Interactive Modelling of Hair with Texture Maps

Will Baker and Scott A. King
Computer Graphics and Vision Research Group
Department of Computer Science
University of Otago
wbaker@cs.otago.ac.nz, sking@cs.otago.ac.nz

Abstract

We present a system for interactively modelling and styling hair on an arbitrary surface using texture maps . Texture maps are used to modulate characteristics of hair strands on the surface, and combinations of these texture maps represent a great deal of information about a hairstyle. We render the scenes in real-time, enabling the user to interactively design and style hair to their liking. Our techniques are designed for real-time facial animation, though they can be applied to the simulation of hair in general.

1 Introduction

Hair modelling and rendering is one of the most difficult problems in computer graphics. Much of the current research in this area focuses on accurate models of hair behaviour and appearance but most do not attempt to provide a way of explicitly styling hair. Why is it necessary to model hair accurately with computer graphics? Most of the pressure for this kind of research comes from the entertainment industry - realistic hair for animations, special effects in movies and computer games.

We present a system that allows a range of hairstyles and hair types to be added to a 3D surface. Creating such a system is not a simple problem - the average human head has over 100,000 hair follicles and on the head alone there are four distinct types of hair - scalp, eyebrow, eyelash and facial hair [1]. Capturing this complexity and range of characteristics is a difficult task, and can be broken down into the following main areas:

- The physical modelling of hair (geometry and physical properties).
- Styling (length, placement and appearance).
- Rendering.
- Animation.

The aim of this research is to solve the first two of these four problems (physical modelling and styling) with the eventual goal of adding hair to an existing real-time facial animation system, *Kare* [2]. We present a new technique for applying and styling hair on an arbitrary mesh, using interactive texture maps to define the appearance of each strand of hair. This technique is well suited to the placement and styling of facial hair, and gives greater control than existing systems.

2 Related Work

2.1 Physical Model

The most immediate problem of modelling hair is capturing its basic shape and behaviour. Hair is affected by gravity and other external forces, giving hair its general appearance. A model must be devised that simulates this behaviour as well as being able to represent the different types of hair (coarse, fine, curly, etc.). Collision detection (preventing strands of hair from passing through the head or other body parts) must also be carried out if the hair is to look convincing.

Anjyo *et al.*[3] present a physical model based on the cantilever beam equation. A cantilever beam is defined as a rigid beam fixed at one end and free to move at the other. A differential equation is used to calculate the bend of the beam due to gravity or other external forces. Collision detection between hair and body is approximated by testing for intersection between the strands and an ellipse; inter-hair collision is disregarded. We modify the cantilever beam model of hair from Anjyo *et al.*[3] for our physical model.

2.2 Styling

In order for hair to look convincing, it is necessary to style it. Hair can be cut, combed, set or coloured and countless hair products can be applied. Hair only appears in certain areas on the head, so there must be some system for specifying where these areas are. Kajiya and Kay [4] and Goldman [5] used a Poisson disk distribution (which models the way in which mammal hairs are distributed [4]). A human head, however, has follicles which are generally equidistant from each other but have a random distribution [6]. Streit and Heidrich [6] suggest that reaction-diffusion textures could be used to simulate this distribution.

Styling techniques have been proposed [7, 8] which give the user differing degrees of control over the look of a hairstyle, allowing cutting, combing and the addition of curl [8] to the hair. Kim and Neumann [7] approximated hair using thin shell volumes. The hair can then have a combing function applied which allows virtual combing, though this method is not interactive which makes it more difficult to achieve an exact style. Later research of Kim and Neumann [8] allows interactive editing of hairstyles based on clusters of hair at different scales. This achieves pleasing results, but to handle the information about each cluster of hair during editing a large amount of memory (a maximum of 200MB [8]) is required.

A different approach is that of Hadap and Magnenat-Thalmann [9] who present an interactive styler based on fluid flow. The interactive styler offers intuitive control of the hairstyle's appearance and a range of different styles can be achieved. However, this system is best suited for long hair and the extent of its ability to generate short hair (such as eyebrows, eyelashes and beards) is unclear. Hadap and Magnenat-Thalmann also state that, because their system requires about a second to update a style after changes, it is not real-time [9].

Here we propose a new way of styling hair, using interactive texture maps to determine the length, colour and combing of the hair. This is an improvement over existing systems because the texture maps used to store the information are small so that fast interactive styling of hair is possible.

2.3 Rendering

The accurate rendering of hair is one of the greatest challenges in computer graphics [10]. There are over 100,000 hairs on the average human head and each of these hairs interacts with light in a complex way. Hair is partially transparent, exhibiting both reflection, which behaves anisotropically, and transmission.

Various methods for rendering hair realistically have been presented [11, 4, 5, 12] which solve the problem satisfactorily. Usually, the lighting model used for hair is that of Kajiya and Kay [4], but the recent Marschner *et al.* [11] solution improves on this model. They not only solve the lighting of hair strands convincingly, but also handle issues such as back-lighting and shadowing. Because their system is based on ray-tracing, these effects can be easily captured by their model without additional processing. However, ray-tracing is an expensive process, so rendering time for their method is slow. Because the *Kare* [2] system operates in real-time a short rendering time is crucial. Yang and Ouhyoung [10] present a method for simulating back-lighting which may be used to decrease rendering time. Lokovic and Veach [13] present a shadowing method they call 'deep shadow maps' which may also prove to be a quicker way to simulate shadows.

3 Physical Model

A strand of hair is modelled as a polyline through several equidistant vertices or nodes. The cantilever beam method [3] is used to calculate the position of each of these nodes based on a hair pore location and an initial direction. The nodes are then connected with straight line segments of equal length to approximate a curve.

Our method differs from the method of Anjyo *et al.* [3] in that we calculate the bend of each hair differently depending on its length. In their method, the bend is calculated for a fixed length of hair (i.e. a fixed number of nodes) and in order to display different lengths of hair the number of these nodes drawn during the rendering process is limited. This means that a short hair strand has the same amount of bend as longer hair strands. A real strand of hair bends differently according to its length - a longer strand weighs more and hence is affected more by the acceleration due to gravity.

Collision detection between hair strands and the surface on which they are grown is approximated using a parametric volume. Because this bounding volume is parameterized, intersection tests are simple and hence the collision detection process is fast. As the position of each node in a hair strand is calculated it is checked for intersection with the bounding volume - if there is an intersection then the node is moved outside of the volume in the direction of the surface normal. This method is most satisfactory when hair is grown from a parametric surface because the volume can fit the surface exactly. When using a triangular mesh this method is only an approximation, though it still yields reasonable results.

4 Styling

We use texture mapping as a device to create different hairstyles, allowing all the characteristics of a hairstyle to be stored in several texture maps. These texture maps can be generated directly on 3D geometry allowing a user to create a hairstyle in 3D and in real time. This geometry can either be a polygonal mesh or a parameterized surface.

In order to represent all characteristics of a hairstyle, our system uses three distinct texture maps. These are:

- *Placement Map* - This defines the density and length for an area of the hairstyle.
- *Colour Map* - This defines the colour of each strand of hair.

- *Comb Map* - This defines the change in initial direction of each strand, simulating the combing of hair.

Each of these maps is stored as a colour image, enabling an entire hairstyle to be captured in a standard image file. The following sections describe each of these texture maps in more detail.

Texture maps are traditionally used to add detail to a model without adding extra geometry. A 2D image is 'mapped' to a 3D model using texture coordinates that are usually defined for each vertex in the model. The texture coordinates map each point in $\Re^3$ to a 2D location in texture space. This 2D location can be used to store details, such as colour, which modulate the colour on the surface of the model. Our system uses a variety of mapping methods.

In our system, instead of the texture maps altering the colour, they change the characteristics of the hair strands growing at the 3D point. For each area covered by a texel on the model, the value for this texel in the texture map is used to grow the appropriately styled hair strands in that area. Figure 1 shows a simple beard style created using a placement and colour map.

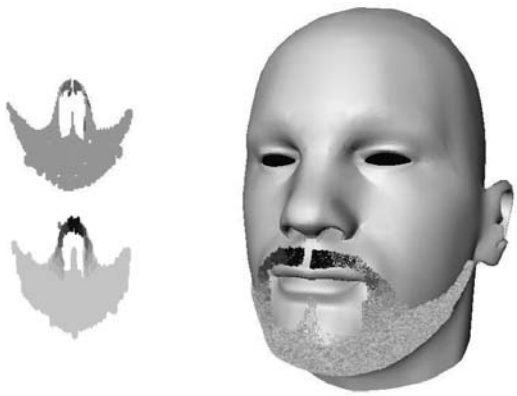

Figure 1: Placement and colour maps with the resulting hairstyle (right). The placement map (top left) is a negative image of the actual map making it easier to visualize than the original. The colour map appears at the bottom left.

The hair placement map is a colour image in which two channels represent density and strand length. When the texture map is applied to the mesh, each texel covers a certain area of the mesh. The density value stored in the texture map determines how many hairs should be grown in this area. The length value in the texture map determines how long each of these hairs are. A length or density of zero indicates that there is no hair being grown in this area. The exact pore location of each hair within a texel is determined randomly to better reflect the distribution of hairs - as mentioned in Section 1.2 this distribution is random rather than uniform. Individual strands of hair can also be added in precise 3D locations, for example adding eyelashes or nasal hair. These strands are not stored in the texture map but instead their geometry is stored separately.

Because the hair strands represented by a texel must necessarily have similar characteristics (with some small random variations), it is beneficial to use a texture map of sufficient resolution. A lower resolution texture map means that each texel covers a larger area on the mesh. If large areas of hair on the mesh share the same characterisitics (especially in the case of colour and combing information) the overall hairstyle will appear less convincing. We found that in practice a texture map of 256x256 pixels has sufficient resolution for satisfactory results.

Texture maps are also used to determine the colour of each hair strand. Each texel of the colour map, when mapped to the 3D mesh, covers a certain area of that mesh. All the hairs within this area will get their colour from that stored in the colour map. The colour map need not be of the same resolution as the placement map, so an area that has the same density and length for each hair need not share the same colour.

Essential to the creation of any hairstyle is the ability to alter the initial direction of the hair. Hair can be combed to change this direction, which is what the comb map is used to simulate. In our system we initialize this direction to be the surface normal (with a random perturbation), which approximates the actual behaviour of hair. In order to simulate the hair being brushed or combed and capture the aberations in the direction the hair exits the scalp, a comb texture map is used.

The comb map consists of a colour image in which each channel corresponds to a magnitude in each of the three primary axes (X, Y, and Z). The combination of these channels gives a vector in three dimensions which can be used to alter the initial direction of a strand of hair. For example, if the RGB value of a pixel in the texture map was (10, 20, 30) then the vector used to alter the initial direction would be (10, 20, 30).

4.1 Interactive Texture Mapping

The texture maps described above can all be created by hand using a standard painting program. This can be a difficult task, especially in the case of the comb map, because it is hard to visualise how the texture map will appear after it has been mapped to the mesh. The comb map is even more difficult to visualise because it is representing a vector in $\Re^3$ with an RGB value in a 2D image. Figure 2 shows such a comb map together with the resulting style.

In order to provide a more intuitive method to create hairstyles, we have made it possible for a user to generate the three texture maps by painting them directly on

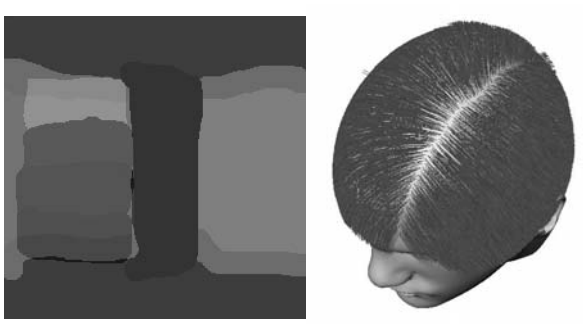

Figure 2: A comb map (left) and the resulting style. The effect of the comb map is difficult to visualise.

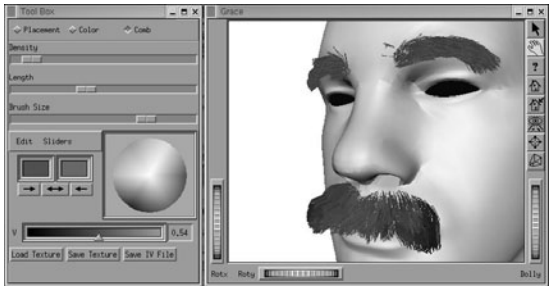

Figure 3: The styling system used in hairstyle generation. The toolbox on the left allows selection of different brush characteristics, and the model view on the right shows the current hair styling.

a 3D model. The mouse cursor acts as a virtual brush which is used to apply different textures to the model. The user's mouse actions, which are in two dimensions, are translated into three dimensions. This translation is achieved by casting a ray through the mouse location on the image plane and into the scene. The intersection of this ray with the mesh, if any, can be calculated and used to find the u,v coordinates of the texel being altered.

Cylindrical texture mapping is used to find where this point is in texture space, and the texture map can then be updated according to the type of brush and the appropriate texels can be refreshed. It is important to note at this point that because it is known which texels have changed it is only those texels that need to be refreshed. Because only a few hairs need to be recalculated each time, the operation is very fast. If all 100,000 hairs needed to be recalculated the system could not work in real-time.

Refreshing the texels that a user has changed is a simple matter because, from the first intersection of the mesh, the face intersected and hence the normal of that face are known. Random pore locations within the changed texel (or texels) can then be generated in texture space. Each pore is tested to ensure that it lies within the intersected face, and then hair is grown based on the texture map values at u and v and the normal of the face. Testing that a pore is inside a face requires that the face the texel is associated with is first translated into texture space. The texture coordinates, u,v, for each vertex of the face can be found easily with a method similar to that described above. The barycentric coordinates for each pore location in texture space are then calculated and used to find the pore location on the face in object space (i.e. on the 3D mesh).

A slightly more difficult task is loading a complete texture map from the disk. The problem is that when processing each texel in the map we do not know which face the texel should appear on. When using a parametric volume this problem is simplified because it is relatively easy to get from a u,v coordinate in texture space to the 3D point on a parametric volume. But a mesh is a different matter as each point does not map so nicely into texture space. Our solution to this problem was to iterate through each face in the mesh, translating it into texture space. The bounding box of the face is then found in texture space, which gives a collection of texels which could possibly contain the face. Hair is then added to each of the texels at random pore locations, but only if each location is within the face (hence texels that contain only some or none of the face will have less or no strands growing from them). To make this method less computationally expensive, information about each face (such as the normal, area and the bounding box) is pre-generated and stored.

There are three different brush types available. The placement brush 'paints' the hair onto the mesh with a specified length. It also controls density and rate of hair growth. The colour brush allows the colour of existing strands to be changed. The colour is selected from the toolbox using the convenient colour wheel. The comb brush allows the user to drag the cursor over the hair and comb it in real-time. The velocity of the cursor's movement affects the amount of combing, allowing intuitive control of each strand's initial direction. The interactive texture mapping system allows varying control of hair strand geometry, enabling a wide variety of hairstyles to be created with relative ease. Figure 3 shows the system in action with the toolbox for selecting brush characteristics on the left and the system view on the right. The model in progress shows variable colour on the moustache and combing on the eyebrows.

A limitation of the system as it stands is that the texture maps are distorted when they are applied to the mesh. Because using a cylindrical map assumes that the object closely fits a cylinder, as you approach the top of the skull the texture is distorted. This is an unfortunate but unavoidable consequence of cylindrical texture mapping. If hair needs to be grown on the scalp a spherical texture map can be used. However, this method has distortions when close to the poles of the sphere so it is inappropriate for hair on the face.

A possible solution to the problem of texture map distortion is to manually (or perhaps automatically) define the coordinates of each vertex of the mesh. This would enable the user to define which portion of the texture map is used to define the hair on each face, thus eliminating distortions. A beneficial side effect of using manual texture coordinates is that areas that require finer detail (such as eyebrows and eyelashes) can be assigned greater areas of the texture map, while areas that do not require detail (such as the middle of the scalp) could be assigned a much smaller area in texture space.

5 Results

Figure 4: Various beard styles created with our system.

Our method of representing hair with several texture maps has been used to create many hairstyles, and performs particularly well in the creation of facial hair. Figures 4 to 7 show some examples of the styles produced by our system. Using our interactive system these styles required minimal time to design - around five minutes each. The styles were created on a Pentium 4, 2.40GHz machine and rendered in real-time.

Figure 4 exhibits the ability of the system to model facial hair. Note how even different beard styles alter the appearance of the head model. Figure 5 shows hair grown on parametric volumes. These volumes are an approximation of the shape of the head so they do not perform well for facial hair but they are satisfactory for scalp hair, especially if collision detection is required. The style in the middle exhibits collision detection using an elliptical bounding volume. Figure 6 shows hair growing on two different mesh models, a torus (left) and the Stanford Bunny (right). The Stanford Bunny, in particular, is a difficult case because the shape of the mesh does not fit well to a cylinder, causing distortions of the texture map as described in Section 4.1.

Figure 7 shows the control that the system allows the user over the hairstyle. It gives a comparison between an actual eyebrow style and the reproduction of that style using our system.

Figure 8 demonstrates another benefit of our texture mapping system. Because all the information about the hair strands are stored using a texture map it is possible to easily store additional information about the hair strands, in this case the growth rate. The figure shows a simulation of hair growth. First a beard is drawn with a short length and given some growth rate. The current time value can then be altered using a slider bar in the toolbox and the beard sprouts in real-time. It is possible, with our system, to have a large number of texture maps, and each could store different information such as the degree of curl, colour change over time (for greying hair or dyed hair with different coloured roots), strand thickness, etc.

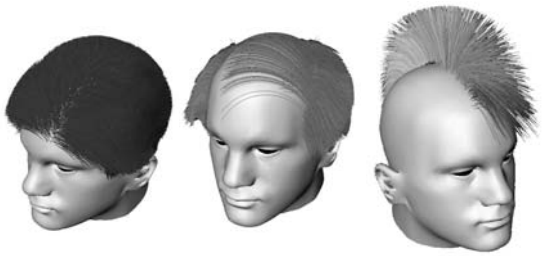

Figure 5: Various styles generated using parametric volumes.

6 Conclusions and Future Work

We describe the use of texture maps to control the placement and styling of hair. We also present a system that enables the user to model and style hair interactively on an arbitrary mesh or parameterized surface. The user can specify the areas on this surface that are to grow hair, as well as the colour and length of the hair in each area. The user can then interactively comb the hair by dragging a virtual comb over the hair and watch as the initial direction of each strand is altered in real-time. This styling method allows a great variety of hairstyles to be created quickly and easily, and then applied to a system such as *Kare* [2]. A particular strength of this system not seen in many others is the ease with which it can handle facial hair (such as eyebrows, eyelashes and beards), a problem not often tackled in other work.

The modelling of the hair is a single coloured polyline whose geometry is generated using the cantilever beam method of Anjyo *et al.* [3]. It is also possible to extend this method so that each hair strand is modelled

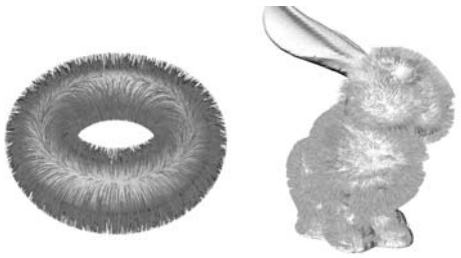

Figure 6: Hair generated using arbitrary meshes.

Figure 7: The system is able to capture fine detail.

Figure 8: Frames from an animation which simulates beard growth by increasing strand length over time.

by a spline curve, resulting in smoother strands even when viewed up close. Also, a continuous model of the cantilever beam equation could be used - we use a discretized approximation for speed but with the increase in CPU power it may now be possible to use a more accurate model.

Further texture maps could also be employed to represent a wider range of characteristics for each strand of hair. The amount that each hair bends, the bending moment, can differ depending on the type of hair (scalp or facial) and other factors such as hair styling products. A texture map could be used to represent the bending moment of each strand, and thus simulate gel and other hair products being applied to the hair. The degree of curling for each strand could also be stored in a texture map, allowing a hairstyle to have different degrees of curl in each area.

Rendering is done either using single coloured polylines, which are simple and fast to render, or using the lighting model of Kajiya and Kay [4]. The apparent thickness of the hair does not change depending on the distance of the camera to the hair. This could be handled during rendering either by giving some geometry, such as a cylinder or prism, to the hair or by drawing lines of appropriate width. Currently the lines are uniformly one pixel wide. More recent lighting models for hair (such as Marschner *et al.* [11]), which handle effects such as backlighting and shadowing, could also be used for high quality offline rendering once the hairstyle has been defined interactively.

7 Acknowledgements

We would like to thank Sui-Ling Ming-Wong for her expertise in spotting those hard to find grammatical errors in all our writing.

References

[1] V. Valkovic. *Trace Elements in Human Hair*. Garland STPM Press, New York, USA, 1977.

[2] S. A. King, A. Knott, and B. McCane. Language-driven nonverbal communication in a bilingual conversational agent. In *Proceedings of Computer Animation and Social Agents 2003*, pages 17–22, 2003.

[3] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 111–120, 1992.

[4] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 271–280, 1989.

[5] D. B. Goldman. Fake fur rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 127–134, 1997.

[6] L. M. Streit and W. Heidrich. Modelling the embryological distribution of follicles. In *Proceedings of Western Computer Graphics Symposium (SKIGRAPH '01)*, 2001.

[7] T. Y. Kim and U. Neumann. A thin shell volume for modelling human hair. In *IEEE Computer Animation Proceedings*, pages 104–111, 2000.

[8] T. Y. Kim and U. Neumann. Interactive multiresolution hair modelling and editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 620–629, 2002.

[9] S. Hadap and N. Magnenat-Thalmann. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation 2000*, pages 87–99, 2000.

[10] T. J. Yang and M. Ouhyoung. Rendering hair with back-lighting. In *CAD/Graphics '97 Proceedings*, pages 219–296, 1997.

[11] Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics (TOG)*, 22(3):780–791, 2003.

[12] T. Y. Kim. *Modelling, Rendering and Animating Human Hair*. PhD thesis, University of Southern California, 2002.

[13] T. Lokovic and E. Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392, 2000.

An Efficient and Topological Correct Polygonisation Algorithm for Finite Element Data Sets

Burkhard Wünsche[1] and Jenny Zheng Lin[2]
Division for Biomedical Imaging & Visualization, Department of Computer Science,
University of Auckland, Auckland, New Zealand
[1]burkhard@cs.auckland.ac.nz, [2]jennylin_nz@yahoo.com

Abstract

Implicit defined surfaces of scalar fields (isosurfaces) are a common entity in biomedical, scientific and engineering science. Polygonising an isosurface permits hardware assisted rendering and simplified geometric operations such as surface analysis and area computation. This paper introduces a novel algorithm for computing isosurfaces for scalar fields defined over (potentially curvilinear) finite elements which are common in numerical simulations and physically-based modelling. The advantages of the method are demonstrated by visualising the myocardial strain in a healthy and a diseased heart.

Keywords: polygonisation methods, isosurfaces, Marching Cubes, curvilinear elements, finite elements

1 Introduction

Implicit defined surfaces of scalar fields (isosurfaces) are common in biomedicine and other sciences. Isosurfaces impart knowledge about the overall distribution of a scalar field and can be used to extract anatomical structures from medical imaging data. The *c-isosurface* of a scalar field s is defined as all points $\mathbf{x}$ for which $s(\mathbf{x}) = c$.

Numerous algorithms (so-called *polygonisation methods*) have been proposed for the efficient computation of isosurfaces (e.g., [1, 2, 3, 4]). Interactive display rates and reduced storage requirements can be achieved by utilising adaptive methods [5], mesh reduction techniques [6] and multi-resolution meshes [7, 8]. A survey and analysis of polygonisation methods and optimisation techniques to achieve faster computation and rendering of isosurfaces is found in [9].

The Marching-Cube algorithm [10] has been one of the earliest and most popular methods. The algorithm requires as input a regular grid of sampled field values and "marches" through the volume cell-by-cell. Each grid cell has eight sample values at its corners. The method constructs a tessellation by computing for each cell the intersection points of the cell's edges with the isosurface and by connecting these intersection points with triangles obtained by a table look-up. The look-up table contains all configurations which fulfill the assumption that the isosurface intersects a cell's edge at most once. Since there are eight vertices in each cubic cell and two values, *positive* and *negative*, there are $2^8 = 256$ ways the surface can intersect the cube. Lorensen and Cline use symmetries to reduce the number of patterns to 15 which are shown in figure 1[1].

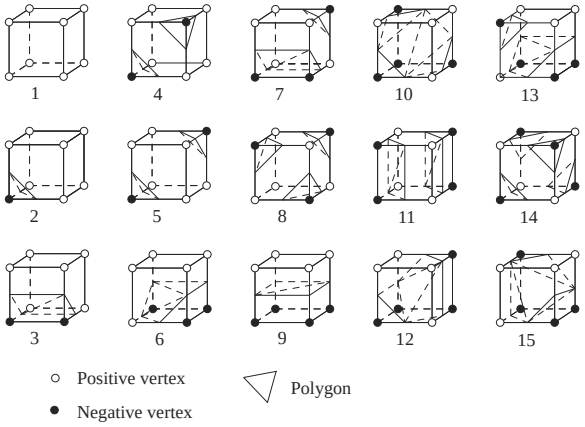

Figure 1: Triangulated cubes.

The main disadvantage of the algorithm is that some patterns in figure 1 are topologically ambiguous as noted by van Gelder and Wilhelms [12]. This may produce a surface with a hole as pointed out by Düurst [13] (see figure 2). The literature offers various solutions to the ambiguity problem [14, 15, 16, 17, 18]. Also in some applications the topology of a biomedical structure is known in advance and specialised polygonisation algorithms can be employed to take this into consideration [19].

In this paper we present a modification of the Marching Cubes algorithm which can be applied directly to curvi-

[1]The cases 12 and 15 are reflective with respect to the xy-plane. This leaves 14 topologically distinct patterns (22 without inversed patterns) [11].