texture mapping [25] is becoming a popular technique to represent the surface details (able to increase the number of input mesh's triangles from millions to billions) without explicit modeling the properties of the object's geometry or materials.

In this subsection we will first track the development of the texture synthesis algorithm and then gallery show the corresponding mapping methods, such as bump mapping [10], displacement mapping [28] or shell mapping [118], to actually modify the topology of the mesh for the sake of adding surface details. Then we will analysis the limitations of these methods, for instance, they are hard to ensure the coherency and consistency especially when the parameterization is not available (especially the parameterization of implicit surface). Finally, we will exhibit a group of procedural noise which can be applied directly to the implicit surface in a parameterization free style and at the same time ensure the coherency and consistency of the repeat pattern maintained in the corresponding procedural textures.

2.3.1 *Image Texture & Texture Mapping*

The most widely used image textures method on the implicit surface is example-based texture synthesis and surface-based texture synthesis. Texture synthesis means from a small path of sample image we can reproduce an arbitrary size of similar texture in order to cover a much larger planar area or manifold surface. One potential benefit of texture synthesis is to amplify the input small texture or tilable texture on demand. A vast amount of work is available on that area and here we only cover a small part which directly related to implicit surface.

Example-based texture synthesis method intend to use 2D image texture as input and synthesize a similar texture over a planar with arbitrary size (see Fig. 2.15). A large varieties of solutions have been proposed since the seminal work of Wei and Levoy [157] to address this issue. The method proposed by Wei and Levoy [157] relys on a fixed size neighbor sampling of the input texture and the $L2$-norm criterion to compare their similarity. In their method, each pixel value of the output texture is decided in raster scan order after querying with a tree-structured vector quantization (TSVQ) on the sample texture and searching all similar pixel neighbors to meet the requirement. After that, the pixel value is determined by randomly chosen one from these neighborhoods. Subsequently, Ashikhmin [2] improved this algorithm by changing the search of entire space of neighborhoods of the sample image into a neighborhood most similar to the current L-shaped neighborhoods which preserves the coherence at the same time. This advance make it well suited for the natural texture synthesis. In summary, in order to make example-based texture synthesis suitable for diverse type of textures and maintain their high quality, there exists several additional issues must be considered, such as image pyramids, fast search techniques, coherence between adjacent pixels and histogram matching [156].

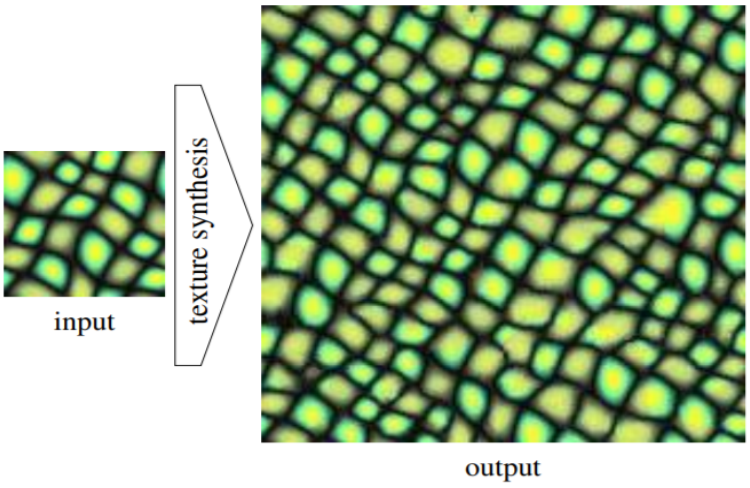


Figure 2.15: From the input texture we can synthesis a new texture that look-alike the input. The synthesized texture is tileable and can be extended seamlessly to arbitrary size decide by the user [156].

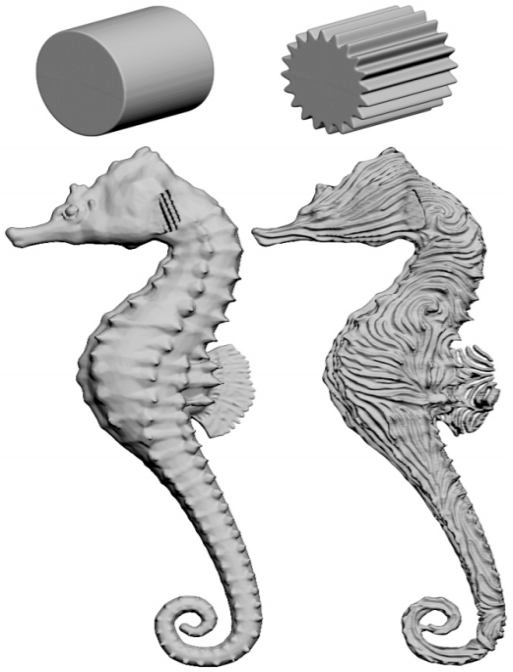


Figure 2.16: A cylinder pair smooth and with grooves (top) are used as input examplar to modify a seahorse model to have similar grooves pattern (bottom) [9].

In addition, this 2D pixel based example texture synthesis algorithm is capable to be extended into a voxel-based 3D space to depict geometric details on the surface of voxel-based 3D models (see Fig. 2.16). For the purpose of this extension, Bhat [9] introduced the following steps to synthesis details on the 3D model: firstly, the 3D object must be voxelized into the interior, exterior and intermediate voxels; secondly, voxels plays the role of pixels and the voxel neighbor is naturally a collection of connected neighbor voxels; thirdly, the traversal order is particularly defined from the top volume slice to the bottom volume slice; finally, each voxel defined a local coordinate to compute the sweep distance field and ensure the consistency of the surface texture. After the previous steps, the problem comes to train the feature vectors and search the closed matches during the synthesis which can be solved in a similar way to 2D example-based texture synthesis. This new approach requires no parameterization step to transfer the actual geometric changes (not just the modification via surface normals or color) from the examplar 3D model to the target 3D model. The benefit is we can synthesis not only micro details of the surface (such as height map) but also the macro structures (see Fig. 2.16) that completely change the topology of the global shape. Whereas, this method is unable to generate a smooth transition between the two textured surfaces, and the details is limited by the resolution of the grid size which make it computational expensive on the high-resolution grid.

For many applications, synthesis textures on a larger planar or 3D voxel is not sufficient, often we want to apply textures onto a particular manifold surface to achieve special effects. In general, there exists two ways to fulfil this task, one way is to mapping the planar texture onto

the surface (see Sec. 2.3.1) and another way is to synthesize the texture directly on the surface. More specifically, the idea of performing texture synthesis on a manifold surface can be divided into two steps: 1. create an orientation field over the whole surface, and 2. execute texture synthesis according to the previous orientation field. The purpose of the orientation field is to determine the direction of the texture to ensure it will flow seamlessly across the whole manifold surface. In brief, the orientation field plays the same role like the rows or columns in the 2D image pixel set. A vector field associates with each point on the surface signify a vector that is tangent to the surface. In effect, this vector field coordinate allows us to step over the whole manifold surface in either of two perpendicular directions just like we are traversal from one pixel to another in 2D planar. Exactly as there exists different ways to synthesis texture on a regular grid, there are also various ways where synthesizing can be performed on the manifold surface. Several methods have been proposed to synthesize textures directly on a 3D model's surface.

The first type of method to synthesis texture on the manifold surface is a point-based method. This method adopt a dense and hierarchy evenly distributed points on the mesh to create an approach similar to the pixel-based neighborhood synthesis. Once those points has been collected, they can be visited sequentially along the vector fields over the surface and the corresponding neighborhoods can be defined to find the best matching neighborhood in the texture exemplar. Finally the target surface is shaded with this method. For instance, Wei and Levoy [158] extended their example-based synthesis method onto manifold surfaces by building a space neighborhoods from a local parameterization of surface mesh vertices. Particularly they use displacement mapping to modify the actual topology of the geometry object. Zhang et al. [170] proposed a novel approach to synthesis progressively-variant textures over the manifold surface. Their main contribution on the surface synthesis is that they introduce texton mask in conjunction with the target texture image and bring a two-layer neighborhood search which can solve the fracture of texture elements in the previous texture synthesis algorithms. With the texton mask the synthesized texture on the target surface maintains the integrity and suitable for the spatially-varying textures (see Fig. 2.17).

Other types of methods proposed to synthesis surface are mainly focus on the mapping between regions of the texture planar and the surface. Their main idea is to split the surface into regular pixel grid or the triangle of the mesh and the mapping is able to perform on the separated patches. For instance, Le and Hoppe [80] propose a method to split the manifold surface into small pixel neighborhoods and a GPU-accelerated neighbor searching algorithm is executed to speed up the synthesis process.

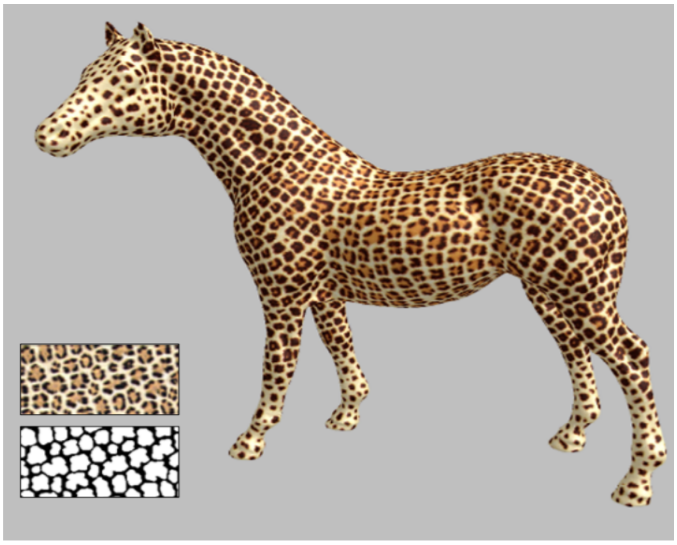


Figure 2.17: Bottom left is the input texture and the corresponding texton mask. The progressive-variant texture synthesized on the body of the horse [170].

Texture Mapping

Texture mapping is a common method that adds realism to the smooth surface of mesh. A naive implementation is called parameterization in computer graphic by mapping a planar area to the surface of 3D model which possibly will bring the problem of visual discontinuity, texture distortion, and specific orientation of textures. In general, there exists a bijective mapping between two surfaces with similar topology. If one of these surfaces is a triangular mesh, the method to find out such a bijective mapping is referred to as mesh parameterization. The surface where the mesh mapped to is usually called the parameter domain. Over the past decades, this method has gradually becomes an irreplaceable tool for many industry mesh processing applications, including detail-mapping, detail-transfer, morphing, mesh-editing, mesh-completion, remeshing, compression, surface-fitting, and shape-analysis, as discussed below [64].

Globally seamless texture mapping is known as difficult where distortion, discontinuities or both always occurs. Ray [120] tackled this problem by patching the object's manifold surface with seamlessly textured triangles. By defining each patch of the surface with a mapping function to a 2D domain, a set of functions named global parameterization is used to control the parameterization process. Then a global energy function is defined to optimize the parameter of each mapping function where the distortion and discontinuity are manipulated. The result is ideal, however, this approach works only for isotropic textures. Moreover, their method requires careful preprocessing of the input texture triangles to comply with specific boundary constraints. In addition, since it defines the bijective mapping function between each surface patch to a 2D domain and employs relatively large number of triangles, make their approach less effective especially when texturing

narrow features. In short, this algorithm works quite well for some (i. e., isotropic texture) but not all kinds of natural textures.

Afterwards, the previous texture mapping between planar area and the surface is replaced by the volumetric mapping. The volumetric mapping refers to constructing the correspondence between solid objects directly, which becomes attractive for the purpose of deformation or morphing. Harmonic volumetric mapping [84] was proposed to build a smooth bijective mapping between the two solid objects. Given a boundary mapping between two solid models, the volumetric mapping is derived by solving a linear system constructed from a boundary method (method of fundamental solution, MFS) [84]. The volumetric mapping is a meshless (neither domain nor mesh connectivity is required) procedural, nevertheless, the property of the inner region is depends on the boundary constraint. Moreover, it has a wide applications in shape registration or material transplant.

One important task in geometry processing is to handle the shape matching and analysis problems. Due to the topological and geometric complexity of the 3D object both problems require either a mapping to process the correspondence between objects or a parameterization technique to project an object onto certain canonical domains. Consequently, more recent research focus on different methods to parameterize meshes, aiming at achieving different parameterization properties (i. e., conformal mapping [20]) as well as various parameter domains (i. e., singularities [173]). Moreover, defining a seamless mapping of two objects enables sharing the surface details between them, or the interpolation of the object with the appearance of several other shapes. As a result, this bijective mapping enables not only transferring the static geometry of surface (i. e., surface details) but also the animation data between objects, i. e., the local surface influence from bones can be transferred to the animation rig, the local affine transformation of each triangle in the mesh can be transferred to other meshes. Moreover, one can change the ratio of interpolation over time and possibly morphing animation is able be achieved. More specifically, in the scenario of spatially-varying or frequency-varying morphs, the change rate varies in the different components or different frequency bands (the coarseness of the features is transformed) of the object.

2.3.2 *Procedural Texture*

Textures plays a very important role in computer graphics and their synthesis is always a hot research topic in the last decades. Generally, texture synthesis methods can be classified into two directions: explicit method and implicit method. One of the most typical explicit method is the example based (pixel value exists in the spatial neighborhoods) texture synthesis as we mentioned above (see Chapter 2.3.1). However, a typical implicit method named procedural texture enables the evaluation of each independent query point on the fly. As a result, procedural textures offers several feasible advantages over traditional explicit texture synthesis. Firstly, texture texels can be evaluated on

demand and in arbitrary order (parallelizable) which make it more flexible and computationally cheaper than explicit ones. Secondly, procedural methods often consumes low memory footprint since it does not need to preserve the texture images in the memory (especially useful for 3D or even higher dimensional textures, i. e., solid texture, dynamic texture) except a few parameters of the equation. Procedural methods provide a compact way to generate complex and diverse (within particular type) effects of visual details through the combination of kernel equations or parameters. Finally, procedural methods is often evaluated independently at each query point in a constant time and offers casual accessibility which makes it suitable to exploit the massive parallel computing ability of the GPU. Unfortunately, implicit methods are usually less general than explicit ones. Because the procedural texture is limited to a few types (classes) of textures and requires some efforts to get the parameter right. Due to the requirement of independent texel evaluation, implicit methods cannot use the statistical example texture synthesis where the evaluation is based on spatial neighbor-pixel dependencies. With the iteration of the graphic hardware, procedural method becomes more feasible for the real-time applications notably noise based procedural texture synthesis.

The term "procedural" in computer graphics means utilizing some rules, equations or algorithms to generate particular effects rather than real feature structure or digital photograph. Procedural noise algorithm is a procedural technique to generate specific pattern for varies graphic applications. The term "procedural noise" is actually a random number consists into unstructured patterns, and is widely used to add details where there is insufficient evident feature structures (i. e., terrain details, object's details).

Efficiently adding rich and realistic visual details to a surface or image has always been one of the major challenges in computer graphics. Ever since the famous image of the marble vase (see Fig. 2.18), presented by Perlin, perlin noise [113] as one of the most successful procedural noise has been extensively used both in research and industry. For instance, perlin noise provides help for the specific class of materials or shapes such as clouds, waves, heat ripples, wood and even the motion of virtual characters, leaving the controllability of such noise difficult to determine.

Then the problem comes to how to design such an ideal pattern. In theory, such random patterns are often displayed in the spatial domain, where the pattern is determined by specifying the value for each pixel point. Whereas an alternative transformation in the frequency domain, a signal can be determined by specifying the specific amplitude and phase for each frequency band. However, for such random patterns, the phase domain is usually random and does not contribute any structure information. Therefore, the noise pattern is often characterized by its power spectrum. The specific high frequency pixel in the power spectrum leads to a higher contribution of the corresponding feature pattern in the spatial domain. As a result, the task involved in designing noise pattern is either corresponds to shaping its power

Figure 2.18: Perlin's famous marble vast, one of the first procedural texture from procedural noise (perlin noise) [113].

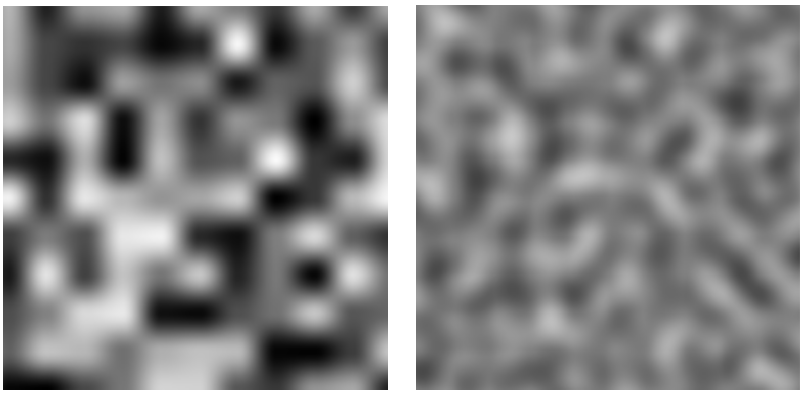
spectrum or filtering a noise by damping higher frequencies in the power spectrum.

In order to manipulate the power spectrum the white noise is usually employed as the raw material to generate unstructured signals with a combination of arbitrary frequencies. This is mainly because the white noise contains all frequencies in the equal possibility and each component involve a random phase, so it is an ideal basis in the frequency domain. More recently, another type of noise named gabor noise [77] was proposed to give us more intuitive and accurate control on the power spectrum, and gradually becomes the mainstream of the procedural noise algorithm. In the next part we provide a unified view of both directions (i. e., white noise based method and gabor noise based method) of techniques to explore the development of procedural noise.

Lattice gradient noise

As we mentioned before, noise is "just" a random number generator and the easiest procedural noise is naming as the white noise. The white noise is generated by assign a random value for each pixel point and thus they are isolated with neighbor-pixels, which will bring the "blocky" effects among neighbor-pixels. Then the value noise is proposed to smooth the transition between neighbor-pixels through an interpolating between four random values of the corner (2D), but there still exists other ways to produce value noise. However, value noise also tends to look "blocky" and with obvious patches on a larger scale (see Fig. 2.19a).

Procedural noise algorithms were initially invented to give a naturalness effect on the image texture. Most natural phenomenons looks random but each neighbor part is connected, i. e., clouds, rocks, trees. In order to simulate this random but correlate relationship, in 1985 Perlin [113] proposed the first lattice-based gradient noise named Perlin noise. Lattice-based noises means the noise functions is defined on the integer lattice (i. e., rows and columns) of the image (2D). The term gradient denote the interpolation of random gradients instead of random values, and the gradient is the result of a 2D random function which returns 2D vector directions. That is, they are able to generate smooth, continuous and self-similar noise patterns without sudden jumps/fractures or sharp edges (see Fig. 2.19b).



(a) (b)

Figure 2.19: The left image shows the value noise and the right perlin noise (also called, 2D Gradient noise).

3D Perlin noise evaluate the noise value at each point in space by generating a pseudo-random unit vector gradient at the eight corner of the integer cubic lattice and then define a smooth interpolating in-between (see Fig.2.20a). The pseudo-random gradient is given by hashing the integer cubic lattice (i. e., corner point (x, y, z)) to produce a set of 256 vectors sown on the surface of the sphere and randomly choosing one gradient from the vector set. The hash function works by applying a permutation to the lattice coordinate to de-correlate the indices into an array of pseudo-random unit-length gradient vectors. Afterwards, Perlin published an improved Perlin noise [115] where he made a few tweaks tricks. He changed the original interpolation function from third-order (cubic polynomial) to fifth-order (quintic polynomial) which has better continuity properties (C^2-continuity to C^4-continuity). For instance, the endpoints of the fifth-order curve becomes more "flat" so it can be gracefully stitches with the next one. In other words, a more smooth transition between the grid cell is achieved. However, this improvement does not affect the value range of the Perlin noise (original value range is $(-\frac{\sqrt{N}}{4}, \frac{\sqrt{N}}{4})$, and N is the dimension of the noise) but it does fluctuate the position and the amount of the maximum in the gradient magnitude. Moreover, instead of selecting the gradient in 256 vectors, one of merely 12 vectors defined by regularly pointing to the edges of a cube are replaced in the

improved Perlin noise. Collectively, these improvements successfully eliminate the undesired directional artifacts in the final texture.

At Siggraph2001, Perlin presented the simplex noise [114] from which he achieved the following improvements over the previous Perlin noise:

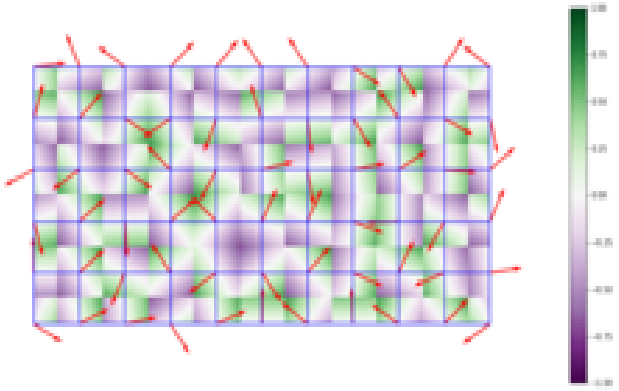
1. an algorithm with lower computational complexity and fewer multiplications especially higher dimension noise considered
2. a noise without directional artifacts
3. a noise with well-defined and smooth gradients
4. an algorithm that is easy to implement on modern GPU

More specifically, in the case of 2D Perlin noise we process the interpolation within 4 points (corners of a square); so we can theoretically infer that in 3D and 4D perlin noise we need to interpolate 2^3 and 2^4 points. That means, for the case of the N-dimension Perlin noise we need interpolate the 2^N corner points. But Perlin smartly noticed that although the most intuitive basic primitive to fill the space is a square, however, the simplest stable shape in 2D is the equilateral triangle (see Fig. 2.20b). The simplex origins from the subdividing of the regular lattice grid and skewing the simplex grid from the two isosceles triangles. Then the simplex shape for N-dimensions is a shape with $N+1$ corners. In other words, the elegant move of the lattice grid to simplex brings lessen one corner computation in 2D, four and eleven lessen in 3D and 4D respectively. Theoretically, the complexity of the classical Perlin noise is $O(n2^n)$, whereas, the complexity of simplex noise is decreased to $O(n^2)$.

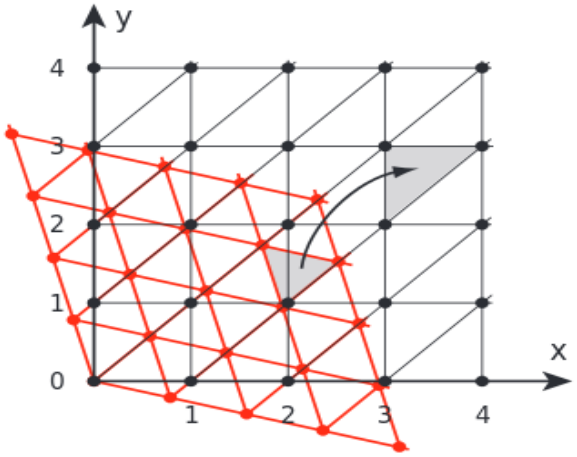
Unfortunately, simplex noise did not widespread applied like Perlin noise for a few reasons.

1. Perlin noise was already "good enough" for many even real-time applications (thanks to the progress of graphic hardware) so there wasn't much necessity for the change.
2. In order to generate perfect natural patterns from the Perlin noise, it requires lots of time to find the magic parameter. However, switching into another algorithm needs same amount of time to search it again.
3. Reimplement simplex noise requires some math tricky to deform the coordinate and it is unintuitive for understanding.

In short, there are mainly five parameters involved in manipulating the perlin/simplex noise (also similar in other procedural noise): amplitude, frequency, octaves, lacunarity, and gain. Frequency meansures the number of waves exist in a given interval, and the amplitude represent the height value of the wave. From a simply inversion of the value in frequency we can get the wavelength and that's why sometimes the term "wavelength" replace the "frequency". Each single set of those five parameters in perlin/simplex noise is called octave.



(a)



(b)

Figure 2.20: (a) The value of each pixel is a scalar produce between gradient vector with the direction vector to the point P. (b) A 2D simplex grid of triangles can be skewed by a nonuniform scaling to a grid of right-angle isosceles triangles, two of which form a square with sides of length 1 [58].

Under this definition, another popular term fractal version of perlin/simplex noise means combining multiple octaves with varying the value of amplitudes and frequencies. Lacunarity and gain measures how fast the value of frequency and ampltidue changes between each octaves. For instance, in most applications, the lacunarity (the times of frequency) will be about $1.95 \sim 2.05$ and the gain (the times of amplitude) is ranging from 0.3 to 0.7, which means within each octave will have twice the frequency and half the amplitude of the previous octave. Empirically, the lower the gain values the larger the terrain features will be, whereas the higher gain values will produce sharp mountain ranges on the textures.

Fractal noise is a self-similar noise when you scale in or out it still looks the same. In general, Perlin and simplex noise are fractal noises when multiple disparate octaves are accumulated together with a consistent range of lacunarity and gain. The fractal noise is the basis to generate terrain features, such as coastlines, which looks similar at different levels of scales. In fact, it is a fully empirical approach to

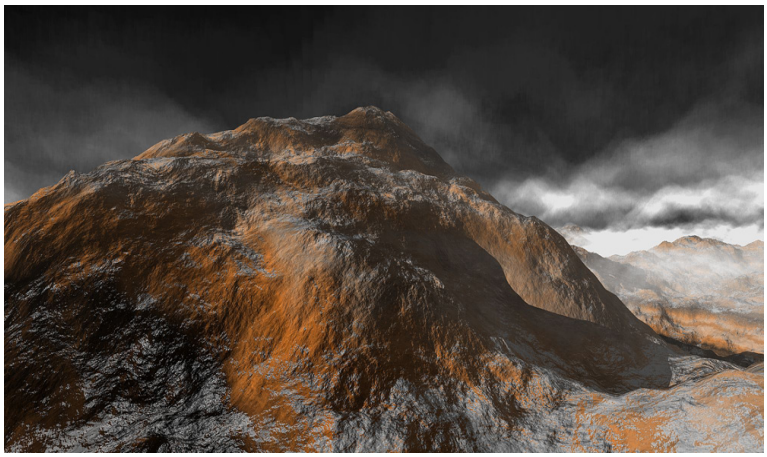


Figure 2.21: The volumetric approach using fractal noise to create compelling and realisitc terrain [154].

general natural terrain from the fractal noise. Furthermore, the number of corresponding parameters within fractal noise multiplied and even harder to generate ideal terrain patterns. Several commercial software are published and provide graphic interface which helps users design the pattern they want, i. e., world machine, terrain generator.

Noise is a powerful modeling tool to generate the geometric details, and therefore how to control the appearance of the noise pattern is the crux for applications. The power spectrum of the noise, which describes the contribution of each frequency band, is naturally a powerful tool to control the pattern generated by the noise. Perlin noise achieves spectral control through a weighted summation of band-limited octaves. However, Cook [29] noticed that perlin noise is only a weakly band-limited noise, and is therefore prone to aliasing and detail loss. Consequently, in 2005, Cook and DeRose [29] proposed the wavelet noise - an almost perfectly band-limited noise, providing good balance between details and minimal aliasing.

In the preprocessing step, a tile of noise coefficients N is created. These coefficients represent the noise as a quadratic B-spline surface. This is done by creating an image R filled with random noise, down-sampling R to create the half-size image R, upsampling R to a full size image R, and subtracting R from the original R to create N. The upsampling is a band-dependent translation and by adding $2^b x$ we can de-correlate the noise bands, and b index the band. The tile of noise coefficients N is thus created by taking R and removing the part that is representable at half-size. The coefficient to each band of the noise control the spectral character of the wavelet noise (see Fig. 2.22). The part left by the sampling process is the part that not representable at half-size, that is the band-limited part. The filter used in the down-sampling and upsampling steps are obtained using wavelet analysis and refinement coefficients of the uniform quadratic B-spline basis function [29]. Additionally, noise bands are orthogonal to each other, which makes spectral shaping more controllable.

Wavelet noise is slightly faster than perlin noise [29], but as an explicit noise they need to store initial image and uses more memory.

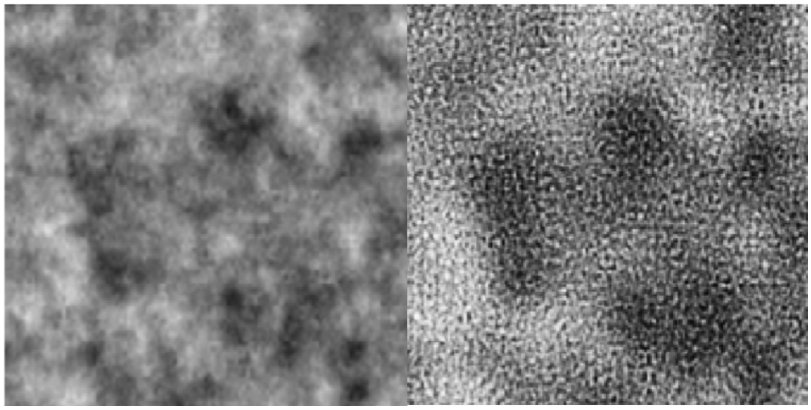


Figure 2.22: The left image is the 2D wavelet noise, 12 bands with gaussian distribution. The right image is 8 bands with white distribution [29].

However, even with a relatively large tile the memory requirements are still quite small. Because the bands are orthogonal, they provide a set of independent controls over the shape of the spectrum. The distribution of the final result is predictable and controllable. Most importantly, the noise is truly band-limited, so that virtually all of the detail can be rendered with minimal aliasing, even when projecting 3D noise onto a 2D surface [29]. Consequently, wavelet noise is well suited for the use as a procedural texture.

Another class of applications of lattice-based noise is physically-based simulations. Several authors have claimed that the noise function can be used for physically-based simulations. For instance, Perlin presented flow noise [116], a Perlin-like noise function for generating time-varying flow textures with swirling and advection. Bridson presented curl noise [17], a Perlin-like noise function for generating time-varying incompressible turbulent velocity fields.

Sparse convolution noise

Although the improved perlin noise and the subsequent wavelet noise achieve great success in generating diverse patterns. However, noise as an important modeling tool and the requirement of controlling its appearance is still increasing. A weighted sum of kernels noise called sparse convolution noise is proposed since the seminar work of Lagae's Gabor noise [77].

Sparse convolution noises are based on the convolution of randomly distributed impulses with a spatial filter function (kernel). As the uniform random distribution of the kernel function result in white noise, then in the frequency domain control over sparse convolution noise is achieved by the kernel function. We will track the development of sparse convolution noise in this section.

Gabor noise

Gabor noise is introudced by Lagae [77] with accurate spectral control using gabor kernel, and provids a setup-free surface texturing method. Gabor kernel combined with sparse convolution to produce a band-limited anisotropic noise with accurate spectral control. As we mentioned the fourier transform of sparse convolution in spatial domain means the multiplication of kernel with a constant in the frequency domain. The main insight of the author is that the impulse should be parameterized and have compact support in the spatial domain to enable an efficient procedural evaluation, and also have compact support in the frequency domain to capable a precise control over the power spectrum (see Fig. 2.23). Then they design the gabor kernel as:

$$gabor\_kernel = ke^{-\pi a^2(x^2+y^2)}\cos[2\pi F_0(xcosw_0 + ysinw_0)] \quad (2.18)$$

and the corresponding fourier transform of the Gabor kernel as:

$$\mathscr{F}\{gabor\_kernel\} = \frac{K}{2\pi}\{e^{-\frac{\pi}{a^2}[(f_x - F_0\cos w_0)+(f_y - F_0\cos w_0)]} + e^{-\frac{\pi}{a^2}[(f_x - F_0\cos w_0)+(f_y - F_0\cos w_0)]}\} \quad (2.19)$$

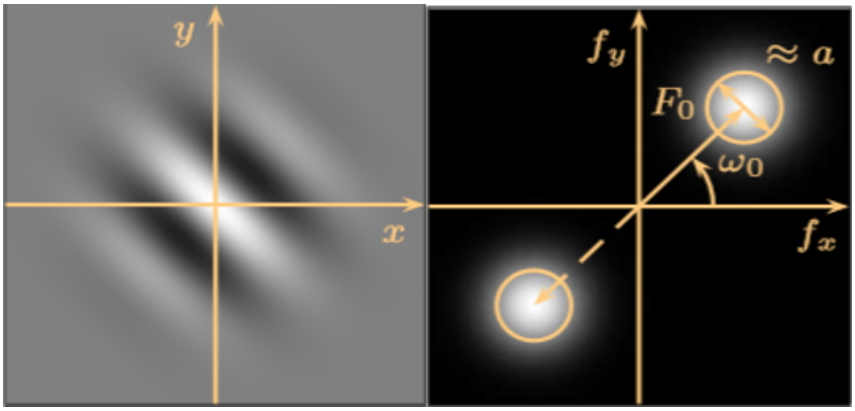


Figure 2.23: The left image is the spatial domain of the gabor kernel. The right image is the power spectrum of the gabor kernel with three parameter $\{F_0, w_0, a\}$ control the frequency, direction and bandwidth [77].

Then we can define the new band-limited anisotropic Gabor noise as a random pulse process convolution with the Gabor kernel as:

$$Gabor\_noise = \sum_i w_i g(x - x_i, y - y_i) \quad (2.20)$$

And the corresponding power spectrum is:

$$\mathscr{F}\{gabor\_noise\} = \lambda E\left[w^2\right]\mathscr{F}\{g(x - x_i, y - y_i)\} \quad (2.21)$$

The power spectrum of band-limited anisotropic noise is the power spectrum of the Gabor kernel scaled by a constant. So we can control the intuitive parameters of the Gabor kernel to control the power

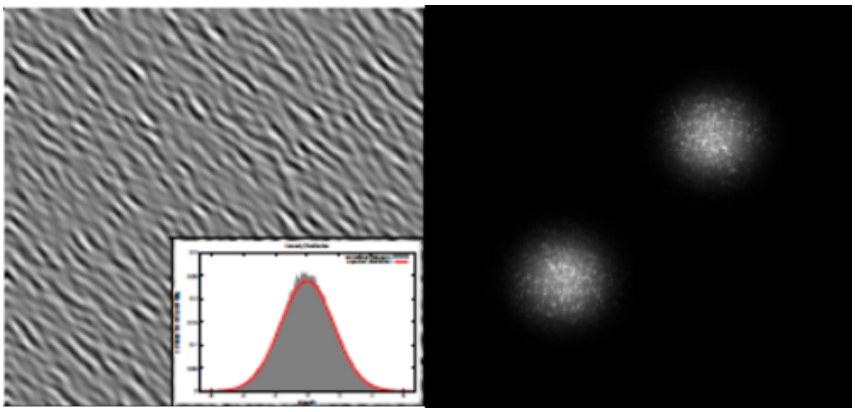


Figure 2.24: The left image is the spatial domain of the gabor noise, the right bottom is the corresponding gray histogram. The right image is the power spectrum of the gabor noise [77].

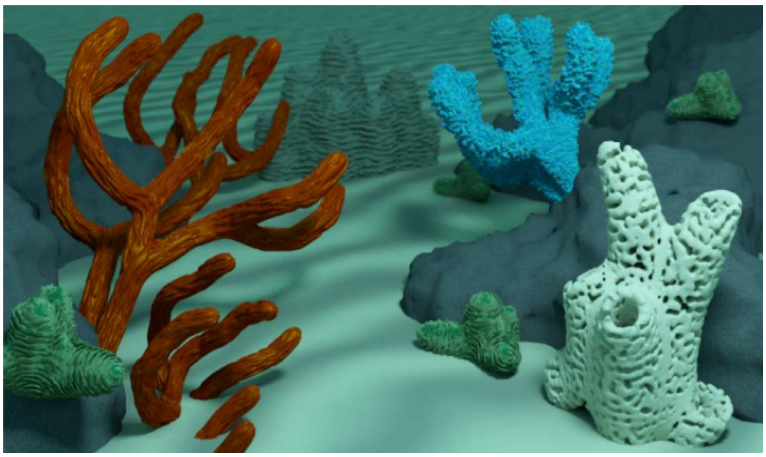


Figure 2.25: A coral reef was generated with skeleton implicit surface and the small scale details were generated from the surface gabor noise [166].

spectrum of the noise (see Fig. 2.24). For instance, Zanni [166] use the gabor noise to deform the implicit scalar field. In order to save computational time, the interval of the iso-surface ($[-\epsilon, +\epsilon]$) is defined the limit the range of scale field to be deformed. Their method is the first time to apply gabor noise over implicit primitives without causing the blur of the details (see Fig. 2.25). The noise can be generalized to arbitrary dimensions and applied to the surface without texture parameterization. However, the "real" environment is such a rich and variety place and the natural patterns are complex and endless. Design those textures by manually turning the parameter of gabor kernel in the power spectrum is cubersome and impractical. How can we approximate this variety of textures in an efficient and elegant way? Then the *gabor noise by example* algorithm was proposed in response to this question. The new algorithm of gabor noise significantly broaden the application of the procedural noise.

Gabor noise by example

The *Gabor noise by example* [46] algorithm generalize the gabor noise by synthesis particular pattern from a given *Gaussian texture*. This work significantly simplify the tedious work of the parameter design involved in the previous gabor noise algorithm. Their major contribution is a bandwidth-quantized Gabor noise which suitable for efficient procedural evaluation and a robust parameter estimation technique that automatically decomposes the exemplar's power spectrum.

The bandwidth-quantized Gabor noise is defined as:

$$bq\_gabor\_noise(p) = \sum_{b \epsilon B} \frac{1}{\sqrt{\lambda_b}} \sum_i \frac{1}{\sqrt{P_{b,i}}} gabor\_kernel(p - p_i) \quad (2.22)$$

The corresponding power spectrum is:

$$S_n(\xi) = \sum_{b \in B} \sum_{g=0}^{G_b - 1} \frac{K_{b,g}^2}{8a_b^2} \mathscr{F}\{gabor\_kernel\} \quad (2.23)$$

Then by solving the parameter estimation problem as Equation 2.24 we can obtain the relevant parameters of the bandwidth-quantized Gabor noise.

$$\begin{cases} minimize & \left\| S_n(\xi) - S_{exemplar} \right\|_2^2 + \nu \left\| \alpha \right\|_1 \\ subject\ to & \alpha \geq 0 \end{cases} \quad (2.24)$$

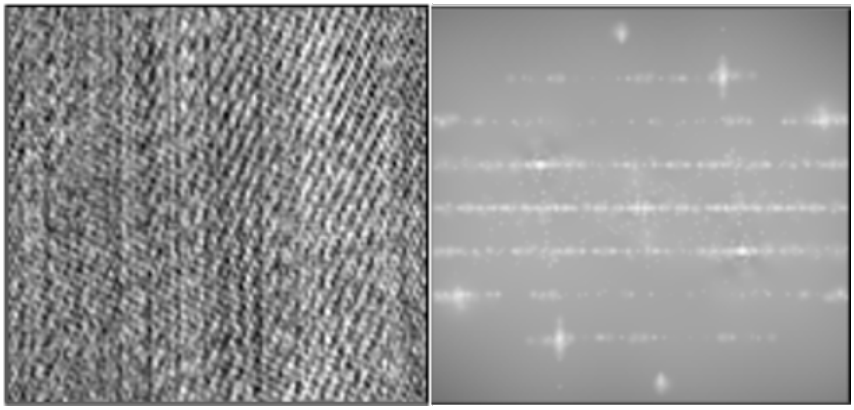


Figure 2.26: The left image is the exemplar texture and the right image is the power spectrum synthesized by the bandwidth-quantized gabor noise [46].

random-phase noise

The limitation of the *gabor noise by exmaple* algorithm is the corresponding patterns are gaussian textures, which only represent a small subset of textures with micro details. However, some textures with regular macro structures are important in lots of applications. Then Gilet [55] propsed *local random phase*(LRP) noise, aim at generating local noise

based on the regular spatial grid. It is a mixture of local noise with random phase (see Fig. 2.27) and defined as follows:

$$LRP\_noise_(p) = \sum_{i=1}^{I} w(\frac{\|p - p_i\|}{\Delta}) \sum_{j=1}^{J} A_{i,j} \cos(2\pi f_{i,j} \cdot p + \varphi_{i,j}) \quad (2.25)$$

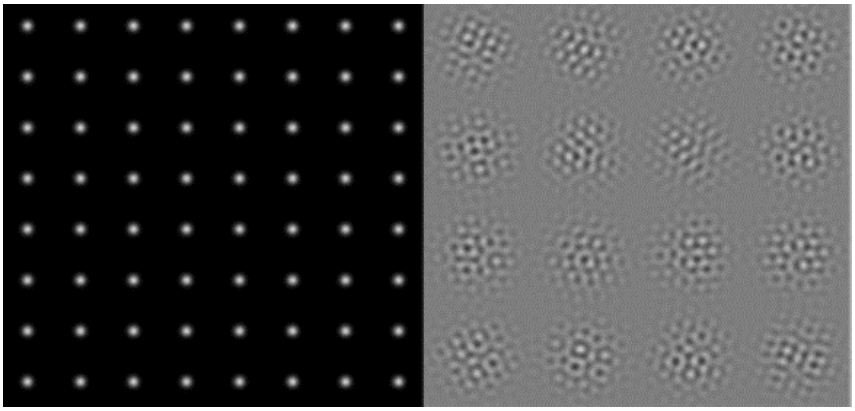


Figure 2.27: The left image is the spatial lattice and the right image is the local noise modulate on the lattice into the local random phase noise [55].

The locality is controlled by a decreasing window w of width Δ and the randomness comes from the parameter of random phase $\varphi_{i,j}$ and frequency $f_{i,j}$. And the total parameter control the spatial and frequency domain is (I, x_i, Δ, w). The corresponding power spectrum is defined as:

$$S_n(\xi) = \frac{\Delta^2}{2} \sum_{j=1}^{J} A_j e^{i2\pi\varphi_j}(\Delta(f - f_j)) + A_j e^{-i2\pi\varphi_j} w(\Delta(f + f_j)) \quad (2.26)$$

The remark is that the random phase is able to generate random features. We can fix the phase $\varphi_{i,j}$ and the amplitude A_j in the power spectrum to approximate the energy of a local region's power spectrum. As a result, the local region that contain the macro structure can be reproduced by the LRP noise, and this similar local pattern comes from the summation of multi-layers of LRP noise.

phasor noise

More recently, Tricard [150] proposed a phase noise to solve the problem of modeling highly contrasted patterns and is particularly suitable for reproducing the high quality microstructure details. For instance, the standard procedural noise is unable to directly control the local characteristics of the pattern (even local random phase noise the local pattern is a pure random process), i. e., contrast, sharpness, and the variation of neighborhood pixels. However, contrary to the previous procedural noise generate the scalar field of the noise value into particular pattern, this new procedural noise - phasor noise - aims at generate the phase field to be subsequently modulate into a periodic

function (i. e., sine\cose wave) (see Eq. 2.30). More specifically, the phasor noise provide a precise control over the profile, orientation and distribution of the produced stochastic patterns, while allowing to grade all these parameters spatially [150]. This well-designed phase field is such that, once fed into the periodic function, it will produces a oscillating field pattern with the predefined main frequency and contrast orientation [150]. The phasor noise is defined as:

$$phasor\_noise(x) = \phi(x) \tag{2.27}$$

Then the famous Gabor noise can be rewritten as a multiplication of intensity function $I(x)$ and periodic function - a clear separation of intensity function and harmonic modulation:

$$\begin{aligned} gabor\_noise(x) &= I(x)\sin(\phi(x)) \\ &= \sum_{j=1}^{n} a_j(x) sin(\phi_j(x)) \end{aligned} \tag{2.28}$$

where $a_j(x)$ is the gaussian weight function and $\phi_j(x) = F_j(x - x_j)u_j$, u is the direction of the kernel function. The phasors are $a_j(x)(cos(\phi_j(x)), sin(\phi_j(x)))$. The corresponding intensity function $I(x)$ and phasor noise $\phi(x)$ (the instantaneous phase of the gabor noise) is defined as:

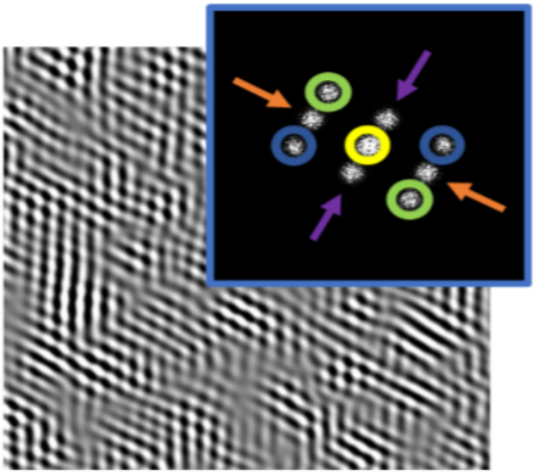
$$I(x) = \sqrt{(\sum_{j=1}^{n} a_j(x)\cos(\phi_j(x)))^2 + (\sum_{j=1}^{n} a_j(x)\sin(\phi_j(x)))^2} \tag{2.29}$$

$$\phi(x) = atan2(\sum_{j=1}^{n} a_j(x)\sin(\phi_j(x)), \sum_{j=1}^{n} a_j(x)\cos(\phi_j(x))) \tag{2.30}$$

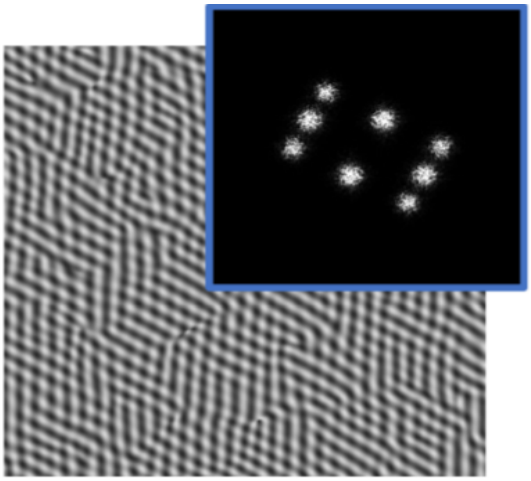
This separation of intensity and harmonic offers two benefits: First, it affords for the definition of a noise that perfectly oscillates without local loss of contrast; Second, we can control the shape of noise oscillations.

If we consider multi gabor kernels, i. e., two kernels, the corresponding gabor noise's variance of spectrum exhibits gaussians in nine locations (see Fig. 2.28a). The spectrum of variance is the spectrum of the squared signal. Five of these gaussians corresponds to the square and the other four origin from the intersection gaussian in two directions (the direction of the two kernels in phasor noise). If the combination of direction is inverse (opposite in sign) and will produce low contrast pattern (see Fig. 2.28a). As a result we can remove the center low frequency signal in the variance of spectrum to eliminate the loss of local contrast (see Fig. 2.28b).

In addition, the profile of each oscillation is directly controllable (i. e., sine wave, sawtooth, rectangular or any 1D profile) for the phasor noise. The shape of the oscillation profile is the consists of the periodic function without any undesirable fluctuations. This affords for a



(a)



(b)

Figure 2.28: Two bi-lobes noise.(a) Gabor noise and its spectrum of variance. The nine Gaussians located at the center Gaussian (in yellow), the bi-lobes Gaussians at $\pm 2u_k$ and $\pm 2u_m$ (in blue and green), and the interaction Gaussians at $\pm(2u_k + 2u_m)$ and $\pm(2u_k - 2u_m)$ (outlined in orange and purple arrows) [150]. (b) Remove the center Gaussians in the Gabor power spectrum of variance and obtained the normalized noise [150].

precise control over the shape of the produced stochastic structures. Its name originates from our reformulation, which exposes a sum of phasors within Gabor noise [150]. Profiles are synthesized as weighted sums of integer harmonics of the base pattern. Being able to orient synthesized patterns while controlling their profile and distribution is especially well suited for the definition of patterns such as hatches, stripes, cracks, ridges, scales and ripples [150].

In short, the advantage of a procedural noise function are the following:

1. a procedural noise function is extremely compact, normally requires a few kilobytes or even less of memory space compared to megabytes of texture images or volumes

2. a procedural noise function is intrinsically continuous, multi-resolution, and not relies on sampled data. So it can produce noise value at any resolution ranging from a distance overview to extremely close inspection,

3. a procedural noise function is non-periodic, able to fill the entire space ranging from 2D, 3D to nD. In other words, it is unlimited in extent and able to cover an arbitrary size of area without seams or unwanted repetition,

4. a procedural noise function is controllable, so it can generate a class of patterns rather then one particular noise pattern. The parameter of the kernel controls the power spectrum of the noise, which characterizes the noise pattern,

5. a procedural noise function is randomly accessible in constant time. That means, it can be evaluated regardless of the location and the previous evaluation. This random accessibility and independent evaluation makes the procedural noise function suitable to harness the power of multi GPUs or multicore CPUs.

Procedural textures offered by the volumetric noise (3D procedural noise) is extremely useful in situations where an impression or effects of natural-looking materials on a manifold surface is required, regardless of the traditional necessities of creating an explicit texture image. Also, rather than exploring how to figure out a mapping from image textures onto the manifold surface, the volumetric nature of noise-based procedural textures allows the evaluation at any locations. In this way, it is possible to effectively carve out the desired pattern on the solid material, which is much more straightforward than work out a reasonable undistorted parametric mapping. Noise-based procedural textures allows the evaluation in a resolution-independent way: it can be kept crisp and detailed (i. e., adding higher frequencies of noise components) when you get nearer to an object rather than the effects of blurring out which occurs in the traditional image-based textures. One limitation of the procedural textures is that you should try to avoid adding very high frequencies which exceed the pixel's sampling rate. Because such super-pixel frequencies do not contribute to the visual quality but result in the unwanted speckling when the texture animates. The key to use noise-based textures efficiently is to implement the noise function on GPUs which really makes use of the parallelizability of the procedural noise.

These sorts of noise-based procedural textures have long been the mainstay in the film industry, where their execution do without real-time frame rates, yet do require high fidelity to the visual effect of many natural materials. In fact, nearly all special-effects in the films of today more or less make use of noise-based procedural textures. For instance, the convincing effects of the ocean waves in "A Perfect Storm" [147], and the atmosphere effects in "The Lord of the Rings" trilogy [146], are just two interesting samples. Both films are highly dependent on the extensive use of the noise function within shaders where written by the language such as Pixar's RenderMan [121].

2.4 VISUALIZATION OF IMPLICIT SURFACE

The task to visualize implicit surfaces has always been considered challenging due to the implicit characteristic of the surface. Broadly, there are two major methods for their visualization: they may be rendered either sampling on the implicit surface and converting the sampling discrete points into meshes from polygonization algorithms, or directly ray-tracing the implicit surface and shading the intersection points. The major problem of the polygonization based rendering algorithm is: they probably can not precisely detect the correct topology or miss the fine surface details. However, for the case of ray tracing, once the field function are defined, the ultimate field function must be used to locate the surface, and the complex shape can be rendered at any resolutions. Moreover, the research of rendering implicit surfaces also present the solution to visualize the scientific or medical dataset. In particular, the implicit surface's visualization algorithms are well suited to depict the data collected from 3D or laser scans (i. e., computed tomography (CT), point cloud and magnetic resonance imaging (MRI)) and digitizing sophisticated objects.

2.4.1 *Polygonization*

The main focus of this section lies on the polygonization techniques used in the implicit surface visualization. The most general and popular research on the polygonization of implicit surface is concerned about presenting fast and accurate meshing methods.

Empirically, based on the sampling strategies they use, methods to polygonize the implicit surface can be classified into four categories: regular grid based methods such as marching-cubes, propagation methods such as marching triangles, cage-based methods and particle systems [164]. These space sampling strategies are able to convert the continuous mathematical descriptive surface into a discrete linear piecewise approximation. In this subsection, we only cover the most typical and suitable method - marching cube - here which is directly related to the subsequent Chapters.

Marching cube was proposed in 1987 by Lorensen [88] and soon became one of the most popular polygonization techniques. Marching cube was initially invented oriented toward volume data visualization rather than implicit surface polygonization, and immediately gained popularity in that area. In the area to visualize the implicit surface, usually we will build a uniform voxel grid and define a field function on each grid point to get an iso-value. Then the marching cube algorithm allows us to generate the correct polygons within the uniform voxel grid, from the given iso-values at its corners. As output, the algorithm will produce zero to five polygons in each voxel (see Fig. 2.29). In the case that the iso-value at the eight corners of a voxel have the same sign (positive or negative), then the voxel is completely inside or outside the manifold implicit surface, and no polygons will export. In all other cases, the polygon lies on the boundary, and will produce

one to five polygons. More specifically, for any point p in 3D space, the field function $f(p)$ will produce a scalar floating-point value. The demarcation point between positive and negative corners - where the field value is zero - is the explicit surface of the implicit surface. It is along this surface the polygonal mesh we wish to construct will produce.

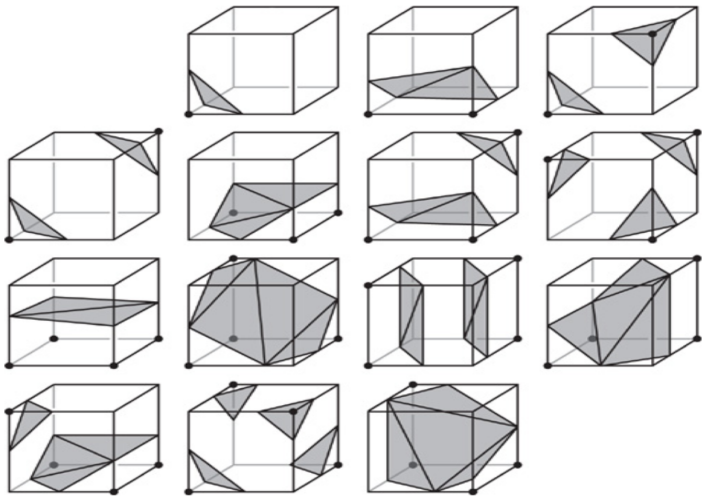


Figure 2.29: The 14 fundamental cases for marching cubes [100].

More recently, the parallel version of marching cube is required for many real-time applications. In order to harness the parallel power of GPU, we can subdivide the space into blocks and in each block we further partition it into even smaller voxels (also called cells). It is within these voxels that we will construct polygons (triangles) which represent the implicit surface. By launching each corner one thread to compute the field value on the GPU, we can compute the iso-value of all blocks simultaneously. Moreover, we can logically joint these eight corner inside/outside bits to produce a byte in the range of $0 \sim 255$ to represent the case of triangles (from zero to five). If the byte value is 0 or 255, then the voxel is entirely inside or outside the surface and, as previously mentioned, no triangles will export. However, if the byte value is in the range of $1 \sim 254$, then $1 \sim 5$ triangles will be generated. Then the byte value is used to index into a lookup table (on the GPU, the table can be preserved in a constant buffer) to determine how many polygons will export for that case and how to build them (index of the vertices order). Each triangle is created by connecting three points (vertices) that lies on the edges of the voxel. More specifically, how to place the vertex of each triangle is determined by the interpolation between the corner point. The output is a sequential list of vertices, every three vertices will produce a triangle and the fourth vertex is the beginning of a new one. Two lookup tables are employed: the first table is indexed by the byte value and tells us how many triangles are needed; the second lookup table is much larger and offers the information required to build up the zero to five triangles within each voxel.

One limitation of marching cube algorithm is the ambiguity in the lookup table and Nielson [101] proposed the dual marching cube to

eliminate this problem. In addition, spatial partition or spatial decomposition techniques are also applied to accelerate the marching cube algorithm. For instance, Velho [153] present a unified and general tessellation algorithm for parametric as well as implicit surface. It generates an adaptive mesh from the controlled subdivision surface. The algorithm starts by building a simplified uniform spatial decomposition to create a coarse triangulation of the surface. Afterward, a refinement step is performed, sampling the edge of the coarse triangles and subdivided the edge to better approximate the shape. Paiva [104] present an algorithm to generate an adaptive mesh which captures the exact topology of the implicit surface. The algorithm use the internal analysis of the implicit value and its gradient to construct three subdivision criteria to build an octree and accelerated the tessellation process.

2.4.2 *Ray-tracing*

As we mentioned the implicit surface is defined as a field function with isovalue c (see Chapter. 2.1). The ray-tracing algorithm visualize the implicit surface by shotting a ray and evaluate the intersection point which enables implicit surfaces to be shaded directly from the field function. This is a big advantage over the traditional method to tessellate the surface into triangles or polygons. This process can be defined as follows:

$$\begin{aligned} &Ray : ray(t) = \vec{O} + t\vec{D} \\ &Field\ function : F(d) = f(\vec{O}, \vec{D}, t) = c \end{aligned} \tag{2.31}$$

where $\vec{O}$ and $\vec{D}$ represent the origin and direction of the ray and we can solve (finding the root of $f(\vec{O}, \vec{D}, t) = c$) the Equation 2.31 for the smallest t (> 0). There are a few method to do this such as polynomial root solving, interval analysis and Lipschitz methods.

Polynomial root solving

For the low degree of polynomial in Equation 2.31, it's possible to find all roots for the degree of two (quadrics), three (cubics) and four (quartics) polynomials in a symbolic method. However, in general the even higher polynomials requires numerical method. More recently, Nishita et al. [103] use Beźier functions to represent the field function along the ray direction and lower the computation. And meanwhile, they have integrated the Beźier clipping to further accelerate the intersection test. This method was also adapted to GPU with an optimization method in [69] and showed vast potential in the particle-based simulation system (see Fig. 2.30).

The major disadvantage of polynomial method is their low efficiency, because the result always returns all roots include positive, negative or complex pairs. Usually, only the smallest positive real result works for the ray intersection test and the other computation is vanished.

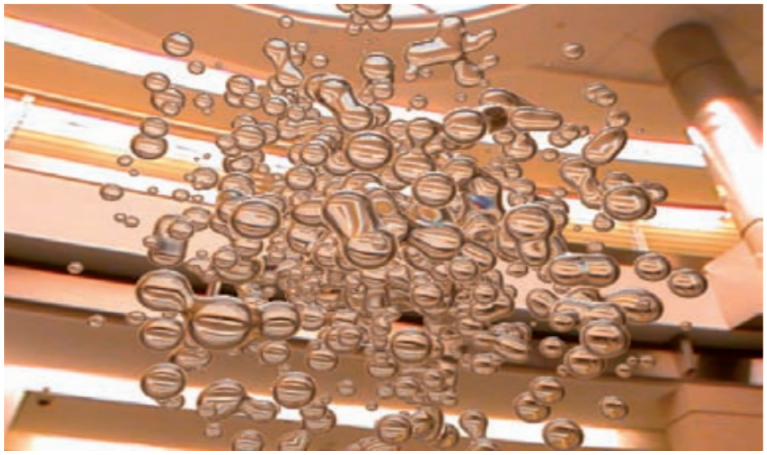


Figure 2.30: Real-time animation with 1000 moving metaballs [69].

Interval analysis

Interval analysis originated as a method to solve the general numerical analysis problems, and [98] was the first to introduce this method into the evaluation of the intersection between a ray and an implicit surface. For instance, they defined an interval $[a, b]$ to represent the range of an ordered pair. Then the closed arithmetic operators can be defined on the interval. After that, they defined the value domain of the function F and its derivative F' equals to two particular intervals. If zero is not inside the value domain (two intervals) of F and F' means the function F is monotonic and the root not exist in the definition interval. Then we can do a divide and conquer operator which subdivide the definition interval at midpoint to repeat the previous step again [98]. Otherwise, the definition interval of the function F contains one single root and can be solved by Newton's law. More recently, [71] optimized the interval and affine arithmetic method and achieve interactive ray tracing of arbitrary form implicit surfaces on CPU and GPU.

Lipschitz methods

Let F be a field function defining an implicit surface. Then if F satisfied the Equation 2.32 and existing some positive constant λ, then we can say that the implicit surface has the Lipschitz property [133].

$$|F(x) - F(y)| \leq \lambda \|x - y\| \tag{2.32}$$

Then we denote the Lip f as the Lipschitz constant only if the smallest λ satisfies (Eq. 2.32). The Lipschitz constant eliminates the influence of steeper area and makes the Lipschitz method the most efficient algorithm.

The Lipschitz bound property is a robust method for ray tracing implicit surface and attracted lots of research interests. Hart [60] introduced the sphere tracing that move forward with the ray to its first intersection, and the step length is guaranteed by the Lipschitz criterion to not penetrate the implicit surface. More recently, Genevaux [54] proposed a novel accelerated version of the sphere tracing algorithm

Figure 2.31: Procedural terrain composed with other terrain primitives are rendered with accelerated sphere tracing algorithm [54].

to compute the intersection between a procedural terrain (its height field defined by a construction tree) and a ray (see Fig. 2.31). Seyb [133] proposed another advance of sphere tracing for directly shading the deformed algebraic surface to ease the computation as well as the complexity of global Lipschitz bound estimation and achieved high performance. Galin [48] extended the previous work by introducing a Segment Ray Tracing through computing the Lipschitz bound λ and also explains the way to compute those bounds for a variety of geometric primitives and operators.

In theory, the intersection testing almost always benefit from the preprocessing step on spatial partitioning. Generally, this step will subdivide the space into regular or irregular size where different data structure can be applied. For instance, the BSP-tree, Kd-tree or Octree [60] were applied to render the implicit surface. Those structures significantly reduced the query times of the field function in the ray-tracing process, which is the most computationally intensive part. However, the main flaw of this spatial partition method is that it possibly miss the fine details on the implicit surface.

2.5 APPLICATIONS OF IMPLICIT SURFACE

Implicit surfaces have been used in various kinds of applications from realistic terrain generation to sculpting characters, we will present some of them in this subsection.

2.5.1 *Procedural Terrain*

Virtual terrain are the dominant visual element in many applications, from real-time simulations to games or movies. Consequently, there exists lots of methods to generate terrain procedurally. Here we only cover the method most relevant to this thesis - the volumetric procedural terrain generation.

The research of terrain generation has a long history due to its wide applications, but how to effectively representing and generating true 3D landscape remains an unsolved problem. This is because most solutions address the problem of heightfield (2.5D) based terrain quite well, however, truely 3D terrain are not frequently present in the literature. Until the last decade due to the fast development of GPU

and the voxel based volumetric procedural method regains popularity which lead to the prosperous of true 3D terrain generation.

Gamito [50] introduced a novel procedural method by deforming the initial heightfield with a vector field. The heightfield is defined as $f : R^3 \rightarrow R$ and by warping the space along the vertical dimension their method is able to generate landscapes with overhangs. The novel deformation define the elevation h as a function of point in the domain $\Omega \subset R^3$ instead of traditional two-dimensional domain, the warped surface of the landscape is therefore procedurally defined as [49]:

$$S = \left\{ p \in R^3 \mid f \cdot w^{-1}(p) = 0 \right\}$$

The warping function $w^{-1} : R^3 \rightarrow R$ deforms the discrete height space and smooth the height jump between the point on the plane with a 3D perlin noise into a concave surface resembling strata of rocks or overhangs. In practice, the warping function can be defined as a procedural displacement function based on a sum of octaves of 3D perlin noise (Fractal Brownian Motion, fBm), and smoothly clamped to a given region of influence in the terrain space.

The first volumetric terrain algorithm running on the GPU is proposed by Geiss [100] to generate complex procedural terrain on the infinite blocks of regular grid points. Conceptually, their terrain surface is a fully implicit function. For any point $p(x, y, z)$ on the voxel grid, the implicit function (density function) produces a height value and a marching cube algorithm is used to visualize the implicit surface (0-isovalue) on-the-fly. Moreover, with the deformation operator (warpping) their volumetric terrain is able to generate diverse overhangs or cliffs from the 2.5D heightfield (see Fig 2.32). This method can be fully implemented on the GPU by computing all voxels in parallel, therefore allowing the interactive terrain authoring. The only limitation on the number of voxels is the memory size of GPU.

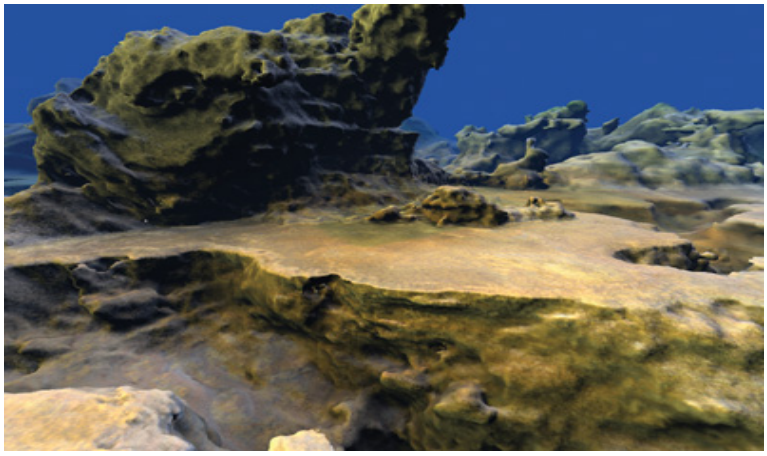


Figure 2.32: Procedural volumetric terrain generate on the GPU at real-time frame rates [100].

Peytavie [117] present a hybrid terrain model that accumulate multi-layer terrain primitives (materials, sand, air, water and so on) into a complex terrain with overhangs, arches and caves. Each terrain primitive is defined as a convolution surface and a stabilization parameter

is used to uniform the action of different layers of materials to be overlaid. The ultimate complex rocky scene is piled free of computationally expensive physically-based simulation [117]. After that, they also provide a high-level tool to help user authoring more complex scene, e.g. locally deforming and carving terrain primitives, or even sculpting the bed rock or large canyons (see Fig 2.33).

Figure 2.33: Arches and caves are generated with different materials from the hybrid terrain model [117].

More recently, Paris [108] proposed a framework to automatically authoring complex realistic terrain primitives on the existing heightfield terrain (see Fig. 2.34). Their amplification workflow works as follows: firstly, for a given heightfield H we can evaluate an approximation implicit surface T, then the 3D terrain primitives (i. e., slot canyons, sea arches) can be freely composed into the implicit surface T through a construction tree. The internal nodes of the construction tree are either binary operator or sub construction tree and the leaves are implicit 3D terrain primitives. Secondly, T is defined as the bedrock of the terrain and managed by a geology construction tree G which defines the property of the bedrock in the form of strata and fault lines [108]. In the construction tree G we can define and edit the area of interest on the bedrock T with 3D terrain primitives or erosion effect from the bedrock resistance $\rho(p)$. Ultimately, an implicit surface based sparse landform construction tree which supports the compact encoding of local 3D terrains primitives are obtained. In the end, an optimized marching cube polygonization method is proposed to speed up the generation of the mesh to visualize the characteristics of the implicit surface. As a result, this terrain model support the editing and simulation of realistic terrain at interactive cycles.

2.5.2 *Animation*

Physically-based cloth animation is prevalent in the field of computer graphics and attracted lots of interdiscipline researchers. In order to fulfil a high quality dynamics for the individual part of cloth, several challenges are needed to be considered. For instance, physically-based cloth model is the first mandatory requirement for the correct animation of cloth's movement. Then efficient collision detection and response algorithms are needed to assure the behave

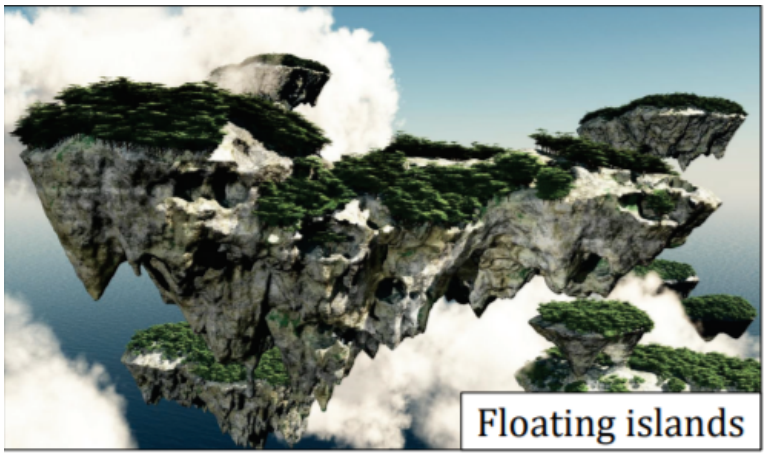
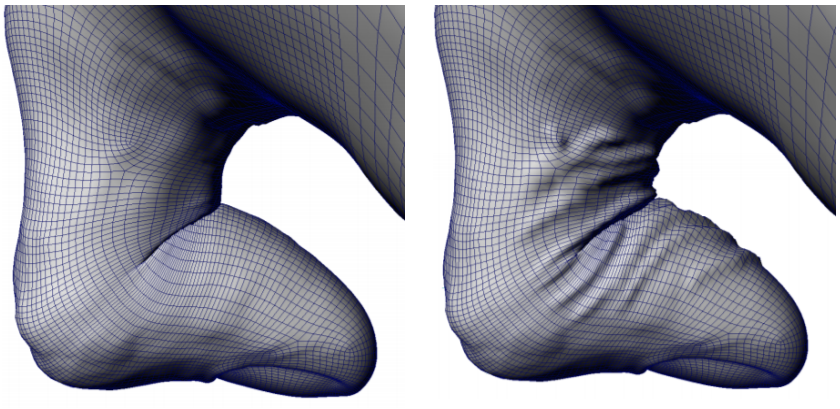


Figure 2.34: The floating islands were created by combining implicitized terrain H with the erosion operator on the construction tree automatically [108].

of cloth as expected. Implicit surface is qualified in both aspects and becomes a promising candidate. Buffet [19] use the implicit surface to approximate the 3D garment and this implicit representation enables a multi-layer of garment to compute collision detection in a constant time. Moreover, the global field value of implicit surface further helps the evaluation of the penetration depth between different layers of the garment. In addition, the penetration depth can also be reused to calculate the deformation amount.

Skinning of virtual character as another important topic in computer animation gained lots of research in the past decades. In [14], the first skinning system use the implicit skeleton to represent the character, and effectively eliminate the artifact around the skeleton joints during the animation. Subsequently, Vaillant [152] improved the skinning system through Hermite Radial Basis Functions (HRBF) to approximate the individual parts (skeletons), and a distinct blending operator is used to consist those parts (primitives). The new parameter surface enables an experimentally parameter setting of the contact property between skin parts in a interactive session, without relying on the actual collision detection step. During animation, the implicit primitives (parameter surface) associated with the character are rigidly transformed and combined, result in a smooth distorted time-dependent implicit volume [152]. The time-dependent property means continuous frames can be evaluated in parallel. As a result, their skinning system is able to generate plausible skin deformations and suitable for real-time animation applications (see Fig. 2.35a). More recently, Turchet [151] extended the previous implicit skinning system with wrinkles which are important visual effects on the surface of deformable objects. The wrinkle curves are invented as a set of line-segments blends into the convolution surface and sum up their contributions into the final field value. The new convolution surface can be easily integrated into the previous implicit skinning system and each line-segment is smoothly blended (see Fig.2.35b).



(a) (b)

Figure 2.35: (a) Implicit skinning [152].(b) Wrinkles on the implicit skinning surface [151].

2.5.3 *Additive Manufacturing*

3D models plays an important role not only in the virtual environment construction but also the modern product fabrication, and continuous enhancing its status as additive manufacturing (AM) technologies are gaining popularity in industrial practice. The most notably advantage of AM is that fabrication cost is mainly determined by the amount of material consumed (nearly no material loss) and independent of the object's complexity. More specifically, additive manufacturing enables the fabrication of objects with unprecedented complexity of interior as well as exterior structures. This capability is easy to understand in terms of global shape and topology optimization and interior microstructure optimization.

Implicit modeling tools, such as field function, blending operators, helps built a watertight volume and able to integrate with physical-based simulation by adding a few constraints (i. e., maximizing a particular object's strength under particular loads while limiting its weight) to create a low-stress design of the global shape effectively [105]. However, the tradition approach is based on the intuitive that eliminating sharp concave corners improves stress behavior, and engineers tend to make specific choices of geometry based on prior experience or just a trial and error method. In contrast, the physical constraint [162] is essential for many mechanical design problems, as it is able to restrict the load stress evenly distributing on objects and detect high stress concentration areas (see Fig. 2.36). In addition, the implicit representation is 3D printer-friendly which allows arbitrary number of high-resolution slices to be generated through the math equation over the classical triangulated mesh format (i. e., STL file).

Moreover, it is also possible for implicit method to fabricate object filled with microstructures - containing intricate internal details and can be evaluated when needed. The first challenge is the microstructurs has to produce the desired large scale rigid behavior. The direct benefit of the microstructure is that the object is lighter and remain-

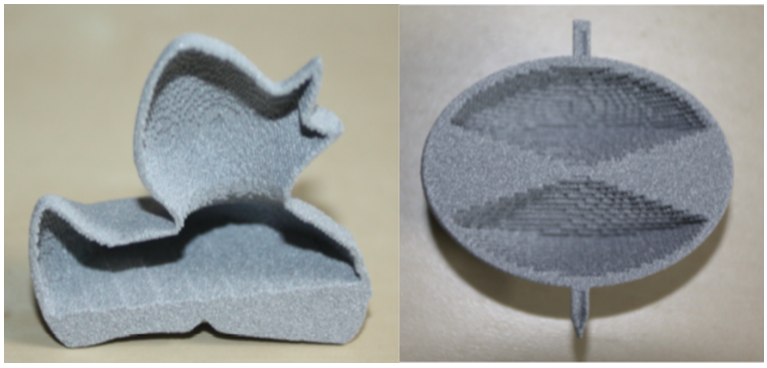


Figure 2.36: 3D printing objects with different mass distribution [162].

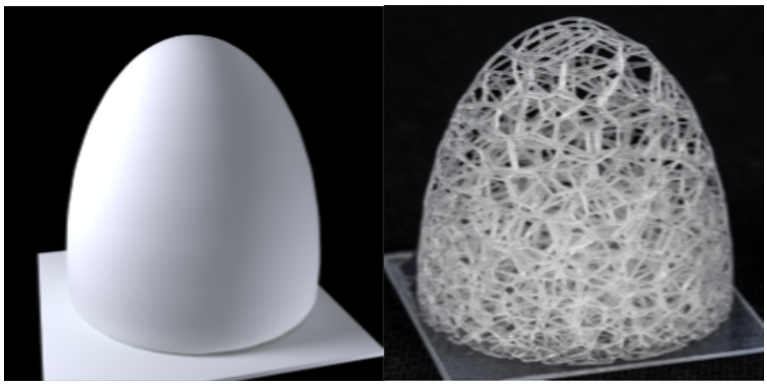


Figure 2.37: A simple graded material applied to a ellipsoid 3D model with inner micro-structure [93].

ing rigid which can ease the consuming of materials, shipping and transportation simultaneously (see Fig. 2.37). The second challenge is the optimal design - minimizing the global stress norm under a set of constraints. However the implicit surface tend to be smooth, "organic" free-form shapes, and usually small variations on the surface will result in significant changes in local stress [93]. As a result, the inner complex structures may progressively vary across the object and meanwhile subject to different mechanical requirements between regions under different local stresses.

Procedural noises is a promising technology to overcome those challenges, where infinite amount of repeated patterns are produced at low, constant memory cost while precise manipulation of the statistical properties of the pattern is achieved. This hints the possibility to generate procedural, stochastic microstructures that directly exhibit the desired rigid property, without further optimization [93]. Tricard [150] proposed a new procedural phasor noise which can be applied to synthesis multi-material patterns for 3D printing (see Fig. 2.38). Their phasor noise has several advantages: firstly, phasor noise broaden the pattern of procedural noise able to deliver. For instance, the new noise introduced a profile function which make it possible to precisely control the properties of spatially vary materials for 3D printing. Consequently, some special materials who have exotic properties, such as negative Poisson's ratio, can be achieved on a single-material printer through the microstructure. This is especially interesting to grade

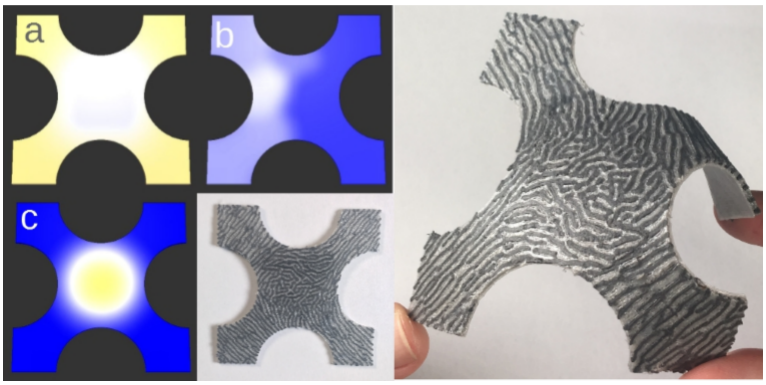


Figure 2.38: **a** ratios of material, **b** orientation and **c** isotropy.The designer control the three fields on the left and use a mixture type of material to print a cross shape plate [150].

materials which have anisotropy effects. Secondly, the patterns can be evaluated very efficiently. This characteristic make it possible for the designer to check the small-scale structures beforehand and further fine tune the parameter of the deformation behavior to achieve more effective design before printing. Also, the structure is very small compared to the size of the objects. Therefore, the traditional rasterization of mesh becomes quickly prohibitively large structures, exerting a bottleneck for fabrication and optimization. In contrast, the procedural noise is continuous and achieves arbitrary resolution on demand to avoid the equally rasterization of the object. Thirdly, the virtual grid affords for complete freedom in orientation. For instance, when we seek to design a multi-material pattern akin to laminates, with the precise control over the direction of material we can acquire unprecedented effects (see Fig. 2.38). In particular, when we focus on optimizing microstructures: assemble periodic tiled small cells with free form orientation can significantly helps produces a particular averaged (homogenized) elastic behavior [105]. We have described a set of applications of procedural algorithm in producing optimized global shape as well as microstructures of the object that cover an extensive range of research on 3D additive manufacturing, such as isotropic/anisotropic materials, minimizing stress concentrations, to give an overview about the advantage of procedural method. In the future procedural method is possible to play an even more important position in all areas of additive manufacturing.

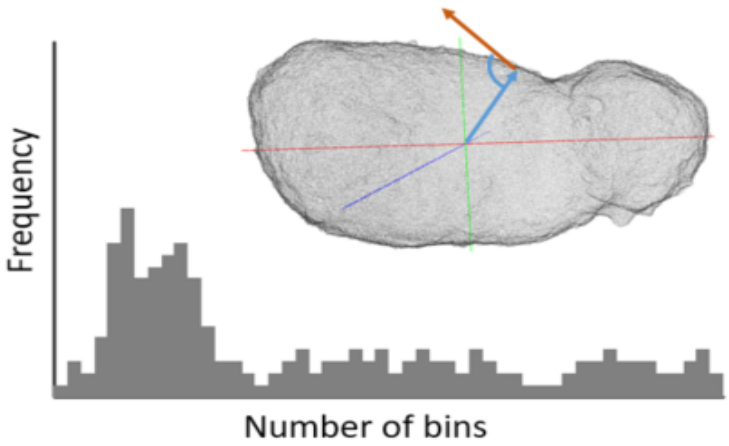
Part II

NOVEL METHODS

3

3D ASTEROID CLASSIFICATION

In this chapter, we will present a novel statistical shape descriptor for arbitrary three-dimensional shapes as a six-dimensional feature vector for generic classification purposes. Our feature vector parameterizes the complete geometrical relation of the global shape and additionally considers local dissimilarities while being invariant to the shape's translation, rotation, scaling. Our approach allows the classification of large-scale shapes even with only small local dissimilarities. Our feature vector can be easily quantized and mapped into a histogram, which can be used for efficient and effective classification. We take advantage of GPU processing in order to efficiently compute our invariant local shape descriptor feature vector even for large-scale shapes. Our synthetic benchmarks show that our approach outperforms state-of-the-art methods for local shape dissimilarity classification. In general, it yields robust and promising recognition rates even for noisy data.



3.1 INTRODUCTION

Robust scene interpretation in the form of detection and classification of previously known 3D objects in arbitrary scenes is a key factor in various computer vision approaches. Efficient and smart shape descriptors are fundamental to object detection and classification. According to Wahl et al. [155] and Rusu et al. [126] such shape representation have to be

1. compact,
2. robust,
3. they should be invariant, i. e. not depending on a global coordinate frame, and they

4. should have the descriptive capacity to distinguish arbitrary shapes.

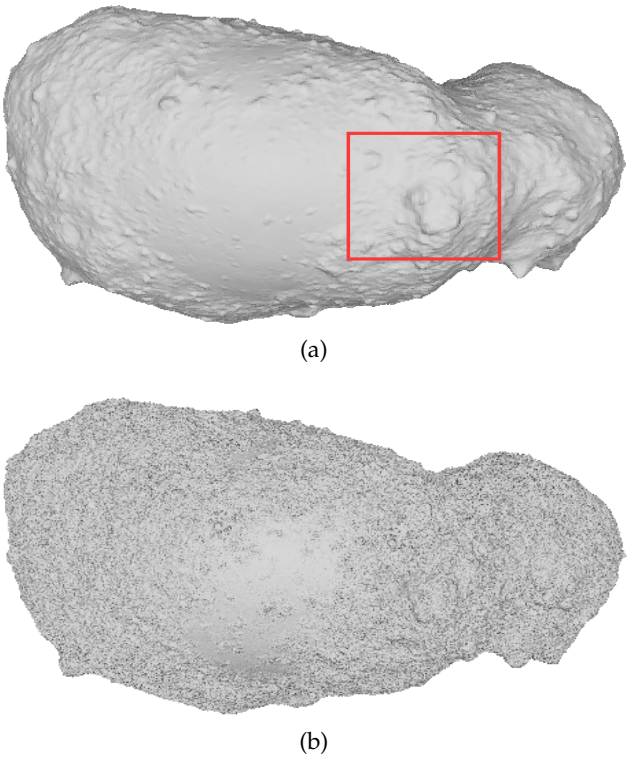
Often, i. e. in robotics, the object detection and classification is done on point cloud streams. Due to a continuously improving 3D sensing technology (stereo systems, laser scanner, or consumer electronics such as the Kinect), these point clouds do not only become larger but additionally contain more details. Consequently, it is necessary that object detection and classification approaches adapt to the increased (local) 3D object detail. Hence, we identified the following additional challenges for such kinds of shape descriptors:

5. they should consider small local dissimilarities,
6. their computation should be manageable to handle also large-scale shapes (i. e., shapes consisting millions of polygons/vertices).

Wahl et al.'s work effectively solve the first four requirements by creating a surflet-pair histogram to represent the shape of 3D objects and matching histograms with KL divergence [62]. However, their approach fails to deal with the small locally dissimilar and large-scale computing problems. The main reason for the performance issues are the sequential computation of the histograms and the usage of the statistical KL method for the classification. On the other hand, Zhang et al. [171] already showed that the GPU can be applied to accelerate the 3D object retrieval process. Moreover, machine learning algorithms have become a very popular and powerful tool for 2D or 3D object classification and regression problems, especially for large, high-dimension histograms. In this chapter we will present a substantial extension to the approach by Wahl. In detail, our contributions are:

- a novel *local* feature that considers small local characteristics of the object,
- a parallelization of the histogram computation,
- a machine-learning-based classification algorithm that can handle large-scale shapes.

Our approach is invariant to translation, rotation and scale of the shapes and moreover, it is robust to noise. We have implemented our algorithm using CUDA that allows it to completely run on the GPU. As a use case scenario we chose the classification of 3D asteroids from point cloud data. This scenario is typical for large-scale objects with local dissimilarities and is currently discovered in spacecraft operation studies for autonomous landing [119]. Additionally, we evaluated regular objects from the NTU database. We compared our approach to several state-of-the-art shape descriptors. Our results show that our approach is capable of classifying both large-scale shapes with local dissimilarities based on their local statistical properties and standard shapes based on their global and local dissimilarities efficiently.



(a)

(b)

Figure 3.1: The dissimilarity of different resolution 3D models of Itokawa.

3.2 RELATED WORK

The goal of shape-based 3D object classification is to formulate shape properties which accurately represent the object. This shape descriptors are used to efficiently classify them while focusing on (1) the compactness of shape descriptors and (2) the robustness of shape descriptors. A detailed overview of shape descriptors for shape-based object classification can be found in [82, 142]. Here, we give an overview of relevant approaches which are directly related to our work.

In early research, Bay [6] established the SURF detector, and [89] proposed SIFT using local invariant descriptors. More recent approaches, e.g. from [27, 32, 42, 87, 155] focused on new shape descriptors. Namely, [155] introduced a four- dimensional global shape descriptor. They defined a 3D coordinate frame for each pair of oriented points (so-called surflet-pairs), and defined a four-dimensional, pose invariant shape descriptor, which describes these surflets. [27] proposed the view-based global shape descriptors Light Field descriptor (LFD) which aims at describing 3D models by a set of two-dimensional representations. In contrast, [42] introduced two-dimensional spherical harmonics based shape descriptors. This approach does not contain a sophisticated classification scheme because the similarity between two shapes is calculated by the Euclidean between two spherical harmonic descriptors. Lo and Siebert [87] proposed Trift which extended the idea of SIFT from 2D image to 2.5D domain. Their idea is to fusion the histogram of range surface topology types with the histogram of the range gradient orientations to form a new feature descriptor.

Another spectral-based shape analysis method called shapeDNA [122] was proposed by Reuter. Their method extracts fingerprints of an arbitrary surface by taking the eigenvalues of its respective Laplace-Beltrami operator. This is the basis for a series of shape descriptors based on such Laplace-Beltrami opearators, such as the wave kernel signature (WKS) [3] or the scale-invariant heat kernel signature (SIHKS) [72].

More recently, shape-based 3D object retrieval algorithm is extensively applied from medical image classification to robot navigation. Due to the success of Convolution Neural Network (CNN) [125] in image retrieval task, numerous cutting edge deep-learning approaches have been transferred into 3D object retrieval domain which significantly boost the performance over traditional shape descriptor methods. However, as is well-known that training such neural network requires massive amount of training data which leads to the development of large-scale repositories of 3D shapes (i. e., shapeNet [26] which consists of more than 50 thousand unique models spread over 55 common object categories) contains much bigger datasets to help develop and evaluate new algorithms.

3D object shape retrieval is benefiting from the recent progress in deep learning methods and based on the method to represent the 3D object we can divide those approaches mainly into multi-projected views [61], point sets [174] (point clouds), 3D voxel grids [132] and traditional shape descriptors [44] groups. The key issue of 3D object retrieval is to construct a particular shape representation which enables distinguishing from different classes and aggregated within the same category. Among these groups, view-based projection methods are the most popular group because they are easy to utilize the additional image information to help learn features for 3D shapes. In view-based 3D shape retrieval, images are obtained by first projected 3D shape from different viewpoints, and then those images are dropped into CNNs to obtain the discriminative shape representation [61]. As a result, the view-based projected approach outperforms on the normal, consistently aligned 3D model's retrieval tasks. However, in the case of unknown and randomly oriented 3D models the point set-based approach which uses local invariant features works better than the view-based approaches [174]. Moreover, when adding the normal 3D model dataset with the perturbed (un-aligned 3D model) noisy data we can notice that, as expected, there is a decline of retrieval performance for all methods, however, the sinking rate is much less for point-set and voxel grid based methods [132, 174].

These approaches indicate that neural network outperforms traditional non-learning 3D shape descriptors approaches. An promising direction for future work is to consider integrating the discriminative power of view-based approaches and the robustness to arbitrary orientation exhibited by i. e., point-set or voxel based methods which use the locally geometry properties. There is still much space for improvement using mixture of view-based, point-set based and volumetric based methods together to handle more challenging tasks of 3D object retrieval.

The approaches mentioned above trace the development on 3D shape analysis from early general shape description to recent spectral shape analysis. However, none of them considers local shape dissimilarities of large-scale objects and and moreover, they are susceptible to noise that often appears in point clouds.

3.3 OUR DESCRIPTORS

Wahl et al.'s four-dimensional geometric feature is the basis of our novel six-dimensional geometric feature. Basically, they proposed surflet-pair histograms to describe the shape of 3D objects. We start with a short recap of this approach.

3.3.1 *Recap: Surflet-Pair-Relation Histograms*

An important advantage of Wahl et al.'s approach is its transformational invariance. In order to realize this, they introduced a canonical coordinate system by extracting features $\vec{U}, \vec{V}, \vec{W}$ (see Fig. 3.2a) as a transformation independent reference. They defined a canonical coordinate system as follows:

1. Extract the whole pairwise points and its normals from the surface mesh of object $(p_i, \vec{n}_i)$. Randomly select surflet-pairs $(p_i, \vec{n}_i)$ and $(p_j, \vec{n}_j)$. If point p_i satisfies Equation 3.1 we simply set p_i as the origin of the canonical coordinate system otherwise p_j (and $\|\cdot\|_2$ denotes the Euclidean norm, i. e., $\|u\|_2 = (u^T u)^{\frac{1}{2}}$, same below).

$$\left\|\vec{n}_i \cdot (p_j - p_i)\right\|_2 \leqslant \left\|\vec{n}_j \cdot (p_j - p_i)\right\|_2 \tag{3.1}$$

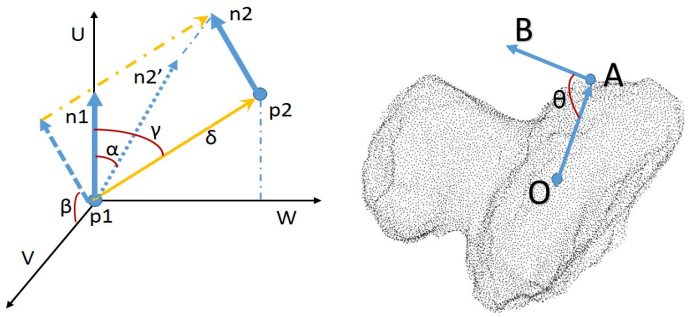
2. Then he constructs the canonical coordinate system by computing $\vec{U}, \vec{V}, \vec{W}$ as the base vectors: Assuming p_i as the origin so n_i is U, we normalize the vector $p_i - p_j$ by $\vec{\rho} = \frac{p_i - p_j}{\left\|p_i - p_j\right\|_2}$ in order ensure that the feature is invariant to scaling. The canonical coordinate system is then given by $\vec{V} = \vec{U} \times \vec{\rho}, \vec{W} = \vec{U} \times \vec{V}$.

From this canonical coordinate system, we derive Wahl et al.'s global features as follows: Given an object represented by a point set (p_i, p_j), for each pair of points we define four features for the complete four-dimensional vector $\vec{G}$:

$$\vec{G}(p_i, p_j) = (\alpha, \beta, \gamma, \delta) \tag{3.2}$$

by

- $\alpha = \operatorname{atan}(\vec{W} \cdot \vec{n_2}, \vec{U} \cdot \vec{n_2})$, $\alpha \in (-\frac{\pi}{2}, \frac{\pi}{2})$
- $\beta = \vec{V} \cdot \vec{n_2}, \beta \in (-\pi, \pi)$
- $\gamma = \vec{U} \cdot \rho, \gamma \in (-\pi, \pi)$
- $\delta = \frac{\|p_1 - p_2\|}{\max(\left\|\vec{p}_i - \vec{p}_j\right\|)}, \delta \in (0, 1)$.



(a) (b)

Figure 3.2: (a) Pairwise points p_1, p_2 and their normal vectors $\vec{n_1}$, $\vec{n_2}$. $\vec{U}$, $\vec{V}$, $\vec{W}$ are the intrinsic reference frame built by the pairwise points. And $\vec{n2'}$ is the projection of $\vec{n_2}$ to UW plane, α , β , γ represents the angles between ($\vec{n_1}$, $\vec{n2'}$) , ($\vec{V}$, $\vec{n_2}$), and ($\vec{n_1}$, $p_2 - p_1$). δ is the distance between the pairwise points. (b) O denotes the geometric center of the object. Vector $\vec{OA}$ and $\vec{AB}$ represents the position vector and normal vector of vertex A, θ is the inclined angle.

3.3.2 *Our Adaptive Hybrid Shape Descriptor*

Our novel invariant local geometric features extend the basic four-dimensional $\vec{G}(p_i)$ by two additional dimensions. They are inspired by the human cognition. Early psychophysical experiments showed that human visual system decomposes complex shapes into parts based on curvature and processes salient features before higher level recognition [135]. This research motivates us to focus on curvature to represent the local shape of the 3D model. This local geometric feature $\vec{L}(p_i)$ for each point can be represented by the following two parameters η and κ:

$$\vec{L}(p_i) = (\eta, \kappa) \tag{3.3}$$

with

- η represents the local normal perturbation of the object's normal
- κ is the Gaussian curvature.

In order to obtain a smooth curvature at each point P_i, we apply the discrete curvature analysis according to [36]. Additionally, we consider the normal perturbation with the parameter κ. The main challenge is to represent the point's normal in a transformational invariant way. Let c be the geometric center of the model and n_i the normal of point p_i. Then we define κ= (p_i-c)n_i, i.e. κ represents the angle between the vector that is spanned by the object's center and the point's position and the normal of the point (see Fig. 3.2b).

The global shape descriptor according to Wahl et al. and our local shape descriptor can be easily combined to our new adaptive hybrid shape descriptor (AHD):

$$AHD\ (p_i, p_j) = (\vec{G}(p_i, p_j), \vec{L}(p_i)) = (\alpha, \beta, \gamma, \delta, \eta, \kappa) \tag{3.4}$$

Since $\vec{G}(p_i, p_j)$ as well as $\vec{L}(p_i)$ are transformation invariant, also $\vec{G}(p_i, p_j)$ is transformation invariant. For a point cloud consisting of n points, $\vec{G}(p_i, p_j)$ can be computed in O(n) and $\vec{L}(p_i)$ in constant time for each individual point p_i.

3.4 TRAINING AND CLASSIFICATION

Our geometric feature described above is the basis for object recognition tasks. To do that, we create a database of histograms for a set of point clouds. The histograms are generated for each point cloud individually by computing our AHD-feature for all pairs of points and then discretizing them into bins. Similar to Wahl et al., we follow Wahl's setting by choosing five bins per dimension. This results in a total number of $5^6 = 15,625$ bins for each object. In the end, we get a 15,625-dimensional vector that represents the object.

3.4.1 *Parallelization*

A nice property of our geometric feature is that the histogram generation can be easily parallelized. Obviously, the parameters for each pair of points can be computed independently (see Algorithm 2). In order to bin the resulting six-dimensional vectors, we additionally have to sort these vectors. In detail, we use a parallel bitonic sort and a parallel reduction algorithm to count the number of entries per bin (see Algorithm 1). Please note, that our *local* features η and κ have to be computed only once per point, whereas the global features $\alpha, \beta, \gamma, \delta$ are computed per pair of points. Consequently, the total parallel running time of our algorithm is in O(n) assuming a perfect PRAM.

Algorithm 1: Compute Histogram (Pointcloud A)

In Parallel forall *points* $p_i \in A$ **do**
    featureSet[i]=computeFeature(p_i, A);
In Parallel sort(featureSet);
histogram = **In Parallel** reduction(featureSet);

Algorithm 2: Compute Feature (Point p, Poincloud A)

compute curvature η(p);
compute angle κ(p);
In Parallel forall *points* $p_i \in A$ **do**
    compute α, β, γ and δ(p, p_i);

3.4.2 *Histogram Cluster Analysis*

Choosing the best classification algorithm is a non-trivial task. For instance, it highly depends on the dataset and the number but also

the identifiability of the clusters. For our use case of asteroid classification (see Sec. 3.5), we first tried to use the linear discriminant analysis in combination with a PCA. The results show that that the distribution can be hardly linearly divided into meaningful clusters (see Fig. 3.3). On the other hand, random forest have shown to achieve high accuracy for the classification of non-linear datasets and they can easily handle multi-class classification challenges [81]. Moreover, the dataset of asteroid is relatively small (several thousands samples) which restricted the usage of neural network especially deep neural network [73]. Consequently, we decided to use random forests which shows the advantage of robustness and accuracy with relative small datasets over competitors.

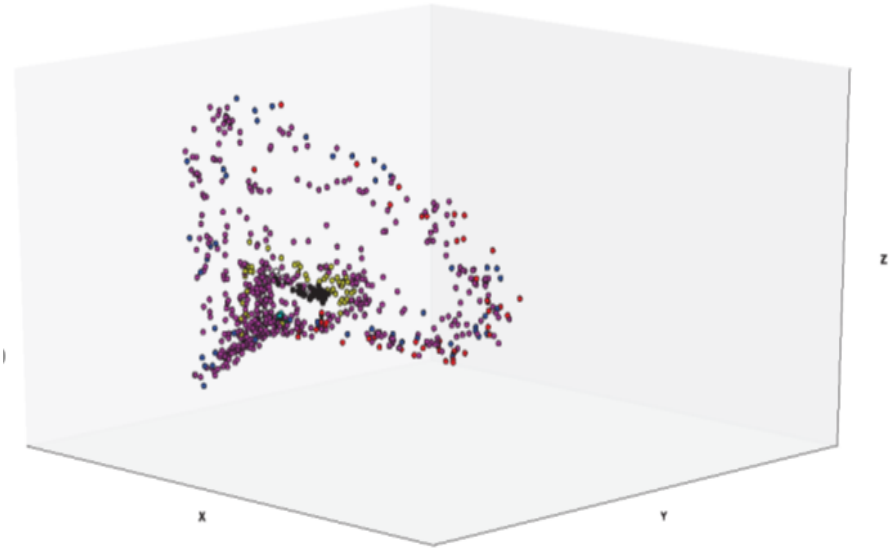


Figure 3.3: Using PCA transform to efficiently reduce raw feature histogram from 15,625 dimensions to 3 dimensions. The asteroid classes are color-coded.

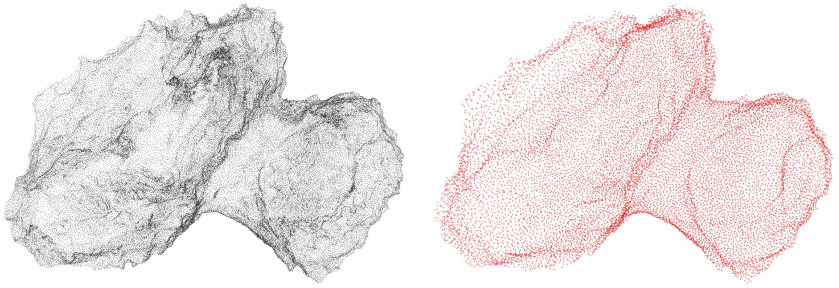
3.5 USE CASE: ASTEROID CLASSIFICATION

As one challenging example for the application of our algorithm we outline celestial bodies, especially asteroids. Asteroids differ in many ways from other (human created) objects because of their complex shapes, internal structures and material properties. For instance, Itokawa has significant porosities which are a key evidence for its belongingness of its corresponding taxonomic class [68]. Therefore, this kind of local dissimilarity pose a competitive challenge to our shape descriptors.

There is an increasing interest in the field of spacecraft flight to perform autonomous surface analysis and safe landing operations [119]. For these autonomous systems it is crucial to efficiently and accurately classify and recognize the shape and local surface details for the landing operations. Thus, the need for the ability of recognizing asteroids in arbitrary scenes based on 3D point clouds without large databases has occupied an important position.

A major drawback in the classification of asteroids is the lack of data. High-quality models of asteroids are usually recorded during rare and costly spacecraft fly-by or rendezvous missions [131]. Consequently, it is hard to obtain a large database for training and classification purposes. In order to overcome this limitation, we decided to use Poisson disk sampling on an asteroid database to increase the number of available data. Poisson disk sampling is one of the most classical methods for the fast resampling of surface points [37] and it is proven to be very robust Corsini [31].

Actually, the Poisson-sampled data draws a special challenge to our algorithm because the generated data is usually a lower-resolution model of the original high-resolution asteroid. Hence, it may lack some details on the surface. Figure 3.4 illustrates the Poisson sampled asteroid models.



(a) (b)

(c) (d)

Figure 3.4: Example 3D asteroids and sampled asteroids. a, c are the raw asteroid named Churyumov (128,002 points) and Eros (99,846 points). b, d represent the poisson-disk sample asteroids each with 25,994 and 18,172 points.

3.6 EVALUATION

We have implemented our adaptive hybrid shape descriptor (AHD) in Python 3.5. We performed our experiments on a machine with Intel Core i7 quad core processor with Hyperthreading enabled, 8 GB of memory, and a Nvidia Geforce GT 640M, operated by Windows 10. We applied three experiments to measure the performance as well as the quality of our shape descriptor approach.

First, we performed a comparison of the sequential CPU algorithm and our massively parallel GPU implementation for the histogram generation and the hybrid-feature computation (see Fig. 3.5). Second, we evaluated the quality of our approach and its competitors for

our previously outlined use case study of asteroid classification for autonomous spaceflight operations. Third, we evaluated the quality of our approach and its competitors based on the standard shape NTU database set.

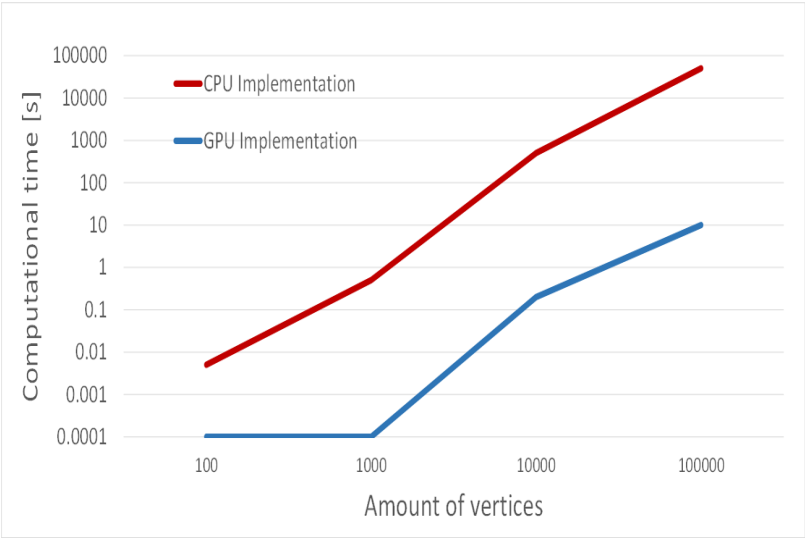


Figure 3.5: Computation times of CPU VS GPU.

We compared the quality to three state-of-the-art methods, namely the 3D Harmonics [43], LightField descriptor [27] and shapeDNA [122]. Here, we used freely available open source implementations. Additionally, we compared our approach to Wahl et al.'s original implementation (Global shape descriptor). In order to find the best parameters for our random forest for this competitive evaluation, we used grid search and selected appropriate parameters for estimators, depth, lea size and split criterion.

For our quality evaluation we use the well-known precision and recall diagram. Each of our evaluation plots precision versus recall averaged over all classified models in the database. The plot axes can be interpreted as follows [42]: For each target model in class C and any number K of top matches, "recall" represents the ratio of models in class C returned within the top K matches, while "precision" indicates the ratio of the top K matches that are members of class C. A perfect retrieval result would produce a horizontal line along the top of the plot, indicating that all the models within the target object's class are returned as the top hits. Otherwise, plots that appear shifted up and to the right generally indicate superior retrieval results.

3.6.1 *GPU-based Histogram Generation*

We compared the performance of a traditional sequential CPU implementation and massively parallel GPU implementation for our histogram generation (see Fig. 3.5). Here, we used Python 3.5. and pycuda for the implementation respectively. Our first evaluation shows that the massively parallel GPU implementation easily outperforms the traditional sequential one with an increasing number of vertices.

Our GPU-based implementation gradually outperforms the sequential CPU implementation by up to a factor of 1000. The GPU timings do not include transferring data between the host CPU memory to the GPU's global memory.

3.6.2 *Asteroid Classification Study*

In the second evaluation study, we evaluated our approach and its competitors for our previously outlined use case study of asteroid classification for autonomous spaceflight operations.

We randomly selected 20 asteroids from the Planetary Data System [95] and utilized our Poisson sampling approach in order to obtain a large set of asteroids 1000 for training, testing, and evaluation purposes (see Fig. 3.4). We add some random noise to all asteroid meshes during evaluation process to simulate realistic situation in space exploration.

Our evaluation shows that our shape descriptor approach with random forest based classification outperforms the competing methods (see Fig. 3.6). This means that our approach is the most discriminative and effective method among all evaluated approaches. Compared with the 3D harmonic descriptor, lightfield, and shapeDNA, our method owns an average of more than 70% precision rate when average the recall axis.

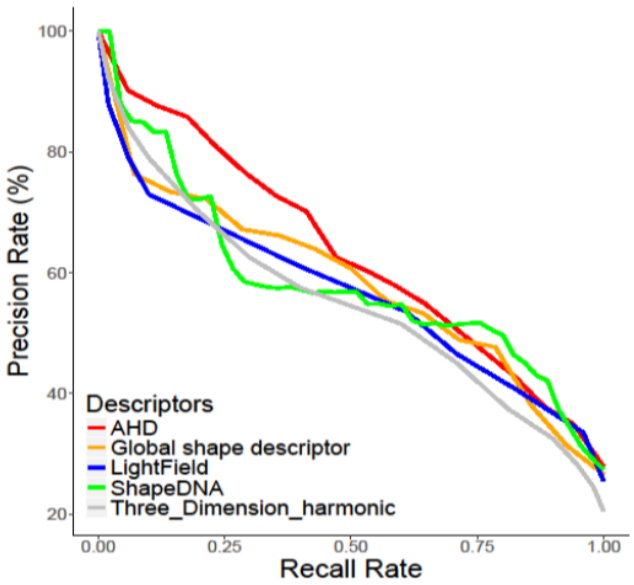


Figure 3.6: Asteroid experiment precision-recall curve performance evaluation of our approach (under three schemes), compared with 3D harmonic and light field descriptors.

Even more, our methods works well for almost all classes of asteroids, except for some difficult shapes as illustrated in the confusion matrix (see Fig. 3.7).

In summary, these good results demonstrate that, although we merely sampled 10∼20% vertices from the raw meshes, our shape

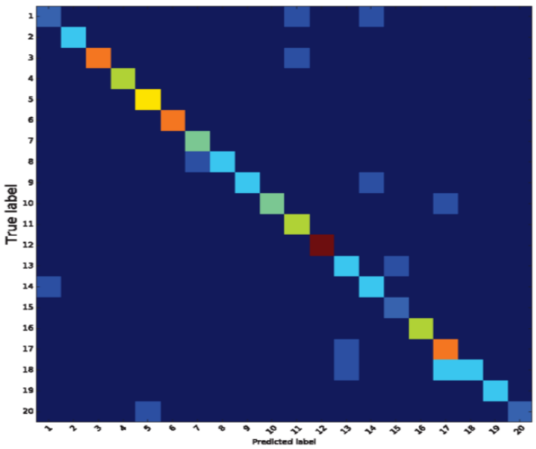


Figure 3.7: Plots the confusion matrix for the experiment of asteroid classification.

descriptor is able to robustly represent the noise asteroid shape while achieving high classification rate.

3.6.3 *Standard Dataset Testing*

In the last evaluation study, we evaluated our approach and its competitors based on the standard shape NTU database set[1] [27]. The NTU database currently contains 10,911 3D models from 352 categories, from which we selected 1,218 representative models from the database as our testing database. These 1,218 3D models are composed into 10 classes which have the most models in each class in the database (see Table 3.1). Several examples of 3D models contained in these 10 most well-annotated classes are shown in Figure 3.8.

Table 3.1: Subset of NTU database

Category	Number of models	Training set
Tree	120	84
Gun	120	84
Enterprise	80	56
Wheel	78	55
Table	115	81
Potted-plant	84	60
Human	192	133
Helicopter	98	68
Fighter-plane	234	164
Four-legged-chair	97	68

We split the above determined 10-class dataset randomly into training, validation and test set and used this data as the evaluation baseline. Our approach outperforms its competitors also in this evaluation study (see Fig. 3.9). Our shape descriptor achieves the best performance with classification accuracy of 62.5% and 57.1% under invariant

1 http://3d.csie.ntu.edu.tw/ dynamic/database/

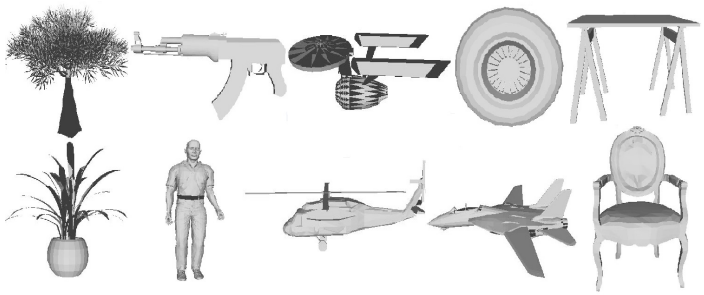


Figure 3.8: The example of all selected class of 3D models in NTU database.

descriptor and global descriptor respectively, after averaging over all the recall axis. While 3D harmonic, lightfield and shapeDNA descriptors achieved 52.8%, 53.5% and 53.1% accuracy under the same conditions. Surprisingly, shapeDNA performed worst. The reason for this could be low quality and incompleteness of some meshes in the NTU database. As a result, shapeDNA is not robust enough to distinguish them. In this evaluation study, our approach does not outperform its competitors to the same extent as in the previous evaluation. We believe that the shapes of the NTU database have less local shape information than the asteroid shapes of our use-case study.

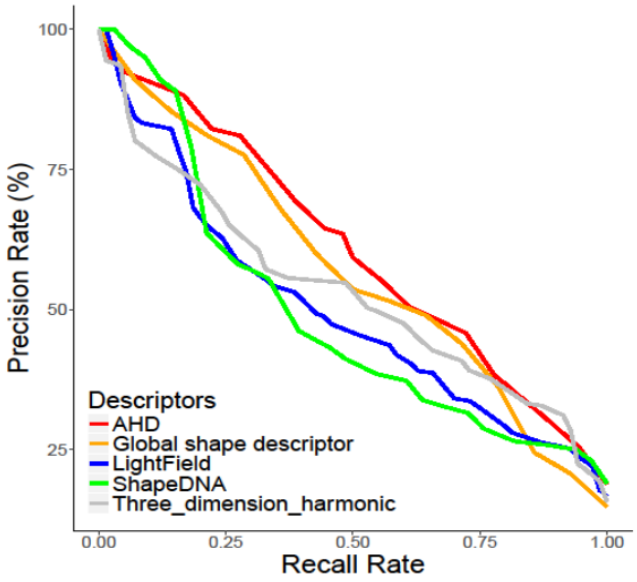


Figure 3.9: NTU database experiment precision-recall curve performance evaluation of our approach (under three schemes), compared with 3D harmonic and light field descriptors.

3.7 CONCLUSIONS AND FUTURE WORKS

We have presented a novel statistically invariant shape descriptor for large-scale shapes with local dissimilarities. The main idea is to combine features that describe the global shape with two novel features that represent the local curvature and the normal perturbation, respectively. This enables our hybrid-feature to classify both large-scale shapes with local dissimilarities based on their local appearance and

standard shapes based on their global and local dissimilarities. Our novel features are robust to noise and invariant to translation, rotation and scale of the shapes. Furthermore, we presented a parallelization of the histogram computation using GPU processing in order to deal with massive data from high-resolution 3D shapes. The results show that our GPU implementation is more than three orders of magnitude faster than the equivalent CPU implementation.

Due to its generality, our approach is applicable to a wide variety of classification domains for three-dimensional shapes. The results from our benchmarks show that our approach is able to efficiently classify large-scale shapes with local dissimilarities in a special asteroid use case but also for common objects.

In the future, we would like to further evaluate our approach with more shape databases, especially in a terrestrial context. However, we are mainly interested in improving our current approach for the outlined asteroid classification use case study. Here, we would like to incorporate reinforcement learning with our hybrid shape descriptor.

Hinton and Krizhevsky [73] proposed unsupervised deep learning method for image retrieval, this method can be a good example for our classification algorithms. Another interesting idea would be to extend our approach with additional shape descriptors to further improve it's accuracy.

4

ASTROGEN - PROCEDURAL GENERATION OF HIGHLY DETAILED ASTEROID MODELS

In this chapter, we will present a novel algorithm, called AstroGen, to procedurally generate highly detailed and realistic 3D meshes of small celestial bodies automatically. AstroGen gains its realism from learning surface details from real world asteroid data. We use a sphere packing-based metaball approach to represent the rough shape and a set of noise functions for the surface details. The main idea is to apply an optimization algorithm to adapt these representations to available highly detailed asteroid models with respect to a similarity measure. Our results show that our approach is able to generate a wide variety of different celestial bodies with very complex surface structures like caves and craters.

4.1 INTRODUCTION

The study of small celestial bodies in the solar system (i. e., asteroids, comets) has become an area of great interest for astronomy science in the past decades. For instance, Galileo and Cassini are successful missions to investigate small celestial bodies and collected abundant interesting data about several asteroids. More recently, JAXA's Hayabusa2 spacecraft will, after studying Ryugu in depth from orbit for about a year, drop three rovers and a lander onto the asteroid's surface this month and hopefully, some samples will be sent back.

Such long distance missions are challenging for several reasons: first, the communication takes a very long time. Hence, it is not possible to immediately react on complications during i. e. the landing phase by the mission control on earth. Consequently, spacecrafts operating in such space environments are usually equipped with some sort of autonomy. Second, the terrain of the asteroids is usually unknown in advance during the planning phase of the mission. The most economic and common way to observe asteroids from earth is to obtain data radar or lightcurve inversion. However, these methods do not deliver surface details. Nevertheless, the landing spacecrafts and rovers has to be designed with such very limited information.

Usually, space missions are planned with support of virtual testbeds (VTBs) before building physical mock-ups (see Fig.4.1). These testbeds consist of physically-based simulation of terrain that provide a real-time, immersive and 3D interaction environments which give engineers the opportunity to interact with the simulated spacecraft or rover to gain comprehensive understanding of possible design flaws during the early design process as well as later mission stages like training and supervision [78]. In order to simulate a large amount of possible scenarios to be prepared for a lot of circumstances, it is essential to have a large number of highly detailed and realistic 3D

asteroid models available in such a virtual testbed. In the case that autonomous algorithms on board of the spacecraft support the landing operation, there is even more need for such 3D models because these algorithms usually have to be trained with a large amount of such data.

The generation of such models has two major challenges: the lack of ground truth data and the missing of appropriate methods to synthesize realistic models of irregular shaped small bodies. Actually, the only available highly detailed asteroid model is that of Itokawa[1]. Moreover, traditional terrain generation algorithms are almost all optimized for spheroidal planetary models where a simple heightmap can be used. This is not directly applicable to irregular celestial objects such as asteroids especially if terrain details such as caves have to be considered.

We present, to our knowledge, the first algorithm that is able to compute realistic highly detailed 3D models of irregular celestial bodies fully automatically. The main idea is to combine different implicit object representations with an optimization algorithm to adapt their implicit parameters to real world models. These parameters can be varied or applied to completely different basic shapes while remaining the overall surface texture.

More precisely, we use a two-tier approach: For a given ground truth asteroid shape, we first approximate the basic asteroid shape by a polydisperse sphere packing. A first optimization step adapts parameters of a metaball approach. In a second optimization step, we learn the surface details by optimizing the parameters of several noise functions that are well chosen to represent typical surface structures of celestial bodies such as craters and caves. This makes it easy to transfer the surface details to other basic shapes. During the training phase, we allow small intervals of all parameters with respect to the distance function. This enables us to further vary the surface details but also the underlying metaball shape.

Our algorithm, called *AstroGen* supports:

- *Full automatic generation* of almost endless variations for a given ground truth asteroid model within a pre-defined error bound. However, the parameters of the synthetization algorithm are easy to understand which makes it easy to manually adjust the generated models.
- *High performance* due to an almost full GPU implementation of all time consuming parts of the algorithm.
- *Arbitrary Resolutions:* due to the implicit representation it is easy to generate polygonal models at any resolution with the marching cube algorithm.

In a use case scenario we have applied AstroGen to the currently only available high resolution asteroid model of Itokawa and generated

1 The data from the Rosetta mission was published simultaneously to this submission. Hence, we could not include this in our algorithm. However, our algorithms would obviously also work with this model.

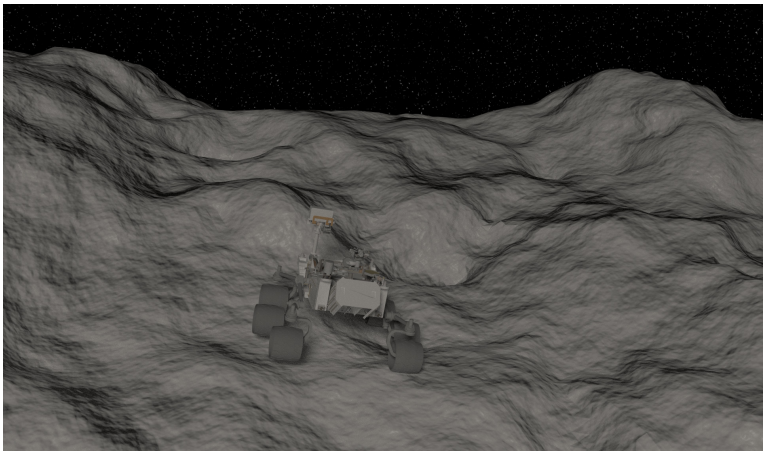


Figure 4.1: Simulated rover cruise on the virtual testbed of unknown asteroid stein [40].

several variations of it. Moreover, we applied the surface details to low-poly models of Stein [40], Ceres [148] and Lutetia [136].

4.2 RELATED WORK

There are four main approaches to generate terrain features: procedural, physics-based, sketch-based and example-based [51]. Physics-based methods generate terrain features geologically correct, but often computationally expensive and lack of scalability in simulation different natural phenomenons. Sketch-based methods require heavy manual intervention while example-based methods are limited by the input data and restricted to two-dimensional terrain features [51]. In contrast, procedural methods are fast and easy to generate arbitrary resolution of realistic terrain in the virtual world, see [74, 137] for a broader overview. Often, such methods rely on some kind of noise function. Perlin noise is known as efficient and its inherent self-similarity, consistency properties are suitable for terrain generation [113]. Enhanced version, like simplex noise [114] reduce some artifacts or generate particular terrain features, such as ridges or rolling hills [39]. The Commercial software Terragen [172] and i. e., a bunch of researcher's papers [51, 107, 149] created diverse and realistic terrain based on noise method. Togelius [149] introduced evolutionary algorithm with noise method to generate terrain map and balanced on several objectives, such as playability and realistic of terrain. [51] focused on Hydrology terrain simulation by using fractal interpolation to connect predefined physical-based terrain features such as river networks, mountain ridges and valleys. [107] proposed a method based on real elevation statistics and utilize value noise – a variant of perlin noise – to generate geotypical terrain. Recently, compact mathematical definition [54] and sparse procedural [57] method are proposed which efficiently combine different terrain primitives and give user more intuitive control about the scene. However, noise-based methods usually tend to create terrain that is uniform at fixed amplitude and frequency values, and often require massive post-process to generate interesting

features and choosing the correct parameters for this post-processing is often unintuitive. Moreover, none of these algorithms supports the generation (or variation) of irregular celestial bodies.

Implicit modeling gained attentions and interests of many scholars due to an intuitive representation of objects in a mathematical way, rather than explicit geometric representation. The mainstream path of implicit surface construction can be divided into two parts: non-skeleton implicit surface and skeleton implicit surface. For the sake of clarity, here we give an overview of skeleton implicit surface.

The idea of skeleton implicit surface origin from metaball modeling [11] which is a side effect of a visualization of electron density field from the early 1980s. This kind of methods can be seen as a point based scalar field integrate blending and deformation operators which allows solid models to be easily described. [13] expanded metaball with convolution implicit surface. Their method is actually an extension of point-based primitives into a line or a plane based primitives. Subsequently, Wyvill and Brian [163] introduced general construction trees called the BlobTree greatly enhance the range of models that can be defined with a skeleton implicit surface system. More specifically, they proposed a hierarchical method which allows arbitrary compositions of models that make use of blending, warping and boolean operations.

The core point of these modeling methods are assembling separate parts with composition operators, and each part have been defined by a geometric primitives' scalar function. [16] proposed an autonomous way through building a sphere-tree to represent the complete approximation of the object. They use medial axis approximation algorithm to cover the object completely with spheres, that do not contain gaps, without adversely affecting the fit to the original object. [144] propose parallel version of sphere packing algorithm which effectively compute the sphere to represent the arbitrary object.

Once fitted the shape with the geometry primitives, composition operators control the way how they combined. For instance, in early research the *max* (*min*) of two scalar functions produces a union operator which is the basis of Constructive Solid Geometry (CSG) [13]. The blending operator [11], in some cases a simple *sum* of the combined scalar functions, smooth the sharp transition between parts produced by the *union*. More recent research the gradient-based blending operators [1, 56] help approximate a mesh by deforming on 3D scalar functions and becomes popular. [56] proposed the idea that scalar functions should not only be combined based on their values but also on their gradients. And their method solve the problem of bulging and topology while merge two shapes. [1] introduced an approach to design gradient-based operator which can achieve desired effect on the resulting surface. Our method address the benefits provided by integrate these concepts into a sphere-based metaball modeling system in the following section.

4.3 OUR APPROACH

The goal of our algorithm is to generate a wide variety of different asteroid models considering the underlying basic irregular shape as well as the surface details. We want to achieve high realism and detailed high polygon models. In order to guarantee realism, AstroGen relies on the usage of real world data, namely surface details from previous space missions, like that of Itokawa and data from earth observation that delivers the rough shape and can be found in extensive asteroid databases [176]. Due to the flexibility of our algorithm, it is easy to also include more data (like that from the Rosetta mission that was just released or from the Hayabusa2 mission).

The two different kinds of data already suggest a two-tier approach. Consequently, AstroGen consists mainly of two stages:

1. We use an *implicit shape* approach to represent the underlying rough shape. The advantage is that it easily allows to make small variations in the rough shape and additionally, we can generate high-poly meshes from it.

2. The surface details are represented by different *noise functions*. Again, this enables us to generate poly meshes in arbitrary resolution.

In order to adapt our asteroids to the real world data mentioned above, we use an *optimization algorithm* to optimize the parameters of our rough shape as well as the surface details. However, we allow small variations in the parameter range to generate an almost infinite number of variations. Finally, we present a method to generate a polygonal mesh from our implicit asteroid representation based on marching cubes.

All these steps can be performed entirely massively parallel on the GPU what guarantees a high performance. However, the optimization steps are required only once per ground truth data. For the generation of a wide variety of asteroid models it is sufficient to simply vary the parameters and generate a mesh using the final step in our algorithm. Figure 4.2 shows an overview on the complete process. In the following, we will present the details of the individual steps of our automatic pipeline.

4.3.1 *Implicit Shape Representation*

Generally, there are two main types of method to represent a free-form surface: *parametric surfaces* and *implicit surfaces*. In the first case, Hermite-spline, B-spline and NURBS are most commonly used. For the latter, metaballs, skeleton and convolution surfaces are the most popular methods. Implicit surface modeling produce smooth and aesthetic shapes that can be seamlessly modified while keeping a consistent structure. In addition, their function definition is compact and require quite simple primitives such as sphere or ellipsoid to construct a model which is suitable for real-time simulation. Finally, it

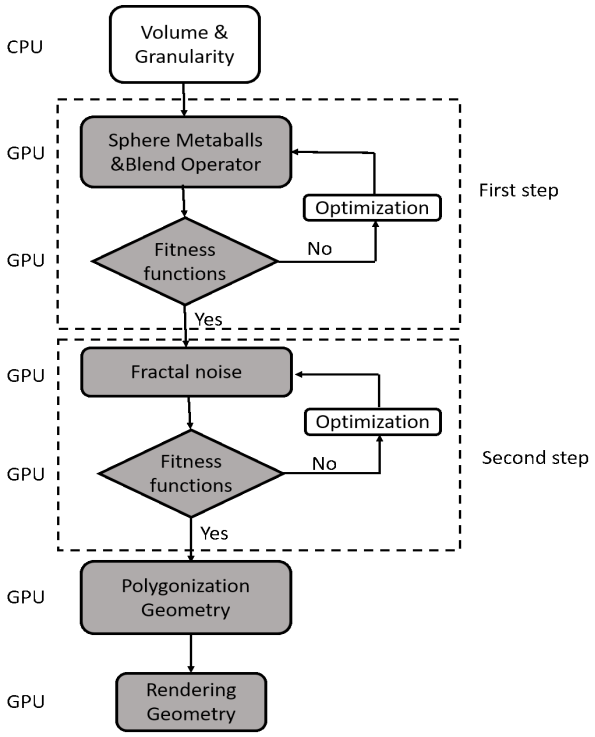


Figure 4.2: The pipeline of two parts simulation.The dark box means the program running on the GPU while the white box on the CPU side.

is possible to adapt this representation to existing polygonal shapes, like the low-poly models from the asteroid database. These are the main reasons, we decided to use an implicit surface representation for the basic shape of our celestial bodies.

In principle, implicit surfaces define a 2-D manifold, where a surface S embedded in the three-dimensional space R^3:

$$S = \{(x, y, z) | F(x, y, z) = T\} \tag{4.1}$$

For a *skeleton implicit surface*, we usually have given a set of n distinct constraint points $c_1, ..., c_n$ and a set of corresponding potential functions F(c_i). These defines a smooth surface M as:

$$M = \{c \in R^3 | F(c) = \sum_{i}^{n} \omega_i F(c_i) + t = T\} \tag{4.2}$$

The challenge is to select an appropriate set of points and potential function.

4.3.2 *Polydisperse Sphere packing*

Recap:Polydisperse sphere packing

Polydisperse sphere packing denotes a method to fill arbitrary object with masses of random size of non overlapping spheres. The main

insight of the algorithm is to define two primary constraints during the filling process. The first constraint is the newly inserted sphere must not intersect with the sphere already exists. The second constraint is that all spheres must inside the object. Generally, the algorithm begins with filling the largest sphere s into the object and incrementally insert new spheres while keep to the two constraints. Then, for the case of largest sphere s inside the watertight constraint object O will produce at least 4 contact points (2D at least 3 contact points), and no extra points on the object inside the sphere. This property means the sphere center is a Voronoi node (VN) of the object. As a result, it is possible to reformulate the greedy object filling problem as an interactive evaluation of a generalized Voronoi diagram (VD) of object with the sphere already exists. The details of the algorithm is explained below [159]:

Algorithm 3: Converge Prototype (prototype p, object O)

place p randomly inside O
while p has not converged do:
$q_c = argmin\|p - q\| : q \in surface\ of\ O$
choose $\varepsilon \in [0, 1]$
$p = p + \varepsilon(p - q_c)$

It is easy to understand, by randomly selecting a single point inside inside the object, the prototype, iteratively move towards one of the voronoi node and it must stays inside the object. The constraint $p = p + \varepsilon(p - q_c)$ (see Alg. 3) guarantees that, because the entire sphere around p with radius $\|p - q_c\|$ is inside the object, then after each iteration, the prototype is ensured inside the object [159]. Nevertheless, moving prototype away from the border, into direction of $(p - q_c)$, leads potentially to bigger spheres in the next iteration step [159]. Commonly, the ε is defined as an attenuation function instead of a constant scalar value which allows the movement range of the prototype shrinking along with each iterations. The filling rate as well as the stability of the approximated Voronoi node is heavily relies on the strategy of the attenuation function and the number of iteration steps (see Fig. 4.3).

Our method

Our idea is based on the approach by Wyvill and Brian [163] that extends the metaballs algorithm [11] by using a so-called BlobTree to represent the skeleton. However, with this method, the points are located on the medial axis of the object. This makes it complicated to add i. e., additional hills or to remove parts to form valleys. Hence, we decided to modify this idea by adopting a sphere packing-based approach. Originally invented for collision detection, the Protosphere [144] algorithm delivers a polydisperse sphere packing of arbitrary 3D objects. This is ideal for metaball-based modeling systems and for our application in particular because the greedy choice of the algorithm

Figure 4.3: A model of armadillo filled with over 15,000 spheres [159].

automatically leads to a level-of-detail representation (hierarchy of spheres) for the model, i. e. it offers an easy trade between the speed and accuracy. Moreover, it is easy to simply add or remove individual spheres to create diverse topologies as well as small varieties on the basic shape. Finally, the algorithm is fast and works completely on the GPU. The voronoi node in the sphere packing algorithm defines our constraint points (metaball center) in Equation 4.2. The Figure 4.4 shows a sphere packing (Protosphere) approximation of the asteroid Itokawa.

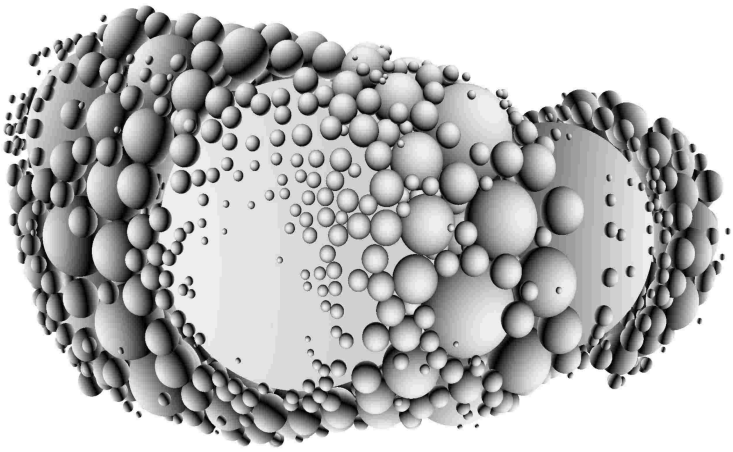


Figure 4.4: The sphere packing [144] representation of asteroid Itokawa by Protosphere algorithm.

Blending

After obtaining the constraint points, we have to define the potential functions in Equation 4.2. Remember, that the surface details are added in the second step, hence, we first want to create an overall smooth

shape. To do that, we slightly modified Blinn's [11] original potential function to fit our constraint points:

$$f(r_i) = \begin{cases} e^{a-br_i^2} & if\ r_i\ \in\ [0,5R] \\ 0, & otherwise \end{cases} \tag{4.3}$$

with

- a and b are the tension factors that control the smoothness in the overlapping areas and the softness of each metaball
- and r_i is the radius of each spheres and $R = \max\{r_i\}$ to limit the influence area

Summarizing, we get the complete potential function of p in 3D space as:

$$f(r) = \sum_{i=1}^{n} f(r_i(p, c_i)) \tag{4.4}$$

While implicit surface deformation is represented by multiple metaballs, bulge, crease and tearing frequently appear in overlapping areas. In order to eliminate multiple metaball influences on an overlapping area we additionally added some blending function according to [163]:

$$f(A \cdot B) = (f^m(A) + f^m(B))^{\frac{1}{m}} \tag{4.5}$$

with

- $f(A)$ and $f(B)$ represent two metaball's potential functions
- and m controls the influence of the overlap in the distance field; With $m = 1$ we get the traditional overlapping method. In case of high convexity we can modestly change this parameter.

Figure 4.5 shows the optimized metaball surface to approximate the asteroid of Itokawa.

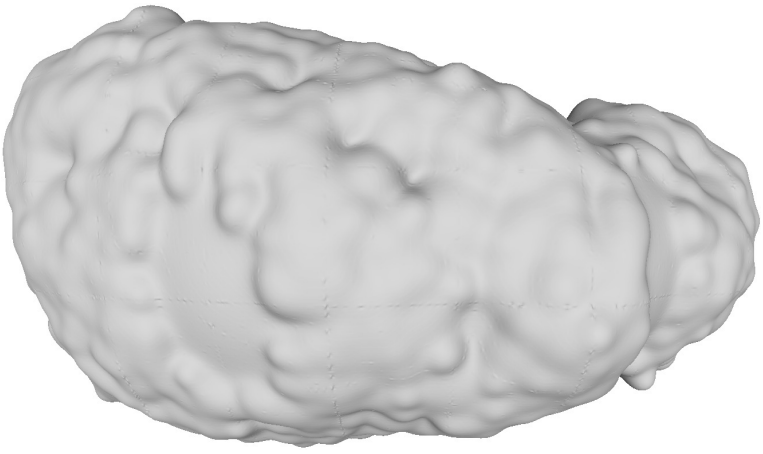


Figure 4.5: The implicit metaball shape of Itokawa (892k vertices).

4.3.3 *Noise Based Surface Features*

The basis of almost every procedurally generated landscape algorithm is a noise function. Due to these noise we can procedurally produce mountains, valley, craters or even pebbles, however these landscape must be created manually. In order to generate a realistic surface features, we also apply several noise functions to the underlying metaball model described above. In the following, we will describe the different layers of noise we used.

Perlin Noise

Perlin noise is a usual type of noise that is often use to generate terrain because it fulfills Tobler's First Law of Geography [97]. Basically, Perlin noise is a lattice-based gradient noise. However, simple Perlin noise often leads to repetitive patterns. The complexity of the generated terrain can be controlled by several parameters. The most important is the number of *octaves*. For a given frequency and amplitude we can generate an octave by doubling the frequency and halving the amplitude or vice versa. For instance, progressively adding lower frequencies (with higher amplitudes) generates larger terrain structures, such as large mountains and trenches [100]. Accumulate eight octaves are called Fractional Brownian Motion (FBM). In our implementation we used FBM. Moreover, we use simplex noise (see Alg. 4), a derivative of Perlin noise that uses a simplex instead of a quadrangular lattice. This improves the performance significantly. Additionally, the combination allows us to enhance the control of the generated details. Figure 4.6 shows the surface created by different combinations of Perlin noise.

Algorithm 4: Compute Density Value (Block A)

In Parallel forall *points* $p_i(x, y, z) \in A$ **do**

freq = 4, amp =1, octaves = 8, density_value = f(blend(r(p_i), r(p_i)))

while octaves > 0

    density_value = density_value + simplex_noise(p_i, freq)

    freq = freq * 1.95

    amp = amp * 0.5

    octaves = octaves - 1

Caves

Using too many octaves of Perlin and simplex noise results in isotropic details that can give the terrain an artificial look. A usual way to overcome this problem is to modulate the original shape (in our application, the rough shape generated by the metaballs) with another noise function. This technique, called *warping* is very common in computer graphics for generating procedural textures or geometry.

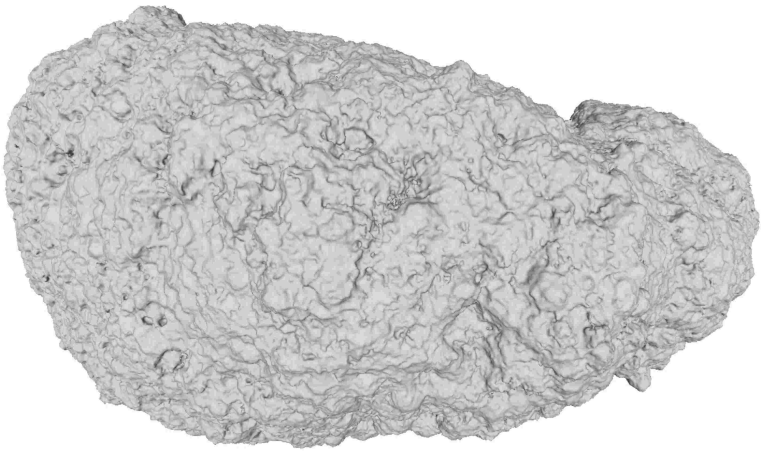


Figure 4.6: The rough shape of the asteroid Itokawa with surface details generated by Perlin noise.

The warping is a small perturbation to the given point p and can be defined as follows (fbm is the fractional brownian motion which denotes the multi-octave of perlin noise):

$$\begin{aligned} q &= vec3(fbm(p+\epsilon), fbm(p+\epsilon), fbm(p+\epsilon)) \\ warping(p) &= p+q \end{aligned} \tag{4.6}$$

where p, q represent the point in the 3D space, ϵ denotes random small deviation to the point p. Using medium frequencies and mild amplitudes results in surreal ropey organic-looking terrains. Lower frequencies and higher amplitudes increase the occurrence of caves, tunnels, and arches [100]. Figure 4.7 shows the Itokawa model with warped coordinates.

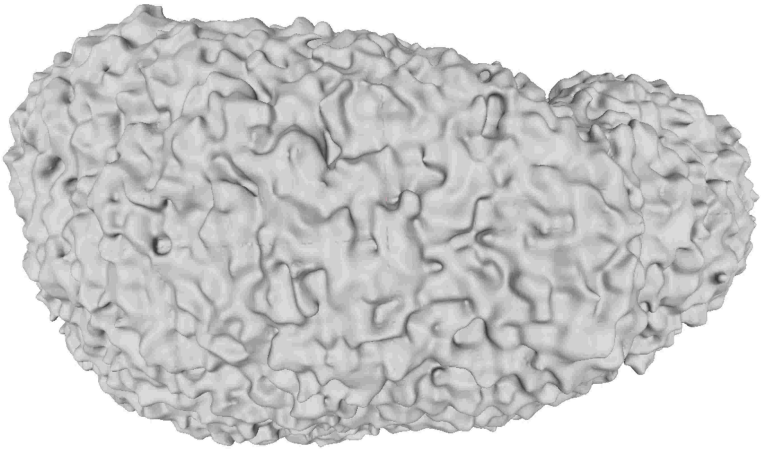


Figure 4.7: The cave occurs through warping the input grid point.

Craters

Craters are one of the most prominent visual terrain elements of celestial bodies without atmosphere. Unfortunately, Perlin noise is not able to generate structures that look like craters, or at least, it is not

known how to set the parameters until now. Consequently, we use another type of noise to support this sort of terrain.

In contrast to Perlin noise, Worley noise [161] (see Alg. 5) is not gradient-based, but value-based. The basic idea of Worley noise is to grow points until another growing point is hit [111]. This leads exactly to terrains that look like craters. We adopted a simplified GPU-based version similar to [59]. Figure 4.8 shows the result when applying Worley noise only (without adding additionally Perlin noise).

Algorithm 5: Generate Craters (Block A)

In Parallel forall *points* $p_i(x, y, z) \in A$ **do**

```
freq = 4, weight =1, octaves = 4
while octaves > 0
    density_value -= worley_noise(p_i,
 freq*weight*worley_noise(p_i, freq))
    freq = freq * 2.05
    weight = weight * 0.5
    octaves = octaves - 1
```

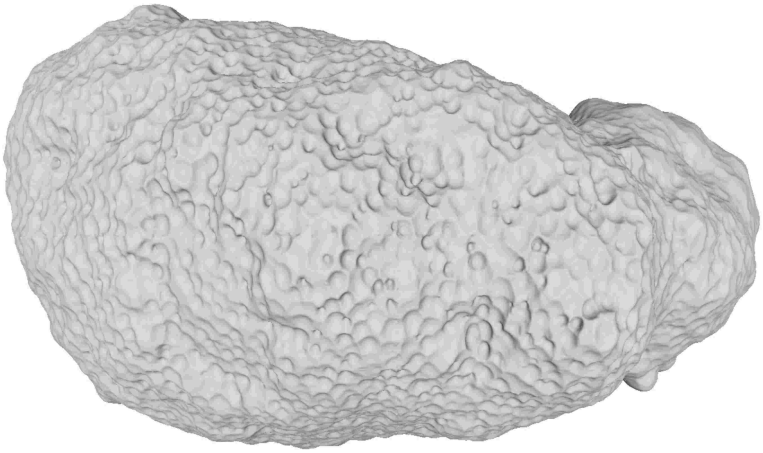


Figure 4.8: The rough shape of Itokawa with pure Worley noise and craters appear.

4.3.4 *Optimizing Noise Parameters*

For simulations in VTBs or the training of autonomous algorithms to steer landing spacecrafts it is essential that the generated asteroids are as realistic as possible. Hence, we do not choose the parameters of our generation methods arbitrary. Manually selecting the parameters is also not an option because they often behave unexpectedly. Consequently, we add an initial optimization step to adapt the parameters automatically.

Table 4.1: The number of parameters of the solution domain in the surface detail optimization for the individual noise functions.

Para	Perlin	Simplex	Worley	Gradient
Weight	1	1	1	1
Frequency	1	1	1	0
Octave	1	1	1	0
Amplitude	1	1	1	0
Coords_w	3	3	3	0
Coords_b	3	3	3	0

Optimization Algorithm

Given some ground truth data, e.g. the detailed model of Itokawa and asteroid databases that provide polygonized rough shapes of asteroids, we can use any optimization algorithm for the parameter adaption provided a good fitness function is available. For several reasons, we decided to apply particle swarm optimization (PSO) [38]. PSO is a stochastic convergence analysis algorithm containing parameter selection and changing. In principle, PSO is a population-based iterative algorithm and the swarm behavior guide the particle in the population to search for globally optimal solutions: The standard PSO algorithm maintains a population of N particles, and each particle defines a potential solution in a D-dimensional solution domain. There exists many factors that influence the convergence property as well as performance of the standard PSO algorithm. In our implementation we rely on fixed parameters according to [127].

The structure of PSO makes it easy to map our parameters to the algorithm. Moreover, PSO is fast and stable with respect to the initial parameter values and it does not require gradient information. We optimize the implicit shape representation and the noise-based (see Sec. 4.3.2) surface details (see Sec. 4.3.3) individually.

Fitness Function

Choosing a good fitness function to compare the result generated by PSO to the real-world model is essential for the success of the optimization. In our implementation we used a histogram-based shape descriptor presented in [85]. It was developed with the special scope of considering large-scale models with local similarities that typically appear in celestial bodies (see Chapter 3 for more details). In other words, our optimization problem can be defined as:

$$minimize: \left\| hist_{generate} - hist_{target} \right\|_2 \tag{4.7}$$

Where $hist_{generate}$ denote the histogram (see Chapter 3) of our procedrual asteroid and $hist_{target}$ represent the target asteroid 3D model's

histogram. The histogram $hist_{generate}$ is a statistical property and we choose the heuristic optimization method - Particle Swarm Optimization (PSO) to execute random search on domain of definition. Moreover, the generation of histogram for each asteroid model is fast and works completely on the GPU, thus, it fits perfectly in our heuristic optimization pipeline. However, it is easy to include other fitness functions.

Implicit Shape Optimization

In order to optimize the rough shape defined by the sphere-packing in combination with the metaball approach (see Sec. 4.3.1), we simply use the parameters from its description for PSO, namely, the number of spheres, the two tension factors of the potential function (see Eq. 4.3) and the blending parameter from Equation 4.5. This defined a 4D parameter space. Figure 4.5 shows the results of this first optimization stage for Itokawa.

4.3.5 *Surface Detail Optimization*

The surface details are represented by three different noise functions – Perlin, simplex and Worley noise – and additionally, the noise modulation of the gradient of the Perlin and simplex noise. In addition to the individual parameters (like frequency, amplitude, the number of octaves) of the particular noises, we add four weight parameters to control their individual amount. Moreover, we include three parameters to scale (scaling) the input grid point's 3D coordinate axis and another three parameters to define biases (translation) to the axis. Another important parameter is the gradient and we directly multiply our fractal noise with its gradient value. What's more, we have another three parameters to control the number (octaves) of perlin noise, simplex noise and worley noise. In total we have 34 parameters to control the possible pattern of the surface details (see Table 4.1).

4.3.6 *Polygonization*

VTBs usually require the 3D objects as polygon meshes instead of implicit representations. However, it is easy to generate such a polygonal mesh by using marching cubes (see Alg. 6) to compute an isosurface. In our recent implementation, we use an hierarchical GPU-based version in order to quickly generate any required resolution. Please note, also our fitness function described above requires a polygonal (point cloud of the surface) representation to generate an appropriate histogram based on the vertices [85]. However, we do not generate a complete high-poly model, but our experiments have shown, that it is sufficient to content with only a subset of around 10% of a full two million vertex model when using Possion disk sampling [37] .

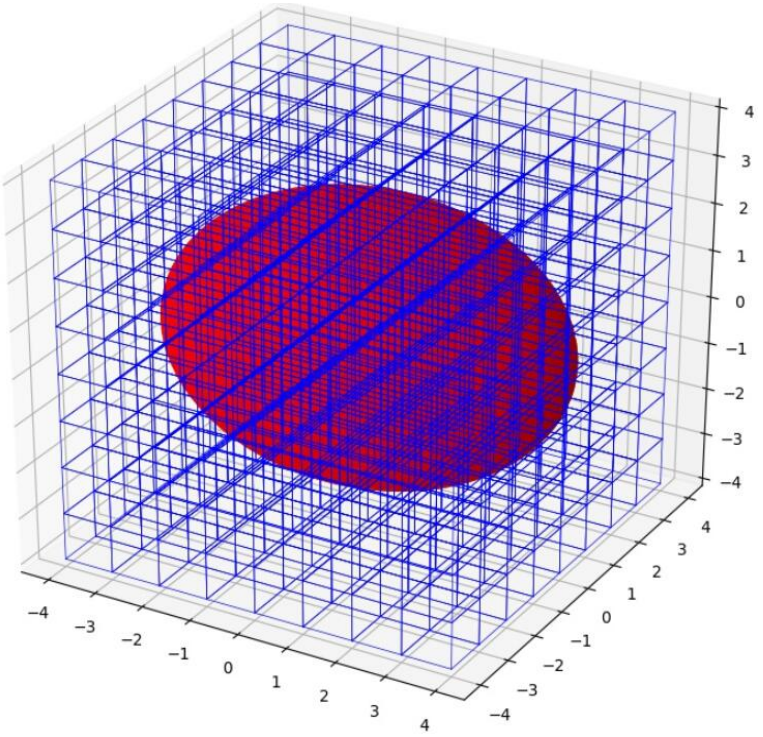


Figure 4.9: We use a hierarchical marching cubes algorithm to polygonize our implicit object representation. The blue grid divides the object into several smaller blocks. The marching cubes algorithm is then performed on these smaller blocks individually.

Algorithm 6: Generate Triangles (Point p_i, Block A)

In Parallel forall *points* $p_i \in A$ **do**
get p_i eight neighbor grid points
compute marching cube case index m
compute number of polygons in current point grid n
for i in n:
check edge connect table[m, i]
interpolate between edges
emit point

4.4 RESULTS AND DISCUSSIONS

We have implemented AstroGen in C++ and CUDA, including the implicit generation and the optimization algorithm. The computation of the noise values is implemented using computer shaders and marching cubes algorithm relies on geometry shaders in OpenGL. We performed our experiments on a machine with Intel Core i7 8-core processor with 8GB of RAM and an Nvidia GTX1080Ti. We have evaluated the performance as well as the quality in two different test scenarios. The basis is the optimization of all parameters for an available high poly-asteroid model of Itokawa. We set the problem space to $D = 34$ according to Section 4.3.5. The population size of PSO was set to $N = 20$ and we allowed at most 100 generations.

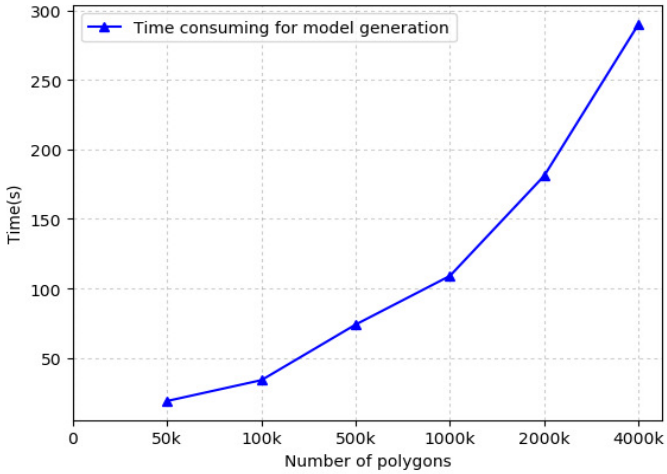


Figure 4.10: The time required by our algorithm to generate 3D models of Itokawa in several resolutions.

First, we investigated the performance to generate 3D objects from our implicit representation with respect to the polygon count. Figure 4.10 shows the mean average computation time for the specific resolutions. The time increases almost linear with an increasing number of polygons. Please note, that our current marching cubes implementation is not yet fully optimized. In the future, we hope to improve the performance i. e. by additionally applying an octree to accelerate the generation time. The training phase took approximatively 8 hours for the implicit shape representation and 100 hours for the noise-based surface features. Most time was spend on the generation of the high-resolution model (90%), while the computation of the fitness function required 10% of the overall time.

In our first test scenario, we applied our method to automatically generate similar asteroids with small variations that are below a $\delta <$ 20% with respect to the fitness function. Figure 4.11 shows the original model and some of the results[2]. Figure 4.11b exhibits most similarity with the original Itokawa model, followed by a rapidly decreasing in similarity but two different terrain patterns in Figure 4.11c and Figure 4.11d.

In our second scenario, we transferred the parameter set for the surface details that were trained with the Itokawa model to other (low-poly) basic shapes from an asteroid database that were never explored with a spacecraft. Figures 4.12 , 4.13 and 4.14 show the results for Stein, Ceres and Lutetia, respectively.

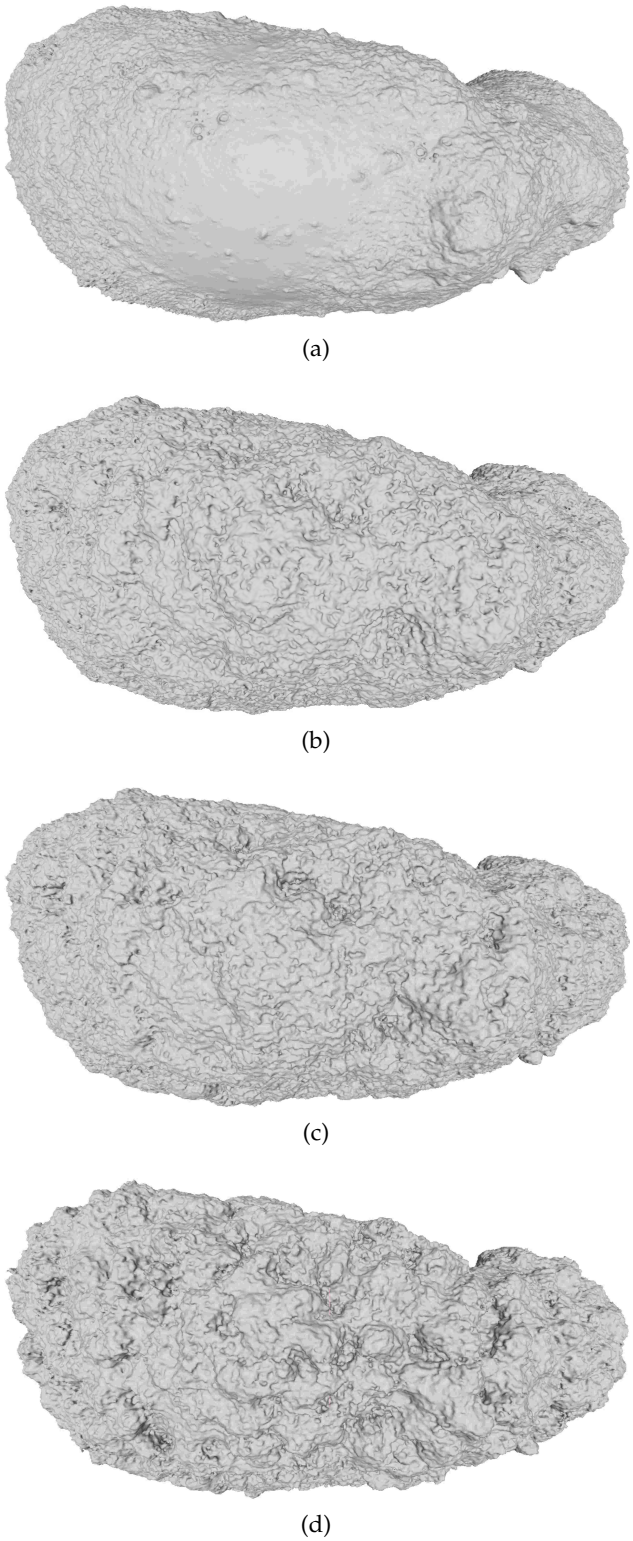
4.5 CONCLUSIONS AND FUTURE WORKS

We have presented the first fully automatic method to generate highly detailed realistic models of small celestial bodies. The main idea is to combine a two-tier approach of implicit shape representation

2 For more results please visit https://github.com/XZ-CG/asteroid-result

and different noise functions with an optimization algorithm. This enables us to "learn" from real world or hand crafted models and automatically generate an almost infinite number of small (or larger) variations. Even more, our results have shown that we can apply the learned parameters also to other, i. e. low-poly, models to generate a similar surface structure. AstroGen runs completely massively parallel on the GPU which indicates a high performance. As the trend in future space exploration tends to focus on objects in deep space, the importance of autonomy increases on-board of spacecraft. Hence, AstroGen could be an important step to enable decision making in virtual testbed by considering different scenarios and for the training of autonomous algorithms in space crafts. Moreover, our sphere based implicit surface be can easily extended to support a mascons-based model to simulate the gravity of asteroid more accurately, even within the Brouillon-sphere [139].

However, AstroGen also offers interesting avenues for future work. For instance, we want to investigate heteromorphic shape reproduction and improve the quality of the mesh in general. For instance, we want to consider different basic shapes, instead of spheres, for the implicit surface reconstruction with metaballs such as cubes, tori or ellipsoids. This can be further combined with a tree-like structure similar to BlobTrees [33]. For the surface details, we want to investigate other noise functions and more parameters, but also thermal erosion, hydraulic erosion algorithms are interesting [76]. This could improve the naturalness of AstroGen. Finally, we want to improve the mesh generation, for instance by adopting dual marching cube [130] to enhance the visual fidelity of the isosurfaces.



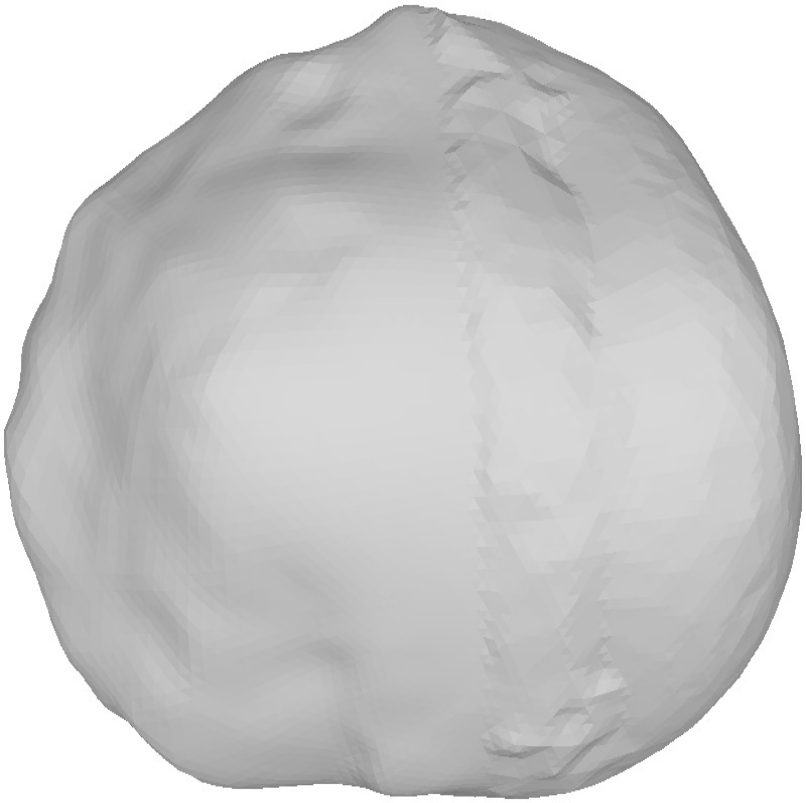
(a)

(b)

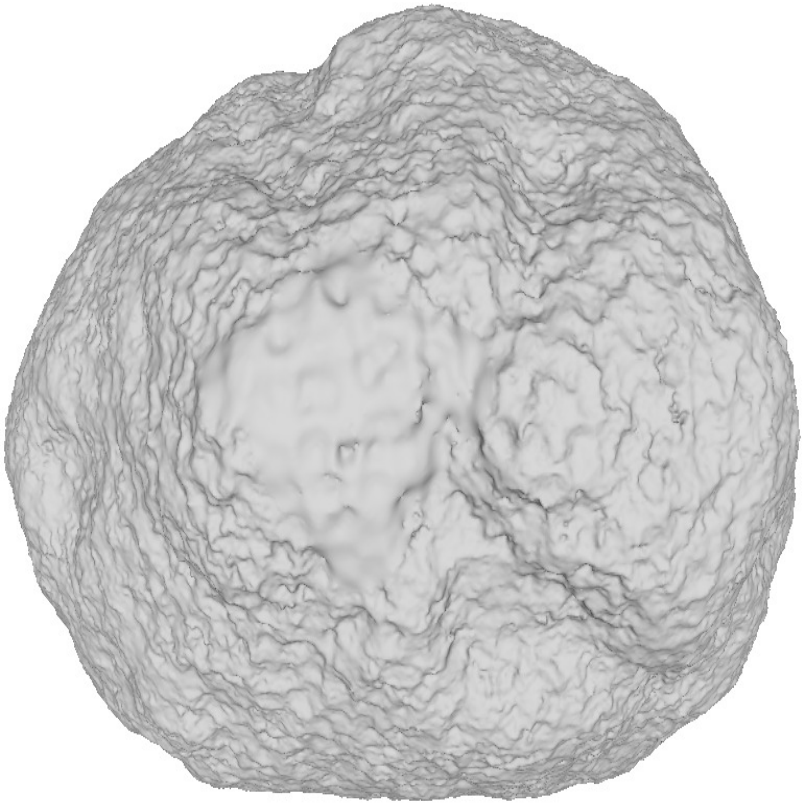
(c)

(d)

Figure 4.11: (a) The original model of the asteroid Itokawa (1,780k vertices). Generated asteroid models with a similarity of (b) 95% (1,986k vertices), (c) 90% (2,173k vertices) and (d) 85% (2,335k vertices).



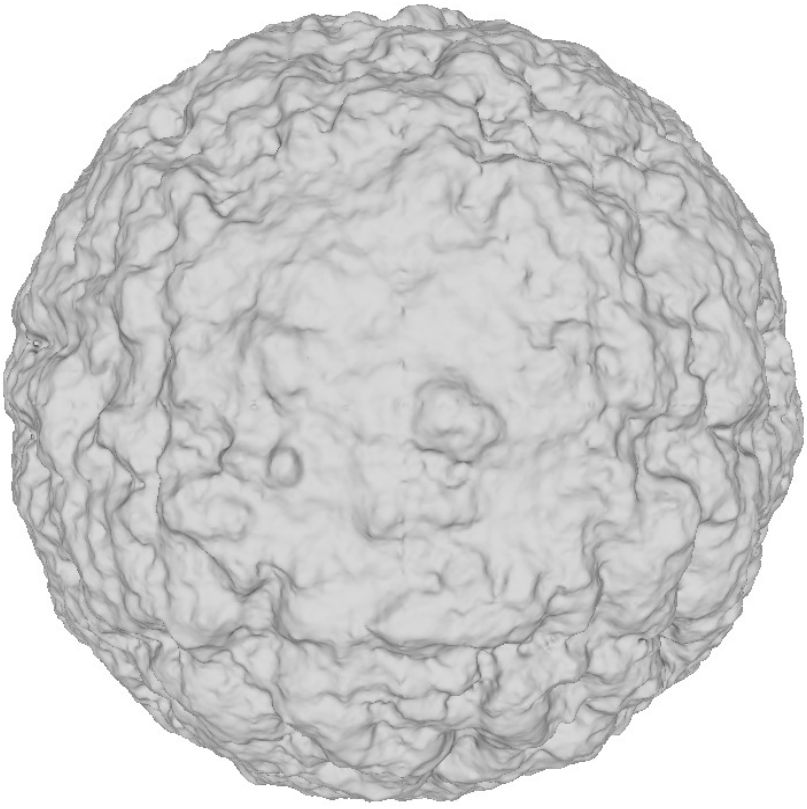
(a)



(b)

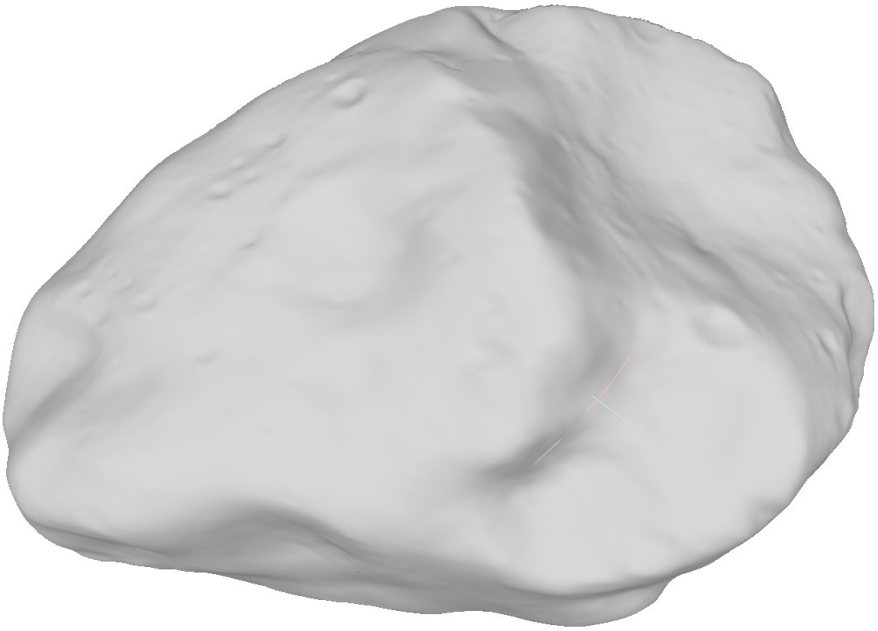
Figure 4.12: The original low-resolution Stein model (10k vertices) and our automatically generated model with surface details (710k vertices).

(a)

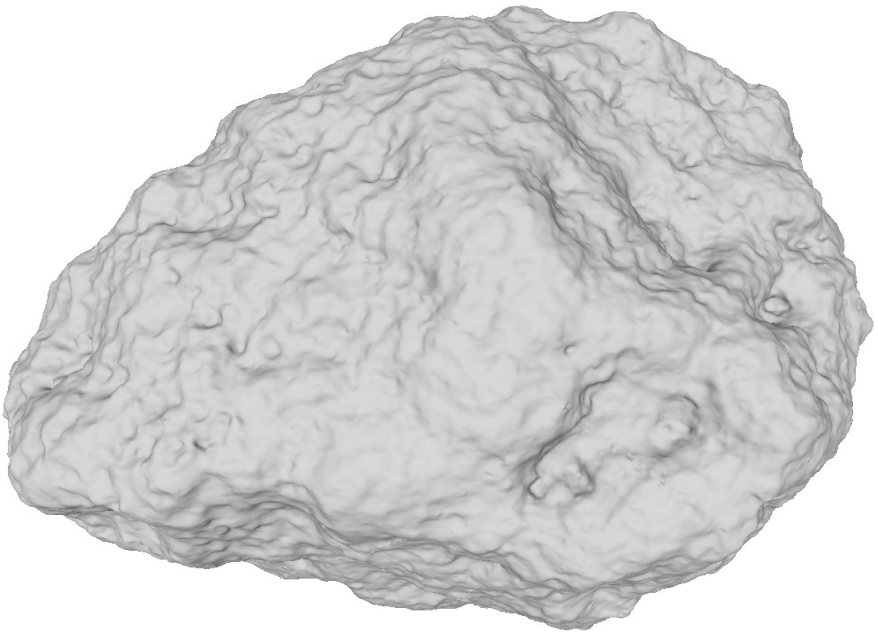


(b)

Figure 4.13: The original low-resolution Ceres model (128k vertices) and our automatically generated model with surface details (1,063k vertices).



(a)



(b)

Figure 4.14: The original low-resolution Lutetia model (122k vertices) and our automatically generated model with surface details (778k vertices).

5

PROCEDURAL 3D ASTEROID SURFACE DETAIL SYNTHESIS

In this chapter, we will present a novel noise model to procedurally generate volumetric terrain on implicit surfaces. The main idea is to combine a novel *Locally Controlled 3D Spot noise (LCSN)* for authoring the macro structures and 3D Gabor noise to add micro details. More specifically, a spatially-defined kernel formulation in combination with an impulse distribution enables the LCSN to generate arbitrary size craters and boulders, while the Gabor noise generates stochastic Gaussian details. The corresponding metaball positions in the underlying implicit surface preserve locality to avoid the globality of traditional procedural noise textures, which yields an essential feature that is often missing in procedural texture based terrain generators. Furthermore, different noise-based primitives are integrated through operators, i.e. blending, replacing, or warping into the complex volumetric terrain. The result is a completely implicit representation and, as such, has the advantage of compactness as well as flexible user control. We applied our method to generating high quality asteroid meshes with fine surface details.

5.1 INTRODUCTION

The evolving space technologies for planetary missions have grown from flyby observation to collect samples from celestial bodies. Studies of celestial bodies has progressed from observe their activities, i. e., orbit and spin to touching physical properties such as surface and subsurface materials, and the latter may provide insight into the Earth and the formation and early evolution of the solar system. For instance, currently there are two ongoing sample return missions: NASA's OSIRIS-REx and JAXA's Hayabusa2 study the asteroids Bennu and Ryugu, respectively. Both spacecraft are orbiting around the asteroid and has begun assessing the safety and sample-ability of potential sample collection site.

To ensure the success of such asteroid sampling missions it is essential to test the missions in advance with a wide variety of diverse scenarios of the hardly known asteroid shapes and surfaces. Further accurate data on robotic performance and sampling must be gathered. In pursuit of this goal, a method to synthesis huge amount of complex asteroid surface details as the virtual testbed is needed. This is typically done in virtual testbeds. In addition, the demanding for diverse high quality asteroid models in space video games, i. e., Kerbal Space Program, or movies has advanced significantly over the past decades. Hence, highly detailed 3D models of asteroids shapes and surfaces are required to cover a wide variety of possible scenarios. Obviously, also the video game or movie industry are interested in complex and

realistic high quality 3D models of celestial bodies. Generating them manually is not an option because of the large amount of test cases. Consequently, automatic shape generation is required.

The generation of diverse virtual testbeds for asteroid sampling mission is challenging for two reasons: first, the uneven distribution of asteroid's microgravity field leads to the terrain can no longer treated as the heightfield on a regular 2D grid but 3D heightfield embedded into a surface. Second, the surface of real asteroid presents locality which reflect their interior properties as well as the impact from outer space. For instance, Hayabusa2's detailed images of Ryugu reveal asymmetries of the asteroid, the western region is smoother than the eastern part [63]. Traditional procedural texture method generate similar pattern globally and require manually re-editing the texture to achieve locality.

As explained in Chapter 2.3, one of the most promising technique to alleviate the challenge of authoring complex asteroid models is procedural content generation. Proceduralism refers to a method that abstracts the underlying shape to be represented into by compact, elegant rule or equation that can be amplified on demand to produce complex models. Moreover, once those rules or equations have defined we can easily manipulate it to create a procedural workflow which will dramatically decrease the amount of manual intervention in complex space scene authoring. On the other hand, the compact essence of procedural method make it suitable for virtual testbed platform.

We introduce a novel noise model that combines *local controlled spot noise* (LCSN) [112] with *Gabor noise by example* (GNBE) [46] to solve the challenge of automatic asteroid surface detail generation. Our main contributions are:

- a new spatial defined kernel formulation of spot noise relying on several intuitive parameters to generate arbitrary sized craters and boulders and
- an integration of GNBE with our 3D noise model which can extract noise parameters from example textures.

The new noise model mix both spot noise focus on spatial macro structures i.e. craters or boulders of asteroid surface, and gabor noise generate gaussian textures as the micro terrain details. With procedural volumetric terrain generator to achieve a fully implicit representation of complex asteroid model.

In order to generate a rough shape of the celestial bodies we have modified AstroGen [83] which is a volumetric modeler that can use existing geometric models as constraints to implicitly approximate the shape by a polydisperse sphere packing [159]. Our method can be a supplement to the procedural workflow (see Fig. 5.2) to generate diverse complex surface details of celestial bodies (Figure 5.1 present the possibilities offered by our new noise model to generate volumetric terrain through the rough shape of several asteroids).

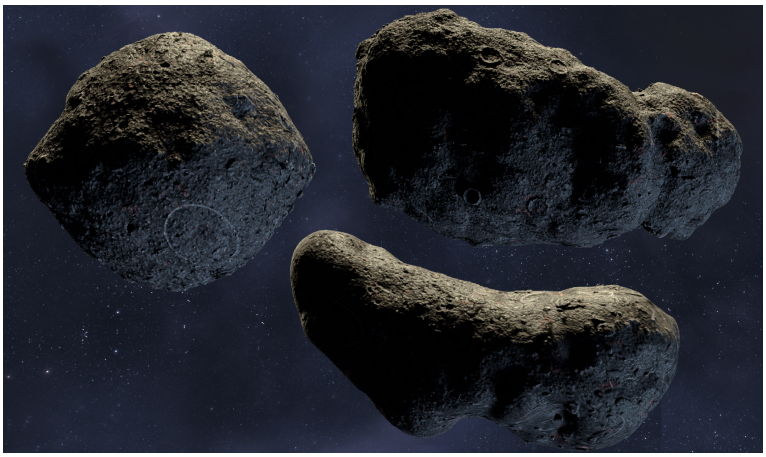


Figure 5.1: Asteroid field. All asteroid models are generated with our volumetric terrain model.

5.2 RELATED WORK

Procedural approaches have been widely used since the seminal work by Perlin in 1980s. Since then, it has shown extraordinary advantage in modeling, rendering and texture synthesis. Our work is related to procedural noise and procedural modeling.

Procedural noise gained attention and interests of many researchers in the past decades due to its efficiency, low storage requirements and its non-periodicity. A detailed overview of procedural noise functions can be found in [75]. Here, we review more recent developments which are directly related to our work.

Galerne and Lagae [46] proposed *Gabor noise by example*, a segmentation approach of the exemplar Gaussian texture's power spectrum into a sparse sum of Gaussian envelops, to generate noise with a similar power spectrum. However, the Gaussian texture fails to preserve large structures which is very important in many applications. Several attempts have been made to deal with structured patterns as well as the efficiency of the algorithm. Gilet et al. [55] introduced *local-random phase* (LRP) noise that demonstrated great ability to generate noise textures with spatial and spectral control. Their noise function was defined on a regular grid to sum localized cosine functions with random or deterministic phases, and the local phases of the input texture can be preserved to generate structure patterns. Subsequently, Pavie et al. [112] proposed *Locally Controlled Spot Noise* as an extension of the LRP noise model that is based on a controlled density profile distribution of Gaussian kernels to generate near-regular to irregular patterns. Galerne [47] proposed *texton noise* which improves the Gabor noise by example significantly in the aspect of parameter estimation and the efficiency. More recently, Tricard et al. [150] presented phasor noise which addressed the contrast oscillation problem and reformulates the Gabor noise into local density and phase, providing fine controls over the fine microstructures through editing the profile function. Cavalier et al. [24] further strengthened LCSN by introducing an intuitive geometric matrix formulation to the kernel and an anisotropic

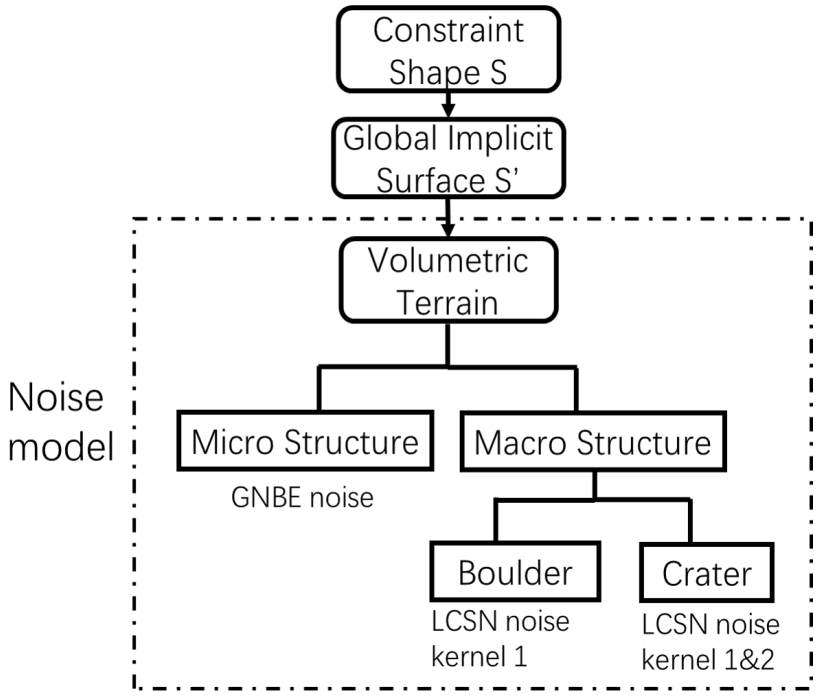


Figure 5.2: The workflow of our new procedural noise model.

filtering scheme which significantly improved the rendering result of procedural textures. We extend 2D Gabor noise by example to 3D procedural modeling, where the heightmap with structure pattern is used as exemplar to generate terrain on the base implicit surface.

Procedural modeling have been used in computer graphics for decades. This method represent models in a mathematical way, and despite the considerable progress in procedural modeling the terrain generation is still an open problem [53]. In large outdoor environments, the geometry of terrain landscapes requires significant storage and rendering bandwidth. In a procedural approach, rather than explicitly specifying and storing all the complex details of a scene or sequence, which abstract them into a function or an algorithm and evaluate that procedural when and where needed. As a results, systems offering procedural modeling techniques are under intensive studies,i. e. [30, 52, 53, 128]. Those techniques can be divided into two directions, fractal noise based terrain and physical based terrain.

First, fractal noise based terrains are created by summing band limited noise octaves each having a randomly varying amplitude. For instance, Musgrave [99] presents the famous fractal terrain using Perlin noise as the base function. [52] focus on Hydrology terrain simulation, they use fractal interpolation to connect predefined physical-based river network, mountain ridges and valleys of the terrain. Santamaria [128] introduce a method for volumetric terrain generation that utilizes Perlin noise for the terrain layer generation. Layers are the base parts of the resulting volumetric terrain, obtained by sampling Perlin noise do determine the layer thickness and shape. However, designing those lattice-based noise summations and mod-

eling to achieve desired results inherently unintuitive and requires extensive manual work.

Second, physics-based modeling enable the terrain landscape to cope with its environment and produce realistic motion and behavior. [53] introduced skeletal implicit surface with feature-based primitives (i. e., rivers, mountains) into a construction tree to describe complex terrains. [30] incorporate several phenomena, sunlight, temperature, prevailing wind direction with snow evolution to simulate the dynamic snow-covered terrain.

Although those algorithms can generate infinite realistic terrains, they either need intensive human intervention or less user control. Therefore, reproducing terrain features using a height map as a source which effectively cover an arbitrarily large area without seams or objectionable repetition while maintain the source height map's features can be a good option. As a result it would be useful if combining and parameterizing noise could be done automatically by using an example height map from which noise parameters are extracted. In this paper, we are the first to integrate implicit surface with Gabor noise by example [46, 47] to synthesize features of an example terrain topology as our base mesh and give user more control compared with the traditional Perlin noise based fractal terrain.

More closely, Martin [92] proposed a method to synthesis the crater and boulder on the surface of asteroid for the landing system.

5.3 OUR APPROACH

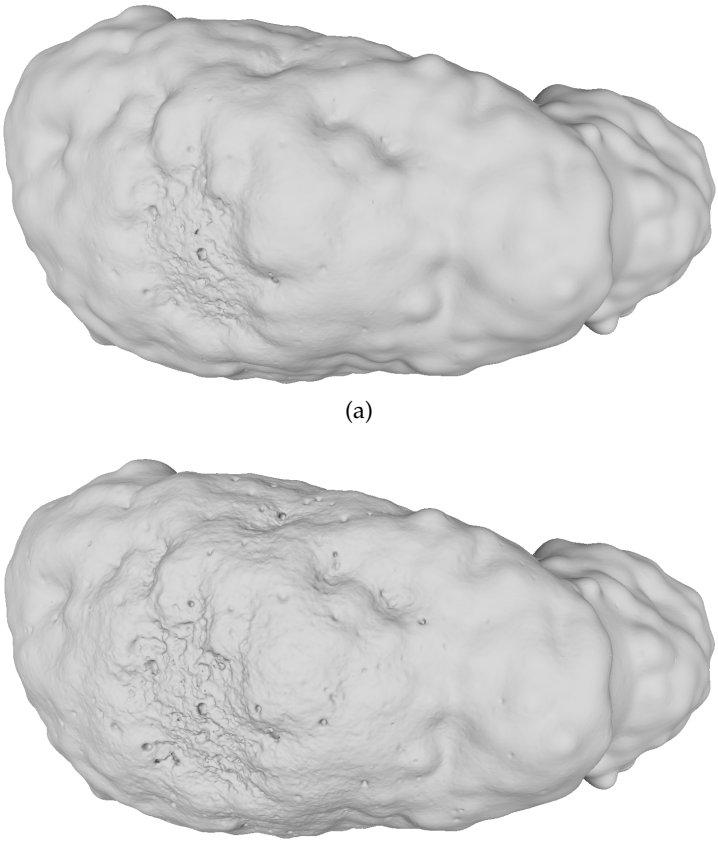
The basis of our implicit asteroid generation is a metaball representation defined by a sphere packing of the rough shape according to AstroGen [83] (details also explained in Chapter 4). Additionally, we add surface details via spot and Gabor noise. Hence, the complete asteroid can be represented as an isosurface of the function $S = \{p \in R^3 \,|\, f(p) = T\}$ with:

$$f(p) = m(p) + \sum_{i=1}^{I} dist(p - p_i) n_i(p) \tag{5.1}$$

where $m(p)$ represents the implicite surface of the metaballs for each point $p \in R^3$ in 3D. Additionally, we assign a noise function n_i to each sphere $i = 1, ..., I$ to control the influence of the individual spheres to the surface details. This guarantees the locality of our approach. Obviously, the influence also depends on the distance $(\|p - p_i\|_2 - r)$ of the sphere's center p_i to the point p. Each sphere i control the weight of noise pattern on its surface based on the distance ($dist(p) = w \cdot \tan^{-1}(dist(p - x_p))$, p_i represent the center of each metaballs, $\tan^{-1}$ to restrict the value into range $[0, \pi/2]$). In Figure 5.3 we showed the spatial heterogeneity effects of our inverse distance weight function.

The noise functions for the surface details $n_i(p)$ are a multi-layer of spot and Gabor noise functions and can be written as

$$n_i(p) = \sum_{j=1}^{J} spot\_noise_j(p) + \sum_{q=1}^{Q} gabor\_noise_q(p) \tag{5.2}$$



(a)

(b)

Figure 5.3: The distance factor $dist(p - p_i)$ control the influence area of noise layers on the surface of each inner metaballs, with the parameter w controls the size of influence area. For instance, in (a) the parameter $w = 1$ and (b) the parameter $w = 2$.

In the following, we will concentrate on the surface details $n_i(p)$. We refer the interested reader to Chapter 4 to find more about the computation of the metaballs shape $m(p)$.

5.3.1 *Macro Terrain Structure*

Recap: Locally controlled spot noise

Here we will go through the *locally controlled spot noise* [112] algorithm to make it better understanding our main idea. The LCSN aim at spatial macro characteristics that cannot be produced by the sole power spectrum definition. As Leeuw [34] mentioned that the spot noise is able to transfer the characteristic of the kernel, such as anisotropic/isotropic or shape contour, into the texture. Thus spot noise is able to produce a wide range of patterns with structural features. In other words, when the kernel contains some structures, this structure is shifted directly to the texture. The LCSN can be defined as:

$$spot\_noise(p) = \sum_{i}^{I} \sum_{j}^{J} w_j(p_{i,j}) kernel((p - p_{i,j})V^{-1}(p - p_{i,j})) \quad (5.3)$$

where $p_{i,j}$ is a random impluse position of the kernel, and matrix $V^{-1} = (MRS)^{-T}(MRS)^{-1}$ consist the information of shift (M), rotation (R), and scaling (S) matrix related to the underlying kernel function (i. e., Gaussian function $kernel = Ae^x$ and its extensions are the most commonly used functions). The outer summation I is a composition of several spot noise and each spot noise is able to model a specific set of patterns (see Fig. 5.4). That means several kernels can be consists into a more complex shape. Moreover, the particular pattern in the kernel of the spot noise can also be modulate into the lattice, by simply changing the shape of the lattice will result in diverse regular patterns (see Fig. 5.5) on the texture. The manipulation of the lattice (angle between the cell) introduce a new level of control over diversity of the generated texture and can be used to regulate the large scale structure.

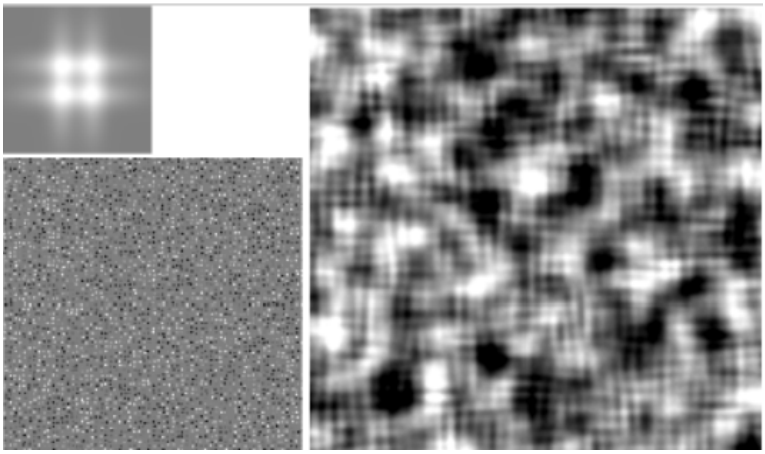


Figure 5.4: The left image is the kernel of the spot noise (summation of four ellipsoid gaussian function) and the right image is the corresponding noise [112].

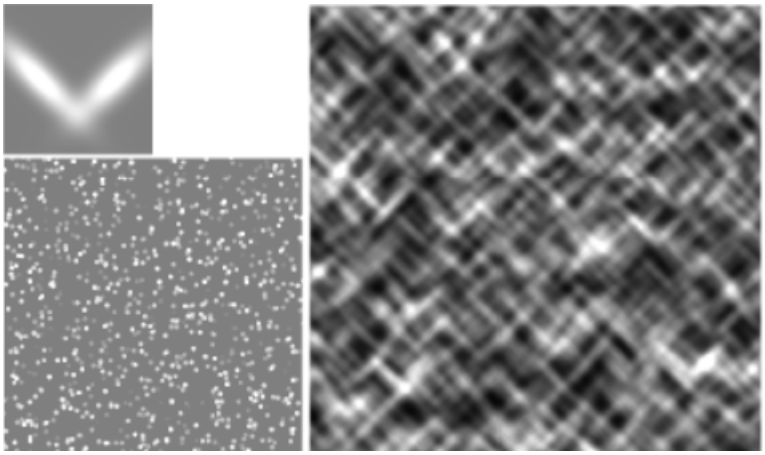


Figure 5.5: The left image is irregular lattice (change the angle of the cell) and the right image is texture from the spot noise kernel modulate onto the lattice [112].

Our Macro Terrain Structure

Recent data from the spacecraft OSIRIS-REx and Hayabusa2 proved the surface of both target asteroids are consistent with a rubble-pile structure, an aggregate of numerous boulder blocks [79, 141]. And another dominant terrain feature on asteroid is the crater, which occurs due to impacts and various in shape and depth of different asteroids [5]. The morphometry of these craters can be classified into two types: for the first type the impact happens on the bedrock and the second volcanic type where the impact on the soft surface and the material accumulate at the border. Here we focus on large and middle size boulders and craters through LCSN.

The LCSN model is a kind of sparse convolution noise introduced by Pavie et al. [112]. It relies on the summation of spatial kernels at uniform positions in texture space and the structural feature of the spatial kernel can be transferred to the texture. In order to generate two different types of craters we introduce two spatial kernels (see Eq. 5.5 and 5.6) as well as new control parameters for the synthesis of various shapes of craters:

$$spot\_noise_j(p) = \sum_{l=1}^{L} w_l(k_1(p - p_l) + k_2(p - p_l)) \tag{5.4}$$

with the two different kernels

$$k_1(p) = e^{-\alpha p^T p} \tag{5.5}$$

$$k_2(p) = \kappa e^{-\beta(\log(\gamma p^T p))^2} \tag{5.6}$$

where k_1 is a Gaussian kernel that generates normal craters where the parameter α controls the depth. In contrast, k_2 generates volcanic craters where κ controls the depth and β and γ define the radii of the outer and the inner ring, respectively. The individual kernels are combined with the weighting factors $w_l \in [0, 1]$ in Equation 5.4. p_l is a simple impulse following a Poisson distribution according to [112].

Moreover, we want to control the shape of the craters to allow also ellipsoidal shapes. Our approach is different from Pavie [112] by introducing the ellipsoidal gaussian kernel. Our basic idea is to simply truncate the pure gaussian kernel outside a certain area named kernel area to control the shape. By simply setting $k_{1,2} = 0$ in the case that $(p - p_l)^T V^{-1}(p - p_l) > \sigma$ for a diagonal matrix V and a user-definable influence radius σ we can easily achieve this.

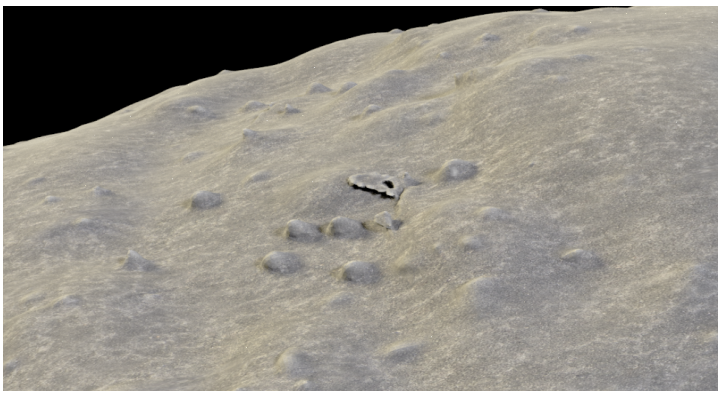
These definitions for our macro details have several advantages:

1. By introducing J layers of LCSN we can generate different sizes of craters simultaneously through the scaling of input point $(p - p_l)$ for each layer. Moreover, doubling the scale of the input point also scales the number of craters. This is according to the literature that states the distribution of large size craters and smaller size craters obey an exponential trend [79].

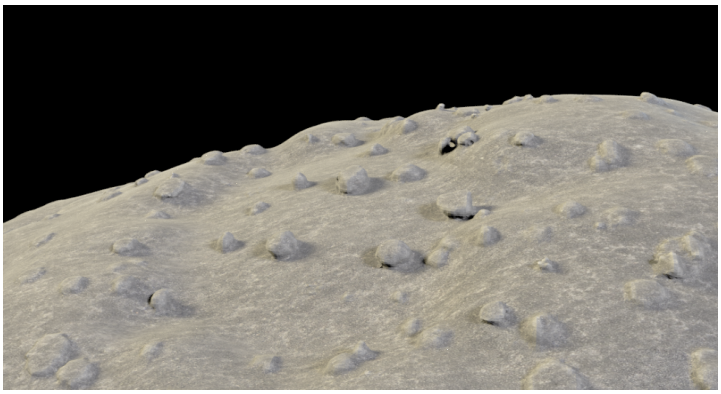
2. The Poisson distribution of impulses p_l controls the position of craters which leads to a random distribution of the craters (see Fig. 5.7a).

3. By introducing the kernel area σ we can achieve shape control while avoiding possible repetitions: normally, the structure generated by the spot kernel relies on a regular grid and within each grid cell, it generates similar pattern. In our case, the density of craters follows the user-definable probability within each grid cell. By defining the kernel area σ and the distribution range of the impulse we can easily control the occurrence of craters to break the repetition on the surface of the asteroid because the impulse position is not limited to one grid cell but includes neighbour areas.

For the generation of boulders we can simply inverse the value of the spot noise and design a different ellipsoidal shapes similar to boulders (see Fig. 5.6). This definition provides intuitive control of the result: the shape of craters (boulders) in each layer can be explicitly authored by setting up the diagonal matrix V, while the distribution of craters in each layer can be achieved by controlling the distribution of impulses and the size of kernel area σ.

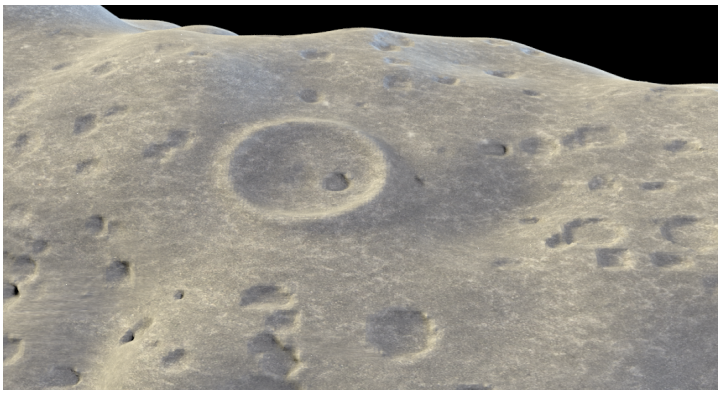


(a)

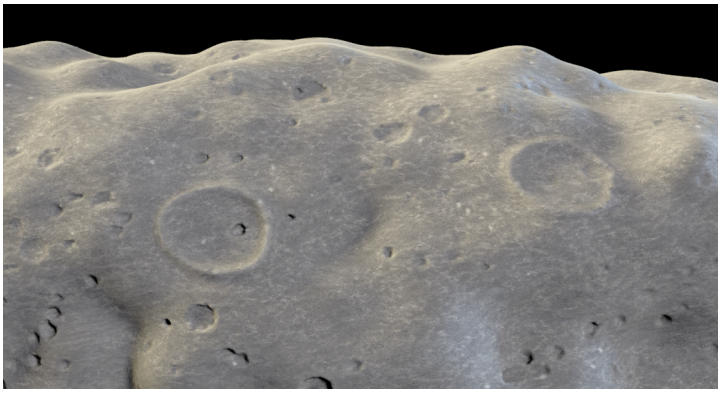


(b)

Figure 5.6: The image shows the corresponding boulders generated by inverting the value of our spot noise model. (a) The ellipsoid boulder generate by the kernel $k1$. (b) The ellipsoid boulder generate by the kernel $k1$ under the restriction of kernel area σ.



(a)



(b)

Figure 5.7: (a) The image shows some craters generated by our spot noise model on the implicit surface. With the intuitive noise parameters we can easily change the shape and density of the crater (The big volcanic crater we set $\kappa = -4.9$, $\beta = -10.0$, $\gamma = 0.2$, for the tiny normal craters we set $\alpha = 0.09$). (b) The image shows the effects of diagonal matrix $M\&V$ to the shape of volcanic crater.

5.3.2 *Micro Terrain Details*

Recap:Gabor noise by exmaple

Noise by example texture algorithm origin from the same property they own, as what is known their spatial domain in statistics can be seen as Gaussian random field. More specifically, both noise and example texture are completely characterized by their power spectrum. Here, the example texture specifically means Gaussian texture, a class of stochastic textures whose phase spectrum information are destroyed [45]. The problem of procedural noise by example can thus be solved by choosing a appropriate noise function and setting the parameters such that the power spectrum of the noise is similar to that of exemplar. The challenge for noise by example algorithm is to efficiently compute a precise approximation of a given Gaussian texture's power spectrum. The Gabor noise by example algorithm can be mainly divided into two step: Gabor noise function and power spectrum estimation.

Gabor noise was initially proposed by Lagae et.al [77], since it has a complete user control on the power spectrum that can be used as the basis to approximate arbitrary power spectra. The origin of their algorithm is the power spectrum of Gaussian texture can be decomposed into G symmetric pairs of Guassians patches. Due to the consistency of gaussian function under Fourier transform the corresponding Gabor noise in spatial domain can be evaluated with a combination of G noises into bandwidth-quantized Gabor noise. The two-dimensional anisotropic bandwidth-quantized Gabor noise [46] is defined as

$$g(x) = \sum_{b \in B} \frac{1}{\sqrt{\lambda_b}} \sum_i \frac{1}{\sqrt{p_{b,i}}} g(x - x_{b,i}; K_{b,i}, a_b, w_{b,i}, \phi_{b,i}) \tag{5.7}$$

where $a_b = 2^{-b}\sqrt{\frac{\pi}{2\ln 2}}$ [15] and $\lambda_b = \sum_g \lambda_{b,g}$ represent the parameters of each Gabor kernel $\{(K_{b,i}, w_{b,i})\}$ are randomly chosen from Gaussian patches G, under probability of $p_{b,g} = \lambda_{b,g}/\lambda_b$. The corresponding power spectrum of the noise $g(x)$ is

$$S_n(\varepsilon) = \sum_{b \in B} \sum_{g=0}^{G_b - 1} \frac{K_{b,g}^2}{8a_b^2} G(\varepsilon; \pm w_{b,g}, \frac{a_b}{2\sqrt{\pi}}) \tag{5.8}$$

From this first step we get the bandwidth-quantized Gabor noise $g(x)$ and its shape in power spectrum $S_n(\varepsilon)$. We need to determine the parameters of $S_n(\varepsilon)$ to partition the Gaussians into the power spectrum to fit with the power spectrum of the example Gaussian textures. For the given $M \times M$ exemplar's power spectrum $S_{ex}(\varepsilon_0, \varepsilon_1)(\varepsilon_0, \varepsilon_1 \in -\frac{M}{2}, \ldots, \frac{M}{2} - 1)$, by placing each discrete frequency $(m_0, m_1 \in -\frac{M}{2}, \ldots, \frac{M}{2} - 1)$ with a Gaussian under each bandwidth b with magnitude K. The goal of the parameter estimation is to find a parameter vector (b, m_0, m_1) satisfying three constraints:

1. the power spectrum of the noise with the exemplar must be close to each other,
2. all Gaussian's parameter must be positive, and
3. α is sparse, i.e. the representation is compact and computation efficient.

Those equations can be solved by using the fast iterative shrinkage thresholding algorithm (FISTA) [7], combined with the duality gap stopping criterion associated to the corresponding convex problem (see Eq.5.9 [46]) with the solver in Matlab. In Equation 5.9 the definition domain is a convex set $[0, +\infty)$ and the S_n as a set of gaussian functions is not convex but the quantized bandwidth b is a fixed set. The discrete, fixed size S_n becomes a convex problem [46]. As a result, the $f(\alpha)$ as the $\mathfrak{l}_2$-norm of S_n is also a convex problem and can be solved by FISTA. For more details of the algorithm see [46].

$$\begin{cases} minimize & f(\alpha) = \|S_n - S_{ex}\|_2^2 + \nu \|\alpha\|_1, \\ subject\ to & \alpha \geq 0 \end{cases} \tag{5.9}$$

Our Micro Terrain Details

Fractal-based terrain was introduced by Mandelbrot [91] in 1980s, which exhibits the self-similarity of terrain. Textures generally are referred to as a surface or area composed of repeating patterns and the fine details of terrain can be reproduced by textures.

Micro details of terrain can be defined as Gaussian random fields [106]. Gabor noise was initially proposed by Lagae et.al [77], since it has a complete user control on the power spectrum that can be used as the basis to approximate arbitrary power spectra. Subsequently, Galerne et al. [46] proposed a GNBE algorithm using a sparse representation of examplar's power spectrum and a bandwidth-quantized Gabor kernel for the efficient computation. According to Gilt et al. [55] and Pavie et al. [112] both LRP and LCSN noise models are able to generate textures from examples. However, LCSN and LRP models focus on synthesizing structured textures and the parameters have to be selected manually. Our goal is a sparse representation of fine terrain details without structures. Actually, GNBE allows a higher level of flexibility in controlling the final results.

First, We briefly present a summary of the GNBE approach and then introduce the details of our terrain generated by the GNBE.

By-example Gabor noise is based on a spectrum segmentation to G symmetric pairs of Gaussian patches, and extract the magnitudes and phases of structural energy to compute the approximation of gabor kernels. Therefore, the gabor noise by example algorithm can be mainly divided into two step: gabor noise function and power spectrum estimation. Galerne use a sparse representation of power spectrum to efficiently compute a precise approximation of a given gaussian texture's power spectrum. Due to the consistency of gaussian function under fourier transform the corresponding gabor noise in spatial domain can be evaluated with a combination of G noises into bandwidth-qauntized Gabor noise. The problem thus be solved by choosing the appropriate bandwidth-gabor kernel and setting the corresponding parameters such that the power spectrum of the noise is similar to that of exemplar.

The anisotropic bandwidth-quantized Gabor noise [46] is defined for each point $p \in R^3$ as

$$gabor\_noise_q(p) = \sum_{b \in B} \frac{1}{\sqrt{\lambda_b}} \sum_i \frac{1}{\sqrt{P_{b,i}}} kernel(p - p_i) \tag{5.10}$$

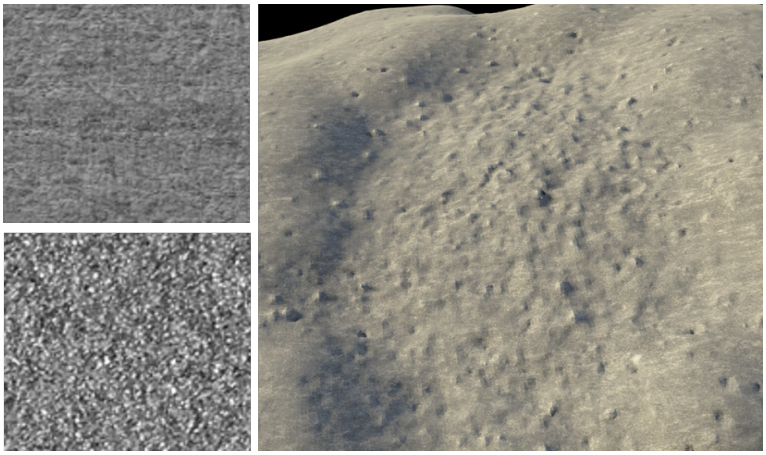
with

$$kernel(p - p_i) = Ke^{-\pi a^2 |p - p_i|^2} \cos(2\pi(p - p_i) \cdot w + \phi) \tag{5.11}$$

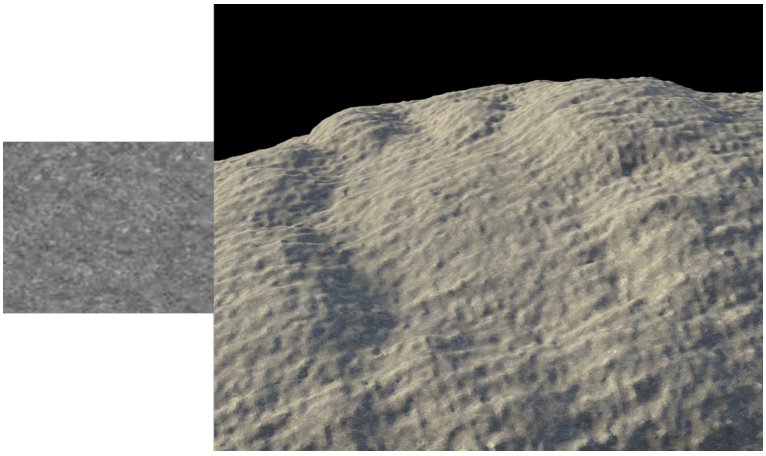
Where p_i denotes the impulse and follows a Poisson distribution. The corresponding impulse density λ in expected number of impulses N per kernel area πr^2, which corresponds to an impulse density of N/π impulses per voxel cell (see Fig. 4.9). The kernel parameter K, w, a and ϕ controls the amplitude, frequency, bandwidth, and phase of each Gabor kernel which are randomly chosen from G Gaussian

patches, under probability of $P_{b,g} = \lambda_{b,g}/\lambda_b$. Then the properties associated with the Gabor kernels, such as their amplitude and frequency, are randomly generated according to the corresponding probability distributions. Through quantize the bandwidth a into a small set of B bandwidths is able to simplify the parameter estimation, and the resolution of the fine details can be controlled.

By decomposing the example's power spectrum into a sparse sum of Gaussian patches using a non-negative basis pursuit denoising we can compute the parameters of the bandwidth-quantized Gabor noise to fit with the power spectrum of the example Gaussian textures [46]. Here, we evaluate the parameter of example textures (see Fig. 5.8 left) and apply the corresponding parameter into our 3D Gabor noise model to generate similar terrain details (see Fig. 5.8 right).



(a)



(b)

Figure 5.8: (a) The left column is the input image, the top image is the texture of a rocky surface and the bottom image shows a sand ground. The right column shows the corresponding terrain details from our example textures. (b) By applying different pattern of example input we can generate different surface details.

Like for the LCSN, we also extend the GNBE into three dimensions. The final result is a noise model on an implicit surface. Inspired by Lagae's et.al [77] surface noise, we simply project the Poisson distribution of impulses onto the tangent space of the implicit surface

and compute our noise model within each grid point's tangent space. Due to the essence of volumetric representation our volumetric terrain can easily generate caves or overhangs on the surface of the asteroid which is impossible for traditional 2D grid heightfield methods.

5.3.3 *Erosion*

In order to provide a proof-of-concept, we added a physically based erosion to our procedural terrain, even though erosion hardly affects asteroids in the real world. Such physical-based simulations are sensitive to distortions, hence a traditional 2D parameterizations does not produce good result in general, while our 3D grid terrain model fits very well with these kinds of simulations.

Temperature variation plays an important role in triggering the fracture of bedrock - referred to as thermal erosion. Therefore, erosion becomes one of the most important terrain modification algorithms. It is a simple simulation function performed on the entire terrain that sums target grid point's height difference for neighboring points, apply a height modification according to the terrain flow ability parameter, and adds some randomization to achieve a better natural look. The simulation is done by a cellular automation residing on the surface of the object. Because the point is built around a 3D surface, the average number of connected neighbors is 6.

The erosion algorithm was initially proposed by Musgrave [99] as a two-tire process: the hydraulic erosion and thermal weathering. Considering the inexistence of hydraulic erosion on asteroid but the thermal erosion is pervasive. Consequently, we have implemented a fractal thermal erosion algorithm proposed by Musgrave [99], which improves the level of realism in the surface details. This erosion model is a method to simulate the loose material falls down to pile up at the bottom. The result incline to generate a terrain with talus slopes of uniform angle. More specifically, at each time step $t+1$, the erosion model will compare the difference between the altitude a_t^v at the previous time step t of each vertex v and its neighbors u to the (global) constant talus angle T [99]. When the evaluated slope $a_t^v - a_t^u$ is larger than T, we will move a fixed number of extra slope onto the neighbor (see Eq. 5.12(1)).

$$a_{t+1}^u = \begin{cases} a_t^v - a_t^u > T : a_t^u + c_t(a_t^v - a_t^u - T) & (1) \\ a_t^v - a_t^u \leq T : a_t^u & (2) \end{cases} \tag{5.12}$$

We have integrated this effective and efficient erosion model with our procedural volumetric terrain which chips away steep terrain features and forms smooth talus slopes. As a consequence, we can see clearly that as the time evolution the sediment located at the area with higher slope eroded and the flat areas become smoother which is the same with the physical world observation (see Fig. 5.9).

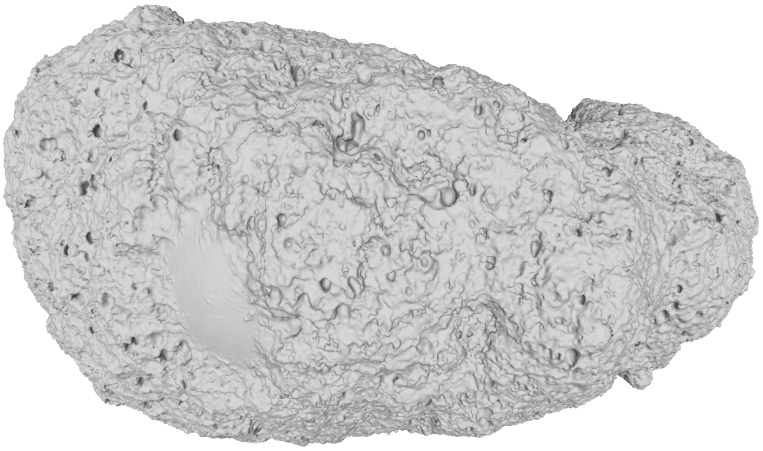


Figure 5.9: We simulate the erosion phenomenon on the surface of mountain-like terrain and we can see the tendency of sediment concentrate to area with higher slope (the mesh with 3,043K vertices).

5.4 RESULTS

We have implemented our system in C++ with the use of OpenGL and GLSL. All examples in this paper were created on a desktop computer equipped with Intel 3GHz Core i7 with 32GB RAM and an NVIDIA GeForce GTX 1080Ti. In order to generate a waterproof mesh we extract the isovalue points from a 3D grid and use a screened Poisson surface reconstruction algorithm [70] to generate the final mesh from the extracted point cloud. The output mesh was streamed into Blender to produce photorealistic asteroid surfaces (see Figures 5.1, 5.7a, 5.6, 5.8, 5.10, 5.11, 5.12).

5.5 CONCLUSIONS AND FUTURE WORK

We have presented a novel noise model for generating diverse terrain types on the surface of small celestial bodies. Our noise model combines LCSN with GNBE to synthesis the macro terrain features like craters or boulders and micro details. Moreover, our method provides local control from the metaball positions in the implicit surface, which avoids the globality of traditional procedural texture methods. Also, our approach makes it easy for non-professionals to almost automatically author diverse complex 3D models by changing just a few parameters.

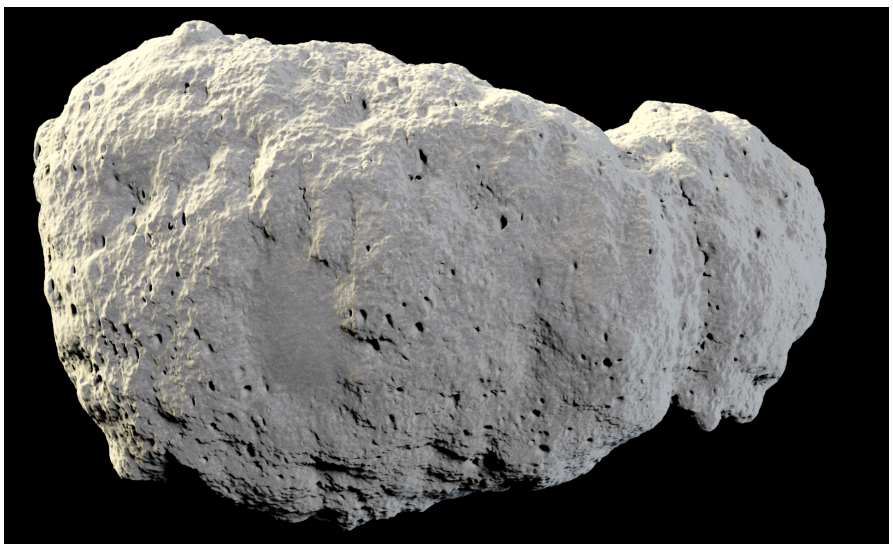
Through our procedural method we can build a fully procedural workflow , which can alleviate the demands for high-quality 3D models in the 3D industry, i. e., entertainment, 3D printing. Our volumetric object representation is very memory efficient: we just have to store a set of spheres together with a set of noise functions. the sphere packing representation automatically leads to a level-of-detail algorithm by considering the largest spheres first.

Our results show the quality of the generated terrains and the comprehensibility of the parameters. The performance is dominated by the evaluation of the Gabor noise and locally controlled spot noise

patches which is computationally more expensive than computing simple Perlin noise. Even though our implementation is currently not suitable for real-time editing, it can certainly be accelerated further, i. e., improving the computation of Gabor noise [143]. However, this has to be considered carefully to remain continuity.

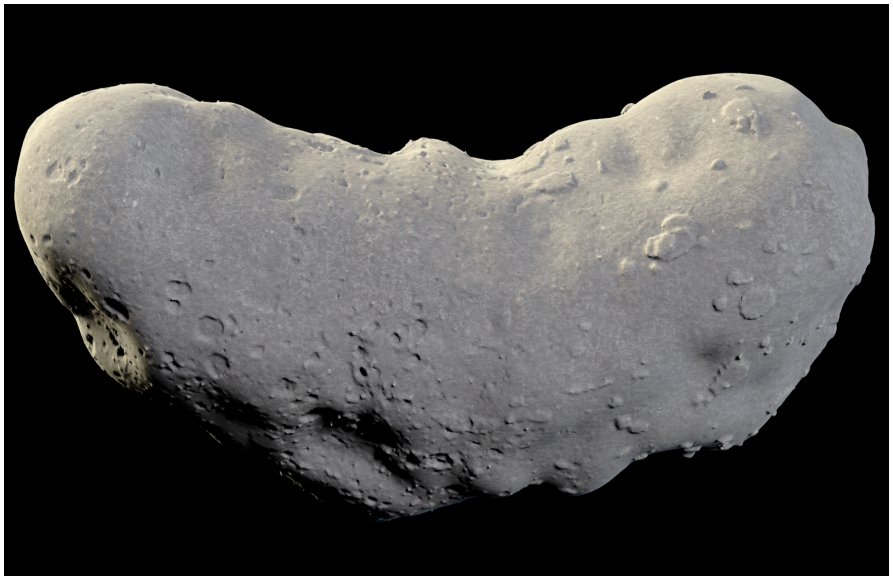
In the future, we plan to extend our method to more general use cases, such as terrain covered with icebergs or complex terrestrial terrains including overhangs and caves. Furthermore, our method can be integrated with other terrain synthesis approaches, such as the simulation of physical phenomena, i. e., erosion. Potentially, it could also be applied to more general physical simulations of elastic objects, with volume preservation or even topological changes, which would be essential, for instance, in medical applications.

(a)



(b)

Figure 5.10: Results from our procedural noise model applied to a low resolution Itokawa model.



(a)

(b)

Figure 5.11: Results from applying our implicite method to represent the asteroid Eros.(a) We show the local control with the right part boulders only and left part a mixture of micro details with different size craters.(b) We show a gravel tiled surface.

(a)

(b)

Figure 5.12: Our procedural noise model applied to a low resolution model of the asteroid Bennu.

Part III

EVERY END IS JUST A NEW BEGINNING

6

EPILOGUE

In this chapter, we will summarize the key contributions presented in this thesis at first. Then, we will venture to discuss possible research benefits in the field of procedural method and close areas. For the sake of simplicity, we will restrict the summary section to the basic concepts or results which appears in the previous chapters. You will find further specific explanation about those concepts or results in each separate sections of the relevant chapters. Nevertheless, you can also find the discussion of technical advance and extension of our procedural method, and conceivable applications in the corresponding chapters. In the future directions section, we try to depict a broaden overview of future research challenges of the procedural method in computer graphics.

6.1 SUMMARY

The work presented in this thesis can be mainly divided into two types of contribution to procedural modeling: firstly, automate the traditional manual implicit modeling process into a machine learning task; secondly, strengthen the accuracy of implicit modeling and widening the application of procedural modeling shapes that can be applied.

Procedural modeling as one of the enabling technology adopted by lots of applications where needs to deal with objects in motion or visualization. Often this method begins from a set of predefined skeleton vertices or hand drawing sketches defined by the designer and then the modeler assemble basic geometric primitives and blend them into a whole. This process is usually cubersome, expensive, low efficiency and can hardly cope with the widespread desire of procedural method. Moreover, the ever-increasing complexity of the photorealistic scene, which enabled by the exponential growth of the computing power (i. e., GPU, cloud computing), also brings continually growing demands on the procedural modeling method to relief the deficiency of more accurate collision detection algorithms.

We have introduced a new procedural method (AstroGen) to automatically implicitization any watertight constraint shape in order to ease the increasing demands in authoring arbitrary shapes as well as collision detections. It is partly based on the sphere-packing algorithm. However, we have optimized the blending function so it can better approximate the given shape as well as reduce manual interventions. The new method is able to make the creation of free-form surface easier and more intuitive.

The lack of details is one of the main problems of implicit surfaces. Another contribution we presented is a new noise model to remedy the drawback of lacking surface details, and from the noise model we

can synthesis desirable spatial geometric details. This noise model is based on the *Gabor noise by example* algorithm for the micro details and the *Locally Controlled Spot Noise* for the macro structures. The mixture of two kinds of noise provide a much larger type of patterns can be generated and especially well-suited for asteroid surface details synthesis. A coherent distribution of the noise pattern with respect to the underlying implicit surface is obtained through the position of inner metaballs. In addition, the procedural noise texture is parameterization free, so the geometric details in the noise function can be applied to the implicit surface seamlessly and with no distortion. Furthermore, the noise model enables a few intuitive parameters to precisely control the shape of the kernel which helps simulate the physically correct virtual testbed when working with real terrain simulation.

6.2 FUTURE DIRECTIONS

Recent AAA games (e. g., The Legend of Zelda [145]) already shows a fully immersive and interactive virtual environment that comparable to the real world. However, the pursue of efficiency and more precise photorealistic scene is endless. Nowadays, the advance in hardware and software offers possibilities that were unimaginable just in a few years. In this section, we will present some short-term objectives targeting the more efficient and realistic procedural modeling algorithms, with particular attention on the shape approximation, visualization and animation that will probably concern the computer graphic research community in the next few years.

6.2.1 *Modeling with Artificial Intelligence*

Graphs are a kind of data structure which models a set of objects (nodes) and their neighbor connections (edges). As an unique non-euclidean data structure the graph neural network (GNN) focuses on node classification, link prediction, and clustering [18]. GNN is a connection model that capture the dependency information within the graph via message passing between nodes. Different from the standard neural networks, graph neural networks utilize the state to preserve neighborhood information with arbitrary depth. Although the early GNN has been found difficult to train for a fixed point [138], recent advances in network architectures, optimization techniques, and parallel computation have enabled successful learning with them [86]. Our previous metaball-based representation is an ideal case for building graphs. For instance, the position of each metaballs (nodes) and the connection of their neighbor balls (edges) are able to build a graph. Subsequently, we can directly apply GNN forward or backward operators to do the node classification and prediction. There are many great advantages to solve the problem in graph domain, because 1) graph is the most intuitive structure to represent the locally connected non-euclidean structure; 2) sharing weights between nodes will significantly decrease the computation cost compared with the traditional

spectral-based graph theory [138]; 3) multi-layer structure of the graph is key to manage hierarchical patterns. All those advantages possible helps our optimization of metaball-based rough shape more accurate and also speed up the convergence process.

6.2.2 *Visualization*

It would be important to improve the efficiency and accuracy of the visualization. The core idea of the ray-tracing algorithm is to test the intersection between the surface and the individual ray. In our research, the virtual scene can be represented with an implicit equation where the intersection can be evaluated in linear time O(n). Moreover, we can also compute the intersection in advance or in demand to further reduce the computation time. In addition, our representation can be integrated with the accelerate data structure (i. e., octree, sphere tree) to further decrease the time consuming of ray-tracing especially in the complex scene. Meanwhile, we can assign various optical properties to the implicit surface to boost the visual qualities during the ray-tracing process.

6.2.3 *Animation*

There is probably some work to be done on animation with procedural representation surfaces. For instance, it will be interesting to introduce constraints such as constant volume deformation during animation. Different deformation states associate with different parameter spaces, and we can build a smooth interpolation between those parameter spaces to achieve continuous movements under the deformation. Another interesting topic is to improve the user's manipulation, for instance it should taken into account how the particular sets of parameters manipulate their counter part skeletons (i. e., a set of parameters to manipulate muscles on an arm), and we can create some animations directly from those parameters. Hierarchical deformation will also require procedural method: for instance, if a skeleton inflated into a surface from an existing shape, this property should be retained during animation. Lastly, our surface details can be applied to animation, such as the avatar's hair can be generated from some physically-based noise model. More specifically, procedural noise model is able to generate wisps or other sharp features that able to freely and smoothly merged or divided during animation.

As a conclusion, we hope that this thesis will help procedural method becomes more popular in all areas of computer graphics.

Part IV

APPENDIX

BIBLIOGRAPHY

[1] Baptiste Angles, Marco Tarini, Brian Wyvill, Loïc Barthe, and Andrea Tagliasacchi. "Sketch-based implicit blending." In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 181.

[2] Michael Ashikhmin. "Synthesizing natural textures." In: *Proceedings of the 2001 symposium on Interactive 3D graphics*. 2001, pp. 217–226.

[3] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. "The wave kernel signature: A quantum mechanical approach to shape analysis." In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1626–1633.

[4] Aurélien Barbier, Eric Galin, and Samir Akkouche. "A framework for modeling, animating, and morphing textured implicit models." In: *Graphical Models* 67.3 (2005), pp. 166–188.

[5] OS Barnouin, MG Daly, EE Palmer, RW Gaskell, JR Weirich, CL Johnson, MM Al Asad, JH Roberts, ME Perry, HCM Susorney, et al. "Shape of (101955) Bennu indicative of a rubble pile with internal stiffness." In: *Nature geoscience* 12.4 (2019), p. 247.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "Speeded-up robust features (SURF)." In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[7] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems." In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[8] Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, and Brian Wyvill. "Implicit blending revisited." In: *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, pp. 367–375.

[9] Pravin Bhat, Stephen Ingram, and Greg Turk. "Geometric texture synthesis by example." In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 2004, pp. 41–44.

[10] James F Blinn. "Simulation of wrinkled surfaces." In: *ACM SIGGRAPH computer graphics* 12.3 (1978), pp. 286–292.

[11] James F Blinn. "A generalization of algebraic surface drawing." In: *ACM transactions on graphics (TOG)* 1.3 (1982), pp. 235–256.

[12] James F Blinn. "A generalization of algebraic surface drawing." In: *ACM transactions on graphics (TOG)* 1.3 (1982), pp. 235–256.

[13] Jules Bloomenthal. "Bulge elimination in convolution surfaces." In: *Computer Graphics Forum*. Vol. 16. 1. Wiley Online Library. 1997, pp. 31–41.

[14] Jules Bloomenthal. "Medial-based vertex deformation." In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2002, pp. 147–151.

[15] Alan C. Bovik, Marianna Clark, and Wilson S. Geisler. "Multichannel texture analysis using localized spatial filters." In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 1 (1990), pp. 55–73.

[16] Gareth Bradshaw and Carol O'Sullivan. "Adaptive medial-axis approximation for sphere-tree construction." In: *ACM Transactions on Graphics (TOG)* 23.1 (2004), pp. 1–26.

[17] Robert Bridson, Jim Houriham, and Marcus Nordenstam. "Curl-noise for procedural fluid flow." In: *ACM Transactions on Graphics (ToG)* 26.3 (2007), 46–es.

[18] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." In: *arXiv preprint arXiv:1312.6203* (2013).

[19] Thomas Buffet, Damien Rohmer, Loïc Barthe, Laurence Boissieux, and Marie-Paule Cani. "Implicit untangling: a robust solution for modeling layered clothing." In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–12.

[20] Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. "Seamless parametrization with arbitrary cones for arbitrary genus." In: *ACM Transactions on Graphics (TOG)* 39.1 (2019), pp. 1–19.

[21] Florian Canezin. "Study of the composition models of field functions in computer graphics." PhD thesis. 2016.

[22] Florian Canezin, Gaël Guennebaud, and Loïc Barthe. "Adequate inner bound for geometric modeling with compact field functions." In: *Computers & graphics* 37.6 (2013), pp. 565–573.

[23] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. "Reconstruction and representation of 3D objects with radial basis functions." In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 67–76.

[24] Arthur Cavalier, Guillaume Gilet, and Djamchid Ghazanfarpour. "Local spot noise for procedural surface details synthesis." In: *Computers & Graphics* 85 (2019), pp. 92–99.

[25] Matthaus G Chajdas, Christian Eisenacher, Marc Stamminger, Sylvain Lefebvre, Pau Panareda Busto, Lefebvre Sylvain, Ares Lagae, Philip Dutré, Laurent Alonso, Edward M Reingold, et al. "Virtual texture mapping 101." In: *GPU Pro* (2010).

[26] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. "Shapenet: An information-rich 3d model repository." In: *arXiv preprint arXiv:1512.03012* (2015).

[27] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. "On visual similarity based 3D model retrieval." In: *Computer graphics forum*. Vol. 22. 3. Wiley Online Library. 2003, pp. 223–232.

[28] Robert L Cook, Loren Carpenter, and Edwin Catmull. "The Reyes image rendering architecture." In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987), pp. 95–102.

[29] Robert L Cook and Tony DeRose. "Wavelet noise." In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 803–811.

[30] Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and M-P Cani. "Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches." In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 497–509.

[31] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. "Efficient and flexible sampling with blue noise properties of triangular meshes." In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012), pp. 914–924.

[32] Tal Darom and Yosi Keller. "Scale-invariant features for 3-D mesh models." In: *IEEE Transactions on Image Processing* 21.5 (2012), pp. 2758–2769.

[33] Erwin PH De Groot. *Blobtree modelling*. University of Calgary, 2008.

[34] Wim De Leeuw and Robert Van Liere. "Divide and conquer spot noise." In: *Proceedings of the 1997 ACM/IEEE conference on Supercomputing*. 1997, pp. 1–13.

[35] *Distance Function 3D*. https://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm. Accessed: 2020-04-23.

[36] Nira Dyn, Kai Hormann, Sun-Jeong Kim, and David Levin. "Optimizing 3D triangulations using discrete curvature analysis." In: *Mathematical methods for curves and surfaces* 38.8 (2001), pp. 135–146.

[37] Mohamed S Ebeida, Scott A Mitchell, Andrew A Davidson, Anjul Patney, Patrick M Knupp, and John D Owens. "Efficient and good Delaunay meshes from random points." In: *Computer-Aided Design* 43.11 (2011), pp. 1506–1515.

[38] Russell Eberhart and James Kennedy. "A new optimizer using particle swarm theory." In: *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE. 1995, pp. 39–43.

[39] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley, William R. Mark, and John C. Hart. *Texturing and Modeling: A Procedural Approach: Third Edition*. United States: Elsevier Inc., 2003. ISBN: 9781558608481.

[40] Jorda Farnham. *SHAPE MODEL OF ASTEROID 2867 STEINS, RO-A-OSINAC/OSIWAC-5-STEINS-SHAPE-V1.0*. NASA Planetary Data System, 2013.

[41] Mark Fox, Callum Galbraith, and Brian Wyvill. "Efficient use of the BlobTree for rendering purposes." In: *Proceedings International Conference on Shape Modeling and Applications*. IEEE. 2001, pp. 306–314.

[42] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. "Modeling by example." In: *ACM Transactions on Graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 652–663.

[43] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. "A search engine for 3D models." In: *ACM Transactions on Graphics (TOG)* 22.1 (2003), pp. 83–105.

[44] Takahiko Furuya and Ryutarou Ohbuchi. "Deep Aggregation of Local 3D Geometric Features for 3D Model Retrieval." In: *BMVC*. Vol. 7. 2016, p. 8.

[45] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. "Random phase textures: Theory and synthesis." In: *IEEE Transactions on image processing* 20.1 (2011), pp. 257–267.

[46] Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. "Gabor noise by example." In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 73.

[47] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. "Texton noise." In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 205–218.

[48] Eric Galin, Eric Guérin, Axel Paris, and Adrien Peytavie. "Segment Tracing Using Local Lipschitz Bounds." In: *Computer Graphics Forum*. 2020.

[49] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. "A review of digital terrain modeling." In: *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 553–577.

[50] Manuel N Gamito and F Kenton Musgrave. "Procedural landscapes with overhangs." In: *10th Portuguese Computer Graphics Meeting*. Vol. 2. 3. Citeseer. 2001.

[51] Jean-David Génevaux, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedrich Benes. "Terrain generation using procedural models based on hydrology." In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 143.

[52] Jean-David Génevaux, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedrich Benes. "Terrain generation using procedural models based on hydrology." In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 143.

[53] Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. "Terrain modelling from feature primitives." In: *Computer Graphics Forum*. Vol. 34. 6. Wiley Online Library. 2015, pp. 198–210.

[54] Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. "Terrain modelling from feature primitives." In: *Computer Graphics Forum*. Vol. 34. 6. Wiley Online Library. 2015, pp. 198–210.

[55] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. "Local random-phase noise for procedural texturing." In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), p. 195.

[56] Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. "A gradient-based implicit blend." In: *ACM Transactions on Graphics (TOG)* 32.2 (2013), pp. 1–12.

[57] Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. "Sparse representation of terrains for procedural modeling." In: *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, pp. 177–187.

[58] Stefan Gustavson. "Simplex noise demystified." In: *Linköping University, Linköping, Sweden, Research Report* (2005).

[59] Stefan Gustavson. *"Cellular Noise in GLSL: Implementation Notes*. 2011.

[60] John C Hart. "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces." In: *The Visual Computer* 12.10 (1996), pp. 527–545.

[61] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. "Triplet-center loss for multi-view 3d object retrieval." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1945–1954.

[62] John R Hershey and Peder A Olsen. "Approximating the Kullback Leibler divergence between Gaussian mixture models." In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–317.

[63] Masatoshi Hirabayashi, Eri Tatsumi, Hideaki Miyamoto, Goro Komatsu, Seiji Sugita, Sei-ichiro Watanabe, Daniel J Scheeres, Olivier S Barnouin, Patrick Michel, Chikatoshi Honda, et al. "The western bulge of 162173 Ryugu formed as a result of a rotationally driven deformation process." In: *The Astrophysical Journal Letters* 874.1 (2019), p. L10.

[64] Kai Hormann, Bruno Lévy, and Alla Sheffer. "Mesh parameterization: Theory and practice." In: (2007).

[65] Samuel Hornus, Alexis Angelidis, and Marie-Paule Cani. "Implicit modelling using subdivision-curves." In: (2003).

[66] Alex Jarauta, Pavel Ryzhakov, Jordi Pons-Prats, and Marc Secanell. "An implicit surface tension model for the analysis of droplet dynamics." In: *Journal of Computational Physics* 374 (2018), pp. 1196–1218.

[67] Xiaogang Jin, Chiew-Lan Tai, Jieqing Feng, and Qunsheng Peng. "Convolution surfaces for line skeletons with polynomial weight distributions." In: *Journal of Graphics Tools* 6.3 (2001), pp. 17–28.

[68] Martin Jutzi, Keith Holsapple, Kai Wünneman, and Patrick Michel. "Modeling asteroid collisions and impact processes." In: *arXiv preprint arXiv:1502.01844* (2015).

[69] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. "GPU-based fast ray casting for a large number of metaballs." In: *Computer Graphics Forum*. Vol. 27. 2. Wiley Online Library. 2008, pp. 351–360.

[70] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction." In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), p. 29.

[71] Aaron Knoll, Younis Hijazi, Andrew Kensler, Mathias Schott, Charles Hansen, and Hans Hagen. "Fast ray tracing of arbitrary implicit surfaces with interval and affine arithmetic." In: *Computer Graphics Forum*. Vol. 28. 1. Wiley Online Library. 2009, pp. 26–40.

[72] Iasonas Kokkinos, Michael Bronstein, and Alan Yuille. "Dense scale invariant descriptors for images and surfaces." PhD thesis. INRIA, 2012.

[73] Alex Krizhevsky and Geoffrey E Hinton. "Using very deep autoencoders for content-based image retrieval." In: *ESANN*. 2011.

[74] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. "A survey of procedural noise functions." In: *Computer Graphics Forum*. Vol. 29. 8. Wiley Online Library. 2010, pp. 2579–2600.

[75] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. "A survey of procedural noise functions." In: *Computer Graphics Forum*. Vol. 29. 8. Wiley Online Library. 2010, pp. 2579–2600.

[76] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony Derose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. "State of the art in procedural noise functions." In: *EG 2010-State of the Art Reports*. The Eurographics Association. 2010.

[77] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. "Procedural noise using sparse Gabor convolution." In: *ACM Transactions on Graphics (TOG)* 28.3 (2009), p. 54.

[78] Patrick Lange, Rene Weller, and Gabriel Zachmann. "Multi agent system optimization in virtual vehicle testbeds." In: *SimuTools*. 2015, pp. 79–88.

[79] DS Lauretta, DN DellaGiustina, CA Bennett, DR Golish, KJ Becker, SS Balram-Knutson, OS Barnouin, TL Becker, WF Bottke, WV Boynton, et al. "The unexpected surface of asteroid (101955) Bennu." In: *Nature* 568.7750 (2019), p. 55.

[80] Sylvain Lefebvre and Hugues Hoppe. "Appearance-space texture synthesis." In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 541–548.

[81] Victor Lempitsky, Michael Verhoek, J Alison Noble, and Andrew Blake. "Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography." In: *International Conference on Functional Imaging and Modeling of the Heart*. Springer. 2009, pp. 447–456.

[82] Chunyuan Li and A Ben Hamza. "Spatially aggregating spectral descriptors for nonrigid 3D shape retrieval: a comparative survey." In: *Multimedia Systems* 20.3 (2014), pp. 253–281.

[83] Xi-zhi Li, René Weller, and Gabriel Zachmann. "AstroGen–Procedural Generation of Highly Detailed Asteroid Models." In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE. 2018, pp. 1771–1778.

[84] Xin Li, Xiaohu Guo, Hongyu Wang, Ying He, Xianfeng Gu, and Hong Qin. "Harmonic volumetric mapping for solid modeling applications." In: *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. 2007, pp. 109–120.

[85] Xizhi Li, Patrick Lange, René Weller, and Gabriel Zachmann. "Invariant local shape descriptors: classification of large-scale shapes with local dissimilarities." In: *Proceedings of the Computer Graphics International Conference*. ACM. 2017, p. 9.

[86] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. "Gated graph sequence neural networks." In: *arXiv preprint arXiv:1511.05493* (2015).

[87] Tsz-Wai Rachel Lo and J Paul Siebert. "Local feature extraction and matching on range images: 2.5 D SIFT." In: *Computer Vision and Image Understanding* 113.12 (2009), pp. 1235–1250.

[88] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm." In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.

[89] David G Lowe. "Object recognition from local scale-invariant features." In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[90] Ives Macedo, Joao Paulo Gois, and Luiz Velho. "Hermite radial basis functions implicits." In: *Computer Graphics Forum*. Vol. 30. 1. Wiley Online Library. 2011, pp. 27–42.

[91] Benoit B Mandelbrot. *The fractal geometry of nature*. Vol. 173. WH freeman New York, 1983.

[92] Iain Martin, Steve Parkes, Martin Dunstan, and Nick Rowell. "Asteroid Modeling for Testing Spacecraft Approach and Landing." In: *IEEE computer graphics and applications* 34.4 (2014), pp. 52–62.

[93] Jonàs Martínez, Jérémie Dumas, and Sylvain Lefebvre. "Procedural voronoi foams for additive manufacturing." In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–12.

[94] Jon McCormack and Andrei Sherstyuk. "Creating and rendering convolution surfaces." In: *Computer Graphics Forum*. Vol. 17. 2. Wiley Online Library. 1998, pp. 113–120.

[95] Susan K McMahon. "Overview of the planetary data system." In: *Planetary and Space Science* 44.1 (1996), pp. 3–12.

[96] G Wyvill C McPheeters and Brian Wyvill. "Data structure for soft objects." In: *The Visual Computer* 2.4 (1986), pp. 227–234.

[97] Harvey J Miller. "Tobler's first law and spatial analysis." In: *Annals of the Association of American Geographers* 94.2 (2004), pp. 284–289.

[98] Don P Mitchell. "Robust ray intersection with interval arithmetic." In: *Proceedings of Graphics Interface*. Vol. 90. 1990, pp. 68–74.

[99] F Kenton Musgrave, Craig E Kolb, and Robert S Mace. "The synthesis and rendering of eroded fractal terrains." In: *ACM Siggraph Computer Graphics*. Vol. 23. 3. ACM. 1989, pp. 41–50.

[100] Hubert Nguyen. *Gpu gems 3*. Addison-Wesley Professional, 2007.

[101] Gregory M Nielson. "Dual marching cubes." In: *IEEE Visualization 2004*. IEEE. 2004, pp. 489–496.

[102] Hitoshi Nishimura. "Object modeling by distribution function and a method of image generation." In: *Trans Inst Electron Commun Eng Japan* 68 (1985), p. 718.

[103] Tomoyuki Nishita and Eihachiro Nakamae. "A method for displaying metaballs by using Bézier clipping." In: *Computer Graphics Forum*. Vol. 13. 3. Wiley Online Library. 1994, pp. 271–280.

[104] Afonso Paiva, Hélio Lopes, Thomas Lewiner, and Luiz Henrique De Figueiredo. "Robust adaptive meshes for implicit surfaces." In: *2006 19th Brazilian symposium on computer graphics and image processing*. IEEE. 2006, pp. 205–212.

[105] Julian Panetta, Abtin Rahimian, and Denis Zorin. "Worst-case stress relief for microstructures." In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–16.

[106] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[107] Ian Parberry. "Designer worlds: Procedural generation of infinite terrain from real-world elevation data." In: *Journal of Computer Graphics Techniques* 3.1 (2014).

[108] Axel Paris, Eric Galin, Adrien Peytavie, Eric Guérin, and James Gain. "Terrain Amplification with Implicit 3D Features." In: *ACM Transactions on Graphics (TOG)* 38.5 (2019), pp. 1–15.

[109] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. "Function representation in geometric modeling: concepts, implementation and applications." In: *The Visual Computer* 11.8 (1995), pp. 429–446.

[110] Giuseppe Patané and Michela Spagnuolo. "State-of-the-art and perspectives of geometric and implicit modeling for molecular surfaces." In: *Computational Electrostatics for Biological Applications*. Springer, 2015, pp. 157–176.

[111] Jen Lowe Patricio Gonzalez Vivo. *The book of shaders*. https://thebookofshaders.com/. 2015.

[112] Nicolas Pavie, Guillaume Gilet, Jean-Michel Dischler, and Djamchid Ghazanfarpour. "Procedural texture synthesis by locally controlled spot noise." In: (2016).

[113] Ken Perlin. "An image synthesizer." In: *ACM Siggraph Computer Graphics* 19.3 (1985), pp. 287–296.

[114] Ken Perlin. "Noise hardware." In: *Real-Time Shading SIGGRAPH Course Notes* (2001).

[115] Ken Perlin. "Improving noise." In: *ACM Transactions on Graphics (TOG)*. Vol. 21. 3. ACM. 2002, pp. 681–682.

[116] Ken Perlin and Fabrice Neyret. "Flow noise." In: (2001).

[117] Adrien Peytavie, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou. "Arches: a framework for modeling complex terrains." In: *Computer Graphics Forum*. Vol. 28. 2. Wiley Online Library. 2009, pp. 457–467.

[118] Serban D Porumbescu, Brian Budge, Louis Feng, and Kenneth I Joy. "Shell maps." In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 626–633.

[119] Alena Probst, Graciela Peytavi, David Nakath, Anne Schattel, Carsten Rachuy, Patrick Lange, et al. "Kanaria: Identifying the Challenges for Cognitive Autonomous Navigation and Guidance for Missions to Small Planetary Bodies." In: *International Astronautical Congress (IAC)*. 2015.

[120] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. "Periodic global parameterization." In: *ACM Transactions on Graphics (TOG)* 25.4 (2006), pp. 1460–1485.

[121] *Renderman Pixar*. https://renderman.pixar.com/. Accessed: 2020-06-23.

[122] Martin Reuter, Franz-Erich Wolter, Martha Shenton, and Marc Niethammer. "Laplace–Beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis." In: *Computer-Aided Design* 41.10 (2009), pp. 739–755.

[123] Antonio Ricci. "A constructive geometry for computer graphics." In: *The Computer Journal* 16.2 (1973), pp. 157–160.

[124] Alyn P Rockwood. "The displacement method for implicit blending surfaces in solid models." In: *ACM Transactions on Graphics (TOG)* 8.4 (1989), pp. 279–297.

[125] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge." In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[126] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. "Towards 3D point cloud based object maps for household environments." In: *Robotics and Autonomous Systems* 56.11 (2008), pp. 927–941.

[127] Nayan R Samal, Amit Konar, Swagatam Das, and Ajith Abraham. "A closed loop stability analysis and parameter selection of the particle swarm optimization dynamics for faster convergence." In: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE. 2007, pp. 1769–1776.

[128] Aitor Santamaría-Ibirika, Xabier Cantero, Mikel Salazar, Jaime Devesa, Igor Santos, Sergio Huerta, and Pablo G Bringas. "Procedural approach to volumetric terrain generation." In: *The Visual Computer* 30.9 (2014), pp. 997–1007.

[129] Vladimir V Savchenko, Alexander A Pasko, Oleg G Okunev, and Tosiyasu L Kunii. "Function representation of solids reconstructed from scattered surface points and contours." In: *Computer Graphics Forum*. Vol. 14. 4. Wiley Online Library. 1995, pp. 181–188.

[130] Scott Schaefer and Joe Warren. "Dual marching cubes: Primal contouring of dual grids." In: *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*. IEEE. 2004, pp. 70–76.

[131] Daniel Scheeres, R Gaskell, S Abe, O Barnouin-Jha, T Hashimoto, J Kawaguchi, T Kubota, J Saito, M Yoshikawa, N Hirata, et al. "The actual dynamical environment about Itokawa." In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. 2006, p. 6661.

[132] Nima Sedaghat, Mohammadreza Zolfaghari, Ehsan Amiri, and Thomas Brox. "Orientation-boosted voxel nets for 3d object recognition." In: *arXiv preprint arXiv:1604.03351* (2016).

[133] Dario Seyb, Alec Jacobson, Derek Nowrouzezahrai, and Wojciech Jarosz. "Non-linear sphere tracing for rendering deformed signed distance fields." In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–12.

[134] Andrei Sherstyuk. "Kernel functions in convolution surfaces: a comparative analysis." In: *The Visual Computer* 15.4 (1999), pp. 171–182.

[135] Philip Shilane and Thomas Funkhouser. "Selecting distinctive 3D shape descriptors for similarity retrieval." In: *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. IEEE. 2006, pp. 18–18.

[136] Holger Sierks, Philippe Lamy, Cesare Barbieri, Detlef Koschny, Hans Rickman, Rafael Rodrigo, Michael F A'Hearn, F Angrilli, M Antonella Barucci, J-L Bertaux, et al. "Images of asteroid 21 Lutetia: a remnant planetesimal from the early solar system." In: *science* 334.6055 (2011), pp. 487–490.

[137] Ruben M Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. "A survey on procedural modelling for virtual worlds." In: *Computer Graphics Forum*. Vol. 33. 6. Wiley Online Library. 2014, pp. 31–50.

[138] Alessandro Sperduti and Antonina Starita. "Supervised neural networks for the classification of structures." In: *IEEE Transactions on Neural Networks* 8.3 (1997), pp. 714–735.

[139] Abhishek Srinivas, Rene Weller, and Gabriel Zachmann. "Fast and Accurate Simulation of Gravitational Field of Irregular-shaped Bodies using Polydisperse Sphere Packings." In: *ICAT-EGVE*. 2017, pp. 213–220.

[140] Alvaro Javier Fuentes Suárez, Evelyne Hubert, and Cédric Zanni. "Anisotropic convolution surfaces." In: *Computers & Graphics* 82 (2019), pp. 106–116.

[141] Satoshi Sugita, Rie Honda, Tomokatsu Morota, Shingo Kameda, Hirotaka Sawada, Eisuke Tatsumi, Matsuichi Yamada, Chikatoshi Honda, Yasuhiro Yokota, Toru Kouyama, et al. "The geomorphology, color, and thermal properties of Ryugu: Implications for parent-body processes." In: *Science* 364.6437 (2019), eaaw0422.

[142] Sarah Tang and Afzal Godil. "An evaluation of local shape descriptors for 3D shape retrieval." In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2012, 82900N–82900N.

[143] Vincent Tavernier, Fabrice Neyret, Romain Vergne, and Joëlle Thollot. "Making Gabor Noise Fast and Normalized." In: 2019.

[144] Jörn Teuber, René Weller, Gabriel Zachmann, and Stefan Guthe. "Fast sphere packings with adaptive grids on the gpu." In: *In GI AR/VRWorkshop (Würzburg, Germany* 4 (2013).

[145] *The Legend of Zelda*. https://www.zelda.com/. Accessed: 2020-06-23.

[146] *The Lord of the Rings*. https://en.wikipedia.org/wiki/The_Lord_of_the_Rings. Accessed: 2020-06-23.

[147] *The Perfect Storm*. https://en.wikipedia.org/wiki/The_Perfect_Storm_(film). Accessed: 2020-06-23.

[148] PC Thomas, J Wm Parker, LA McFadden, Cc T Russell, SA Stern, MV Sykes, and EF Young. "Differentiation of the asteroid Ceres as revealed by its shape." In: *Nature* 437.7056 (2005), p. 224.

[149] Julian Togelius, Mike Preuss, and Georgios N Yannakakis. "Towards multiobjective procedural map generation." In: *Proceedings of the 2010 workshop on procedural content generation in games*. ACM. 2010, p. 3.

[150] Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. "Procedural Phasor Noise." In: (2019).

[151] Fabio Turchet, Oleg Fryazinov, and Marco Romeo. "Extending implicit skinning with wrinkles." In: *Proceedings of the 12th European Conference on Visual Media Production*. 2015, pp. 1–6.

[152] Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. "Implicit skinning: real-time skin deformation with contact modeling." In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–12.

[153] Luiz Velho. "Simple and efficient polygonization of implicit surfaces." In: *Journal of Graphics Tools* 1.2 (1996), pp. 5–24.

[154] *Voxel World 3D*. http://paulbourke.net/fractals/noise/sample10.html. Accessed: 2020-05-22.

[155] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. "Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification." In: *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*. IEEE. 2003, pp. 474–481.

[156] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. "State of the art in example-based texture synthesis." In: 2009.

[157] Li-Yi Wei and Marc Levoy. "Fast texture synthesis using tree-structured vector quantization." In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 479–488.

[158] Li-Yi Wei and Marc Levoy. "Texture synthesis over arbitrary manifold surfaces." In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 355–360.

[159] René Weller and Gabriel Zachmann. "ProtoSphere: a GPU-assisted prototype guided sphere packing algorithm for arbitrary objects." In: *SIGGRAPH ASIA*. 2010.

[160] Holger Wendland. *Scattered data approximation*. Vol. 17. Cambridge university press, 2004.

[161] Steven Worley. "A cellular texture basis function." In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 291–294.

[162] Jun Wu, Lou Kramer, and Rüdiger Westermann. "Shape interior modeling and mass property optimization using ray-reps." In: *Computers & Graphics* 58 (2016), pp. 66–72.

[163] Brian Wyvill, Andrew Guy, and Eric Galin. "Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system." In: *Computer Graphics Forum*. Vol. 18. 2. Wiley Online Library. 1999, pp. 149–158.

[164] Jihun Yu and Greg Turk. "Reconstructing surfaces of particle-based fluids using anisotropic kernels." In: *ACM Transactions on Graphics (TOG)* 32.1 (2013), pp. 1–12.

[165] Cédric Zanni. "Skeleton-based implicit modeling and applications." PhD thesis. Grenoble, 2013.

[166] Cédric Zanni, Paul Bares, Ares Lagae, Maxime Quiblier, and Marie-Paule Cani. "Geometric Details on Skeleton-based Implicit Surfaces." In: 2012.

[167] Cédric Zanni, Adrien Bernhardt, Maxime Quiblier, and M-P Cani. "SCALe-invariant Integral Surfaces." In: *Computer Graphics Forum*. Vol. 32. 8. Wiley Online Library. 2013, pp. 219–232.

[168] Cédric Zanni, Michael Gleicher, and M-P Cani. "N-ary implicit blends with topology control." In: *Computers & Graphics* 46 (2015), pp. 1–13.

[169] Cédric Zanni, Evelyne Hubert, and M-P Cani. "Warp-based helical implicit primitives." In: *Computers & Graphics* 35.3 (2011), pp. 517–523.

[170] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. "Synthesis of progressively-variant textures on arbitrary surfaces." In: *ACM Transactions on Graphics (TOG)* 22.3 (2003), pp. 295–302.

[171] Qian Zhang, Jinyuan Jia, and Hongyu Li. "A GPU based 3D object retrieval approach using spatial shape information." In: *Multimedia (ISM), 2010 IEEE International Symposium on*. IEEE. 2010, pp. 212–219.

[172] Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. "Terrain Synthesis from Digital Elevation Models." In: *IEEE Transactions on Visualization and Computer Graphics* 13.4 (2007), pp. 834–848.

[173] Jiaran Zhou, Changhe Tu, Denis Zorin, and Marcel Campen. "Combinatorial Construction of Seamless Parameter Domains." In: *Computer Graphics Forum* (2020). ISSN: 1467-8659. DOI: 10.1111/cgf.13922.

[174] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499.

[175] Xiaoqiang Zhu, Xiaogang Jin, Shengjun Liu, and Hanli Zhao. "Analytical solutions for sketch-based convolution surface modeling on the GPU." In: *The Visual Computer* 28.11 (2012), pp. 1115–1125.

[176] J. Ďurech, V. Sidorin, and M. Kaasalainen. "DAMIT: a database of asteroid models." In: *Astronomy & Astrophysics* 513.A46 (2010). DOI: 10.1051/0004-6361/200912693. URL: http://www.aanda.org/articles/aa/abs/2010/05/aa12693-09/aa12693-09.html.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

Xizhi Li, René Weller and Gabriel Zachmann: *Procedural 3D Asteroid Surface Detail Synthesis*. In *Eurographics'2020*, Norrköping, Sweden, May 25-29, 2020

Xizhi Li, René Weller and Gabriel Zachmann: *AstroGen - Procedural Generation of highly detailed Asteroid Models*. In *ICARCV'2018*, Singapore, November 18-21, 2018

Xizhi Li, Patrick Lange, René Weller and Gabriel Zachmann: *Invariant Local Shape Descriptors:Classification of Large-Scale Shapes with Local Dissimilarities*. In *CGI'2017*, Yokohama, Japan, June 27-30, 2017