Survey on Polygonization of Implicit Surfaces

B. De Araujo¹ and P. Jepp¹ and J. Jorge¹ and B. Wyvill²

¹INESC-ID, Lisbon, Portugal
²Victoria, CA

Abstract

Implicit surfaces are commonly used in image creation, modelling environments, and scientific data visualization. In this paper we survey techniques for fast visualization of implicit surfaces. The main classes of visualization algorithms are identified along with the advantages of each in the context of the different types of implicit surfaces commonly used in Computer Graphics. We focus closely on polygonization methods as they are the most suited to fast visualization. Classification and comparison of existing approaches are presented using criteria extracted from current research. This enables identification of the best strategies from the point of view of a number of specific requirements such as speed, accuracy, quality or stylization.

1. Introduction

Implicit Surfaces are a popular mathematical model used in Computer Graphics. They are used to represent shapes in Modelling, Animation, Scientific Simulation and Visualization [GVJ09]. Implicit representations can be extremely compact, requiring few high-level primitives to describe complex free-form volumes and surfaces. They provide a smooth and compact model requiring few high-level primitives to describe free-form volumes and surfaces. They also present a solution of choice for visualizing scientific and medical data. They are well suited to representing data gathered from 3D scans (e.g. Computed Tomography (CT) and Magnetic Resonance Imaging (MRI)) and digitizing complex models.

Precise and fast visualization of implicit surfaces is a difficult, time consuming process. The most faithful surface reproduction technique, Ray Tracing, is also the most costly. Abstract, stylized techniques are relatively fast but are used for specific purposes such as illustrative visualization. The most general and popular approach to represent implicit surfaces is using a polygonal approximation. The study presented in this paper is primarily concerned with providing fast visualizations. Therefore, we focus more on polygonization techniques but we also include a summary of both stylized methods and ray tracing.

As is common in computer graphics there exists a trade-off between accuracy and speed. The traditional speed versus accuracy balance is compounded in implicit surface polygonization approaches by the desirability of a high quality mesh. The mesh may be required for re-use in other applications. Therefore as well as being an accurate representation of the surface it must also use a good budget of well-shaped triangles. Choosing an appropriate polygonization strategy is therefore dependent on the primary motivation, i.e. speed of interaction, fidelity or mesh quality.

There are many polygonization approaches to choose from. Although methods are generally framed with a primary motivation they also include consideration of other factors. Choosing the most relevant approach is easy to identify using the comparisons presented in this paper. The algorithms are organized from the point of view of those primarily motivated by providing either a fast representation of the surface or an accurate one. In addition we identify approaches that are also concerned with providing well-shaped triangles and faithful representations of the features and topology of the surface.

We start by describing the rendering process of implicit surfaces introducing how implicit surfaces are represented, which attributes can be extracted from their formulation and how they can be sampled and vi-
sualized. Then Section 3 presents the basic approaches highlighting main techniques, discussing the main issues of existing rendering techniques and proposing a taxonomy of existing issues. Section 4 surveys polygonization method motivated by speed issues, followed by Section 5 regarding quality-oriented methods. Finally a discussion of the surveyed polygonization approaches is done related to: topological correctness; sharpness and smoothness fidelity and conversion quality of the resulting polygonal approximation.

2. Rendering Implicit Surfaces

Implicit models are generated in a number of ways: based on scan data, representing either a 3D volume (e.g. MRI and CT) or a 2D surface (e.g. 3D laser scans); polygon meshes or point clouds; or are mathematically defined from a set of functions (e.g. constructed from primitives). Visualising the models involves sampling the space to identify the location of the surface before rendering it.

2.1. Defining the IS

Volume rendering is the visualisation of a set of volumetric data rather than a single iso-surface. Volumetric data is often obtained from scans such as CT and MRI data. The volume is rendered using a transfer function to convolve samples and, due to the large number of samples, is computationally intensive; efficient methods use the GPU [SCCB05]. For a description and survey, with regards to implicit surfaces, see the PhD Thesis of Christian Sigg [Sig06] and a survey of Octree volume rendering methods by Knoll [KWPH06].

Data obtained using 3D scans (or indeed from polygon meshes or point clouds) for 2D surfaces (rather than 3D volumes) can be used to visualise of an object. Piecewise functions capture the local shape of the surface and are blended to create the larger model. If the discrete sample points are dense enough unconnected (non-polygonal) geometry can be used to approximate the surface. This method also benefits greatly from GPU processing for fast visualisation. A survey of point based rendering techniques including implicit surfaces is included in the PhD thesis of Patrick Reuter [Reu03] and also Botsch and Kobbelt [KB04].

Implicit models can also be built from mathematical definitions using compositions of functions [BBB’97]. Since the introduction of the Blobby molecule [Bli82] using a weighted sum of density functions, several radial basis functions have been proposed such as the soft-objects [WMW86], meta-balls [NHK’85], blobby model [Mur91] and generalized distance functions. The following list presents the mathematical formulation of some of these basis functions.

\[ \text{ImplicitFunction} : F(X) = \sum_i w_i f_i(X) \]

\[ \text{BlobbyMolecule} : f(X) = \exp(-ar) \]

where \( r = \text{distance}(X,X_i) \) and \( X_i \) is the location of an atom.

\[ \text{Metaballs} : f(X) = \begin{cases} 1 - 3\left(\frac{r}{R_i}\right)^2, & 0 \leq r \leq \frac{R_i}{3} \\ \frac{2}{3}(1 - \left(\frac{r}{R_i}\right)^2), & \frac{R_i}{3} \leq r \leq R_i \\ 0, & r > R_i \end{cases} \]

\[ \text{BlobbyModel} : f(X) = \begin{cases} \left(1 - \left(\frac{r}{R_i}\right)^2\right)^2, & 0 \leq r < R_i \\ 0, & r \geq R_i \end{cases} \]

\[ \text{SoftObject} : f(X) = \frac{4}{9} \left(\frac{r}{R}\right)^6 + \frac{17}{9} \left(\frac{r}{R}\right)^4 - \frac{22}{9} \left(\frac{r}{R}\right)^2 + 1 \]

By controlling the blending of these radial basis functions, complex objects can be created by incremental modeling such as BlobTrees [WGG98, AGCA06] or using a reconstruction process such as Variational Implicit Surfaces [TO99], FastRBF [CBC’01], Compactly Supported Radial Basis Functions [MYC’01,OB505,TR06] and Multi-Partition of Unity [OBA’03].

2.2. Mathematical basics

Independently of its origin, i.e. based on discrete data or a mathematical model, implicit surfaces are defined by a level-set function \( F(X) : \mathbb{R}^3 \rightarrow \mathbb{R} \) such as the surface is represented by the set of points \( \{X \in \mathbb{R}^3 : F(X) = \text{cst}\} \). Depending of the mathematical model, \( \text{cst} \) represents the iso-value of the surface and most of existing implicit surfaces are defined such that \( \text{cst} = 0 \). The implicit function classifies points in space in relation to the surface such that \( \{X \in \mathbb{R}^3 : F(X) < \text{cst}\} \) and \( \{X \in \mathbb{R}^3 : F(X) > \text{cst}\} \) represent points located inside or outside the object and \( \{X \in \mathbb{R}^3 : F(X) = \text{cst}\} \) the set of points lying on the surface. Using this definition several features can be extracted to gather properties from the surface which are described below when \( \text{cst} = 0 \) but can be generalized for \( \text{cst} \in \mathbb{R} \). The gradient vector \( \vec{G} \) and the normal unit vector \( \vec{N} \) are defined at point \( X \) using the first order partial derivative of \( F \). Applied to all points lying over the surface, i.e, \( X \in \mathbb{R}^3 \) such that \( F(X) = 0 \), they allow to get the normal vector of the surface to that point:

\[ \vec{G} = \nabla F = \left[ \frac{\partial F}{\partial x} \quad \frac{\partial F}{\partial y} \quad \frac{\partial F}{\partial z} \right]^T \]  

and \( \vec{N} = \frac{\vec{G}}{\|\vec{G}\|} \)

The curvature information of the function \( F \) in \( \mathbb{R}^3 \) at point \( X \) is held by the Hessian matrix \( H \), which is
defined by second order partial derivatives of $F$:

$$H = \begin{bmatrix}
\frac{\partial^2 F}{\partial x^2} & \frac{\partial^2 F}{\partial x \partial y} & \frac{\partial^2 F}{\partial x \partial z} \\
\frac{\partial^2 F}{\partial x \partial y} & \frac{\partial^2 F}{\partial y^2} & \frac{\partial^2 F}{\partial y \partial z} \\
\frac{\partial^2 F}{\partial x \partial z} & \frac{\partial^2 F}{\partial y \partial z} & \frac{\partial^2 F}{\partial z^2}
\end{bmatrix}$$

However, the Hessian matrix $H$ represents the curvature evolution of the scalar field $F$, which supports the implicit function, rather than the surface defined by $F = 0$. Indeed, we need to study curvature values and directions on the plane tangent to $F = 0$ at $X$. In order to compute these, we need to use the matrix $C$ defined by the partial derivatives of the normal $\vec{N}$ instead of the Hessian:

$$C = \begin{bmatrix}
\frac{\partial N_x}{\partial x} & \frac{\partial N_y}{\partial y} & \frac{\partial N_z}{\partial z} \\
\frac{\partial N_y}{\partial x} & \frac{\partial N_y}{\partial y} & \frac{\partial N_y}{\partial z} \\
\frac{\partial N_z}{\partial x} & \frac{\partial N_z}{\partial y} & \frac{\partial N_z}{\partial z}
\end{bmatrix}$$

The matrix $C$ can be defined using the Hessian matrix $S$ and the gradient vector $\vec{G}$ as we can see from the following equation:

$$C_{ij} = \frac{H_{ij} + \|\vec{G}\| - \frac{G_i \dot{G}_j}{\|\vec{G}\|^2}}{\|\vec{G}\|^2}$$

where $\dot{G}_j = \frac{\partial \vec{G}}{\partial x_j}$, $H_{ij}$, $H_{ij}$, $H_{ij}$ $\vec{G}$, $\|\vec{G}\|^2$. Since $C$ is defined in terms of the normalized gradient, one of its eigenvalues is zero value and the corresponding eigenvector is normal to the surface (and thus collinear to $\vec{G}$). The other two eigenvalues $k_1$ and $k_2$ are called the principal curvatures and their respective eigenvectors are the principal directions defined to lie in the plane tangent to the surface at $X$. According to [Gra96] and [Koe90], the following curvature measures can be computed from $k_1$ and $k_2$ beyond both maximum and minum curvatures:

- Mean Curvature $\frac{k_1 + k_2}{2}$,
- Gaussian Curvature $k_1 k_2$,
- Deviation from Flatness $\sqrt{k_1^2 + k_2^2}$ and Shape Index $= \frac{\pi}{2} \arctan\left(\frac{k_{max} + k_{min}}{k_{max} - k_{min}}\right)$

2.3. Sampling

Sampling the implicit surface is extremely important. Since the surface is defined by $\{X \in \mathbb{R}^3 : F(X) = c\}$, common root finding methods such as bisection method, secant method, false position method and Newton-Raphson [PFT86] can be used to sample and identify points of the surface depending on the function continuity. Most of existing implicit surface visualization algorithms rely on Newton-Raphson method or interval of analysis. However, finding and illustrating a surface and its features is dependent on the level of sampling. The surface will only be identified where the sampling level is small enough to identify features. Guaranteeing feature identification requires additional information about the surface being rendered, Kalra and Barr discuss sampling for implicit surfaces [KB89]. The Nyquist theorem states that the maximum frequency of the signal must be known in order to define a sampling rate that is greater than or equal to twice this value [BBB’97]. Lipschitz constants also require information about the maximum rate of change of the function [BBB’97]. Therefore, in order to guarantee that all details are identified, the distance between samples should be related to the size of the smallest surface feature. As too many samples will have a considerable time overhead [vOW04] a balance between speed and accuracy is important. Morse theory [Har97] have been used for critical point analysis extracting more topological information of the surface from its implicit definition. There is no general method of guaranteeing that sampling rates are appropriate for all surfaces. Information about the surface is necessary to define an adequate sampling rate. Sampling the surface and visualisation can be carried out in a single step using ray tracing, and particle-based methods. Polygonisation using surface tracking uses a single step whereas methods using spatial decomposition (see next section) generally perform sampling and rendering separately.

2.4. Visualization

Ray tracing implicit surfaces is the most accurate method of visualization. The implicit surface can be sampled directly by identifying the intersection point along a viewing ray. Ray surface intersections are calculated using root finding and methods and spatial data structures, such as octrees, are used for acceleration. It is the most computationally intensive visualisation method taking the longest time to produce results [BBB’97]. A comprehensive survey of ray tracing for implicit surfaces is beyond the scope of this article, see the PhD thesis of Aaron Knoll [Kno09].

Particle based rendering was originally introduced for fast sampling and visualisation of implicit surfaces. More recently it has been used as the foundations for sampling for repolygonization (see Section 3.4) and also for non-photorealistic rendering. For a summary of particle based visualisation approaches for implicit surfaces see the PhD thesis of Pauline Jepp [Jep07].

Polygonization of implicit surfaces implies the conversion from a continuous mathematical description to a discrete piecewise approximation. It is a lossy process because accuracy is dependent on the sampling frequency, e.g. the size of triangles.
3. Basic Approaches

In this section we will briefly describe the fundamental approaches used for sampling and analysing the model space for implicit surfaces. The location of the surface is first identified before being visualised. We will explain spatial decomposition, surface tracking, inflation/shrinkwrap and particle-based approaches.

3.1. Spatial Decomposition

In order to polygonise an implicit surface, the space containing it should be organised in a coherent manner. The space is divided up into semi-disjoint cells such as cubes (voxels: volume elements) or tetrahedra which will enclose the entire object. In general, only the iso-surface is of interest so only cells containing parts of the surface are stored. Stored cells are used for creating a polygonisation. Polygon vertices are calculated from surface intersection points along edges of spatial cells using root finding and convergence, see Section 2.3.

There are three main methods of spatial decomposition: subdivision, continuation and enumeration [BBB97]. We will give a short description of each of these methods in this section. These methods are still used as the foundations for many modern polygonisation algorithms. Indeed, advances in polygonisation have been focused on improving speed or using different shaped decomposition cells.

Subdivision of object space is a recursive process that identifies cells containing the surface [BBB97]. First, a cell containing the entire object is identified, it is as a regular shaped, convex hull or bounding box of the object. This cell is subdivided into equal parts, any of these cells that are identified to contain part of the surface are then recursively subdivided to an agreed level, in this example four times. The result is a collection of cells that contain surface intersections. Lines that approximate the shape are calculated from edge surface intersection points. The higher the number of subdivisions, the greater the level of detail. For three dimensional objects an octree is the most commonly used with the 2D lines illustrated in the figure being replaced by 3D polygons.

Continuation methods use a pre-decomposed spatial organisation and identify a single seed cell that contains part of the surface [BBB97]. From this seed cell the surface is tracked through adjoining cells using shared edges with surface intersections. In order to avoid recalculating edge surface intersections or cell data a storage medium is required. Stacks, queues and hash tables are all commonly used. Once edge intersection points have been found a table is used to polygonize cells. Cell vertices are identified as being "hot" if they are inside the surface and "cold" when outside.

A table contains the configurations for polygons based on cell polarity patterns. Wyvill et al. were the first to publish this method and used several performance improvements [WMW86], including the hash function to only store cells containing parts of the surface, and a smaller look-up table with repetitions and ambiguities removed. The combinations of creating polygons depends on vertices being either hot or cold (inside or outside the surface). When diagonal corners are identified as being inside the surface there is no clear identification of the behaviour of the surface in that cell. One of the simplest ways to gain more information is to sample the field function in the centre of the face [WMW86].

Exhaustive enumeration is related to processing volume scan data, such as CTs and MRI [BBB97]. Each and every cell is examined in turn to determine the surface intersection points. The Marching Cubes algorithm [LC87] is the most famous of this type of method. It examines planar slices in turn until the whole volume has been processed. Marching Cubes uses the same idea of a look-up table presented in Figure 1 for polygons according to cell vertex polarity, although the table did not have the same ambiguity reductions as the method by Wyvill et al. [WMW86]. The term Marching Cubes has become synonymous with the continuation method of Wyvill et al. Cell partitioning or spatial decomposition techniques are the most popular methods for rendering implicit surfaces because they are the fastest.

3.2. Surface Tracking

Alternatively, surface tracking methods do not subdivide the space to create polygons, but query the surface directly. It is a family of techniques that generate polygonal approximations by starting from a point lying on the surface and generating triangles by following the

Figure 1: Marching Cubes look-up table: the 256 different configurations of vertex cell polarity are generalized by 15 cases due to existing rotations and symetries
Marching Triangles was one of the first such approaches and many advances are made from this basic technique.

Marching Triangles was proposed by Hilton et al. [HSIW96, HI97] and generates a 3D triangulation that verifies the Delaunay constraint, i.e. for each triangle of the mesh, the circumscribed sphere passing through each vertex does not contain other vertices. Applying this constraint generates a mesh with a uniform but not regular vertex distribution over the surface. Marching Triangles can be applied starting from either a seed point (or triangle) or from an incomplete mesh (with holes) to be refined. The algorithm generates new vertices by expanding each triangle edge of the mesh generation boundary as depicted by Figure 3 where the edge $ij$ is expanded to create the vertex $new$. This vertex is placed such as the new triangle verified the Delaunay constraint. The process is repeated resulting in the creation of a mesh that covers all the surface.

Inflation and Shrinkwrap

The shrink-wrap algorithm [vOW04] is analogous to the process of using a plastic, mouldable film that is shrunk (using heat) to tightly cover an object; the film is the plastic wrap that is shrunk into shape. The shrink-wrap algorithm starts with an iso-surface that is homeomorphic to a sphere; it is calculated by adding a large offset value to the isosurface value. Each iteration of the algorithm reduces the offset value and then corrects vertices so that they lie on the new isosurface; more closely approximating the desired surface at each step. This process is continued until the offset is reduced to zero. During each iteration new vertices are created to accommodate the surface details. Edges are subdivided and the midpoint is corrected along the direction of the gradient to lie on the surface. Ultimately the shrink-wrap method converges to the surface as the resolution increases.

Inflation is the opposite approach where polygonal components (second image of Figure 4) are created inside the implicit volume and inflated until they create a shape that closely approximates the surface [SH95] as shows Figure 4. Components are created, adapted and joined in relation to critical points: maximum, minimum and saddle points visible on the leftmost image of Figure 4. One of the benefits of the inflation algorithm over the shrink-wrap approach is that internal pockets are identified, although in general these regions are only visible when trimming, clipping or transparent polygons are used.

Particle System

Historically, particle based approaches of viewing implicit surfaces were used to speed up visualisation time. The bottleneck in displaying implicit surfaces (using any approach) is the number of field function evaluations used by the sampling process. Particles should, in theory, query the defining function less often than ray-tracing or using a polygonization approach. More recently, advances in hardware and software, however, have enabled polygonization of intermediate complexity implicit models to be calculated in real time. Particle systems are still used for visualising larger or more complex data sets, or for creating stylized representations.

Witkin and Heckbert (WH) [WH94] used a combination of attraction and repulsion forces to sample, visualise and deform an implicit model. Particles are born at a single point, attracted to the surface and repelled from one another until they spread to cover the object. It can be viewed as an energy minimization to achieve equilibrium in particle placement. The WH model is the basis for most particle systems that visualise implicit surfaces [RRS97, DTG95]. Levet et al. [LHRS05] use a WH
approach to create a point based rendering technique that also takes advantage of GPU processing. The point based approach creates a curvature driven anisotropic sampling. Discrete samples are individually rendered without connectivity information. This gives the surface an appearance similar to a solid mesh but without the topological constraints introduced by the polygonization process. Although the technique is effective and fast, it is only applied to non-complex models. WH approaches can visualise intermediate complexity implicit surfaces in real time. Complex models, however, can take considerable time for particles to spread across the surface. Jeppe et al [JDWS08] created a Multi-Agent System specifically aimed at visualising complex implicit surfaces. The method uses voxel based spatial decomposition that identifies features of a complex surface faster than a WH approach.

Most of the particle based methods [WH94, dFd-MGT92, DTG95] are well designed for adaptive sampling and take into account the smoothness of the surface and the existence of sharp features. Note that particle based approaches are similar to the polygonal vertex relaxation and Laplacian smoothing methods used to improve the quality of a mesh [NK97, OBP01, KS01, O802, PLLdF06, PFG07, vOW04], see Section 5.1.

3.5. Non Photo-Realistic Rendering

Non-Photorealistic Rendering (NPR) aims to present an accurate but partial view of a model or to focus on specific features of a surface. As such, the purpose is not to create a high fidelity visualization. NPR techniques are, however, particularly good at accurately representing feature outlines (e.g. silhouette or discontinuities) due to the fact that the field function is sampled directly. Some of these techniques are extensions of particle systems. However NPR techniques allow abstractions of images and can be used to focus the viewer’s attention on important shape features. Equally, they can be used to simulate illustration styles or media, for example styles common in medical and scientific illustration. Pen and ink style illustrations use feature outlines to convey shape and form. Bremer and Hughes [BH98] present a predictor-corrector method of identifying silhouette outlines and can also create short surface strokes. Foster et al [FJW05] use a similar method extended to calculate discontinuity feature outlines and cover the surface with stipples or short strokes using a WH based particle system. Other interesting surface lines are ridges and ravines used by Belyaev et al [BPK98] to present more feature outlines from implicit surfaces. The result is a more detailed representation of the surface but is at the cost of expensive calculations based on differential geometry and critical point theory. Faster methods [BA05, OBS05] present a good trade-off between computational cost and accuracy. Other useful features to display are mathematical properties of surfaces such as singularities [SH05, RRS97]. A hybrid system using polygonization and NPR methods is ShapeShop [SIW06]. An underlying voxel representation is used to create polygons and to identify locations of feature outlines. Outlines and stipples are subsequently corrected to lie closer to the surface using a gradient defined attractor force similar to the one presented in WH.

3.6. Discussion

As mentioned ray tracing implicit surfaces in real time is not widely achievable. The most viable method of real-time visualization (using commodity hardware) of non-stylized implicit surfaces is using polygon meshes. Polygonal approximations enable us to explore trade-offs between fidelity of representation and interactive performance rates. Compared to the other techniques (see Table 1) polygonal representations are easy to render for fast visualization. They also generate a representation that can readily be used in several other domains of Computer Graphics.

The level of fidelity of polygonal approximations is

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Outline usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantages</td>
<td>Discrete based sampling</td>
</tr>
<tr>
<td></td>
<td>Uniform or random sampling</td>
</tr>
<tr>
<td></td>
<td>Skeleton based sampling</td>
</tr>
<tr>
<td>Issues</td>
<td>Partial visualization</td>
</tr>
<tr>
<td></td>
<td>Hard to capture sharp features</td>
</tr>
<tr>
<td></td>
<td>Number of iterations required to converge</td>
</tr>
<tr>
<td>Table 1: Rendering comparison</td>
<td></td>
</tr>
</tbody>
</table>

© 2010 The Eurographics Association and Blackwell Publishing Ltd.
dependent on a number of factors: it is a balance of speed, accuracy and quality. Some polygonization algorithms are primarily motivated by the fastest method of visualization; others by the fidelity of the representation with respect to features. Others aim at creating a high quality mesh that can easily be used or re-used in other applications.

As current graphics hardware is designed for optimal handling and processing of triangles, real-time visualization is possible for complex models with billions of triangles. Faster triangle handling allows more focus of attention on areas that improve the quality of the mesh and the accuracy of the approximation.

The issues of polygonization methods are discussed in this paper classifying the methods according to their motivations: either quickly finding the surface and its features (Section 4); or creating a better mesh based on surface properties, i.e. motivated by the quality of triangles as a representation of the surface (Section 5).

4. Fastest Methods

As mentioned in Section 3.1, cell partitioning or spatial decomposition techniques are the most popular methods for rendering implicit surfaces in particular Marching Cubes (MC) [LC87] because they are the fastest. Although the main motivation is speed the polygonization methods also need to be a close approximation to the surface and therefore it must also consider topology and feature sensitivity. Bloomenthal [Blo88] presented an improvement of MC using a tetrahedral decomposition of space rather than a cubical one. This approach has since become known as Marching Tetrahedra. It was motivated by the fact that some of the cubical polygonization patterns are ambiguous. Marching Tetrahedra subdivides each cubical cell into 6 tetrahedra. This enables a simpler pattern table since the possible permutations are based on 4 vertices of the tetrahedra rather than 8 for a cube. The implementation of this algorithm was published in Graphic Gems [Blo94] and became the most used algorithm to convert implicit surfaces into polygonal meshes. Ning and Bloomenthal [NB93] discuss cell decomposition algorithms and conclude that tetrahedral decomposition produces a consistent topology. It also, however, uses twice the number of triangles than a single-entry cubical lookup table, so has the potential to be comparatively slower.

The Marching Tetrahedra algorithm was optimized by Triquet et al. [TMC01, TGM03] to allow “near interactive” rates of polygonization. Triquet et al. demonstrate that the Bloomenthal Marching Tetrahedra included redundant calculations and did not include optimization or acceleration steps. The authors use the following speedup techniques similar to those proposed by Wyvill et al [WMW86]:

- voxels vertices are shared, referencing previously calculated values avoids re-evaluation
- pre-calculated surface-intersection points on edges can also be referenced

Redundant implicit function evaluations are avoided marking them with time stamps and storing computed values in function of query point parameter. These techniques resulted in a new pattern table with 6 new patterns compared to the original Marching Tetrahedra table with 20. It also reduced the polygonization time by a factor of 6. This work used compactly defined basis functions similar to meta-balls and soft objects. Therefore most of the speedup was achieved by taking advantage of the field function structure and could not be applied to all implicit representations.

A similar approach taken by Cuno et al. [CEOC04] used a hierarchical adaptation of the Marching Tetrahedral to polygonize variational implicit surfaces. The method aimed at minimizing the number of implicit evaluations and quickly pruned the space. This is achieved, however, by simplifying Variational Implicit Surface [TO99] models. The pseudo-Euclidean distance computation was changed to be based on the location of the normal and boundary constraints of the VIS model.

Zhang et al. [ZWW06] proposed simple speedup techniques (similar to those used in [WMW86]) that cuts half the meshing time of Marching Cubes or Marching Tetrahedra. The algorithm simplifies the edge-surface intersection computation. This is achieved by using an estimation value based on the interpolation of vertex values instead of the convergence mechanism such as the Newton-Raphson method (which uses the implicit value and its gradient). This algorithm presents simple and effective speedups, however, at the cost of several assumptions that do not guarantee the resulting mesh correctly approximates the surface. The algorithm is only targeted to implicit surfaces that estimate a distance field. The time reduction is achieved by avoiding calculation of empty cells. This method is more suited to achieving a fast coarse representation of implicit surfaces.

Finally, regarding speed, with the increase of ability of GPUs, implicit surfaces can be easily represented using graphical shader capabilities. In the method presented by Kipfer and Westermann [KW05] performance gain is achieved using two approaches. The first avoids redundant computations of edge surface intersections. The second uses features of the GPU to reformulate the iso-surface identification and reduces numerical computations and memory access operations. A span-space data structure is also used to avoid processing non surface intersecting elements. Kanai et al. [KOKK06] present a fast method to directly render sparse low degree implicit surfaces. The approach is
based on ray-casting with ray intersections and blending operations performed in the fragment program on a GPU. The Accelerated Marching Cubes method presented by Johansson and Carr [JC06] uses graphics hardware to achieve improvements of as much as 1300%. The method pre-computes topology for each cell and stores it in a span space structure to avoid redundant CPU computations at runtime. It also improves iso-surface acceleration by caching the topology on the GPU.

4.1. Topological Guarantees

One of the main problems of MC-style approaches is the topological inconsistencies that may arise from the cubical cell classification and its corresponding triangulation, see Figure 5. This point is critical particularly when dealing with iso-surfaces extracted from Medical Scan data. The problem arises from the use of tri-linear piecewise interpolation of the scalar data values at cubical vertices rather than actual field values.

Figure 5: Polygonization mesh result using Marching Cubes and Marching Tetrahedra for the Happy Buddha Model: MC with 9320 triangles, MC with 40740, MT with 28164, MT with 122516, MT with 508668

In order to improve the topological correctness of the polygonal approximation produced by the MC algorithm, several approaches [MSS94b, MSS94a, Che95, HSSZ97, LR03] present alternative lookup tables for cubical cell classification, and also for tetrahedral subdivision [Blo94]. Other solutions have proposed additional correction steps to solve topological ambiguities which may arise between adjacent cells [Mat94] or the existence of internal saddle points inside cells [Nat94]. Nielson [Nie03] proposes an alternative method which guarantees the topological correctness of iso-surfaces at the cost of a more complex lookup mechanism. This mechanism is based on a three level classification i.e. edges, faces and cell interiors. More recently, a method based on iso-surface critical point analysis was proposed by Renbo et al. [RWY05]. Triangles for each cell are created by classification of the nature of critical points, rather than using look-up tables.

Non-manifold surfaces create problems for polygonization strategies. Bloomenthal [BF95] proposed a strategy for polygonizing non-manifold implicit surfaces that used a new tetrahedron classification, as shown in Figure 6. Yamazaki et al. [YKI02] also proposed an approach for non-manifold implicit surfaces by extending the Marching Cubes algorithm to correctly handle discontinuous fields. Features such as holes, boundaries and intersections are dealt with by enhancing the distance field using bounding volumes to simplify the calculation between points. Although features are correctly approximated the mesh quality depends on a user-defined subdivision level, if the cell size is small enough the triangulation is good and identifies sharp features, otherwise the mesh is of poor quality.

Plantinga and Vegter [PV04] presents an extension to MC using an octree based space partitioning. Triangle sizes are adapted to the topology of the surface. Global properties of the implicit function are determined from interval arithmetic and are used as criteria for octree subdivision. Plantinga and Vegter [PV07] also present an isotopy for octree-based meshing and extend their approach to apply tetrahedra to Marching Cubes cells. The results, however, are limited to simple algebraic functions or meta-balls. The output mesh does not present a smooth transition of triangle sizes and reveals jagged triangle edges from adaptive subdivision. The method for creating tetrahedra reduces this problem but does not completely solve it. A similar approach was followed by Alberti et al. [ACM05] to triangulate implicit algebraic surfaces. Instead of using Marching Cubes they proposed a subdivision algorithm using a method similar to interval analysis. The method isolates singularities and guarantees the topology in smooth areas. The algorithm results in a topological approximation that produces similar meshes to MC. The topology is guaranteed up to a given threshold due to the spatial partition being defined to identify singularities.

Inflation methods presented in Section 3.3 also guarantee topological correctness of the polygonal approximation [SH97] thanks to critical point analysis. The topology is guaranteed by tracking the critical points of any $C^2$ continuous implicit surface. These points are found using interval analysis, starting from an initial bounding box which is subdivided as an octree data structure using a zero Newton-based search. They can be classified according to the eigenvalues of the Hessian matrix, which is extracted from the implicit function.
been studied for more than two decades. Prior meth-
consisting of millions of points. However, computing a topo-
or blends of basis functions and have been applied to datasets
clouds. These surfaces are defined as weighted combinations
struct topologically accurate continuous surfaces from point
features approximation and right) spatial subdivision versus
Figure 7: (left) Dual Marching Cubes [SW04] result for sharp
features approximation and right) spatial subdivision versus
Varadhan [VKZM06] visibility maps for an MPU model

4.2. Feature Sensitive Techniques
One of the main concerns regarding MC-style algo-
rithms is their poor ability to correctly approximate sharp features and other discontinuities, including holes. Most of the artifacts are related to the discrete sampling approach followed by the cubical cell subdivision of the space. Several approaches try to over-
come this limitation by improving approximations to the implicit function value. Both the Extended Marching Cubes [KBSS01] and the Dual Contouring [JLSW02] use the spatial partitioning to get a better estimation of the field value, and also refine cell pattern classi-
fications according to the detection of sharp features. This approach was improved by Azernikov and Fischer [AF05] who proposed an adaptive meshing of implicit surfaces by combining an anisotropic grid for sampling sharp features with Dual Contouring.

The main problems of the previous works are: they are only able to approximate at most one sharp feature per cell; and also thin sharp features are discarded if they are located over an edge cell with no sign change. Varadhan et al. [VKKM03] present an alter-
native Extended Dual Contouring algorithm which mixes both approaches to generate an accurate polygonization from volumetric data. This new algorithm presents a more robust intersection test and applies adaptive subdivision techniques to get a better approximation of sharp features. The authors also used additional criteria for the adaptive octree based sampling to guarantee one intersection per cell edge and identify any thin features [VKSM04].

Schaefer and Warren [SW04] proposed an alternative solution by introducing the Dual Marching Cubes algo-

This classification allows identification of the type of the critical point i.e. a maximum, minimum or a saddle point and also the sign of the change. The same extension was done regarding Shrinkwrap algorithms as demonstrated by Bottino et al. [BN096] extension to support implicit surfaces with arbitrary topological genus. This is done by adjusting the topological structure of the mesh with critical points.

Ho et. al. [HWC'05] propose a different solution named Cubical Marching Squares to solve together sharp features, topological inconsistency and inter-cell dependency when using traditional MC. They propose to analyse each cell by unfolding it as six marching squares. Then they look for ambiguous scenarios on each edge of each square. If an edge ambiguity is de-
tected or it contains a complicated surface (heuristically defined analysing sample normals), the cell needs to be subdivided such as an octree structure. The process is recursively repeated until no ambiguities are detected or a maximal level of subdivision is achieved. After the subdivision step, resulting cell faces are processed generating line segments based on a lookup table for marching squares. With the help of sample normals, ambiguous scenarios are solved by checking sharp fea-
tures. Finally the surface is extracted and the final mesh is generated connecting the segments of each face per cell. The achieved results propose an adaptively refined mesh and present less average geometric error comparing the mesh to the input data than both Extended Marching Cubes and Dual Contouring.
5. Quality-Oriented Methods

The algorithms described in Section 4 are motivated by quickly finding the surface and its features. In this section we discuss the approaches primarily motivated by creating better polygon meshes.

Piecewise representations such as MC style algorithms create undesired patterns of triangles that are related to the spatial subdivision. Triangles are created within voxels boundaries so these patterns are evident in the final mesh. Creating meshes without these artifacts requires techniques that focus on mesh quality and regularization. Quality is of particular concern when the mesh is to be used for other applications such as simulation, modelling and deformation.

Mesh quality can be viewed in two ways: creating either regular or adaptive meshes. Regular meshes should have triangles of similar sizes and where vertices have similar valences. Adaptive meshes use triangles that can be adapted in size and density to reflect properties of the surface, i.e. small triangles in areas of high curvature and larger triangles in more flat regions.

5.1. Regular Meshing

Regular meshes aim to create a set of evenly-sized polygons where vertices have equal valence. Regularization aims to remove the unwanted artifacts associated with spatial subdivision techniques. Regular meshes are also very easy to use in other applications such as subdivision, texture mapping, etc.

Surface tracking approaches as depicted in Figure 8 can achieve good results by generating meshes of evenly-sized and quasi-equilateral triangles.

Hartmann [Har98] proposes a similar surface tracking algorithm to Marching Triangles [HSIW96, HB97] (Section 3.2) where points located at mesh boundaries are organized into fronts that are expanded to cover the surface. Starting from a seed point the first expansion creates a new front formed by boundary vertices. Triangles are created and appended relative to topological characteristics. Fronts need to be split or merged with others to avoid overlap. This requires collision tests, which can be costly. More than one triangle can, however, be generated by each point expansion compared to the single edge expansion used by Marching Triangles.

Cermak and Skala propose another variant of Marching Triangles named Edge Spinning [CS02]. Expansion is applied using an oriented rotation around one of the edge extremities. Each new triangle is tested against existing active edges to avoid mesh overlap. Existing points are reused to create a new triangle after finding the closest point lying in an active edge (boundary). Isosceles triangles are created using a constant radius for edge expansion. The projection of the new edge extremity on the surface is done by binary subdivision. The convergence is slower for binary search even though it does not require gradient computation, as with a Newton Step. The Edge Spinning algorithm presents more almost-equilateral triangles than Marching Triangles. The expansion usually creates only one new triangle, however some scenarios can create two using adjacent points or three using the nearest colliding and adjacent points.

The meshes produced by Marching Tetrahedra (Section 4, [Bio88]) approaches were of poor quality. Chan [CP98] proposed a simpler subdivision for the tetrahedra to produce more regular triangles. The ratio of longer to shorter edges was reduced and a smaller number of resulting triangles was observed. There is a marked improvement in the quality of the mesh as the triangulation is more smooth and without a subdivision pattern. For scan data, however, the method may not be compatible as it requires a sampling inside the voxel, which is not available.

The requirement for an internal voxel sample was partially eliminated by Treece et al. [TPG99], see Figure 9. A method is presented for regularized Marching Tetrahedra applied to volume data extraction. This approach combines Marching Tetrahedra for iso-surface extraction and vertex clustering to produce the regularized triangle set. Furthermore, by applying this additional step the final triangle size of the mesh is reduced from 26 to 30 percent. Liu et al. [LYF05] also presents a regular meshing method for algebraic and skeleton based implicit surfaces designed to handle dynamic implicit descriptions, see Figure 9. They use a particle system similar to Witkin and Heckbert [WH94] (Section 3.4) with repulsion and fission mechanisms to produce a uniform sampling of the surface. Then a ball-pivoting algorithm is used: starting from a triangle created using three initial points, travel along the surface by pivoting from sample to sample. This process creates new
triangles and finishes when all the samples are visited producing a regular mesh.

Recent review of traditional Marching Cubes have produced more regular meshes and avoid small and degenerate triangles. Dietrich [DSS’99] proposes the Marching Cubes using Edge Transformations (Macet) algorithm by introducing a new stage between the detection of active edges and the intersection calculation. This stage transforms the end points (cell extremities) of the active edge combining a gradient and a tangential transformation. The algorithm chooses the best of the two transformations using an iterative process. Finally, a different intersection method is proposed based on bissection since the assumption of alignment of the edges with the grid is no longer valid. This algorithm has been extended with the notion of Edge groups in [DSC’09] producing less skinny triangles than the original Macet algorithm. This is done using a probabilistic analysis which identifies MC configurations that generate skinny triangles. In such scenarios, an extra vertex is added to the cell for triangle generation.

A different approach is used by Raman [RW08] called SnapMC. The scalar field value of intersection points close to grid corners are are changed to snap these points onto the grid vertex. Then the traditional marching cubes approach is applied on the new scalar grid using an extended lookup table. Finally, the snapped vertices are moved back to one of the original iso-surface intersections. Although this algorithm is two times slower than the original MC, it successfully reduces the number of triangles by 20 to 40 percent. The method is faster and more robust than the Dellso [DL07] and Afront [SS06] algorithms, which are presented on the following sections 5.2.2. It does not, however, generate an adaptive mesh.

Several implicit surface meshing algorithms are based on Constrained Delaunay Triangulation. Chew [Che’93] presents a technique to create high-quality triangular meshes for curved surfaces by generalizing Delaunay triangulation. A high-quality mesh refers to triangles that have angles between 30 and 120 degrees and for which the triangle density can be controlled according to the curvature of the surface. Starting from a crude triangulation the Delaunay property is checked for all triangles. The mesh is updated by subdivision steps or edge-flipping until the desired quality is reached.

Regular approaches are not adapted to local surface properties. Large numbers of small triangles are required to approximate surfaces with a large variations in curvature. This method is, therefore, more sensitive to numerical instability than space decomposition methods (depending on the complexity of the implicit surface).

5.2. Adaptive Meshing

Meshes are described as adaptive when triangles are created relative to characteristics of the surface. Size and density of polygons are related to surface properties such as curvature or features. In adaptive polygonization methods there is a larger number of triangles in regions of high curvature or in the presence of sharp features. Conversely, in more flat regions triangles are larger and there are less of them.

This section describes several algorithms that produce adaptive meshes. First, we describe approaches that are based on adapting existing meshes then we discuss approaches that are designed to create new adaptive polygonizations directly from the surface.

5.2.1. Remeshing

Remeshing approaches create new triangles from a previously defined mesh. This is generally motivated by either speed considerations or availability of data e.g. medical scans.

Of the remeshing algorithms there are two main types: those that do not make further calls to a defining function, which are designed to be faster and can be used where no further information is available, e.g. scan data; and also those that must make additional evaluations, e.g. with defining functions.

Without further calls to implicit function

Several methods perform global modification of an existing mesh. Feature sensitive re-meshing techniques [VRK01, WHDS02, AFRS03] can be applied to meshes generated using MC-style approaches without using any further implicit surface information.

Vorsatz et al. [VRK01] present several triangle operators that are applied to improve the mesh. This approach was also used by Kobelt et al. [KB03] to generalize sensitive re-meshing techniques as a reverse engineering tool and complement previous work i.e. Extended Marching Cubes [KBS01].

Re-meshing can be restricted to specific areas, for example around sharp features. Attene et al. [AFRS03]...
present a re-meshing method sensitive to sharp features based on a simple filter named an Edge-Sharpener. The filter starts by classifying mesh edges, vertices and triangles. Smooth areas are clustered to identify the triangles located near or at sharp features. Triangles at sharp features are subdivided and their vertices are optimized using a process similar to Ohtake and Belyaev’s normal error minimization [OB02]. (Ohtake’s method works correctly over regular or almost regular meshes so it cannot be used over adaptive triangulations. Also it is dependent on the initial mesh sampling correctly identifying all sharp features).

Spatial decomposition techniques can also introduce non-existent topological features, such as unwanted handles, in the polygonal mesh. This problem can be solved using the technique from Wood et al. for remeshing [WHDS97]. First, handles are found using a Reeb graph representation. Then the significance of handles is evaluated and erroneous ones are identified and removed. Reducing and removing unnecessary topological features allows a better approximation for other mesh processing techniques, in particular mesh compression algorithms.

Requiring further calls to implicit function

Another approach to generate adaptive meshes of implicit surfaces is to improve the polygonal approximation at the cost of additional field evaluations. These methods use the result of an existing polygonization algorithm (most of the time using Marching Tetrahedra). Several steps are performed on the mesh to improve its quality, to be more sensitive to sharp features, to become adaptive or topologically correct.

Rosch et al. [RRS97] present a post remeshing technique based on WH style particle [dFIMGT92, WH94] energy minimisation where repulsion and attraction forces are applied to mesh vertices to optimize their placement and therefore the quality of the mesh. Particles are modelled as a surface constrained mass-spring system. Surface constraints include curvature and singularities.

Neugebauer and Klein [NK97] combine a subdivision based method with the Discrete Marching Cubes from Montani et al [MSS94a] which uses an MC alternative look-up table [MSS94b] generating less triangles and solving some ambiguous configurations. The original coarse mesh was subdivided to produce an adaptive triangulation that better approximates the implicit surface. The coarse mesh is improved by shifting the vertices into the centre of gravity of surrounding polygons. Then, vertices located on a coplanar surface or along almost collinear edges are removed. Finally, a fixed number of subdivision iterations is applied to mesh triangles, as shown in Figure 10.

Figure 10: Adaptive meshing using subdivision over Marching Cubes triangles [NK97]

Ohtake et al. [OBP01] used a combination of techniques to produce good approximations of sharp features (illustrated in Figure 11). The optimization is made on mesh vertices combining the usage of three forces similar to a WH particle system. Two forces are used to optimize vertices using the implicit function value and its gradient. A third is applied to improve mesh regularity using a Laplacian smoothing operator. The resulting mesh better approximates sharp features and is of higher quality regarding regularity at the cost of several iterations of the local operator.

Figure 11: Ohtake remeshing technique over Marching Cubes: (left) [OBP01], right) [OB02]

Ohtake et al. also proposed a different approach [OB02] based on the Dual Primal mesh concept, which consists of two steps. The first creates the dual mesh based on vertices created from triangle centroids that are projecting over the surface. This is in contrast to directly using the vertices obtained by Marching Cubes as was done by the Neugebauer and Klein approach [NK97]. Then, the second step optimizes the modified mesh vertices by minimizing a quadratic energy function using a Garland-Heckbert error metric. During these two steps curvature-weighted vertex re-sampling (similar to the Laplacian smoothing) and adaptive mesh subdivision procedures are performed, depending of the normal deviation. Although this method produces good results it requires a fine initial mesh to retrieve all shape measures and results in many calls to the implicit function during the post re-meshing process. A similar approach is followed by Peiro et al. [PPG07] that also mixes Laplacian smoothing with local modifications such as side swapping. It also includes an optimization step using a curvature estimation of the surface for triangle side collapsing.
5.2.2. Meshing

Many adaptive techniques do not need an initial polygonal representation. They examine the definition of the surface directly to create the mesh. The traditional approaches of spatial decomposition and surface tracking are both used to create adaptive meshes. The algorithms, however, are adapted to have more consideration of surface properties and create triangles accordingly.

Spatial Decomposition

Velho et al. [VdFG99] present a unified and general tessellation algorithm for parametric and implicit surfaces. It generates an adaptive mesh using controlled subdivision. The algorithm starts by creating a simplified uniform spatial decomposition creating a coarse triangulation. Then a refinement step is performed sampling the edges and subdividing the triangulation to better approximate the shape.

Galin and Akkouche [GA00] propose a method leading to an adaptive mesh for visualization of skeletal based implicit surfaces for modelling operations. The algorithm starts by creating an octree using a subdivision criteria based on the Lipschitz condition [KB89]. The Lipschitz property allows culling of empty cells and identification of cells that needed to be subdivided. Cell polygonization is performed using a lookup table similar to Marching Tetrahedra [Blo94]. Adjustments are proposed to correctly deal with ambiguities and adaptive cell sizes. When ambiguities are detected cell subdivisions are required to avoid cracks on the final mesh.

Paiva et al. [PLLdF06] present an algorithm to generate an adaptive mesh that captures the exact topology of the implicit surface. The algorithm starts by building an octree using three subdivision criteria based on the interval analysis of the implicit value and its gradient. The first criterion discards empty cells using interval arithmetic. The second uses the gradient value to identify topological features. Finally, the third criterion estimates the curvature from the interval analysis of the gradient. This method is similar to the approach of Azemikov and Fischer [AF05] (Section 4.2) that creates an octree. The mesh is then generated using an enhanced version of Dual Marching Cubes [SW04] (Section 4.2), which uses a Lewiner et al. [LLVT03] Marching Cubes implementation. Finally, mesh vertices are shifted by vertex relaxation using the tangent plane defined by the normal and the barycenter of neighbouring points. This method presents adaptive meshes with guaranteed topology. It does not, however, correctly approximate singularities as it requires $C^2$ continuity.

Bouthors and Nesme [BN07] present an adaptive meshing method for dynamic implicit models also relying on dual representation. The first is a mechanical mesh which is a coarse mesh approximation generated by Marching Cubes that is optimized using a particle system. Optimization creates a regular sampling of the surface and better approximates sharp features using the method from Ohtake et al. [OB02]. Finally the visualization is offered through a second mesh named the geometric mesh. This mesh is obtained by applying successive subdivision steps on the mechanical mesh. The subdivision criteria uses the normal vector and implicit gradient analysis. This approach presents smooth and adaptive mesh results as depicted by Figure 12, however only implicit models with few primitives (i.e. meta-balls, convolution, skeleton based models) can be supported. Major topological changes of the implicit function cannot be approximated by this method for example disjoint elements.

Gois et al. [GPJE08] use a Partition of Unity Implicits method to produce an implicit surface representation. This method produces an adaptive triangulation for complex topological models and correctly identifies sharp features. This method couples the polygonization process with the implicit representation thanks to an adaptive structure named $f^1$ triangulation. The $f^2$ triangulation allows the association of spatial decomposition with an adaptive surface extraction algorithm. The algorithm starts by subdividing the implicit function domain using the $f^3$ triangulation. For each $f^3$ block a local approximation is generated and a recursive subdivision of the domain is performed. Sharp features are detected using the same classification as Kobelt [KBSS01] and approximated using a method similar to Ohtake’s MPU [OBA03]. The triangulation is obtained from the $f^3$ definition and a mesh enhancement is applied displacing vertices. This method generates an adaptive representation and correct sharp feature approximation. The triangulation, however, is of poor quality when compared to other similar spatial decomposition techniques, due to the nature of the $f^3$ triangulation.

Most of the particle based methods (described in Section 3.4 [WH94, dFdMGTV92, DTC95]) are well designed for adaptive sampling and take into account the smoothness of the surface and the existence of sharp features. Note that particle based approaches are sim-
ilar to the vertex relaxation and Laplacian smoothing methods used to improve the quality of a triangulation [NK97, OBP01, KS01, OB02, PLLdF06, PFC'07].

Tetrahedral cell decomposition was used to achieve an adaptive mesh by Hall and Warren [HW90]. Algebraic implicit surfaces are polygonized using a recursive tetrahedron-based adaptive spatial subdivision method. This approach was extended by Hui and Jiang [HJ99] to present an adaptive marching tetrahedral algorithm for bounded implicit patches. The patch is initially enclosed by a tetrahedron which is subdivided according to vertex value classifications. Ambiguous tetrahedron/surface intersections are further subdivided, which results in an adaptive polygonization. Crespin [Cre02] also proposes an algorithm based on tetrahedra. This method, however, generates a dynamic triangulation for variational implicit surfaces [TO99] using incremental Delaunay Tetrahedralization. An extended bounding box is created using the constraint points of the implicit model, and it is subdivided into tetrahedra instead of cubical cells. A refinement criterion based on the tetrahedron’s circumscribing sphere is used to subdivide the tetrahedron, which are then triangulated in a method similar to Bloomenthal’s approach.

These methods are able to produce adaptive meshes thanks to non uniform spatial subdivision. The method presented by Paiva et al. [PLLdF06] is the only one, however, that uses local implicit characteristics such as curvature as criteria for the subdivision. This is also partially true regarding the method used by Gois et al. [GPJE'08] since it generates its own implicit surface representation. The resulting mesh produces ill-shaped triangles due to the nature of the $F_1^3$ triangulation. The other methods try to identify the correct topology of the surface by applying subdivision for a more robust surface/cell intersection classification.

Surface tracking

Surface tracking algorithms such as the Marching Triangles from Hilton et al. [HI97] and Hartman’s [Har98] approach present more controlled and regular meshes than spatial decomposition methods. This is because the samples evaluate and follow the surface rather than examining the space surrounding the object. Regularity is, however, a problem when the edge step is not small enough to correctly approximate a high curvature surface area. On the other hand, on a low curvature area it may produce too many triangles. The following approaches extended surface tracking methods to produce an adaptive mesh that better approximates the implicit surface.

Akkouche and Galin [AG01] present an extended version of Hilton and Illingworth’s Marching Triangles [HI97]. It produces an adaptive mesh and supports local re-polygonization when minor changes are applied to the implicit function. The algorithm is divided into two phases. First is the growing phase that adds validated, non-intersecting triangles to the mesh when expanding an edge i.e. there is no triangle in the circumscribing sphere with the same orientation. Then the second step closes all the cracks that remain from the growing stage. The adaptive mesh is achieved by constraining triangle edge lengths to local surface characteristics using both a mid-point projection heuristic and Delaunay triangulation properties. This method does not, however, correctly approximate sharp features.

Karkanis and Stewart [KS01], illustrated by Figure 13, present a similar method to Akkouche and Galin. They use a local curvature measure to define the edge length of the expansion rather than edge midpoint projection over the surface. The radius of curvature is the local measure that is computed using the minimum geodesic lying in the normal plane. The radius is computed using the angle between the surface normal and geodesic normal. In contrast to Akkouche’s method, gap closing can create additional points to produce a smoother edge length transition. It uses vertex relaxation, convex filling, edge flip and subdivision.

Figure 13: Adaptive meshes using surface tracking: right) Karkanis using geodesic based curvature estimation [KS01], left) Cermak adaptive edge-spinning [CS04]

Even with the use of a more reliable curvature estimator, both Akkouche and Karkanis’ methods presented above are not able to correctly support sharp features. McCormick et al. [MF02] present an improvement to Marching Triangles that is more sensitive to sharp features, such as edges and corners. This is done by constraining the Marching Triangle approach with the detection of $C^1$ discontinuities. The detection of features creates a set of seed points for the algorithm to create correct approximations using line fitting for edges.

Araujo and Jorge [dAJ05a] present a surface tracking algorithm to generate adaptive triangulation of smooth variational implicit surfaces. A front based propagation approach is used, based on Hartmann [Har98], instead of Marching Triangles. In a single step this algorithm generates a mesh adapted to the local curvature of the implicit surface. Starting from a seed point, the front
propagation generates new points updating the border of the triangulation. New points are created at a given distance from the front related with an heuristic based on the curvature of the implicit surface. The curvature is extracted from the Hessian matrix, which is computed with the second order partial derivative of the implicit function. Compared to the Karkanis [KS01] curvature estimator using the minimum geodesic, this solution provides a more reliable curvature estimator with less computational cost. On the other hand, the triangle density of the mesh can be controlled using the curvature heuristic scalar. However, it does not correctly reproduce sharp features as a fixed thresholded step value is used to create the new points when the curvature cannot be extracted. This method was adapted to support MPU and other algebraic implicit models in [DAJ05b]. However instead of relying on front propagation, this algorithm is based on point expansion. By doing so, it avoids the cost of managing fronts that can be merged or split, such as in Hartmann’s original algorithm. This approach takes advantage of the MPU octree to propose a faster method of collision detection during mesh generation.

Cermak and Skala [CS04] also present an adaptive extension of the edge-spinning algorithm (right image of Figure 13). The logic of the algorithm remains the same as in the original version. The radius, however, is used to place the new vertex when the edge spin is variable. The radius is computed using a curvature estimation based on normal vectors. A root finding adjustment is performed to better approximate sharp features computing the intersection of three planes: two from the existing tangent and the other from the circle. The accurate location of the sharp point is made by binary subdivision. The resulting mesh, depicted in Figure 13, has a better approximation of sharp features than other surface tracking methods. The improvements also decrease the number of triangles with poor aspect ratio. Few results, however, are presented by the authors and the method’s reliability with more complex implicit surface models is not illustrated. This algorithm has been extended in [CS07] to polygonize disjointed implicit surfaces. This is done by sampling the surface using a regular grid with a cell size proportional to the smallest object.

Using the same sampling approach, Cheng et al. [CDRR04] propose an adaptive method relying on Delaunay Triangulation to polygonize smooth and compact surfaces without boundaries. First, they start by sampling the surface to define a set of seed points for the algorithm. Points are added to the Delaunay Triangulation using a geometric sampling. The quality of the triangulation is improved using Chew’s approach [Che93] (Section 5.1). Finally the adaptive mesh is obtained by smoothing the triangulation according to a threshold. The threshold uses vertex dihedral angles to represent a local estimation of the curvature.

Following the ideas of Cheng et al. [CDRR04] Dey and Levine [DL07] proposed an adaptive meshing method for iso-surfaces. Starting from an initial 3D Delaunay Triangulation they recover a “rough” version of the surface. Delaunay based criteria are applied at each vertex insertion and poles estimate the scale of local features. A refinement process is then performed over the mesh from the Delaunay Triangulation. This method does not require that Delaunay is applied at each insertion only the previous mesh and pole values are used to produce the final adaptive mesh.

Using the approach of Chew [Che93] (Section 5.1) Boissonat and Oudot [BO05] proposed an adaptive meshing algorithm for sampled surfaces. It can be applied to implicit surfaces that are $C^2$ continuous and for which gradients do not vanish. First, they search for the set of points on the surface that have horizontal tangent planes, i.e. the critical points of the function with respect to the height function. These critical points are identified using the generalized normal form modulo if the implicit surface is polynomial, or using interval analysis if the function is not. Following a surface tracking approach, triangles are added by finding a point with the smallest radius of curvature. At each step the Delaunay property is certified over the incremental mesh. This method presents a topologically correct adaptive meshing. However, no global timing are presented in the article, only comparative relationships with the Delaunay triangulation. It also does not correctly approximate sharp features.

A similar approach is presented by Schreiner et al. [SS06] with their Afront algorithm. First, they sample the surface according to the curvature using a guidance field function [SSFS06]. Then these points are used as seeds for a front propagation algorithm. The resulting mesh combines high triangle quality, adaptiveness and fidelity. However, depending of the implicit definition, models polygonized within seconds using Marching Cubes require up to fifteen minutes using this approach. Also following a surface tracking approach, Xi [XD08] produces a curvature dependent semi regular mesh approximating sharp features correctly while supporting disjoint implicit components. Point seeds are created on all disconnected elements of the surface by a regular grid sampling. These sharp features are detected creating new point seeds. These seed are prioritized depending of their curvature and subsequently expanded generating the triangular mesh. For each new triangle, the Delaunay face property is verified forcing the property that the intersection of the local Delaunay sphere with the iso-surface is a topological
This is done by evaluating the normal variation among several points inside the Delaunay sphere.

Meyer et al. [MKW07] generate an adaptive mesh with nearly-regular triangles mixing Delaunay criterion with a dynamic particle system. Starting from an iso-surface description, the medial axis of the surface is extracted using all grid points. A sizing field is then created based on the maximum curvature extracted from the Hessian matrix of an implicit function approximating the iso-surface. The sizing field is sampled generating particles that are distributed with a density related with the curvature of the surface and are projected on the surface using the iterative Newton-Kraphson gradient descent method. Finally the distribution of particles is triangulated using the Delaunay triangulation method. Several examples of biological data sets are tested producing high quality adaptive meshes. However the method is time consuming (complex models require several hours) and unsuited to reproduce sharp features since the sizing field is smoothed and the implicit representation is continuous.

Recently Gelas et al. [GVPN09] introduced a two stage adaptive mesh algorithm that minimizes distance error whilst preserving topology and sharp features. It does so by applying a Delaunay triangulation of a random sampling of the surface. This is then refined by minimizing the local geodesic distance using a quadratic error metric, and flipping the edges which fail the Delaunay criterion. This algorithm only uses both implicit and gradient values and estimates the geodesic by analyzing the normal of several sampled vertices. The final result is an adaptive mesh with good triangle edge ratios and also good reproduction of sharp features.

6. Discussion on Polygonization

Table 2 presents the classification of existing polygonization approaches comparing primary motivations of: speed; topology reproduction ability; quality of sharp feature approximation; surface curvature approximation (smoothness); and the quality of the mesh. Each of these topics are discussed in the following sections.

6.1. Speed Issues

If the primary motivation is for a fast polygonization, we can see that the fastest algorithms belong to the spatial decomposition set of algorithms.

Surface tracking algorithm performance times are highly dependent on the surface being polygonized. Timings are not consistent and the algorithm ends when the complete surface is covered by the mesh. There are a number of surface tracking algorithms that can, depending on the simplicity of the model, be as fast as spatial decomposition algorithms. Less implicit surface evaluations are needed for simple surfaces and a better triangle distribution is produced. The main concern regarding surface tracking algorithms is to guarantee that mesh overlap is avoided during expansion. This is achieved using fast collision detection tests.

Algorithm speeds for re-meshing approaches depend on the complexity of the mesh and the area to be covered. The number of triangles, or vertices, and the quality of the initial mesh are important. A full spatial decomposition does not always need to be performed before applying remeshing techniques. Traditional spatial subdivision requires a standard, surface-wide level of detail to identify small features of the surface. Some remeshing approaches use a lower resolution subdivision and target specific areas of the surface. Therefore some approaches may have comparable timing results to spatial decomposition techniques [NK97, OB02].

6.2. Topology Issues

Topology concerns are particularly important for iso-surfaces extracted from scan data. Polygonization algorithms can not solely guarantee the isotopy between an implicit surface and its polygonal representation. The most successful spatial subdivision algorithms concerning topology have been specifically designed with scan data in mind [Blo94, MSS94b, Che95, HSSZ97, LB03, Nie03, RWY05]. Wood [WHDS02] presents the best remeshing technique, although it does not provide any guarantees.

Topological guarantees are mostly offered by methods studying the critical points of the implicit surfaces [SH97, BNO96]. Regarding mesh generation for iso-surfaces, this problem is overcome using space decomposition improvements to deal with the topological ambiguities [LLVT03, Nie03, Nie04, RWY05, RW08, DSS’09].

When considering topology our analysis has also included the potential for algorithms to identify non-connected elements. Some spatial decomposition techniques [WMW86, Blo94] can not identify disconnected surfaces as they are dependent on a seed cell that intersects the surface to start the polygonization process. Also some surface tracking techniques use the same principal as far as identifying a point on the surface to start the calculation, and also do not correctly identify all topological concerns [HSIW96, CS02, AG01, KS01]. The topological basis used by the delaunay based methods [BO05, CDRR04, DL07] could be used to overcome these issues.
Table 2: Polygonal approaches comparison. Scale: –−− < − −< −< =< +< ++< ++++. The Mesh Type column describes: patterns from spatial decomposition (P), Adaptive (A) and Regular (R). For Speed the approach is considered variable VA when the process cannot be compared to finite spatial decomposition. The classification is empty when it is not possible to evaluate the category due to the lack of support. (Abbreviations: Curv. Curvature, MC. Marching Cubes, MTet. Marching Tetrahedra, MTri. Marching Triangle, ReMesh. Re-Meshing, Subdiv. Subdivision, S.D. Spatial Decomposition, Topo. Topology.)

<table>
<thead>
<tr>
<th>Motivation</th>
<th>Mesh Type</th>
<th>Speed</th>
<th>Topology</th>
<th>Features</th>
<th>Smoothness</th>
<th>Mesh Quality</th>
<th>Techniques</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>[WSIW96]</td>
<td>P</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>S.D.</td>
<td>M.C.</td>
</tr>
<tr>
<td>[LC87]</td>
<td>P</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td></td>
</tr>
<tr>
<td>[BLS96, BLS97]</td>
<td>P</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td></td>
</tr>
<tr>
<td>[HS05, HS09]</td>
<td>PA</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[MS94b]</td>
<td>P</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>Non-Manifold</td>
</tr>
<tr>
<td>[CS95, LLTV95, LB90]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[VdS97]</td>
<td>PA</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[HS82]</td>
<td>P</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[MK91]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. lookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[R95]</td>
<td>PA</td>
<td>+</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td></td>
</tr>
<tr>
<td>[MS98]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td>Extend. MC. Distance Field Simpl.</td>
</tr>
<tr>
<td>[LS92]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td>For Blobs-like IS</td>
</tr>
<tr>
<td>[KV02]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[AS95]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[PR92]</td>
<td>PA</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>−</td>
<td>−</td>
<td>M.C.</td>
<td></td>
</tr>
<tr>
<td>[PC90]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[K96]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[K96]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[K96]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[K96]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
<tr>
<td>[BB95]</td>
<td>P</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.C. NoLookup</td>
<td>For Iso-surface data</td>
</tr>
<tr>
<td>[K96]</td>
<td>PA</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>−</td>
<td>−</td>
<td>M.Tet.</td>
<td></td>
</tr>
</tbody>
</table>

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.
6.3. Sharp Features

Implicit polygonization techniques often miss important qualities of the surface, in particular sharp features. Some methods from both spatial subdivision and adaptive mesh approaches are designed specifically to identify and adapt the polygonization to features.

Spatial subdivision techniques were extended to specifically identify sharp features and some techniques have excellent results [Che95, LLVT03, LB03, Nie03, VKKM03, VKSM04, VKZM06].

Re-meshing techniques are often designed specifically to improve sharp feature approximation [VRKS01, OB02, KB03, AFRS03]. From the surface tracking approaches, only [CS07, GVPN09] proposes an adaptation to improve sharp feature approximation.

6.4. Smoothness

Spatial decomposition techniques and regular meshing methods are rarely concerned with the curvature of the surface and representing it by varying triangle sizes. Adaptive meshing techniques are precisely motivated by capturing local shape characteristics. Generation of polygons is guided by curvature estimation. The best techniques are designed with curvature as a motivation for polygonal construction [AG01, KS01, VRKS01, KB03, MF02, OB02, dAJ05a, CS07].

Most methods approximate curvature rather than calculating it from the Hessian matrix of the implicit function. Although the Hessian matrix can produce robust curvature values the cost is differential geometry calculations, which are as expensive as an implicit evaluation [dAJ05a]. Most correct topological approximation of implicit surfaces relies on critical point analysis [SH97, Wo04, Bo05]. Several approaches have used Hessian based curvature analysis for other purposes such as feature line extraction [PPP88, BLBK03, OBS04].

6.5. Mesh Quality

Referencing Tables 2 and 3 we can highlight two classes of techniques that produce the best polygonal approximations in terms of mesh quality. These are surface tracking algorithms [KS01, AG01, MF02, dAJ05a, CS07, XD08, GVPN09] and re-meshing techniques over Marching Tetrahedra or Marching Cubes triangulations [OB02, VRKS01, KB03, RRS07, NK97, OB01, PFC07, AFRS03]. These algorithms present better adaptive meshes and do not exhibit the triangulation pattern created by cubical space decomposition. It also possible to produce such quality as well as reproducing sharp features correctly as it is done by [GVPN09]. Surface tracking techniques produce well defined adaptive meshes and are more suited to use local shape information such as curvature. On the other hand, re-meshing techniques are identified to be the more complete method allowing to not only produce an adaptive mesh but also to improve the sharp feature approximation [VRKS01, OB02, KB03, AFRS03]. Finally particle based methods combined with Delaunay as shown high quality meshes in [MKW07]. However, this method is slow and does not handle correctly discontinuities.

Table 3: Most Recommended methods for each category

<table>
<thead>
<tr>
<th>Topology</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SH97]</td>
<td>[VRK01, KB03]</td>
</tr>
<tr>
<td>[Nie03, Nie04]</td>
<td>[OB01]</td>
</tr>
<tr>
<td>[Bo05]</td>
<td>[AFRS03]</td>
</tr>
<tr>
<td>[RS01]</td>
<td>[VRK01, KB03]</td>
</tr>
<tr>
<td>[VKZM06]</td>
<td>[GVPN09]</td>
</tr>
</tbody>
</table>

7. Conclusions

In this paper we have summarized visualization methods for implicit surfaces with a focus on fast methods. Ray tracing is a high fidelity, direct visualization technique that produces the most faithful representations of implicit surfaces. It is also the slowest and not generally appropriate for fast visualization. There are also stylized representations using NPR and particle systems. These methods are applicable in specific circumstances like illustrative visualization and are not widely used for general purposes.

The most common techniques for fast visualization, however, are polygonization. This is supported by current graphics hardware design focusing on processing of triangles. Polygonization techniques strive to maintain an acceptable balance between the speed of generation and the accuracy of the mesh as a representation of the implicit surface. Techniques are, however, broadly defined by their primary motivation: either speed of visualization or re-usability of the mesh.

In this paper, we have discussed issues related with the type of visualization, i.e. topological correctness, feature sensitivity, smoothness and visualization or conversion quality. Implicit surface polygonization algorithms have focused on improving the process of conversion into a piecewise representation.

Polygonal approaches are able to satisfy a number of concerns related to creating an accurate surface representation. First they can guarantee topological correctness and creates a high quality approximation of
surface features such as edges and corners. They can also create adaptive representations that fit local shape features such as curvature. The quality of generated meshes is good and can be used for alternative representations by several applications i.e. in the scope of computer graphics or scientific simulation.

We have presented the results of this investigation from the point of view of choosing an appropriate algorithm for specific purposes. If the primary concern is speed or quality of mesh. The results are presented in Table 2 for comparison and evaluation.

From the existing methods spatial decomposition is generally the fastest, and surface tracking and post remeshing techniques result in the most usable representations. Remeshing specific areas of the surface can be fast and provide adaptive meshes, but at the cost of an extra processing step. Surface tracking is the most adapted strategy to use local implicit information during the mesh generation and create a polygonal representation in a single step.

8. Acknowledgements

References


[BPK98] Belaev A., Pasko A., Kunii T.: Ridges and ravines on implicit surfaces. cgi 00 (1998), 530. 6


symposium on Geometry processing (New York, NY, USA, 2004), ACM Press, pp. 245–254. 8, 17


© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.


[ZWW06] Zhang Y., Wang X., Wu X. J.: Fast visual-