# Fast poligonization and texture map extraction from volumetric objects based on surface fairing using a modified discrete laplacian operator

Carlos Eduardo Vaisman Muniz Anselmo Montenegro Esteban Walter Gonzalez Clua

Universidade Federal Fluminense, Instituto de Computação, Brasil

## ABSTRACT

This work proposes a new method for the polygonization and texture map extraction from volumetric objects based on a polyhedral surface extraction and surface fairing approach. The surface faring approach is based on a mesh signal processing technique that uses an approximation of an adaptive low-pass filtering based on a function that combines the discrete laplacian operator and an adaptation of the Lanczos kernel. Differently from other works that propose surface fairing approaches we deal mainly with voxelized models created by artists for games which led us to design a filtering approach that smoothes strongly in high frequency regions to remove the jagging effects of the sampling and less intensively in the low frequency regions to preserve the natural behavior of the model. The overall aim of this work is to propose a simple methodology in which it is efficient and easy to use voxel manipulation techniques to produce a final boundary representation of the model together with the appearance function codified in a texture atlas. The proposed method produced very promising results and enabled us to extract smooth texturized models that preserve the features of the original volumetric object at interactive times.

**Keywords**: polygonization of volumetric models, mesh fairing, texture atlas, volumetric models

**Authors' contact**:
{cmuniz,anselmo,esteban}@ic.uff.br

## 1. INTRODUCTION

The use of boundary representations (b-rep) is recommended for the storage of complex and large 3D objects describing solids composed by homogeneous inner material. On the other hand, it is not an efficient way to store detailed surfaces with dense variation of attribute values, such as colors, normals, etc. This limitation has been minimized with the use of functional or procedural-based texture mapping [Catmull and Clark, 1978].

Most existing 3D graphics editors uses b-rep, which is an intuitive approach for modeling the geometry of the object. However, these programs require an entire different user interface or even different programs to manipulate texture maps. It does not allow the user to preview the result of the changes that the texture is receiving in real time and it is also hard to visualize which part of the object is being modified as the texture is being painted.

This problem can be circumvented with the direct manipulation of voxel-based models [Levoy, 1988]. It allows users to know which part of the object is being painted and to preview their changes in real time. For this reason, it is a more intuitive approach to add attribute details into a model than mesh modeling combined with functional or procedural-based texture mapping.

In order to make this voxel based modelling approach viable, this work proposes a new method for the polygonization and texture map extraction from volumetric objects based on a polyhedral surface extraction and surface fairing approach.

The surface faring approach is based on a mesh signal processing technique that uses an approximation of an adaptive low-pass filtering based on a function that combines the discrete laplacian operator and a modification of the Lanczos kernel. We compared our approach with Taubin's work [Taubin, 1995] and it produced results more efficiently being able to smooth a model with only one pass for the voxelized models.

The aim of this work is to propose a simple methodology in which it is easy to use voxel manipulation techniques to produce a final boundary representation of the model together with the appearance function codified in a texture atlas to be used in games, allowing users to see which part of the object is being painted and eliminating tasks related to texture generation.

This technique produces results in interactive times and was tested with the help of the program Voxel Section Editor III [VXLSE3, 2010], which has a vast amount of free graphical models, being constantly created using the file format supported by it.

The results produced are quite promising and enabled us to extract smooth texturized models that preserve the features of the original volumetric objects in interactive times.

This paper is organized as follows. In section 2 we present the motivation of our work and the works related to the method proposed. In section 3 we pose the problem and discuss its main challenges and unique characteristics that make it a relevant problem to be solved. In section 4, we present the proposed method, first by showing the overall methodology and, in the sequel, the details of each step that composes it. In section 5, we present some results of the application of our method and finally, in section 6, we present some remarks and propose some future work.

## 2. Motivation and previous works

The method proposed in this work aims at fast polygonization of volumetric data representing a solid that describes a three-dimensional object as well as the extraction of its appearance function in the form of a texture atlas.

Here we are mainly interested in a particular class of the problem of polygonization of solids in which the objects of interest are composed of homogeneous inner material. This is the usual case of computer graphics models used in games, interactive environments and the majority of objects in mechanical simulations.

One of the main challenges of this work is to deal with 3D objects represented by shapes modeled by artists. In this context, one must be aware that an artist may use all possible artistic effects and resources to describe what he considers as an expressive representation of the shape.

Frequently, the shapes produced in an artistic environment are abstractions that are neither appropriate in a mathematical sense nor adequate for computational processes. For example, they are seldom discrete representations of 2D manifolds.

In face of such characteristics we propose a simple and efficient solution based on the smoothing and simplification of the polyhedral surface determined by the faces extracted from the boundary of volumetric data.

Our work is founded on top of three main techniques: Polygonization of volumetric data, surface smoothing and texture atlases extraction. Works related to each one of these techniques are detailed in the next section and compared with our approach.

### *2.1 Polygonization of volumetric data*

Polygonization of volumetric data is a very well investigated problem in the literature of computer graphics and geometrical modeling.

Among the most known methods we highlight the Marching Cubes Method [Lorensen and Cline, 1987] and its different variants including the Extended Marching Cubes [Kobbelt et al, 2001] and dual methods as, for example, the Surface Nets [Gibson, 1998] and the Dual Contouring [Ju et al, 2002].

The Surface Nets method generates one vertex positioned near, or on the contour, for each cube that intersects it.

An example of adaptive method, Dual Contouring is able to produce a contiguous polygonal mesh from a signed octree by minimizing a quadratic error function at each heterogeneous leaf cells which yields vertices that are combined into polygons by using a recursive algorithm.

These methods uses information collected from 3D scanning devices, which includes implicit definitions from the real surface of the object. This information is not available on voxel based artistic assets. That's why all methods based on Marching Cubes are useless for this work.

Our polygonization method is very simple compared to the previous ones. It extracts the polygonal faces at the boundaries of the dual grid defined by the centroid of the voxels. Moreover, the objects are defined by a relatively dense sampling of the volume of occupation of the object so that the boundary voxels gives us, in most cases, but not always, a reasonable detailed description of the overall shape of the contour. In some situations, the artists modeling process may produce sets of voxels that only indicates a surface without describing it precisely in terms of the geometry of the boundary faces of the voxels. How this relates to our work will be discussed later.

### *2.2 Surface smoothing*

Surface smoothing is a vastly investigated topic in geometric modeling. The most used techniques are based on two main approaches: discrete differential operators [Zhang et al, 2005] and optimization-based techniques [Freitag and Plassmam, 2000].

The technique we used is based on the discrete laplacian operator which was proposed in different works and investigated more formally in the work of Taubin [1995] for processing signals defined on discrete surfaces.

The surface fairing method proposed by Taubin is based on the approximation of low-pass filtering for discrete surface signals.

In order to smooth a signal one can project it in the space of low-frequencies using Fourier Descriptors. Nevertheless, there is no extension of the Fast Fourier Transform for $n$-dimensional signals which makes the process computationally inexpensive. In order to deal with this drawback Taubin proposed the use of an approximate projection on low frequency spaces using the Gaussian Filtering, a technique associated with space-scale filtering.

Let $s=(s_1,s_2,\ldots,s_n)^T$ be a discrete time $n$-periodic signal defined on the vertices of a polyhedral surface $S=\{V,E\}$. Besides, let $0 \leq \lambda \leq 1$ be a real value and $\Delta s$ the discrete Laplacian of $s$. Then, a smoothed version $s'$ of $s$ is given by expression (1).

$$s' = s + \lambda \Delta s \quad (1)$$

In matricial form, assuming $K=-\Delta s$ this is written as:

$$s' = (I - \lambda K) \quad (2)$$

In intuitive ways, the new signal s' is obtained by moving each of its points in the direction of the corresponding laplacian which causes a reduction in the differences between each point and its neighbors which leads to smoothing. Another way to understand that is to see equation (1) as a discreticized version of the diffusion equation (3).

$$\frac{\partial s}{\partial t} = \nabla \circ \nabla s \approx s' - s = (I - \lambda \Delta s) \quad (3)$$

The discrete Laplacian can be defined more precisely for a *n*-dimensional signal on a discrete surface in equation (4) where $\varphi$ is a function defined on the edges $E$ of $S$, $w_{ij}$ is a set of weights, and i* is the *first order neighborhood structure* on $V$, where for each pair of vertices $(v_i,v_j)$, $v_i$ is a neighbor of $v_j$ if $v_i$ and $v_j$ share a common face. The neighborhood structure is symmetric if every time $v_i$ is a neighbor of $v_j$ then $v_j$ is a neighbor of $v_i$.

$$\begin{cases} \Delta x_i = \sum_{j \in i^*} w_{ij}(s_j - s_i) \\ w_{ij} = \dfrac{\varphi(v_i, v_j)}{\sum_{h \in i^*} \varphi(v_i, v_j)}, \sum_{j \in i^*} w_{ij} = 1 \end{cases} \quad (4)$$

Let $W$ be the matrix weights with $w_{ij}=0$ if $v_j$ is not a neighbor of $v_i$. Then, matrix $K$ can be rewritten as $K = I - W$ and the smoothing of the signal as:

$$s' = Ks \quad (5)$$

It can be shown that the smoothing process based on an iterative process $s^n = K^n s$ of the Gaussian filtering does not consist in a low-pass filtering leading to mesh shrinkage.

We can understand this effect by knowing that it is possible to project $s$ onto the unit length linear independent eigenvectors $(u_1,u_2,\ldots,u_n)$ of $K$ with corresponding real eigenvalues $0 \le k_1 \le k_2 \le, \ldots, \le k_n \le 2$ (see appendix of [10]), for any choice of weights obeying first order neighborhood as shown in equation (6) .

$$s = \sum_{i=1}^{n} \xi_i \mu_i \quad (6)$$

Hence, we have $K^n s = \sum_{i=1}^{n} \xi_i k_i^n \mu_i$ and in order to define a low-pass filtering we should have $k_i^n \approx 1$ for the smaller eigenvalues which correspond to the low frequencies and $k_i^n \approx 0$ for the higher frequencies. It can be observed that this is not true by the interval of definition of the $k_i$.

In order to guarantee the low-pass filtering effect, Taubin proposes the definition of a new operator $f(k)$ where $f$ is a *transfer function* which enforces the desired properties, that is $f(k)^n \approx 1$ for low frequencies and $f(k)^n \approx 0$ for higher frequencies.

Taubin proposes $f(k)^n \approx (1 - \lambda k)(1 - \mu k)$ which avoids shrinkage. In our work, we wanted to smooth the surface in a minimum number of iterations - if possible in one iteration - while preserving the object features. Hence, differently from other works, we used a function based on the Lanczos kernel [Duchon, 1979] (equation (7)).

$$L(s_i) = \begin{cases} \dfrac{a \sin(\pi s_i) \sin(\pi s_i / a)}{\pi^2 s}, -a \le s_i \le a, s_i \neq 0 \\ 1, s_i = 0 \\ 0, \text{otherwise} \end{cases} \quad (7)$$

### 2.3 Texture Atlases Construction

One fundamental problem in geometric modeling, computer graphics and visualization is how to manipulate attribute functions or maps. In special, the problem of attribute map extraction is one of the most relevant.

The definition of attribute maps usually relies on parameterizations $g : R^2 \rightarrow R^3$ of the plane onto a subset of the $R^3$. Unfortunately, global parameterizations are rarely available or are unknown in many cases, something that leads to the use of attribute atlases.

The problem of texture atlases generation is usually posed as an optimization problem where the goal is to minimize the number of the charts and the distortion of each mapping with or without restrictions imposed. Examples of such approaches are the works of Jonas Sossai et al [Sossai and Velho, 2007] and Cohen-Steiner [Cohen-Steiner et al, 2004]. Both works adopted variational approaches.

In our work we propose a very simple and naïve approach because our intention is to extract the texture atlases in interactive times.

## 3. The Problem Definition

Now let us define more precisely the problem we are supposed to solve by using the method that will be detailed in the next section.

Let $O_1=(S_1,F_1)$ be a solid whose geometrical support $S_1$ is defined by a spatial decomposition of the Euclidian space $R^3$, that is $S_1$ is represented by a set of *voxels* $V$ that describe the shape of $O_1$ and $F_1$ is the attribute function describing the appearance (colors) associated to each element $v$ in $V$. The problem consists in reconstructing a new graphical object $O_2=(S_2,F_2)$ where $S_2$ is represented by a polyhedral surface and $F_2$ is represented as a texture atlas obtained from $F_1$ by a mapping $g:F_1 \rightarrow F_2$. We assume that the result of the reconstruction process satisfies the following properties: (a) $S_2$ is a smoothed version of $S_1$; (b) $S_2$ preserves most of the features in $S_1$; (c) The topology of $S_1$ is preserved in $S_2$.

In a few words, the problem corresponds to the problem of converting one representation based on spatial decomposition into a representation based on intrinsic decomposition of the object in parts that are composed by

simple elements: the faces of the polyhedral surface. The conversion also must take the attribute function from one representation to the other.

Before we proceed, some remarks must be done regarding the choice for the method we will present in the next section.

First of all, the 3D graphical objects we are interested in are solids with homogenous substance, that is, internally the material is constant. Consequently, the extraction of different level sets is not a major concern in this work; we are only interested in the boundary of the solid.

The second important point is that the solid is described by a characteristic function in an enumerative way – there is no implicit function and we have no knowledge where the real surface is – we know only a subsampling of the 3D solid.

Last but not least, the topology of the original surface may not be preserved when the voxelized model is constructed. In fact, the artist may suggest a surface shape whose representation cannot be directly converted into a manifold associated to the surface originally thought. For example the cannon of a tank is basically a cylinder, but may be represented by the artist as a strip of voxels connected by one vertex (or edge) if we consider a cubic representation of it (Figure 1).

In face of such unique characteristics, conventional methods are not the best choice for the extraction of the polyhedral surface. Many are too computationally intensive and are not able to deal with topological issues mentioned before.

All these issues have led us to devise our own methodology for the polygonization of the volumetric representation of homogenous solids widely used in games and virtual environments.

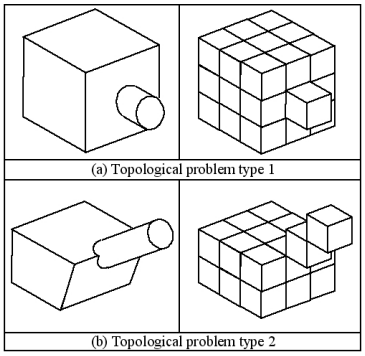


Figure 1. (a) Topological problem of type 1 – a voxel with oposite faces that are neighbor to the exterior of the volume. (b) Topological problem of type 2 – besides having the same property of type 1 the voxel is connected to the model only by vertices or edges.

## 4. THE PROPOSED METHOD

In this section we describe our method. We start with a brief overview and, in the sequel, we present in details the mesh extraction approach we used, the surface fairing method which is one of the main contributions of the work, followed by the attribute smoothing technique and texture atlas extraction. In the last subsection, we point out the simplification method we used to simplify the mesh in order to obtain the final result.

### *4.1 The method overview*

There are five primary steps in our approach to solve the fast polygonization problem. First we acquire the volumetric model. Then, we extract a cube-based surface from it as a mesh composed by a set of quad faces, where each has its own surface normal and color. The third step is to smooth this surface by applying a signal processing technique. Such technique is responsible for smoothing the geometry keeping most of its original details. In this step the mesh is converted into triangles and the surface normal vectors and colors per vertex are computed. The following step is to extract the texture atlas of the model and generate a diffuse texture map. The result will have many useless faces and vertexes that need to be removed in the surface simplification step. The diagram in figure 2 presents the workflow of this technique.

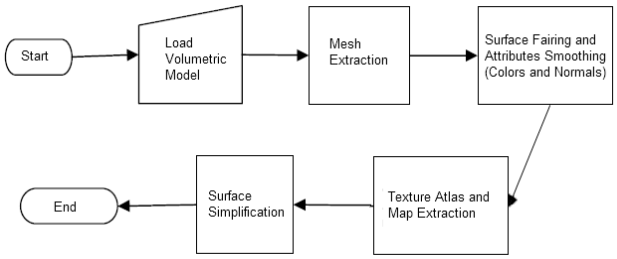


Figure 2. Method workflow.

The next sections will explain each step with more details, starting from the mesh extraction.

### *4.2 Mesh extraction*

This step starts by detecting which voxels are inside the volume, outside or in the surface. Then, it detects which faces of the voxels from the surface should be part of the final model. The vertices of these faces must be ordered in the counterclockwise direction and each quad should hold the color of the original voxel that it belongs to. Once the initial model is generated, unnecessary vertices are removed.

In order to detect which faces are inside and outside the volume, it uses a new buffer to store the volume. This buffer has the dimensions of the original volume plus two for each axis. The volume is centralized in this new buffer and each element should receive the value one for the non-used voxels and zero for the used. This step proceeds with a flood and fill algorithm that starts at the position (0,0,0) with the value zero. That will provide the internal voxels that were not

originally painted by the user. Once this buffer is merged with the painted voxels from the original volume, it obtains all internal and external voxels. Figure 3 presents this procedure in two dimensions.

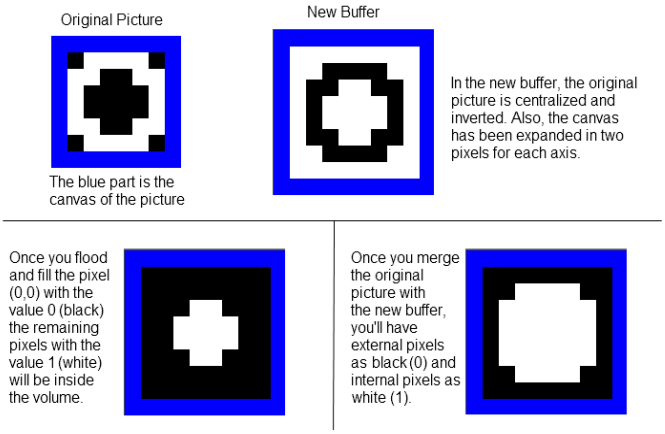


Figure 3. Detection of inner pixels in 2d.

In order to discriminate the internal and surface voxels we check if each voxel has a neighbor that is outside the volume.

From now on, each voxel will be treated as a cube. The list of vertices should contain all vertices corresponding to the voxels that belong to the surface of the model. A new buffer might be required to avoid repeating this process.

In order to build the faces as quads of the cube, it is necessary to detect if each candidate face will be part of the model or not by checking if only one of the voxels where it belongs is part of the surface of the volume. Then, the vertices of each face must be ordered in counterclockwise direction. To find the correct direction, if suffices to know if this is a back or front face of the surface voxel that originated it.

Due to the culling procedure explained in the last paragraph, some vertices will not be used by any face and they must be excluded.

### *4.3 Surface fairing*

The surface fairing step is responsible for smoothing the geometry of a mesh composed of quads using an approach based on signal processing techniques. In this case, the signal is the whole description of the geometry.

As we mentioned before, we do not use an exact low-pass filtering approach, based on the projection of the signal in the space of low frequencies using the Fourier Descriptor technique. Instead, we approximate such low-passing filtering by using a smoothing mechanism based on the displacement of each vertex $v$ in the mesh so that it moves in the direction of its neighbors. In this sense it is similar to Taubin's work but it is essentially different due the way we deal with the frequency characteristics of the data we are interested in.

The coordinates of each vertex $v$ and the coordinates of its neighbors $v_i$ are taken, combined and weighted into new coordinates of $v$. The neighborhood $v_i$ of a vertex $v$ is obtained by scanning all vertices that belongs to the faces adjacent to the $v$. The usage of a mesh composed by a set of quads instead of triangles, for this operation, results in a more accurate neighborhood because we increase the connectivity of the vertices and the level of influence they have on each other.

The weighted differences between each vertex $v$ and its neighbors $v_i$ is given by the discrete laplacian operator whose application to the coordinates of $v$ yields an estimate of the curvature of the signal at $v$. This curvature estimate can also be seen as a measure of the details of the signal.

This can be understood intuitively by considering two point of views: one geometrical and another based on the frequency of a wave.

According to the geometrical interpretation of the operator, the higher frequencies correspond to sharp solid angles, while the lower frequencies correspond to the smoother regions of the surface. If the frequency is zero, the surface is flat at that vertex and the amplitude is always zero.

We can also understand the results produced by the operator as a measure of the frequencies of the signal. The position of a vertex $v$ of the mesh can be understood as the crest of a wave represented by the connection of v with its neighbors $v_i$. The base of this wave is a plane $\pi$ determined by all its neighbors while the amplitude is the distance of $v$ to $\pi$.

In order to compare the frequencies among different vertices, we can scale all the waves to a common amplitude. The smaller the period of the scaled wave the higher is the frequency at that vertex. Another way to understand the estimation of the frequency at any vertex, is to consider that all waves should have a common amplitude value that will be reached in a time $t + t_i$. The faster the wave takes to reach this amplitude, the smaller will be the period and, as a consequence, the higher will be the frequency in that vertex.

Therefore, the laplacian term, which comes from the diffusion equation, can be seen intuitively as an estimate of the frequencies of the vertex.

Let $(x,y,z)$ be the current vertex coordinates, $(x_i,y_i,z_i)$ be the coordinates of its $n$ first order neighbors and $\lambda$ a constant proportional to the diffusivity constant in the diffusion equation. The estimated frequencies $(\omega,\xi,\psi)$ corresponding to the directions in $x$, $y$, and $z$ components in space can be obtained by using the laplacian where the weights are given by the lengths of the edges between $v$ and one of its neighbors, as shown in equation (8):

$$\begin{cases} \omega = \dfrac{\lambda}{n}\displaystyle\sum_{i=0}^{n-1} \dfrac{(x_i - x)}{\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}} \\ \xi = \dfrac{\lambda}{n}\displaystyle\sum_{i=0}^{n-1} \dfrac{(y_i - y)}{\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}} \\ \psi = \dfrac{\lambda}{n}\displaystyle\sum_{i=0}^{n-1} \dfrac{(z_i - z)}{\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}} \end{cases} \quad (8)$$

The constant λ increases or decreases the effect of the mesh processing technique to be used and it has to be determined experimentally. In this work, we have used the value 1.3, which still ensured that the value obtained for every frequency was below 1. If λ is set to 1, the maximum frequency found in a quad based mesh is 0.5.

The use of equation (8) will provide acceptable results if the crest of the wave is detected correctly. We may consider, for the cube-based models we used, the Euclidean distance, in which the distances of each vertex to one of its first order neighbors is either 1 or $\sqrt{2}$. However, to ensure a more precise solid angle measure we recommend to consider all vertices with a fixed distance independent of the length of the edges. This avoids an anisotropic measure of the angles which would depend on the distance and direction from a vertex.

The determination of the displacements of each vertex *v* is done by evaluating a function on the frequencies obtained as show before. Intuitively, this corresponds to obtain the displacement in space as a function of the estimated frequency.

Here our work is essentially different from Taubin's work. We are interested in reconstructing the signal instead of just smoothing it. The effects of the filter are stronger at the regions of the signal corresponding to the higher frequencies which in our case correspond to the jagging effects introduced by the discrete sampling of the solid in the form of a voxelized model. In the regions where we have medium or low frequencies the smoothing process is less intense.

In order to achieve such effect, which can be understood as an adaptive low-pass filtering, we adopt a function based on a modification of the Lanczos kernel as shown in equation (9)

$$L(u) = \begin{cases} 1 - \dfrac{3\sin(\pi u)\sin(\pi u/3)}{\pi^2 u^2}, u > 0 \\ \dfrac{3\sin(\pi u)\sin(\pi u/3)}{\pi^2 u^2} - 1, u < 0 \\ 0, u = 0 \end{cases} \quad (9)$$

Finally, the new position of the vertex is calculated using the equation (10):

$$\begin{cases} x' = x + L(u) \\ y' = y + L(v) \\ z' = z + L(w) \end{cases} \quad (10)$$

Compared to the simple Gaussian Filtering which produces too much blurring, the adapted Lanczos kernel is able to retain the details of the mesh, while it still smooths. Besides, we achieved such results in only one pass, which enabled us to build a interactive application that is able to polygonize any voxelized model.

### *4.4 Attribute smoothing*

This step smooths the surface normal vectors and colors of the model, yielding new surface normal vectors and colors for each vertex. The mesh extraction generates a model without surface normal vectors and with colors per face. At this stage, we convert the existing quads into triangles.

The procedure for smoothing the normal vectors starts with the detection of the normal vectors of a face, which is done by the cross product of its vertices in counterclockwise direction. Then, the surface normal vector of a vertex is calculated as the mean of the surface normal vectors of the faces adjacent to such vertex. This calculation must ignore repetitive directions.

Similarly, the color of a vertex *v* is computed as the mean of the colors of the faces adjacent to it.

### *4.5 Texture atlas extraction*

In order to extract the texture atlas, the model needs to be splitted into a set of smaller partitions. Then, these partitions are projected into small planar blocks, which are merged into only one image. This operation requires the knowledge of both normal vectors per face and per vertex.

The process of generating a partition starts with the creation of a new buffer that will decide the order in which the faces will be scanned.

In this work, we have ordered the faces using the following criterion: the first ones in the list are those that are most aligned with one of the axis x, y, z, picking the highest absolute value of the direction of the surface normal. Once the first face *fc* of the first partition is chosen, the rotation angles $\theta_x$ and $\theta_y$ are obtained using the equation (11) with the coordinates *x*, *y*, *z* of the normal vector from *fc*. The angle $\theta_x$ must return a value between [-π/2, π/2] while $\theta_y$ returns a value between *[0,2π)*:

$$\begin{cases} \theta_y = \dfrac{-x}{|x|}\arccos\left(\dfrac{z}{\sqrt{x^2+z^2}}\right), x \neq 0 \\ \theta_y = 0, x = 0 \\ \theta_x = \dfrac{-y}{|y|}\arccos\left(\sqrt{x^2+z^2}\right), y \neq 0 \\ \theta_x = 0, y = 0 \end{cases} \quad (11)$$

These angles are used to generate a matrix to project the faces of the partition onto a 2-dimensional image plane. The other faces of the partition are added by scanning the neighbors of the face *fc* and verifying if the angle between the normal of the current face with the face *fc* is lower or equal than a pre-determined angle $\theta_{min}$. In this work, $\theta_{min}$ is 60 degrees. If the face is added, its neighbors must be scanned. A new partition is created when no other face can be added to the current partition according to the insertion criterion.

Each vertex must belong to only one partition. If a vertex belongs to faces from different partitions, it must be cloned. Each face also belongs to a single partition. During the generation of the partitions, it is important to collect the local projected positions ($u$,$v$) of each vertex generated with the projection matrix. This data should be used to find out the sizes of each partition.

Once the partitions are set and projected onto rectangles, which will be merged into a big squared block. This step starts with the translation of the origin of the coordinates of all blocks and their respective vertices to position (0,0). We will need at least 3 buffers: the first buffer *Hb* will order the partitions by horizontal size; the second *Vb* will order by vertical size and the third *Btb* will work like a binary tree of partitions. The buffer *Btb* codifies the relative positions among the partitions as they are inserted in the big squared block.

The algorithm picks the last two elements of *Hb* and the last two elements of *Vb* and determines which combination will occupy less space in the texture atlas. The areas of the partitions are given by axis aligned bounding boxes. The winning team is merged into a new and bigger partition, one of the elements is translated to the right/bottom of the other by using the *Btb* and they are eliminated from both *Hb* and *Vb* while the new partition is added to all buffers. This loop ends once there is only one partition left in both *Hb* and *Vb*.

The final partition is converted to a square and all partitions inside it are translated to become centralized vertically or horizontally, depending on the lower dimension. Then, the coordinates from all vertices is found with the sum of its position in its partition with the starting position of the partition where it belongs divided by the maximum dimension of the largest partition.

### *4.6 Surface simplification*

The model generated so far includes several unnecessary vertices and faces, due to the discrete topology that originated it. In order to be useful in video games and real time simulation applications, its geometry must be simplified.

Any method that simplifies the mesh keeping its texture attributes can be used, such as Heckbert et al [Garland and Heckbert, 1998], Hoppe [1999], Lu et al [2007], Zhang et al [Zhang and Wu, 2008], among others.

## 5. RESULTS OBTAINED

This technique was implemented in the Voxel Section Editor III. It is a free open source [VXLSE3SVN, 2010] volumetric image editor with an active fan base that provides several free models [PPM, 2010]. It was originally conceived to edit volumetric models for the games Command & Conquer Tiberian Sun and Command & Conquer Red Alert 2 [Classics, 2010].

Over 20 models were used to test this technique. Most of them were created by Westwood Studios using 3ds max and their own voxel exporter, while others were created with the Voxel Section Editor III. The tested models features up to 64000 used voxels and over 250000 unused ones.

The tests were conducted with at least 4 different platforms. The slowest one was a tablet PC HP TX1120 US, consisting of a dual core 1.8 GHz processor with 4gb RAM, GeForce Go 6150 and Windows Vista. The table 1 shows the time taken for each step using this system, as well as the surface fairing using Taubin's Smooth method as implemented in MeshLab 1.2.3 [MeshLab, 2010].

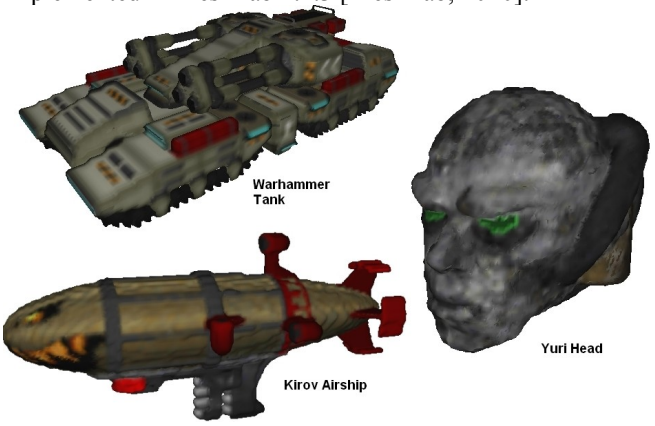


Figure 5. Texturized models generated with this technique.

The table 1 displays the execution time from the mesh extraction, surface fairing, attribute smoothing and texture atlas generation steps. In the end, it also shows the time that 10 iteractions of Taubin's Smooth takes to be executed with these models in MeshLab.

| Execution Time (in ms) | | | | | |
|---|---|---|---|---|---|
| Model Name | Mesh Extraction | Surface Fairing | Attribute Smooth | Texture Atlas | Taubin's Smooth |
| Demo Truck | 35 | 11 | 14 | 316 | 47 |
| Kirov | 397 | 52 | 56 | 1237 | 203 |
| Yuri Head | 327 | 124 | 131 | 5522 | 485 |
| Warhammer | 359 | 186 | 201 | 6393 | 735 |

Table 1. Execution time of each step of the method and Taubin's smooth.

On every model, all operations together takes less than one second to execute, except for the texture atlas extraction. This means that this technique allows the result to be previewed quickly by the user, because the extraction of the texture atlas would be more useful to export the volume as a geometric model once the user has finished its edition.

Voxel Section Editor III is a volume object editor and the only way it opens a polygonal model is by importing into a volume. It also has restriction that it prevents models with sections bigger than 255x255x255 voxels from being opened. This is why this work was tested with low polygon models. The demo truck (Figure 6) has one section with dimensions 21x21x48 voxels, with a total of 21168 voxels, where 7893 are opaque. The Warhammer tank (figure 5) features 429900 voxels in two sections with 66x30x130 and 60x25x115 voxels respectively. Only 63150 of these 429 thousands were painted by the user. The table 2 displays the amount of vertices and triangles generated by this method for each model.

| Model Size | | | | |
|---|---|---|---|---|
| Model Name | Original Volume | | After Texture Extraction | |
| | Voxels | Opaque Voxels | Vertices | Triangles |
| Demo Truck | 21168 | 7893 | 8987 | 13624 |
| Kirov | 554880 | 35841 | 36005 | 52620 |
| Yuri Head | 391680 | 42733 | 98324 | 121468 |
| Warhammer | 429900 | 63150 | 89750 | 185552 |

Table 2. Contents of each model.

Figure 6 shows the result of the conversion of Red Alert 2's demo truck volume into a fully textured 3D model, while figure 7 shows the same model smoothed with 10 iteractions of Taubin's Smooth.


Figure 6. Result obtained with the conversion of a truck.

Figure 7. Result obtained with 10 iteractions of Taubin's smooth..

The desired result of the signal processing operation was to smooth the model keeping as many details as possible. The use of modified Lanczos kernel provided satisfactory results, with the features being only lightly smoothed. Moreover, it produced the expected results in curved areas and also dealt appropriately with topological problems of type 1 (see section 3), as shown in figure 8 and figure 9. The topological problems of type 2 (see section 3) were not solved. Differently from Taubin's work, this result was obtained with a single iteration.

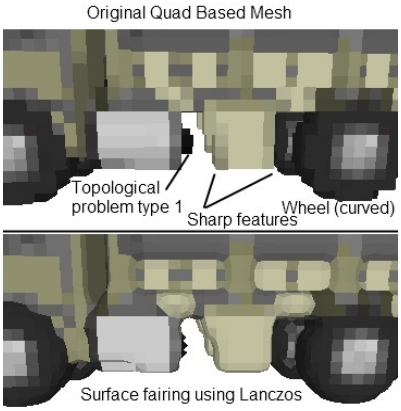


Figure 8. Surface fairing results.

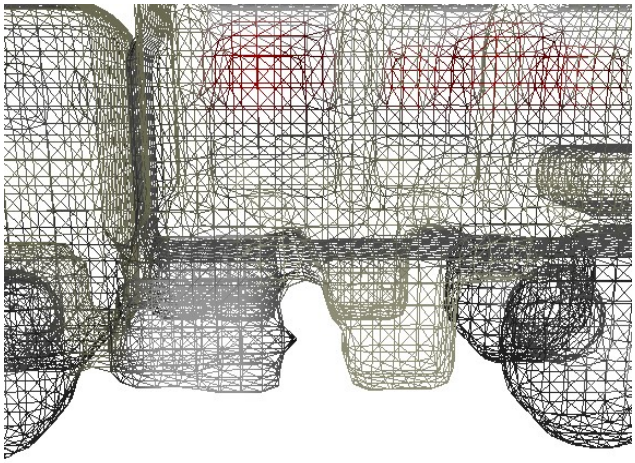

Figure 9. Surface fairing results in wireframe.

The smooth level is controlled with the constant λ. However, if it is raised too much and the frequency returns values above 1, it will increase the noises in the mesh.

The texture map generated from the texture atlas obtained with this work successfully covers the whole model, which is enough for the needs of a game engine. However, it is not intuitive for edition by the final user and it features unnecessary micro partitions that will only confuse whoever decides to modify it. Other methods, as for example those presented in section 3 use more sophisticated approaches for the extraction of the patches belonging to the texture atlases but do not produce results in interactive times as the greedy method we proposed. Cohen's work [COHEN-STEINER, 2004] takes approximately 5 min just to compute an approximated version of a mesh with 400k triangles via a variational method which is just one step in the texture atlases construction framework proposed in [SOSSAI, 2007.]. The texture for the truck is displayed in the figure 10.

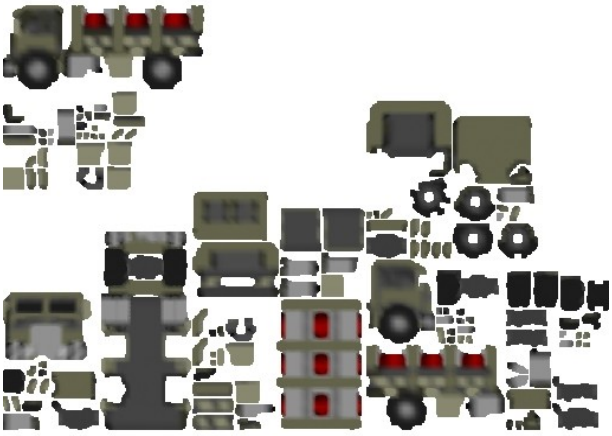

Figure 10. Centralized partitons in the final diffuse texture map generated for the truck.

## 6. Conclusion

This work presented a method for the polygonization and texture extraction from volumetric objects based on a polyhedral surface extraction and surface fairing.

We proposed a novel surface fairing approach that is appropriate to smooth meshes obtained from voxelized models with homogeneous inner material.

The surface fairing approach we proposed is based on an adaptive low-pass filtering based on a function that combines the discrete laplacian operator and a function of a modified version of the Lanczos kernel.

Besides, we also presented an effective way to extract texture atlases in a few seconds differently from previous approaches based on variational methods.

By using our method, we were able to build a complete 3d modeling application that is able to produce mesh-based texturized representations from volumetric data at interactive times.

As future work we intend to pursue the proposal of a solution for the topological problem of type 2. This solution can be obtained by using the subdivision ideas presented in [Bischoff and Kobbelt, 2006]. We will also investigate the extraction of other attribute maps like material maps, normal maps and relief textures.

## 7. Acknowledgment

We are grateful to Faperj for the support of this work.

We are also grateful to the users of Project Perfect Mod for helping with tests and supporting this research.

Command & Conquer, Tiberian Sun, Red Alert 2, Westwood Studios and the voxel format used in this experience are property of Electronic Arts.

## 8. References


*BISCHOFF, S., KOBBELT, L., 2006.* Extracting Consistent and Manifold Interfaces from Multi-valued Volume Data Sets; Bildverarbeitung für die Medizin 2006; Springer Berlin Heidelberg; pp. 281–285.

*CATMULL, E., CLARK, J.., 1978*. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes; Computer Aided Designs; Nov. 1978; vol. 10, No. 6; pp. 350-355.

*CLASSICS, 2010*. Command & Conquer Classic Download Page. The games Command & Conquer Tiberian Sun, Red Alert 1 and Command & Conquer 95 are available for download in t he official site of the publisher, Electronic Arts. http://www. commandandconquer.com/classic

*COHEN-STEINER, D., ALLIEZ, P., DESBRUN, M., 2004*. Vari ational Shape Approximation; ACM Trans. Graph, 23(3):905-914, 2004.

*DUCHON, C. E., 1979*. Lanczos Filtering in One and Two Dimensions; Journal of Applied Meteorology; Jan. 12-Mai 07, 1979; vol. 18, American Meteorological Society; pp. 1016-1022.

*FREITAG, L., PLASSMAM P., 2000*, Local optimization-based simplicial mesh untangling and improvement, Intl. J. Numer. Meth. Engin., 49 (2000), 109--125.

*GARLAND, M., HECKBERT, P., 1998*. Simplifying surfaces with color and texture using quadric error metrics; Visualization ’98 Proceedings. 1998, IEEE, pp. 263–269.

*GIBSON, S. F. F., 1998*. Using distance maps for accurate surface reconstruction in sampled volumes, Volume Visualization Symposium - IEEE, pp. 23-30, 1998.

*HOPPE, H., 1999*. New quadric metric for simplifiying meshes with appearance attributes, Proceedings of the conference on Visualization '99: celebrating ten years, October 1999, San Francisco, California, United States, p.59-66

*JU, T., LOSASSO, F., SCHAEFER S., WARREN, J., 2002*. Dual contouring of hermite data. ACM Transactions on Graphics, vol. 21, No. 3, pp. 339-346.

*KOBBELT, L. P., BOTSCH, M., SCHWANECKE, U., SEIDEL H. P., 2001*. Feature sensitive surface extraction from volume data ; Computer Graphics; Siggraph 2001 Conference Proceedings; Ago. 2001; ACM Siggraph; pp. 57-66.

*LEVOY, M., 1988*. Display of Surfaces from Volume Data; IEEE Computer Graphics and Applications; Mai. 1988; vol. 8, No. 3; pp. 29-37.

*LORENSEN W. E., CLINE H. E., 1987*. Marching Cubes: A High Resolution 3D Surface Construction Algorithm; Computer Graphics; Siggraph '87 Conference Proceedings; Jul. 27-31, 1987; vol. 21, No. 4; ACM Siggraph; pp. 163-169.

*LU W., ZENG, D., PAN J., 2007*. QEM-based mesh simplification with effective feature-preserving; Proceedings of the 2nd international conference on Virtual reality; July 22-27, 2007; Beijing, China

*MESHLAB, 2010*. MeshLab. An open source, portable and extensible system for the processing and editing of unstructured 3D triangle meshes. http://meshlab.sourceforge.net

*PPM*, 2010. Project Perfect Mod Voxels. Forum where members of the Command & Conquer modding commmunity post their volumetric models. http://www.ppmsite.com/forum/index.php?f=111

*SOSSAI, J. Jr, VELHO, L., 2007*. Projective Texture Atlas Construction for 3D Photography; The Visual Computer: International Journal of Computer Graphics; Aug 2007; vol. 23, Springer-Verlag New York, Inc.; pp. 612-629.

*TAUBIN, G., 1995*. A Signal Processing Approach To Fair Surface Design, Computer Graphics; Siggraph '95 Conference Proceedings; Set. 1995; ACM Siggraph; pp. 351-358.

*VXLSE3, 2010*. Voxel Section Editor III. Free open source volume modelling and editing tool. http://www.ppmsite.com/index.php?go=vxlseinfo

*VXLSE3SVN, 2010*. Voxel Section Editor 3 SVN. Source code of Voxel Section Editor III. http://svn.ppmsite.com/listing.php?repname=OS%20Voxel%20Tools&path=/vxlseiii14x/&rev=0&sc=0

*ZHANG, S., WU, E., 2008*. A shape feature based simplification method for deforming meshes; Proceedings of the 5th international conference on Advances in geometric modeling and processing,; April 23-25, 2008; Hangzhou, China

*ZHANG, Y., BAJAJ, C., GUOLIANG, X., 2005*. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In 14th International Meshing Roundtable, 449--468.