Heterogeneous Object Scalar Modeling and Visualization Using Trivariate T-splines

A Thesis Presented
by
Hao Huang

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
Master of Science
in
Computer Science
Stony Brook University

May 2010

Stony Brook University
The Graduate School

Hao Huang

We, the dissertation committee for the above candidate for the degree of Master of Science, hereby recommend acceptance of this dissertation.

Hong Qin, Dissertation Advisor
Professor, Computer Science Department

Xianfeng Gu
Associate Professor, Computer Science Department

Joseph Mitchell, Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Heterogeneous Object Scalar Modeling and Visualization Using Trivariate T-splines

by

Hao Huang

Master of Science

in

Computer Science

Stony Brook University

2010

Volume modeling has become an important research topic. For many years people have been focusing on modeling homogeneous solid object. Because of the rapid hardware and technological development in volumetric data acquisition during the past decade, effectively modeling heterogeneous volumetric objects or datasets becomes imperative in solid modeling and 3D graphics. A heterogeneous model consists of a solid model and a number of spatially distributed material attributes. Heterogeneous object scalar modeling focuses on representing and capturing a broad range of complex appearances in solid model. Discrete representations such as voxels and regular or irregular grids have been predominating in scientific visualization and finite element analysis for the last thirty years. Prior state-of-the-art in continuous representations include B-splines, triangular splines and simplex splines, however, they either lack of local adaptivity and refinement, and hierarchical structure, or have irregular domain and the process of choosing partitioning into simplices is extremely hard. To overcome these problems, in this paper we advocate a continuous representation scheme for heterogeneous material modeling which built upon trivariate scalar T-splines, whose domain is principal-axis-aligned regular and control grids/lattice permit T-junctions. By using trivariate T-splines, lines of control points need not traverse the entire control grid and local adaptivity and refinement can be obtained. Moreover, heterogeneous volumetric objects can be adaptively refined in a hierarchical manner (hierarchical refinement) by this scalar modeling method. In addition to object modeling and material representation, we also design a volume rendering algorithm via ray-casting. Since trivariate T-splines afford a continuous representation, we can benefit from this precise and compact mathematical formulation that will facilitate the modeling and visualization tasks in engineering applications. Several techniques, such as empty space skipping, intersection refinement, adaptive sampling, and attribute integrating are proposed to get both efficient and accurate visualization results. We conduct experiments that have demonstrated the utility of our trivariate scalar T-splines in modeling, graphics, and visualization.

For my beloved family, advisors, friends and colleagues.

Contents

List of Tables

List of Figures

Acknowledgements

It is a pleasure writing this acknowledgment to all the people who have been of great help and support in my path to writing this thesis. First of all, I would extend gratitude to my advisor Prof. Hong Qin for his guidance and critique without which it would be impossible to complete this thesis.

I am very grateful to Dr. Dantong Yu, for providing help and all the interesting discussions in technical matters.

I would also like to thank the members of my committee, Prof. Joseph Mitchell and Prof. Xianfeng Gu, for their time, suggestions and comments.

In addition I want to thank my past and present labmates, Hongyu Huang, Bo Li and Tingbo Hou, for sharing their experiences, knowledge, and all kinds of help to my research. I also want to thank my roommates, Yunlong Wang, Bowen Song and Yan Yan for their kindness and understanding.

And finally, I would like to thank my family, my girlfriend for their encouragement and support in all my endeavors.

Chapter 1

Introduction

1.1 Problem Statement

Volume modeling has been an important research subject in 3D graphics and visualization. In solid modeling field, for more than three decades researchers have been mainly focusing their effort on modeling homogeneous solid object [36, 58, 21, 34, 39, 55], which typically have only geometric data or whose characteristics or properties are uniform on and inside the whole model.

During the past decade, the rapid advancement of specialized hardware devices and their effective integration with volumetric data acquisition techniques have engendered exciting research for heterogeneous volumetric object modeling and visualization. Heterogeneous objects comprise different constituent materials and can exhibit continuously-varying material composition and microstructure, which usually represent results of physical simulations, geological and medical datasets, digital prototypes in CAE environments. In such scenario, material composition can be controlled at different regions within modeled object so as to achieve desirable material property to meet diverse modeling, design, and display requirements. Moreover, heterogeneous objects can fulfill the critical functional requirement since they can synthesize different distribution and achieve various properties of multiple materials in one monolithic component. In addition, heterogeneous objects can overcome traditional material limitations such as material incompatibility (stress concentration, non-uniform thermal expansion, etc.). Motivated by the above-mentioned advantages, heterogeneous objects are being adopted in the design of high efficiency engines, ceramic turbine components, biomaterials, mould and die tools for industrial use [28, 37, 50, 41, 55].

1.2 Literature Review

Various modeling schemes have thus far been proposed to support the creation of different graded material of heterogeneous models. Pratt [39] once discussed the pros and cons of different representation schemes. Recently Kou and Tan [25] have compared the advantages and disadvantages of various representation schemes. Modeling of multidimensional point sets with multiple attributes was discussed in [35].

A non-manifold B-Rep scheme was used in [27] to subdivide an object into components made of unique materials. Each component is homogeneous inside and has an assigned index of material. A more general object model proposed in [26] was designed to include all the characteristics and attributes of an object. The model of attributes is a collection of functions mapping the object geometry to several attributes. The object model, combining the point set and attributes models, is represented by a trivial fiber bundle. Discrete representations such as voxels and regular grids have been predominating in scientific visualization and finite element analysis for the last thirty years, mainly because of their simplicity and direct connection with data acquisition. Some researchers adopted voxel-based representation in which the modeling space is divided in to small cells and each cell has unique geometrical and material representation [23, 34, 11]. Voxel arrays in volume modeling and graphics can be considered as attribute models with the default geometry represented by a bounding box. Constructive Volume Geometry (CVG) [9] combines geometry and attributes in a systematic manner. The model was presented as algebra of 3D spatial objects utilizing voxel arrays and continuous scalar fields for representing both geometry and photometric attributes (opacity, color, etc.). Siu and Tan [51] introduced the concept of "grading source" and material composition function to represent heterogeneity. Explicit mathematical functions like linear, parabolic or exponential were used to describe material variation. Implicit mathematical functions were used to address heterogeneity in many representations [45, 7, 50, 53]. Rvachev et al [45, 7] generated inverse distance weighing interpolation with R-function and distance fields. Kumar et al [27, 28] proposed a set based approach with separate sets representing geometry model and attribute model respectively. The intersection of these two sets represented the object. Adzhiev et al [1] proposed constructive hypervolume model based on function representation which is used to represent multidimensional point sets and heterogeneous attributes. An r-m sets-based method was proposed in [28], which handles heterogeneous objects by using r-sets as the basis of representing the geometry and material distribution. A mesh based method was reported in [23, 22], which employs four-node isoparametric quadrilateral elements to model the material distributions. An implicit function-based method was proposed in [6, 7] which parameterizes the space by distances from the material features. Some methods used by researchers are feature-based methodology [54, 42, 30] and axiomatic design methodology [8]. On standardization of heterogeneous object representation, Patil et al [40, 38] suggested information model to represent heterogeneous object for ISO 10303.

B-spline-based methods [57, 21, 31, 41, 42, 60] model the object heterogeneity by specifying scalar/vector values of a set of control points and interpolating them with the B-spline shape functions. Some researches [43] on heterogeneous object modeling showed that B-spline-based method has excellent representation coverage due to the large number of control points. On the other hand, the large degrees of freedom make it inconvenient to edit the model. Hence, intuitive and efficient heterogeneous object modeling method was needed for designing a B-spline based heterogeneous object model, which could alleviate the inconvenience induced by the large number of design variables [47]. Qian and Dutta [41] proposed B-spline tensor solid representation for heterogeneous material representation. The method was applied for designing and the analysis of a turbine blade, by using heterogeneous B-spline lofting [42, 60]. Schmitt [48] proposed a volume sculpting scheme with multiresolution capability based on trivariate B-spline functions to define both object geometry and material attributes. In recent work [5], an application of heterogeneous B-spline tensor product modeling, namely heterogeneous B-spline fairing and heterogeneous B-spline fit, was developed, and heterogeneous surface models were created for the proximal femur, skull and mandible slices using this approach. Besides the above approaches, most recent research also includes the level-set based method [55] and trivariate simplex splines-based method [20].

The ideas of volumetric texturing and rendering were also applied in [31, 34] to model volumetric distribution of attributes. Gradient material distribution represented by scalar functions was combined in [34] with a BRep model of object's 3D geometry. Trivariate NURBS splines were used in [48] to represent both geometry and attributes.

Besides modeling, volume visualization (rendering) is also a main driving force that guides the research direction on heterogeneous materials. In Muller's paper [32], different volume rendering techniques were presented illustrating their fundamental features and differences as well as their limitations. Rossl [44] developed a new approach to reconstruct and visualize non-discrete models from uniform-gridded volume samples. In particular, they used quadratic trivariate super splines defined over regular grids with a uniform tetrahedral partition. In 2009, Finkbeiner [2] demonstrated that non-separable box splines deployed on body-centered cubic lattices are suitable for fast evaluation on present graphics hardware. Therefore, they developed the linear and quintic box splines using a piecewise polynomial form as opposed to their already-known basis form. The paper written by Hua et.al [19] documents a powerful heterogeneous solid modeling paradigm for representing, modeling, and rendering of multi-dimensional, physical attributes across any volumetric objects, and its technical core is founded up simplex splines.

1.3 Objectives and Contributions

For a B-spline curve, the finer control for details can be gained through knot insertion which is a local refinement process and restricts the influence of the control point to a local area. For a tensor-product B-spline surface and volume, however, knot insertion is not a local process because the insertion of one knot into the existing knot vector causes creation of new control points along the entire row or column. Forsey [16] firstly proposed Hierarchical B-splines, where a single control point can be inserted without propagating an entire row or column of control points, in order to get hierarchical refinement. In 1995 he extended Hierarchical B-splines to multiresolution surface reconstruction [17]. Certain modeling methods (also see [61]) enable designers to create a complex smooth surface of arbitrary topology whose geometric details can be added by refining the patches wherever necessary. However, these splines (also including triangular splines) may not be regular and the process is not a true local refinement while introducing additional, unnecessary control points in nearby regions. Simplex splines [20] have local adaptivity, but its domain is irregular, so the process of choosing the partition into simplices is extremely hard.

To overcome these problems, T-splines [49] was introduced whose domain is principal-axis-aligned while the grids/lattices permit T-junctions. T-spline can be thought of as a NURBS surface for which a row of control points is allowed to terminate without traversing the entire surface, which create T-junctions. By using T-junction, individual control points can be inserted only where they are needed to provide additional control, or to create a smoother tessellation. In other words, T-splines have the ability to eliminate superfluous control points and perform true local refinement. Furthermore, T-splines support merging of several B-spline surfaces that have different knot vectors into a single gap-free model. Thus, T-splines inherit all of the respective strengths of B-splines/NURBS while eliminating most of their weaknesses. In 2006, Yang et al. [59] studied the evolution of T-spline level sets (i.e, implicitly defined T-spline curves and surfaces). In [13] the authors not only used particles on the evolving surface with a goal to discretize the evolution equation, but also discussed volume and range constraints which can be added to the framework. However, these approaches do not focus on representing heterogenous solid objects.

In this paper, we strongly advocate a continuous representation scheme for heterogeneous material solid modeling, which is built upon trivariate scalar

T-splines with the following contributions:

- Our **continuous representation** scheme for heterogeneous material modeling admits a regular structure, and its principal-axis-aligned nature matches perfectly with data acquisition formats and data processing architecture on standard graphics hardware. The compact mathematical formulation with a regular structure accelerates all the data management and processing tasks in real-world applications.

- Our method can afford real **local refinement** since individual control point can be inserted in order to provide finer control over any region of interest via trivariate T-splines. Heterogeneous volumetric objects can be adaptively refined in a hierarchical manner (i.e., **hierarchical refinement**) by this scalar modeling method.

- The modeling, processing, and visualization pipeline is streamlined and accelerated because all the intermediate T-spline computations are relevant and can be reused throughout various stages.

- We develop novel visualization techniques for trivariate T-splines. Volume ray-casting can be formulated in a close-form by integrating the rendering equation with T-spline basis functions. Several practical techniques such as empty space skipping, intersection detection and refinement, adaptive sampling, and attribute integration enable high-fidelity visualization in an efficient way.

1.4 Thesis Outline

We begin Chapter 2 by briefly reviewing basic modeling techniques such as parameterization (hyperpatch), then comes the basic knowledge of B-spline and T-splines. In Chapter 3, we describe common themes in advanced material models using trivariate scalar T-spline volume modeling, that includes least-squares fitting on trivariate scalar T-splines. Experimental results are also shown and analyzed at the end of this chapter. Then in Chapter 4 we introduce methodology such as volume classification, volume ray-casting and detailed techniques to do volume rendering on trivariate scalar T-spline. Finally in Chapter 5, a general review and conclusion is drawn.

Chapter 2

Knowledge on B-splines and T-splines

2.1 Parameterization

Parametric representation is the most general way to specify surfaces and volumes. Besides modeling and remeshing, parameterization techniques have a wide variety of applications including texture mapping, detail transfer, fitting and morphing. Most of researches focus on "patch gluing" where a certain level of smoothness along the patch boundaries is desired.

Hyperpatch is a continuous mathematical functional representation of patch bounded collection of points with three parameters, which can be represented as

$$x = x(u, v, w), \tag{1}$$

$$y = y(u, v, w), \tag{2}$$

$$z = z(u, v, w), \tag{3}$$

where u, v and w are parametric variables.

When this equation is used to represent not only geometry but also material information, the parameters (u, v, w) vary within a certain domain in the parametric UVW-area. The first partial derivatives with respect to the parameters are usually denoted as $\vec{r}_u$, $\vec{r}_v$ and $\vec{r}_w$, and similarly for the higher derivatives, $\vec{r}_{uu}$, $\vec{r}_{vv}$, $\vec{r}_{ww}$, $\vec{r}_{uv}$, $\vec{r}_{uw}$, $\vec{r}_{vw}$.

Suppose the geometry of a point p is give by $p = \{x, y, z\}^t$ So hyperpatch can be also used to represent material composition; denote material parameter as m:

$$p = x, y, z, m, [u, v, w] \in [0, 1], \tag{4}$$

where all the elements x, y, z, and m are functions of (u, v, w). The vector m represents the material distribution within the geometric solid,

$$m = \{m_1, m_2, ..., m_n\}.$$

where m_i is a scalar to describe the material fractions of the ith material at a point (u, v, w), n is the number of materials. Scalar m_i can be replaced by any material property such as color, density, elastic modulus etc. depending upon the application area can be suitable modified.

2.2 B-spline Curve

A degree n B-spline curve (see Figure 1) can be decomposed into a sequence of degree n Bezier curves that join automatically with c^{n-1} continuity. The parameter values at which the Bezier curves meet are referred to as knot values. The sequence of knot values in a B-spline curve is referred to as a knot vector.

A piece-wise polynomial B-spline $C(u)$ can be represented as the linear combination of the basis functions weighted by the components of control points:

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) p_i, \tag{5}$$

where n is the number of control points, $N_{i,p}(u)$ are the pth degree B-spline basic functions, given by the recurrence relation:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1}, \\ 0 & otherwise, \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p-1} - u}{u_{i+p-1} - u_{i+1}} N_{i+1,p-1}(u), \tag{6}$$

where u_i are the knot values satisfying the relation $u_i \leq u_{i+1}$, and

$$U = \{u_0, u_1, u_2, ..., u_{n+p+1}\}$$

is the knot vector.

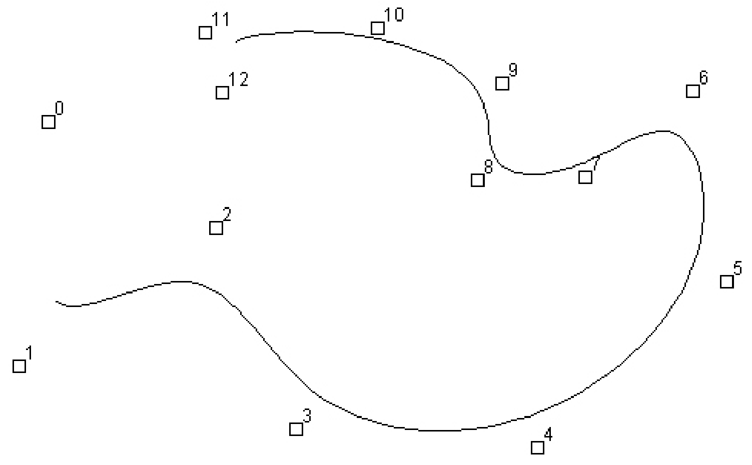

Figure 1: B-spline.

By fixing one of the knot values, we are assigning a parameterization to the curve. By changing the fixed knot value, it is possible to change the parameterization of the curve. Every edge of a cubic B-spline control polygon (except the first and last edge) corresponds to a Bezier curve and each vertex of the control polygon corresponds to a starting and ending point of a Bezier curve. There is also an end-condition knot interval needed before the first control point.

Uniform cubic B-splines are those cubic B-splines with uniform knot-vector. The blending function can easily be precalculated, and is equal for each segment in this case. If every control point there is a corresponding weight, we can represent a rational B-spline curve as the following equation:

$$C(u) = \frac{\sum_{i=0}^{n} w_i N_{i,p}(u) p_i}{\sum_{i=0}^{n} w_i N_{i,p}(u)}. \tag{7}$$

A NURBS (non-uniform rational B-spline) curve is defined by its order, a set of weighted control points, and a knot vector (whose knot intervals are not necessarily all equal in size). NURBS curves are generalizations of both B-splines and Bezier curves, the primary difference being the weighting of the control points which makes NURBS curves rational (non-rational B-splines are a special case of rational B-splines).

2.3 B-spline Tensor Product Surface and Solid Volume

The equation for B-spline surface is quite similar to the equation for B-spine curve, just switch from one parameter component to two parameter components. By means of tensor products, B-spline surfaces can be constructed starting from a bidirectional net of $(n+1) \times (m+1)$ control points and knot vectors (n for u direction and m for v direction) as follows:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) p_{i,j}. \tag{8}$$

A bivariate surface can be obtained by this equation over the two independent parameters u and v, where $p_{i,j}$ are control points of the heterogeneous surface. $N_{i,p}(u)$ and $N_{j,q}(v)$ are the p-th degree and q-th degree B-spline basis functions defined in u and v directions respectively. There are two knot vectors for a B-spline surface, one corresponding to the u direction and one corresponding to the v direction. Similar to NURBS curves, a NURBS surface is a B-spline surface whose control points have weights attached to them and whose knot intervals are not all equal.

The equation of the gradient at any point $p(u,v)$ along the direction u represented as p^u and along direction v represented as p^v, can be expressed as follows.

$$p^u(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} N^u_{i,p}(u)N_{j,q}(v)p_{i,j}, \tag{9}$$

$$p^v(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u)N^v_{j,q}(v)p_{i,j}. \tag{10}$$

Equation (8) is mostly used for geometry representation, but can be also used for defining material modeling along the parametric variables u and v. Here the equations (9) and (10) are used for obtaining material gradient in parametric directions u and v respectively. These material gradients may be used for finite element mesh generation.

We can extend B-spline volume equation (with three different parameters) as modeling tool to represent a point in a solid volume:

$$M(u,v,w) = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l} N_{i,p}(u)N_{j,q}(v)N_{k,r}(w)p_{i,j,k}, \tag{11}$$

where $p_{i,j,k}$ are control points for the solid volume, p, q, r are the order of the B-spline basis functions $N_{i,p}(u)$, $N_{j,q}(v)$, $N_{k,r}(w)$, in the direction of u, v, w respectively, which are similar to the bivariate B-spline surface, and defined by the knot vectors of

$$U = \{u_0, u_1, u_2, ..., u_{n+p+1}\},$$
$$V = \{v_0, v_1, v_2, ..., v_{m+q+1}\},$$
$$W = \{w_0, w_1, w_2, ..., w_{l+r+1}\}.$$

.

2.4 T-splines

If we want to refine a B-spline curve, we can use knot insertion, which is a local refinement because the influence of the new points are restricted in a local area. But if we want to refine a tensor-product B-spline surface or volume, knot insertion is not local at all because if one knot is inserted into a

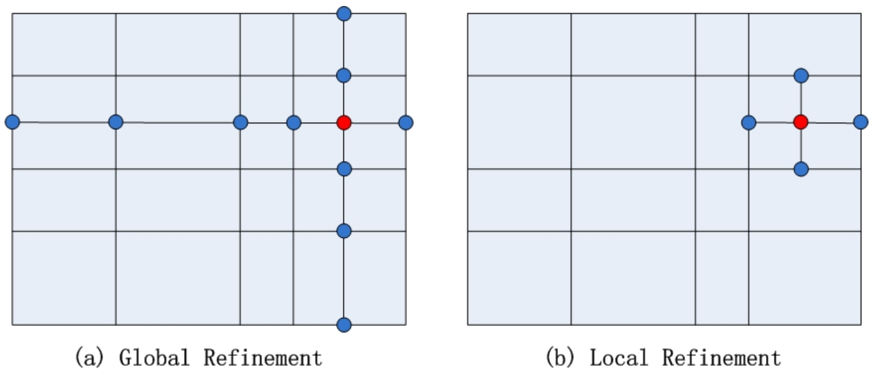

Figure 2: Global Refinement and Local Refinement.

surface or volume, new control points should also be created along the whole row or column (see Figure 2(a)).

T-spline surface are a generalization of non-uniform B-spline surfaces. T-spline control grids permit T-junctions, which allow lines of control points to only traverse part of the control grid (see Figure 2(b)). A T-spline control grid is called a T-mesh. Figure 3 shows an example of a simple T-mesh. The T-junctions are marked with red dots.

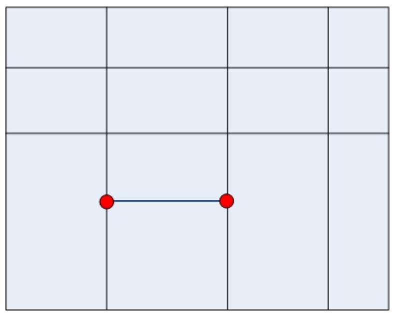

Figure 3: Simple T-mesh.

The formulation for a T-spline surface is

$$T(u,v) = \frac{\sum_{i=0}^{n} B_i(u,v) p_i}{\sum_{i=0}^{n} B_i(u,v)}, \tag{12}$$

where p_i represent control vector/scalar in multidimensional space and $B_i(u,v)$ are T-spline basis functions corresponding to control point p_i and formulated by

$$B_i(u, v) = N_i(u)N_i(v), \tag{13}$$

$$B_i(u, v, w) = N_i(u)N_i(v)N_i(w), \tag{14}$$

$w_i = [w_{i0}, w_{i1}, w_{i2}, w_{i3}, w_{i4}]$

where $N_i(u)$, $N_i(v)$ are the cubic B-spline basis functions associated with the knot vectors $u_i = [u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}]$ and $v_i = [v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}]$ respectively. The knot vectors u_i and v_i are extracted from the T-mesh neighborhood of p_i.

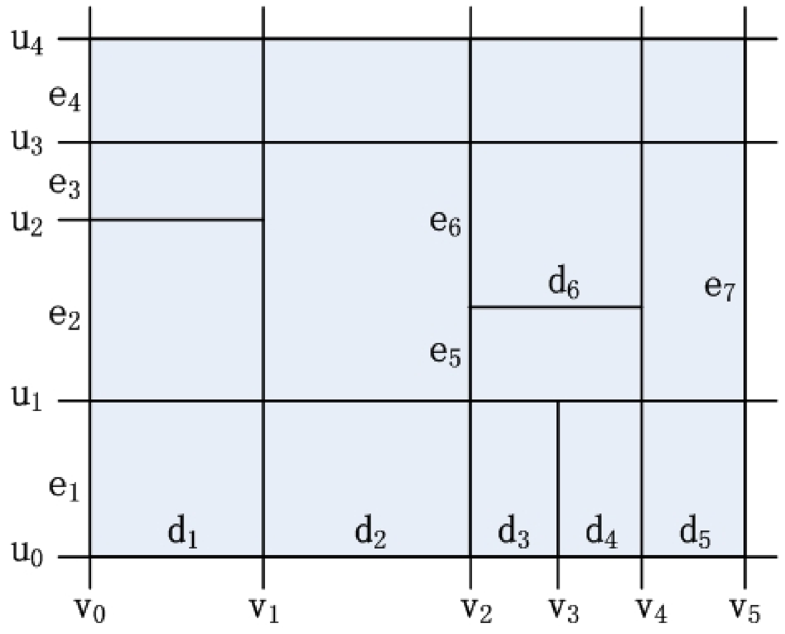

Figure 4: T-mesh.

The parameterization of a T-spline is represented by knot intervals, with a knot interval being assigned to each edge in the T-mesh. Figure 4 shows a portion of a T-mesh in (u, v) parameter space with d_i and e_i representing the knot intervals.

There are two rules that a T-spline must follow. First, the sum of the knot intervals on one side of a face must be equal to the sum of the knot intervals on the opposite side of the face. For example, in Figure 4, $e_2 + e_3 = e_5 + e_6 = e_7$, and $d_3 + d_4 = d_6$. The second rule is that if T-junction on one edge of a face can be legally connected to a T-junction on the opposite side of the face, that edge must be included in the T-mesh. An edge can be legally connected if the sums of the knot vectors on the opposite sides of the newly created faces are equal. For example, an edge would need to be added to split face if $e_2 = e_5$ and $e_3 = e_6$.

2.5 T-NURCCs

T-NURCCs (Non-Uniform Rational Catmull-Clark surfaces with T-junctions) are a superset of both T-splines and Catmull-Clark surfaces. One of the most important aspects of T-NURCCs is that they allow non-valence four control points called extraordinary points. Local refinement can be obtained in the neighborhood of an extraordinary point.

The surfaces created by Catmull-Clark algorithm (using bi-cubic uniform B-spline surfaces) are called Catmull-Clark surfaces. Local refinement can be performed around extraordinary points in Catmull-Clark surfaces. Normal Catmull-Clark surfaces only provide global refinement, which increases the number of control points exponentially. However, T-NURCCs can do this in a much better way. Similar to T-splines, T-NURCCs provide finer control by only individual control points insertion. Hence control points are only added linearly but not exponentially.

An important aspect of T-junctions is that they allow T-splines to be locally refinable, meaning that individual control points can be inserted in order to provide finer control over details. For example, when modeling a face it is desirable to have more control for modeling the nose than for modeling the forehead. This is shown by the head models in Figure 5, where there are many more control points to model the nose than to model the forehead.

2.6 Materials Representation

By many literatures (e.g. [31]), materials can be represented either with the same order, dimensions or knot vectors as geometric representation (just as

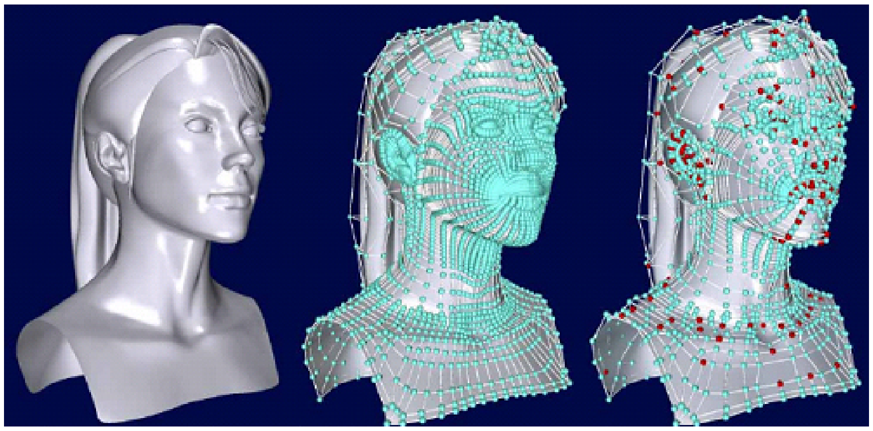

Figure 5: Head Models (a) Using NURBS and (b) Using T-spline [49].

what we introduced in Equation (8) and (11)), or totally different. In this case, all aspects of the materials representation are independent of the geometric trivariate representation. The only requirement is that all the volumes (both geometry and materials) share the same parametric domain,

$$A(u, v, w) = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} N_{i,p_2}(u) N_{j,q_2}(v) N_{k,r_2}(w) Ap_{i,j,k}. \tag{15}$$

This representation has two major advantages. First, material representation can be decoupled from geometric representation, which means a model with complicated geometries but simple materials, or simple geometries but complicated materials can be represented at the resolution that best suits them. Hence it can save a large cost in storage and computation time. Second, it can provide a robust representation for a model contain moderate noise, which is very usual in measured data. Hence, compared with polygonal meshes, or higher dimensional analogues such as voxels, splines are a terse representation and can be a smooth function with fewer control points.

Chapter 3

Trivariate Scalar T-spline Volume Modeling

3.1 T-spline Volume Modeling

We now use trivariate T-spline volume for heterogenous attribute (scalar) representation. In order to reduce the computational cost when calculating intersection between ray and modeled volume later, UVW parameterization coordinates ($[0,1]^3$) are set linearly correlative to the XYZ geometry coordinates. To do this, we compute a minimum bounding box first. Axis-aligned bounding box is usually a suitable box that enclose the whole model completely. The minimum bounding box for a given model is its minimum box subject to the constraint that the edges of the box are parallel to the (Cartesian) coordinate axes. It is simply the Cartesian product of intervals, and each of which is defined by the minimal and maximal value of the corresponding coordinate for the points in the model. So the minimum bounding box is defined by x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, and z_{max}. Mapping between UVW coordinates and XYZ coordinates is given as follows:

$$u = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{16}$$

$$v = \frac{y - y_{min}}{y_{max} - y_{min}}, \tag{17}$$

$$w = \frac{z - z_{min}}{z_{max} - z_{min}}. \tag{18}$$

However, we must point out that visualization is carried out in XYZ coordinates, and the inverse mapping should be done before we visualize models on the display device.

We can easily extend T-splines/T-NURCCs to three dimensions of T-spline volumes. First, a three dimensional space version of a T-mesh, called T-lattice, can be defined as a tiling of rectangular cells in R^3. A single (control) point can be inserted to T-lattice by adding T-junctions point. Compared to B-spline volume, the refinement is locally controlled instead of globally controlled without the creation of an entire row of control points. A local knot coordinate system can be easily imposed onto a T-lattice by using knot intervals. We choose an origin for the parameterization where all the parameters equal to 0. Then we assign each edge in the first direction a r knot value, each edge in the second direction a s knot value and each edge in the third direction a t knot value. Each of the knot values will also collectively participate to form a sum of knot intervals. By doing this, specific knot coordinates have been assigned to each control point.

When we finish building a knot coordinate system onto a T-lattice, we can deduce knot vectors u_i, v_i, w_i for each basis function. The edges on three different directions are called u-edge, v-edge and w-edge respectively. A T-junction is a vertex shared by one edge in some direction and two edges in other directions at the same time. For example, $P1$ in Figure 6(a), $P2$ in Figure 6(b) are both T-junctions. If a T-lattice is simply a rectangular parallelepiped with no T-junctions, the T-spline volume reduces to a NURBS volume. In each minimal cell, the sums of knot intervals in the same directions must be equal, which is similar to T-mesh. Thus for the cell in Figure 6(a) and 6(b) in the vertical direction we have

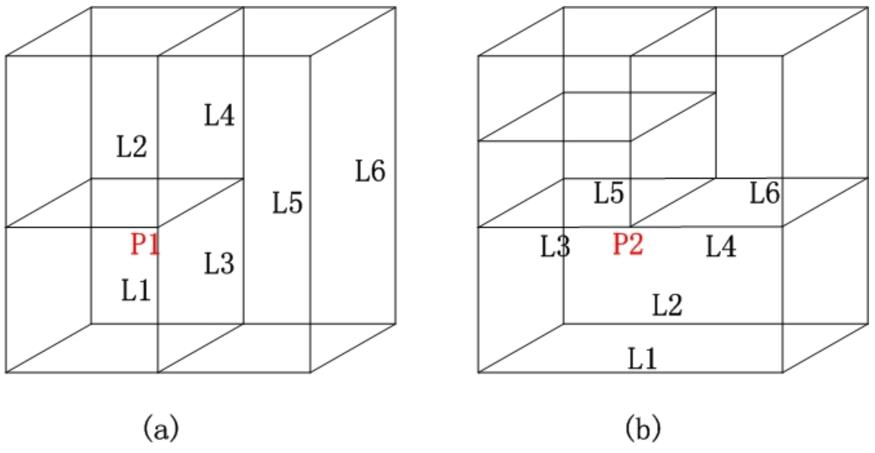

Figure 6: Examples of T-lattice.

$$L_1 + L_2 = L_3 + L_4 = L_5 = L_6 \qquad in \quad Figure \quad 6(a), \quad and$$
$$L_1 = L_2 = L_3 + L_4 = L_5 + L_6 \qquad in \quad Figure \quad 6(b).$$

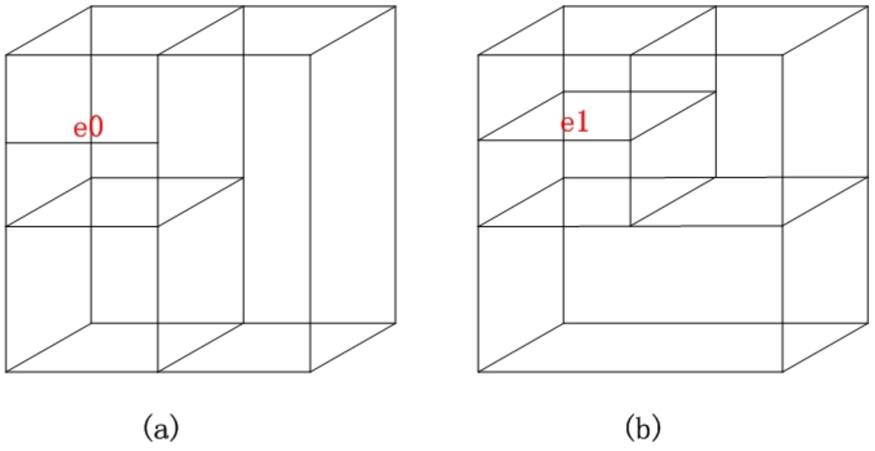

Figure 7: Illegal Edge (a) and Legal Edge (b) in T-lattice.

Every edge must be a cell edge. In Figure 7(a) e_0 is not an cell edge therefore it is not a valid edge either. But e_1 in Figure 7(b) is a valid edge because it is a cell edge. Based on Section 2.4 and 2.5, each P_i is calculated using basis functions $B_i(u, v, w)$ in UVW coordinates. And each basis function is defined in terms of knot vectors u_i, v_i, w_i. For example, the knot coordinates of P_i are (u_{i1}, v_{i1}, w_{i1}). If we consider a ray origins from (u_{i1}, v_{i1}, w_{i1}) and goes along axis u, the ray can be represented as $R(a) = (u_{i1}+a, v_{i1}, w_{i1})$ in the UVW parameter space. Then u_{i2} is the u coordinates of the first edge intersected by the same ray. The other knots in v and w direction can be found similarly.

T-spline volume enforces continuity in the whole parametric volume. The sum of knot intervals in the same direction in the same cell must be equal, and each edge must be an edge of at least one cell. It is easy to prove that any octree-subdivided lattice satisfies the above rules, and hence, is a valid T-lattice [52]. Therefore, we use octree-subdivided lattice as T-lattice directly. In material attribute representation, the aspects of each attribute (e.g., order, dimension of the control mesh and knot vectors) can be set independently with other material attributes. The only requirement is that all the representations share the same parametric domain (u, v and w).

However, parameterizations almost always introduce distortion in either angles or areas, and a good mapping in application is the one which minimizes these distortions in some sense. The readers are referred to [15, 14] for excellent surveys. For our method in the paper, we use least-squares fitting for parameterization purpose.

3.2 Least-Squares Fitting on Scalar Trivariate T-splines

Linear least-squares fitting on splines is a well-researched and very popular topic. Related literatures span across several areas such as linear algebra [15, 14] and performing matrix operations effectively [15, 14]. This section introduces linear least-squares fitting and its application to trivariate T-splines.

In general, least-squares fitting is a mathematical procedure for finding the best-fitting function to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the function. That is, given a spline function $f(p_0, p_1, ..., p_m; q_i)$, we adjust the control points $(p_0, p_1, ..., p_m)$, in order to minimize the squared-distance from the function to sample data (every p_i represents the parameterization coordinates u_i, v_i, w_i of every control point). The mathematical representation in the discrete case is given as follows:

$$R = \sum_{i=0}^{n} [Q_i - f(p_1, p_2, ..., p_m; q_i)]^2 = \sum_{i=0}^{n} r^2, \tag{19}$$

where Q_i are the n discrete (scalar) samples corresponding to $f(p_1, p_2, ..., p_m; q_i)$, where the model function f is decided by m control points $p_1, p_2, ..., p_m$. The sum of the squares of the offsets is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity.

Now we apply this general framework to trivariate T-spline volume using LSCM (Least Squares Conformal Maps [29]). Suppose $p = (p_1, p_2, ..., p_m)$ are the control points (scalar) of T-lattice, and each Q_i and p_i represent a set of scalar $(m^1_{u_i,v_i,w_i}, m^2_{u_i,v_i,w_i}, ..., m^d_{u_i,v_i,w_i})$ in a certain position in UVW coordinates, the T-spline volume is

$$M(u, v, w) = \frac{\sum_{i=0}^{m} B_i(u, v, w) p_i}{\sum_{i=0}^{m} B_i(u, v, w)}. \tag{20}$$

We want to minimize the summation of distance between data scalar points $Q = Q_1, Q_2, ..., Q_n$ $(m < n)$ and the volume $M(u, v, w)$:

$$D_{squa-dis}(p_1, p_2, ..., p_m) = \sum_{i=1}^{n} (M(u_i, v_i, w_i) - Q_i)^2. \tag{21}$$

Here (u_i, v_i, w_i) is the parameterization of Q_i. Then the energy function is augmented with an additional term:

$$\begin{aligned} D_{add}(p_1, p_2 ... p_m) = & \\ & \sum_{i=1}^{n} (M^2_{uu}(u_i, v_i, w_i) + M^2_{vv}(u_i, v_i, w_i) + M^2_{ww}(u_i, v_i, w_i) \\ & + 2M^2_{uv}(u_i, v_i, w_i) + 2M^2_{uw}(u_i, v_i, w_i) + 2M^2_{vw}(u_i, v_i, w_i)). \end{aligned} \tag{22}$$

So the optimization problem reduces to the minimizing of the following energy

$$D_{energy}(p_1, p_2, ..., p_m) = D_{squa-dis}(p_1, p_2, ..., p_m) + \sigma D_{add}(p_1, p_2, ..., p_m). \tag{23}$$

Here σ is a non-zero constant and usually between 0.001 to 0.1. In this way we cast the fitting process as a global optimization problem with this energy function that measures the deviation of the approximation from the original data. In order to obtain control points, we can solve the linear system $\mathbf{A^T A Q} = \mathbf{A^T P}$ (the solution is also for Equation (20)), where the coefficients of the $m \times n$ matrix $\mathbf{A}$ are given by $a_{k,i} = B_i(u_k, v_k, w_k)$. Just as Equation (23) the linear system can be changed to

$$\begin{aligned}(\mathbf{A^T} + \sigma(\mathbf{A^T_{uu} A_{uu}} + \mathbf{A^T_{vv} A_{vv}} + \mathbf{A^T_{ww} A_{ww}} \\ +\mathbf{2A^T_{uv} A_{uv}} + \mathbf{2A^T_{uw} A_{uw}} + \mathbf{2A^T_{vw} A_{vw}}))\mathbf{Q} = \mathbf{A^T P},\end{aligned} \quad (24)$$

where the coefficients of the $m \times n$ matrix $\mathbf{A_{uu}}$, $\mathbf{A_{vv}}$, $\mathbf{A_{ww}}$, $\mathbf{A_{uv}}$, $\mathbf{A_{uw}}$ and $\mathbf{A_{vw}}$ are the second-order derivatives:

$$\begin{aligned} &B_{iuu}(u_k, v_k, w_k), \quad B_{ivv}(u_k, v_k, w_k), \quad B_{iww}(u_k, v_k, w_k), \\ &B_{iuv}(u_k, v_k, w_k), \quad B_{iuw}(u_k, v_k, w_k), \quad B_{ivw}(u_k, v_k, w_k). \end{aligned}$$

Equation (23) can give us the control points of our trivariate scalar T-spline volume.

In heterogeneous model scalar fitting, the parameters must be assigned very carefully, because the poor parameterization causes distorted unnatural materials, especially near the boundary between different materials, where being called "mutation". To achieve tolerance within certain threshold, we use a similar approach to the one used by Weiss et al. [56], which was originally developed for reverse engineering. When local material attributes have large fitting-errors even after parameters are refined, the system automatically inserts knots to increase the degrees of freedom for fitting by splitting the region at the center of region. More details about T-spline knot-insertion can be found in [4]. Then, a T-spline volume is calculated again with updated parameters. The above steps are iterated until the fitting error falls below a predefined threshold value. In our experiments, this method successfully generates natural and smooth parameterization with a few iterations (typically less than 5). In this way models can be obtained that exhibit excellent accuracy to conciseness trade-offs.

3.3 Experimental Results and Analysis

We introduce our experimental results in this section. A prototype system is implemented on a PC with 3.5GHz P4 CPU and 4GB RAM. The system is written in C++ and OpenGL. Figure 8 illustrates the trivariate T-spline heterogeneous scalar fitting method applied to Fuby model. We first conduct the material attributes fitting, with the red color indicating its fitting error larger than 1%. Figure 9 illustrates the trivariate T-spline heterogeneous scalar fitting method applied to Tooth model. The red parts highlight regions where the opacity fitting error larger than 1%.

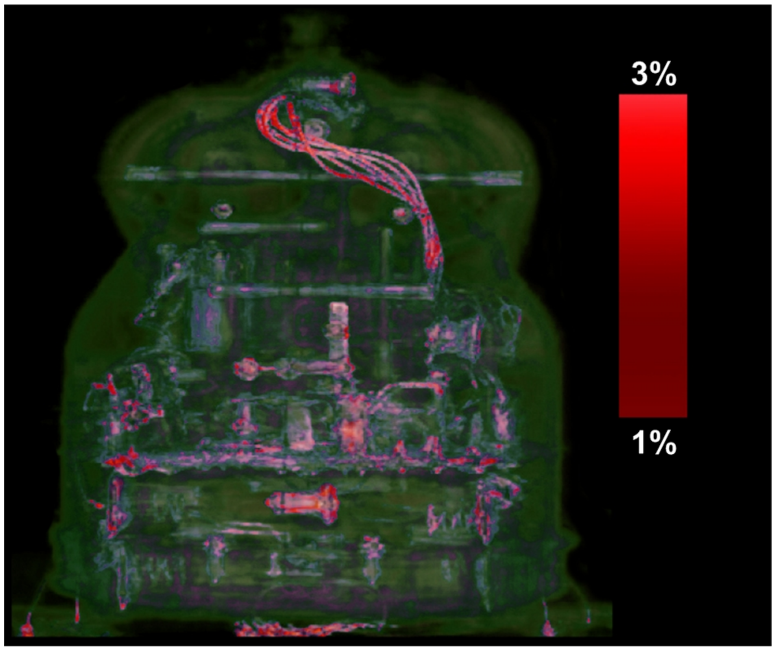

Figure 8: Color Fitting Error on Fuby Model.

As seen in Figure 8 and Figure 9, the parametric materials of the Fuby model and Tooth model have a very salient color/opacity fitting error (colored in red) on the material boundary. In other words, the significant fitting errors mainly lie on the mutation part. With the trivariate T-splines knot insertion and the refinement technique we documented above, we can refine the parametric domain easily according to the point distribution and sharp features.

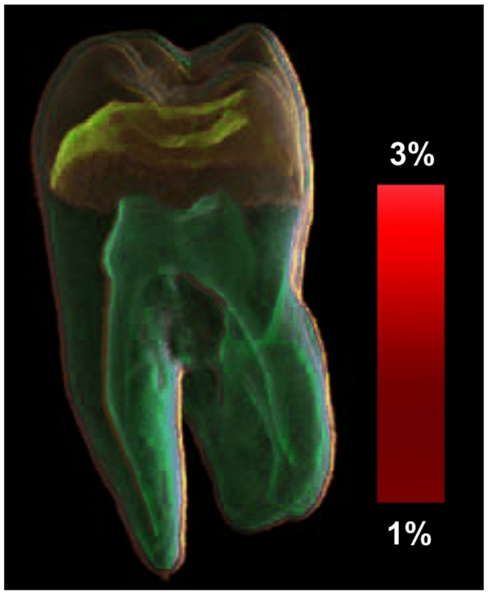

Figure 9: Opacity Fitting Error on Tooth Model.

We can also resample the point data, and get more sample points on the material boundary, in order to get a more precise model. Furthermore, since our method is based on trivariate T-splines, we can easily perform cutting and patching work, which is usually a main advantage when using T-splines. Table 1 summarizes the statistics of the performance of our color/opacity fitting algorithm in three models.

Figure 10 and Figure 11 show another two simpler models and significant color/opacity fitting error (red part).

Table 1 shows the statistics of the performance of our color/opacity fitting algorithm on the above four model.

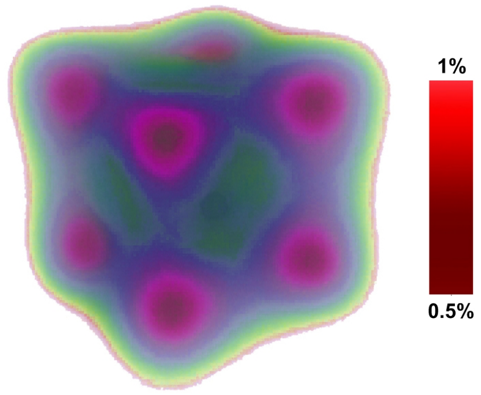

Figure 10: Opacity Fitting Error for Dice Model.

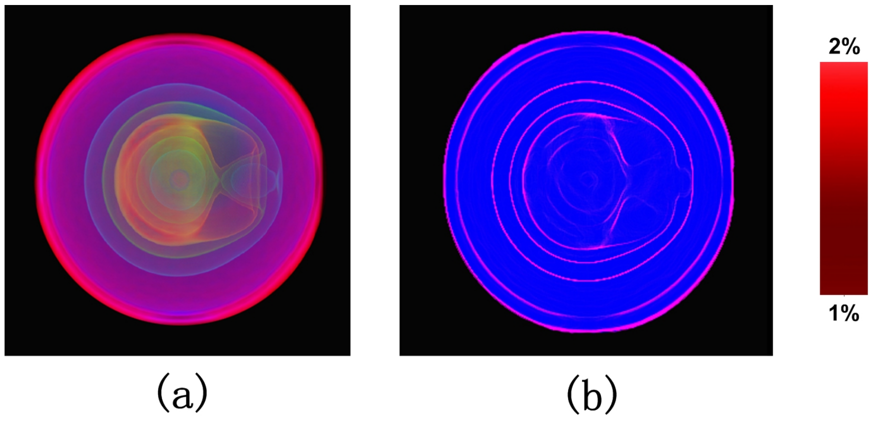

Figure 11: Color Fitting Error for Particle Model.

Table 1: The Statistics of Data Fitting.

Dataset	Tooth	Fuby	Dice	Particle
Datasize	100^3	256^3	20^3	32^3
Control Points	$1.2 * 10^5$	$2 * 10^6$	$8 * 10^2$	$3.5 * 10^3$
Max Color Fitting Error	1.35%	2.17%	0.11%	1.77%
Average Color Fitting Error	0.09%	0.19%	0.01%	0.12%
Max Opacity Fitting Error	2.01%	1.28%	0.79%	0.24%
Average Opacity Fitting Error	0.07%	0.10%	0.16%	0.02%
Running Time (s)	121.7	1017.2	12.5	35.3

Chapter 4

Heterogeneous Scalar Visualization

4.1 Introduction of Ray-casting

Volume rendering is a technique used to display a 2D projection of a 3D discretely sampled data set on the screen. It has been developed to overcome problems of the accurate representation of parametric volume with many different material attributes, here we only use RGBA (for red, green, blue, alpha). A direct volume renderer requires every sample value to be mapped to opacity and a color. This is done with a transfer function which can be a simple ramp, a piecewise linear function or an arbitrary table. Once converted to an RGBA value, the composed RGBA result is projected on correspondent pixel of the frame buffer. The way this is done depends on the rendering technique. Many rendering techniques have been developed during the last five decades (see Figure 12).

Ray casting is one of these methods, which uses ray-surface intersection tests to computes 2D images from 3D data sets so it can provide results of very high quality, usually considered to provide the best image quality. Ray-casting considers the model along arbitrary ray **r**, which is generated for each desired image pixel:

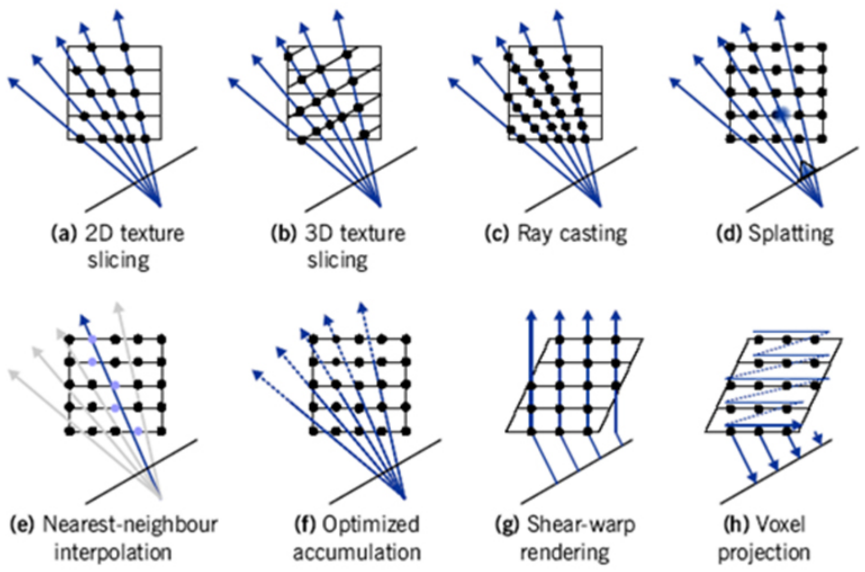

Figure 12: Common Methods in Volume Rendering.

$$r = r(t) : \quad t \longmapsto q_0 + tr_0, \quad t \geq 0, \tag{25}$$

where $q_0 \in IR^3$ is the position of the viewer and $r_0 \in IR^3$ is the (normalized) viewing direction determined as the difference of the current pixel position in the projection plane and q_0. Using a simple camera model, the ray starts at the center of projection of the camera (usually the eye point, which is q_0 here), and passes through the image pixel on the imaginary image plane floating in between the camera and the model to be rendered. Firstly, we need to clip the ray by the boundaries of the object model. Hence our goal is to find the smallest parameter $t^* \geq 0$ therefore $q^* = r(t^*)$ is the point closest to the viewer position. A standard ray-casting algorithm generates rays through all pixel positions, examines the model along each ray in order to find the closest intersection point q^* with the surface. Finally, the gradient for proper visualization value at the current pixel position is evaluated.

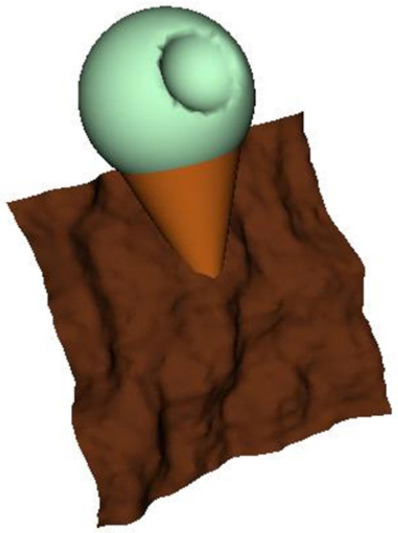

Figure 13: Volume Classification.

4.2 Volume Classification

In some applications it is very useful to segment a volume dataset into various components based on the similarity of material attributes, so that each component can be processed separately.

We usually conduct volume segmentation before volume rendering. One of the most familiar technique is known as thresholding. A ”seed” is usually planted in a part of volume which is interested to users, and then it can grow until it fills the region encompassed by an isosurface of a given threshold value. During the process, all points inside this part should be labeled. Here we introduce a secondary storage of volume with identical dimensions, which is used to store labels indicating which feature is present at the corresponding points in the volume.

The whole volume model could consist of several feature parts, and sometimes there would be overlapping area among different parts generated by thresholding. Figure 13 shows the eye ball, conoid and a rugate surface in different colors yet these features have identical or overlapping ranges of grey-level data values. These area can be removed by eroding or shrinking the connected features. Thresholding sometimes might not identify the whole parts of a given feature. We can solve this problem by using more "seeds" at a suitable position. For instance, in Figure 13 $air = 0$, $eyeball = 1$, $conoid = 2$ and $rugate\ surface = 3$.

4.3 Volume Ray-casting

In our paper, we use volume ray casting (sometimes called volumetric ray casting or volume ray marching), which is classified as image based volume rendering technique, as the computation emanates from the output image, not the input data as is the case with object based techniques. Volume ray casting, which processes volume data, is different from simple ray casting, which only processes surface data.

The technique of volume ray casting can be derived directly from the rendering equation. The first few steps are the same as simple ray casting technique, which aims on finding the closest point on the model surface to the viewer position along a certain ray. But in volume ray casting we should also find the farthest point on the model surface that ray exits from. Moreover, we will sample the ray between the closest and farthest points at regular intervals throughout the volume. The data is interpolated at each sample point in a standard volume ray casting algorithm, the transfer function applied to form an RGBA sample, the sample is composite onto the accumulated RGBA of the ray, and the process repeated until the ray exits the volume.

Since it needs accuracy, which means a lot of calculations, volume ray casting provides results of very high quality at the price of long runtime, usually it is considered to be both the best and the slowest technique.

Here we introduce volume ray casting algorithm in three basic steps (also see Figure 14):

1. **Raycasting**. For every pixel of the final 2D projection image, a ray $\mathbf{r}$ is generated from the position of viewer to this pixel, and go through the whole volume model. As mentioned in Section 4.1, a ray could be represented as $q_0 + tr_0, \quad t \geq 0$. In our implementation, q_0 can be represented as u_0, v_0, w_0 since we are using the UVW parameterization

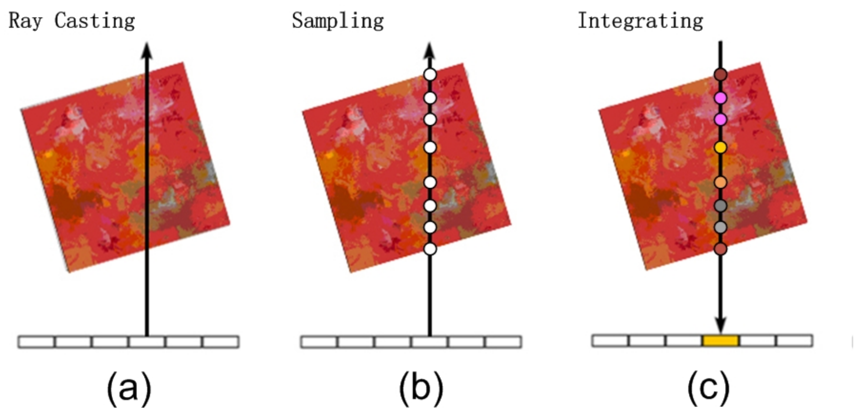

Figure 14: Three Basic Steps of Volume Ray Casting.

coordinates. As mentioned in Section 3.1, in order to save the calculation time, u, v, w are set linearly correlative to x, y, z respectively. Because only in this way can we keep the ray formulas still in linear format, which are much easier to calculate the coordinates of the intersection points. At this stage we can firstly skip most of the empty area in T-lattice (details in Section 4.4.1), and then focus on the rest parts to find the first and the last volume position being touched by the ray (details in 4.4.2).

2. **Sampling**. In volume ray casting, we should not only find out the intersection positions of ray and volume, but also trace along the ray direction into the volume, to sample the inside points as well. Standard volume ray casting technique samples the ray at regular intervals throughout the volume until it exit the model. And because the volume points are discrete and not necessarily aligned along the ray, trilinearly interpolations are usually used to sample the final point from its surrounding data points. In this paper, we will calculate the sample points exactly and accurately based on the advantage of our modeling using trivariate T-spline. So we can get material attributes of a certain point accurately instead of interpolating using data points surrounding the sample points. Moreover we will sample the points along the ray at adaptive intervals instead of regular intervals in order to save computational cost (details

in Section 4.4.3).

3. **Integrating**. The shaded information, i.e. color and opacity is computed for each sampling point during sampling process. After all the sample points have been shaded, they are integrating along the ray, resulting in the final (materials) attributes for the pixel that is currently being processed. This process is derived directly from the rendering equation and is very similar to blending many information of different points into one point (voxel shown on the screen). Details can be checked in Section 4.4.4.

4.4 Details in Implementation

4.4.1 Space Skipping for Empty Regions

In its standard use, any casted ray is traversed and executed across the entire bounding volume of the data set, including all the regions with zero alpha (i.e., totally opaque) or irrelevant attributes (i.e., totally empty) along the ray. To improve, we shall skip opaque and/or empty regions so that the only part for display purpose includes those relevant regions with meaningful opacity.

We mentioned in Section 3.1 that the bounding box is exactly the T-lattice box of the volume model, and it is obvious that blocks in T-lattice do not actually re-arrange the volume. Here we will only focus on the boundary blocks (i.e., blocks containing the boundary of the volume model) for calculating intersections. In this way a ray-box intersection involves less computation and hence speeds up the process. We assign each block a flag to indicate whether they are "active" or not. Here "active" means that the ray would probably intersect with the volume part inside the block. For each block, min-max values in three directions (u, v and w) are simply stored in an additional data structure for culling purpose. When a new ray is generated, blocks are culled against the ray using their min-max information and a range query [10] which can determine whether they are "active" or not. For each ray, we know which block would be penetrated, and we set them as "active" and those would not be intersected as "inactive". After we get all the "active" blocks, we put a seed inside the "active" area and let it "grow" in all directions, once it hit the "inactive" blocks it stop growing in that direction and the last "active" block is marked as boundary block. One important fact is that sometimes there are more than two boundary blocks that are intersected by one ray, which means the ray could pass through a region of interest twice or more. Hence for each ray we examine all the boundary blocks for intersection detection. In this way we only focus on the intersections inside boundary blocks and skip the other ones.

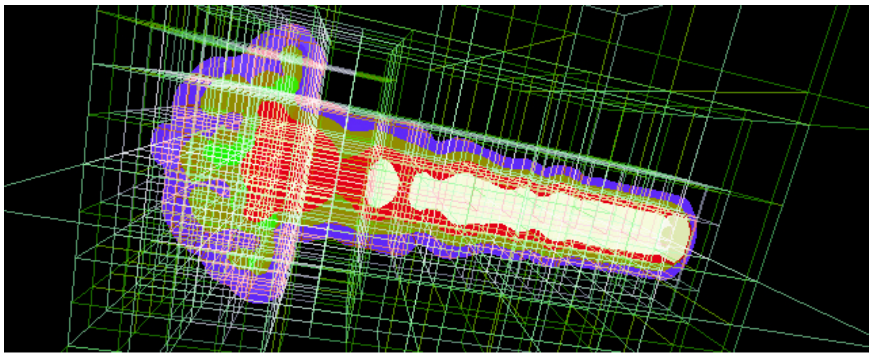
Figure 15: Octree Subdivision

4.4.2 Intersection Detection and Refinement

Many literatures [3, 43, 12, 33, 24] have discussed the rendering issue for spline models. The intersection point of a ray and the volume must be calculated iteratively because the underlying T-spline volume is modeled by a multivariate rational polynomial. The Newton method generally converges very fast and is well suited if the initial guess is close to the final solution. In Section 4.4.1 we introduced the concept of boundary blocks of trivariate T-spline models in order to skip empty or opaque space. We only focus on the domain part where the opacity is in (0,1). Now suppose we already know a block would be intersected by a given ray. We adaptively subdivide the block into parts during the preprocessing step. This subdivision is done by halving the volume along the u, v or w axis (just like the octree structure, see Figure

15). Subdivision will be repeated until the data point density of each part is below to a threshold. By intersecting the ray with these parts we get very good initial points for the following iterative formulation based on the Newton method. We use the following formula to find the intersection point:

$$p_0 = q_0 + t_0 * \vec{r_0}. \tag{26}$$

We now need to compute the coefficients t' for p_0 in order to obtain the first approximate values of enter point u_0, v_0, and w_0, and we can get

$$\frac{u_0 - \underline{u}}{\overline{u} - \underline{u}} = t', \tag{27}$$

$$\frac{v_0 - \underline{v}}{\overline{v} - \underline{v}} = t', \tag{28}$$

$$\frac{w_0 - \underline{w}}{\overline{w} - \underline{w}} = t', \tag{29}$$

where $\underline{u}, \underline{v}, \underline{w}$ denote the first intersection and $\overline{u}, \overline{v}, \overline{w}$ denote the last intersection. Next we use u to describe v and w as $v = V(u)$ and $w = W(u)$. Now $O(u, v, w)$, which denoted as the opacity in $p_{u,v,w}$, can be changed to $O'(u)$. Suppose we are now finding the enter point p_{enter}, and $O'(\underline{u}) = 0$. We can set a coefficient ς small enough to let $O'(u) = \varsigma$. The solution of this formula is the intersection of the ray and the volume surface. Next we calculate the partial derivative of this formula to compute a new approximation point. The whole iteration continues until two succeeding points differ less than a given threshold ε, which is calculated by the maximum of the differences between u, v and w values. In this way we can get the enter point p_{enter}, and the exit point p_{exit} of the ray could be calculated in a similar way.

There exists special case where the iteration cannot terminate, which only appears when rays fail to intersect with the model inside the boundary block, as a result the iteration cannot converge, and we make use of the following strategies:

- We set a threshold H as the maximum number of iterations. If the convergence criteria are not met in few than H iterations, we assume that the ray does not intersect with this part of the model.

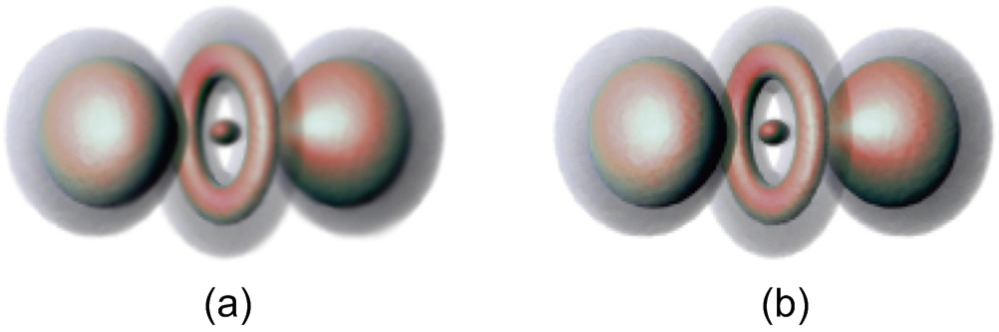

Figure 16: Intersection Detection on Coolball.

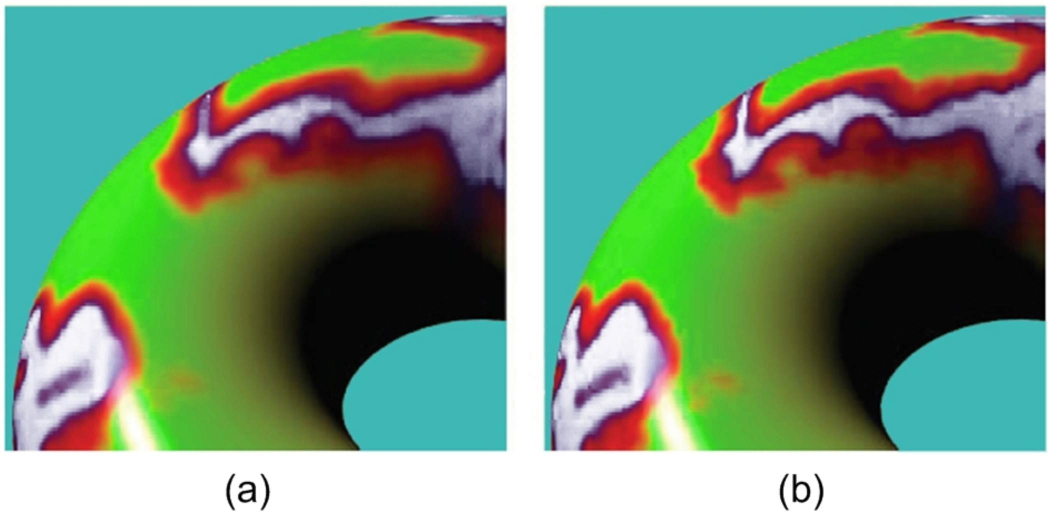

Figure 17: Intersection Detection on ColorLoop.

- The following way to handle the miss-hit ray is that we suppose the intersection probably lies inside the domain of a neighboring block when the iteration leaves the u, v, w domain of the current block. So we check the border of the domain to see whether the iteration approaches to the solution from that side. If we still can not find the iteration point inside the domain in the next iteration step we assume that the ray fails to intersect with any part of the neighbor.

Using these two rules, we can identify all the rays which do not intersect with the actual model of current interest.

We conduct intersection detection by using T-splines and discrete point cloud for comparison. Figure 16 and Figure 17 show the different result between them. Figure 16a shows the result from discrete point cloud. Compared to Figure 16b using trivariate T-splines, the result is far away from precise, which is because from discrete point cloud we cannot get the accurate intersection. In Figure 17, we can also see by using intersection detection with trivariate T-splines (Figure 17a), visualization quality is improved.

4.4.3 Adaptive Sampling

There is a tradeoff between quality and computation cost during sampling in ray-casting. If we select more sample points, the more accurate the visualization result would be, but the more computation would be needed. Since our volume object is modeled by trivariate T-splines, the representation is continuous. If we want the visualization result as accurate as possible (especially that we do not want to miss any silhouette), we can set the interval arbitrarily small in order to get more sample points. However, it would lead to an huge number of samples and the computation cost is extremely large. Here we use adaptive sampling to get a good tradeoff between quality and computation cost, we use the following formula:

$$p_i(u, v, w) = q_0 + t_i \vec{r}(u, v, w) \tag{30}$$

to calculate the point position along each ray inside the model. Parameter t_i needs to be set between t_{enter} and t_{exit} (here we suppose that the ray only intersect with the volume model once, and the process is similar if the ray intersects twice or more). p_{enter} and p_{exit} of the ray can be detected using the method explained in Section 4.4.2. Other points along the ray inside the model can be easily obtained by simply calculating the proportion between t_{enter} and t_{exit} using Equation (29).

After the actual intersections on the model boundary p_{enter} and p_{exit} have been detected, we start to sample points along the internal ray segment. Different from the standard way to sample points with a fix interval, here we use three different sampling intervals, τ_0, τ_1 and τ_2, which descend gradually (we are currently using sampling intervals τ_1 and τ_2 as a constant multiple of τ_0, $\tau_1 = \tau_0/5$ and $\tau_2 = \tau_0/10$). The first sampling interval τ_0 is used inside the part where materials are almost homogeneous. The other two sampling intervals τ_1 and τ_2 are used to improve the quality near material boundary. We do not detect the material boundary explicitly because the cost is tremendous. Instead, we compute the average data fitting error for the points inside a small sphere each time. Here we set up two thresholds δ_1 and δ_2:

- If the fitting error is smaller than δ_1, we use τ_0.
- If the fitting error is equal to or larger than δ_1 but smaller than δ_2, we use τ_1.
- If the fitting error is equal to or larger than δ_2, we use τ_2.

For quality control purpose, τ_0, δ_1, and δ_2 are all set by the user. In our current implementation, we set δ_1 and δ_2 based on color fitting error (here we neglect opacity fitting error) for each model. Parameter δ_1 is set as the average color fitting error and δ_2 is set as the mean value of average and maximum color fitting error of the whole model.

Using the approach here, we can see that a fastest computation speed can be obtained when using τ_0 as the sampling rate. And slower speed is used in material boundary with guaranteed accuracy. This way we can get a better tradeoff between quality and computational cost.

We test our adaptive sampling on two models. By using adaptive sampling, the samples of the right pictures (running time is 30s) are 25% less than the left one (using uniform sampling, running time is 37s) in Figure 18. But the samples of the right pictures (using adaptive sampling, running time is 54s) just 12% less than the left one (using uniform sampling, running time, running time is 61s) in Figure 19, because the model used here is not smooth and uniform as the model used in Figure 18. The similarity of both images in Figure 18 and Figure 19 indicates that visual quality is preserved in the adaptive, reduced sampling.

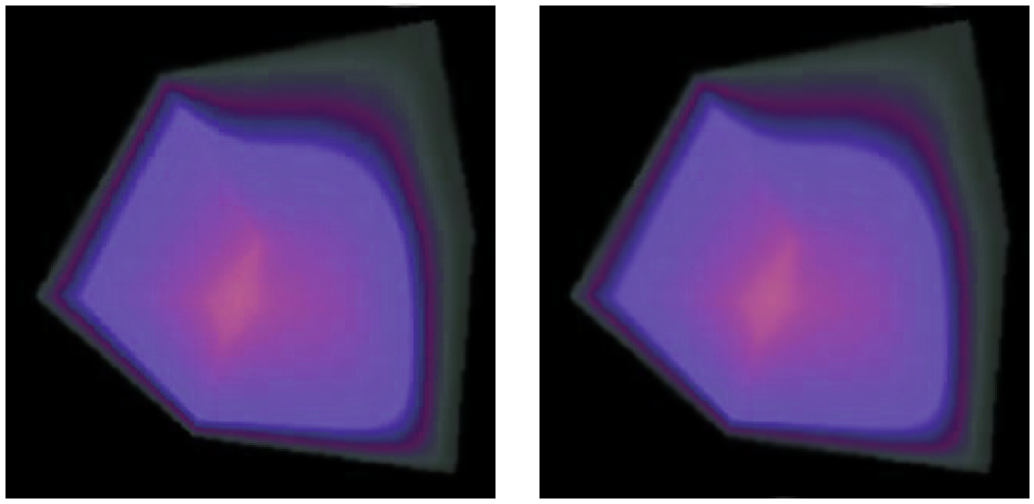
Figure 18: Adaptive Sampling on Cube A.

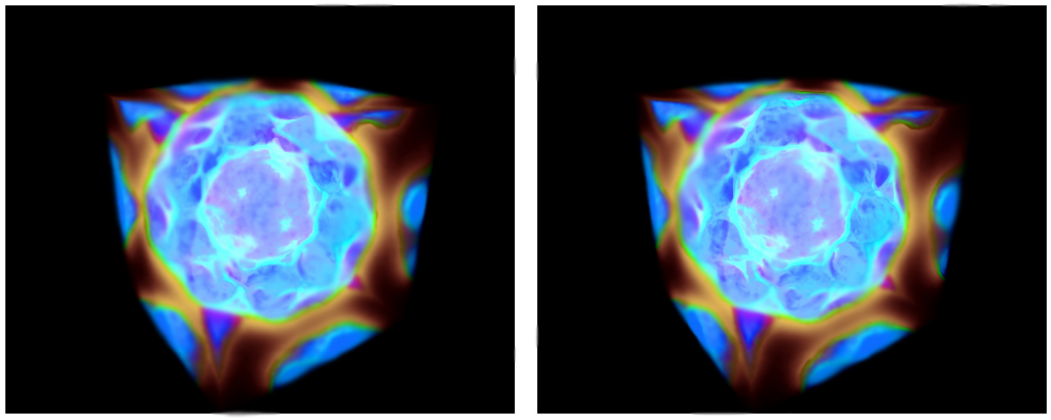
Figure 19: Adaptive Sampling on Cube B.

4.4.4 Attribute Integrating

Volume rendering enhances visualization of imaged model by providing semi-translucent rendering. In our paper the object is visualized by integrating material attributes along the path of each casted ray. Users can use the thresholds (opacity, color, or brightness) to control the visualization effect of rendering. Here we assume each sample along a given ray has only two material attributes, color $\tilde{C_i}$ and opacity $\tilde{\alpha_i}$:

$$(\tilde{C_0}, \tilde{\alpha_0}), (\tilde{C_1}, \tilde{\alpha_1}), ..., (\tilde{C_N}, \tilde{\alpha_N}) \quad \tilde{C_i} \in [0,1]^3, \tilde{\alpha_i} \in [0,1],$$

where the first sample is set as a point right before p_{enter}, the last sample is background (black and fully opaque) and set right after p_{exit}. An approximation of the volume rendering equation using the emission-absorption model [46] is employed to achieve pre-integration:

$$I(r) = \int_0^r q(t) e^{-\sigma(0,t)} dt, \tag{31}$$

where I is the intensity at r along a ray, $r \in [0, B]$ ($r = B$ is the background location), the emission function $q(t)$ describes the photons emitted by the volume along the ray, and the absorption function $\sigma(0,t)$ can be viewed as the optical depth defined as

$$\sigma(t_1, t_2) = \int_{t_1}^{t_2} \kappa(\tau) d\tau, \tag{32}$$

where κ is the opacity function. In order to compute the integral equation, the interval $[0, B]$ includes all the sampling points derived from Section 4.4.3 and two additional points on the two sides as $t_0 = 0$ and $t_N = B$.

Now we use an n-point Gaussian Quadrature rule [18] as an approximation of the definite integral, by using a weighted sum of function values at specified points within the domain of integration:

$$I(t) = \sum_{k=0}^{N} q_i \Delta t e^{-\sum_{j=0}^{i-1} \kappa_j \Delta t} = \sum_{k=0}^{N} q_i \Delta t \prod_{j=0}^{i-1} e^{-\kappa_j \Delta t}, \tag{33}$$

which matches the α-compositing formula:

$$C_a^b = \int_a^b q(t) e^{-\sigma(a,t)} dt = \sum_{i=a}^{b} \alpha_i C_i \prod_{j=a}^{i-1} (1 - \alpha_j) \tag{34}$$

and the voxel opacity α_i is

$$\alpha_i = 1 - e^{-\sigma(t_i, t_{i+1})}. \tag{35}$$

Equation (33) and (34) can be rewritten into a recursive front-to-back compositing equation:

$$\tilde{C_k} = \tilde{C_{k-1}} + (1 - \tilde{\alpha_{k-1}})C_k, \tag{36}$$

$$\tilde{\alpha_k} = \tilde{\alpha_{k-1}} + (1 - \tilde{\alpha_{k-1}})\alpha_k, \quad k = 1, 2 ... N - 1, \tag{37}$$

where $\tilde{C_k}$ is the pixel color, $\tilde{\alpha_k}$ is the pixel opacity, and $\tilde{C_0} = C_0$, $\tilde{\alpha_0} = \alpha_0$. We apply this framework to each ray cast into the volume, so the color and opacity of each voxel is computed in the front-to-back order. In this way we can terminate the ray as soon as the composite transparency falls below a threshold. This is also called "early ray termination".

Figure 20 and Figure 21 show experimental results of our volume rendering algorithm on two models.

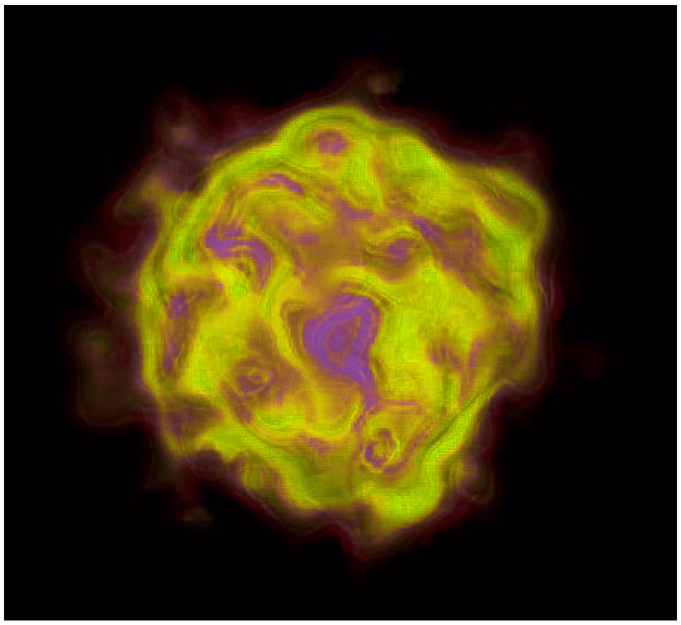

Figure 20: Volume Rendering on StarCloud

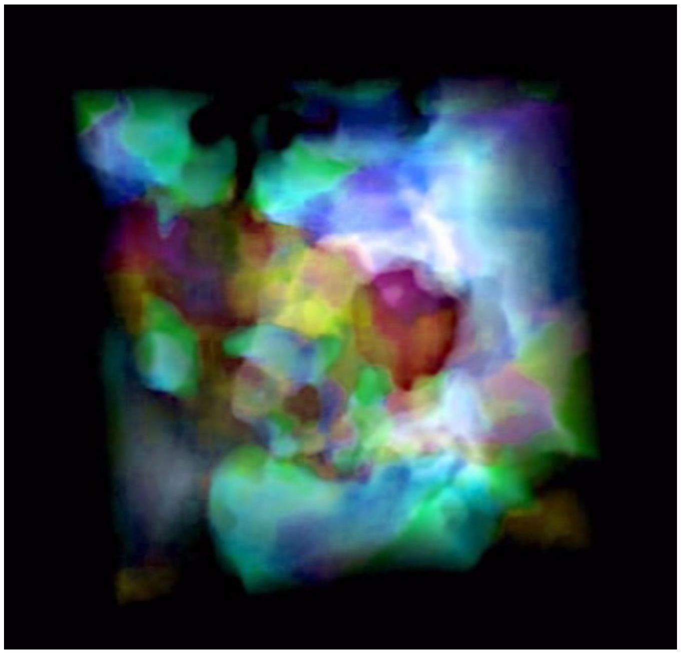

Figure 21: Volume Rendering on Polychrome

Chapter 5

Conclusion

In this paper, we have detailed the algorithm to represent and visualize a heterogeneous model using trivariate scalar T-splines.

Most of research use B-splines and simplex splines to model heterogeneous objects. However, knot insertion is not a local process in B-splines and it will increase the number of control points exponentially, hence B-splines only have global refinement. Simplex splines can perform local refinement, however, its domain is irregular and hence the process of choosing partitioning into simplices is hard.

To overcome these problems, trivariate T-splines are used in this paper to build up heterogenous material representation. Since control grids/lattice of permit T-junctions, lines of control points need not traverse the entire control grid and we can do not only local refinement but also more smooth merging on the heterogeneous models. T-spline has principal-axis-aligned regular domain and is a more efficient way to model object. Also, heterogeneous volumetric objects can be adaptively refined in a hierarchical manner (hierarchical refinement) by this modeling method.

Moreover, we have also advocated a framework to render trivariate T-spline volume via ray-casting, adaptive sampling, and accurate intersection. Since trivariate T-splines afford a continuous representation, we greatly benefit from this precise and compact mathematical formulation that will facilitate the modeling and visualization tasks in various graphics applications. We use several technique, such as empty space skipping, adaptive sampling, intersection refinement and attribute integrating to get both efficient and accurate models.

Finally experimental results have started to show great promise of our trivariate scalar T-splines in graphics, visualization, and engineering design.

Bibliography

[1] V. Adzhiev, E. Kartasheva, T. Kunii, A. Pasko, and B. Schmitt. Hybrid cellular-functional modeling of heterogeneous objects. *Journal of Computing and Information Sciences in Engineering*, 2(312-321), 2002.

[2] T. M. B. Finkbeiner, A. Entezari and D. V. D. Ville. Efficient volume rendering on the body centered cubic lattice using box splines. *Technical Report CMPT2009-04, Graphics, Usability and Visualization (GrUVi) Laboratory*, 2009.

[3] W. Barth and W. Sturzlinger. Efficient ray tracing for bezier and b-spline surfaces. *Computers and Graphics*, 17(4)(423-430), 1993.

[4] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. Evans, S. Lipton, M. Scott, and T. W. Sederberg. Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(229-263), 2010.

[5] A. D. Bhatt and R. M. Warkhedkar. Reverse engineering of human body: A b-spline based heterogeneous modeling approach. *Computer-Aided Design and Applications*, 5(1-4)(194-208), 2008.

[6] A. Biswas and V. Shapiro. Approximate distance fields for curves and surfaces. *Technical report SAL, 2001-3. University of Wisconsin-Madison, Mechanical Engineering Department*, 2001.

[7] A. Biswas, V. Shapiro, and I. Tsukanov. Heterogeneous material modeling with distance fields. *Computer Aided Geometric Design*, 21(215-242), 2004.

[8] K. Z. Chen and X. Feng. An computer-aided design method for the components made of heterogeneous materials. *Computer-Aided Design*, 35(453-466), 2003.

[9] M. Chen and J. Tucker. Constructive volume geometry. *Computer Graphics Forum*, 19(4)(281-293), 2000.

[10] Y. J. Chiang, C. T. Silva, and W. J. Schroeder. Interactive out-of-core isosurface extraction. *In Proceedings of IEEE Visualization 98*, (167C174), 1998.

[11] J. R. Cho and D. Y. Ha. Optimal tailoring of 2d volume fraction distributions for heat resisting functionally graded materials using fdm. *Computer methods in applied mechanics and engineering*, 191(3195-3211), 2002.

[12] A. Efremov, V. Havran, and H. P. Seidel. Robust and numerically stable bzier clipping method for ray tracing nurbs surfaces. *Proc. 21st Spring Conference on Computer Graphics,ACM Press*, (127-135), 2005.

[13] R. Feichtinger, B. Jttler, and H. Yang. Particle-based t-spline level set evolution for 3d object reconstruction with range and volume constraints. *In V. S. S. Cunningham, Proc. WSCG 2008*, (49-56), 2008.

[14] M. S. Floater. High order approximation of rational curves by polynomial curves. *Comp. Aided Geom. Design.*, 23(621-628), 2006.

[15] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling*, (157–186), 2005.

[16] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. *Computer Graphics (SIGGRAPH 88 Proceedings)*, (205-212), 1988.

[17] D. R. Forsey and D. Wong. Multiresolution surface reconstruction for hierarchical b-splines. *Tech. rep., University of British Columbia*, 1995.

[18] A. Gil, J. Segura, and N. M. Temme. Numerical methods for special functions. *SIAM*, 2007.

[19] J. Hua, Y. He, and H. Qin. Multiresolution heterogeneous solid modeling and visualization using trivariate simplex splines. *Proceedings of the ninth ACM symposium on Solid modeling and applications*, (47-58), 2004.

[20] J. Hua, Y. He, and H. Qin. Trivariate simplex splines for inhomogeneous solid modeling in engineering design. *ASME Transactions: Journal of Computing and Information Science in Engineering*, 5(149-157), 2005.

[21] J. Huang and G. M. Fadel. Heterogeneous flywheel modeling and optimization. journal of materials and design. *Journal of Materials and Design*, 21(111-125), 2000.

[22] T. Jackson. Analysis of functionally graded material representation methods. *Ph.D. thesis. Cambridge (MA): Massachusetts Institute of Technology*, 2000.

[23] T. R. Jackson, N. M. Patrikalakis, E. M. Sachs, and M. J. Cima. Modeling and designing components with locally controlled composition. *Materials and Design*, 20(63-75), 1999.

[24] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics*, 15(6)(1571-1578), 2009.

[25] X. Y. Kou and S. T. Tan. Heterogeneous object modeling: A review. *Computer-Aided Design*, 39(284-301), 2007.

[26] V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. A framework for object modeling. *Computer-Aided Design*, 31(9)(541-546), 1999.

[27] V. Kumar and D. Dutta. An approach to modeling multi-material objects. fourth symposium on solid modeling and applications. *ACM SIGGRAPH '97*, (336-345), 1997.

[28] V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. *ASME Journal of Mechanical Design*, 120(4)(659-667), 1998.

[29] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, (362-371), 2002.

[30] H. Liu, T. Maekawa, N. M. Patrikalakis, E. M. Sachs, and W. Cho. Methods for feature-based design of heterogeneous solids. *Computer Aided Design*, 36(1141-1159), 2004.

[31] W. Martin and E. Cohen. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. *Sixth ACM Symposium on Solid Modeling and Applications*, (234-240), 2001.

[32] C. M. Muller-Karger, E. Rank, and M. Cerrolaza. P-version of the finite -element method for highly heterogeneous simulation of human bone. *Finite Elements in Analysis and Design*, 40(757-770), 2004.

[33] H. Pabst, J. Springer, A. Schollmeyer, R. Lenhardt, C. Lessig, and B. Froehlich. Ray casting of trimmed nurbs surfaces on the gpu. *Proc. IEEE Symp. Interactive Ray Tracing*, (151-160), 2006.

[34] S. M. Park, R. H. Crawford, and J. J. Beaman. Volumetric multi-texturing for functionally gradient material representation. *Sixth ACM symposium on solid modeling and applications*, (216-224), 2001.

[35] A. Pasko, V. Adzhiev, B. Schmitt, and C. Schlick. Constructive hypervolume modelling. *Graphical Models, Special issue on volume modeling*, 63(6)(413-442), 2002.

[36] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modelling: concept, implementation and applications. *The Visual Computer*, 11(8)(429-446), 1995.

[37] L. Patil, D. Dutta, A. Bhatt, K. Jurrens, K. Lyons, and M. Pratt. Representation of heterogeneous objects in iso 10303 (step). *ASME international mechanical engineering congress and exposition*, (5-11), 2000.

[38] L. Patil, D. Dutta, A. D. Bhatt, K. Jurrens, K. Lyons, M. J. Pratt, and R. D. Sriram. A proposed standards-based approach for representing heterogeneous objects for layered manufacturing. *Rapid Prototyping Journal*, 8(3)(134-146), 2002.

[39] M. J. Pratt. 3d modelling of material property variation for computer aided design and manufacture. *Proceedings of the SIAM conference on Mathematics of industry: Challenges and frontiers*, (58-83), 2003.

[40] M. J. Pratt, A. D. Bhatt, D. Dutta, K. W. Lyons, L. Patil, and R. D. Sriram. Progress towards an international standard for data transfer in rapid prototyping and layered manufacturing. *Computer-Aided Design*, 34(1111-1121), 2002.

[41] X. Qian and D. Dutta. Physics-based modeling for heterogeneous objects. *ASME Transactions: Journal of Mechanical Design*, 125(416-427), 2003.

[42] X. Qian and D. Dutta. Feature-based design for heterogeneous objects. *Computer Aided Design*, 36(1263-1278), 2004.

[43] A. Raviv and G. Elber. Interactive direct rendering of trivariate b-spline scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 7(109-119), 2001.

[44] C. Rossl, F. Zeilfelder, G. Nurnberger, and H. P. Seidel. Visualization of volume data with quadratic super splines. *Proceedings of the 14th IEEE Visualization*, (52-67), 2003.

[45] V. L. Rvachev, T. I. Sheiko, V. Shapiro, and I. Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18(195-220), 2001.

[46] P. Sabella. A rendering algorithm for visualizing 3d scalar fields. *Proc. SIGGRAPH, Computer Graphics*, 22(4)(51-58), 1988.

[47] K. Samanta and B. Koc. Feature-based design and material blending for freeform heterogeneous object modeling. *Heterogeneous Object Models and their Applications, Computer Aided Design*, 37(287-305), 2005.

[48] B. Schmitt, A. Pasko, and C. Schlick. Constructive sculpting of heterogeneous volumetric objects using trivariate b-splines. *The Visual Computer*, 20(130-148), 2004.

[49] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and t-nurccs. *ACM Transactions on Graphics 2003*, (477-484), 2003.

[50] K. H. Shin and D. Dutta. Constructive representation of heterogeneous object. *Journal of Computing and Information Science in Engineering*, 1(205-217), 2001.

[51] Y. K. Siu and S. T. Tan. Source-based heterogeneous solid modeling. *Computer-Aided Design*, 34(41-45), 2002.

[52] W. Song and X. Yang. Free-form deformation with weighted t-spline. *The Visual Computer: International Journal of Computer Graphics*, 21(139-151), 2005.

[53] W. Sun and X. Hu. Reasoning boolean operation based modeling for heterogeneous objects. *Computer-Aided Design*, 34(481-488), 2002.

[54] W. Sun, F. Lin, and X. Hu. Computer-aided design and modeling of composite unit cells. *Computer Science and Technology*, 61(289-299), 2001.

[55] M. Y. Wang, S. K. Chen, X. Wang, and Y. Mei. Design of multi-material compliant mechanisms using level set methods. *ASME Transactions: Journal of Mechanical Design*, 127(5)(941-956), 2005.

[56] V. Weiss, L. Andor, G. Renner, and T.Varady. Advanced surface fitting techniques. *Computer Aided Geometric Design*, 19(19-42), 2002.

[57] Z. Wu, H. S. Seah, and F. Lin. Nurbs-based volume modeling. *International workshop on volume graphics*, (321-330), 1999.

[58] B. Wyvill, E. Galin, and A. Guy. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2)(149-158), 1999.

[59] H. Yang, M. Fuchs, B. Juttler, and O. Scherzer. Evolution of t-spline level sets with distance field constraints for geometry reconstruction and image segmentation. *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, (37-42), 2006.

[60] P. Yang and X. Qian. A b-spline-based approach to heterogeneous objects design and analysis. *Computer-Aided Design*, 36(95-111), 2007.

[61] A. Yvart, S. Hahmann, and G. P. Bonneau. Hierarchical triangular splines. *ACM Transactions on Graphics*, 24 (4)(1374-1391), 2005.