Visual Hull Surface Estimation

Phillip Milne Fred Nicolls Gerhard de Jager
Digital Image Processing Group
Department of Electrical Engineering
University of Cape Town
South Africa
phillip@dip.ee.uct.ac.za

Abstract

This paper describes a technique used to approximate the surface of an object's visual hull. The hull's basic structure is initially represented by a partitioned 3D space using voxels. The marching cubes algorithm then assigns polygonal patches to surface voxels depending on a certain criteria. It is less computationally intensive to manipulate a few triangular patches than a number of six-faced voxels. The visual hull model is further refined by applying a binary search to the vertices of each surface that improves the positional accuracy of each vertex. The method is applied to a small plastic cat using a 5-camera system and results are shown.

1 Introduction

In computer graphics and machine vision, it is often necessary to model real world objects. The information contained in a 3D model can be useful in systems that require any one of the following:

- Multimedia Content: making movies or computer games with 3D objects.
- Classification: deciding on the type of shape of an object.
- Recognition, e.g. recognising a person's gestures.

Information on the structure of an object is contained in any 2D image of that object (an image of a small plastic cat can be seen in figure 1). The information from a single view can be used towards approximating a 3D model of the object. The accuracy of the model is improved with an increase in the number of different views available.

Figure 1: Multiple image views of a small plastic cat.

This paper discusses some of the concepts needed in order to build a 3D model of an object from silhouette images. These silhouette images are obtained from an accurately calibrated camera system [4] and used to generate a voxel based model. This model is used as an initial estimate of the visual hull of an object. Surface voxels are then replaced with triangular patches. This has the effect of smoothing the visual hull. A binary search further refines the accuracy of the visual hull.

2 The Visual Hull

An object's *visual hull* is a geometric representation of its structure. This representation is an upper bound estimation and is not necessarily an exact copy of the object. Visual hulls cannot capture concavities, such as the inside of a tea cup.
The visual hull can be computed from the silhouettes of an object taken from differing viewpoints. Silhouettes are extruded to create cone-like volumes which intersect to form the object's visual hull. The accuracy of the visual hull is refined as more views are added.
Sample silhouette images can be seen in figure 2.

These silhouettes were obtained by thresholding grey scale images.

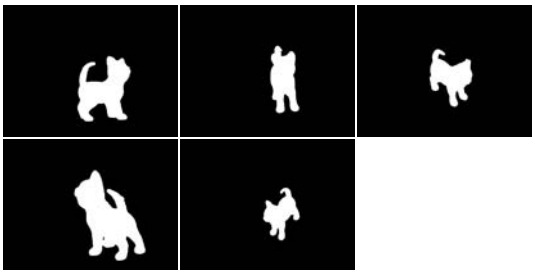

Figure 2: Five silhouette images of a plastic cat

3 Voxel Representation

One method to represent an object's visual hull is to create a spacial occupancy map using a number of volume elements or *voxels*. The concept involves starting with an initial cubic volume and "carving away" parts that are not included in the visual hull.
Voxels are the 3D equivalent to the pixels that make up a 2D image. A single voxel element is a scaled cube with 3D coordinates in some coordinate system. It has ordered vertices (as shown in figure 3) that can each be assigned a different value. These values decide the occupancy of the voxel.
The three voxel occupancy categories are:

- Voxel is completely inside the visual hull
- Voxel is completely outside the visual hull
- Voxel has the surface of the visual hull running through it

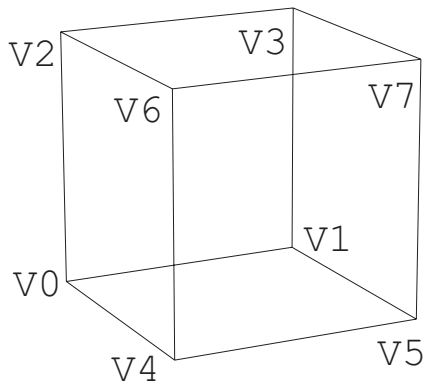

Figure 3: A voxel with ordered vertices.

The size of the smallest voxel determines the resolution of the visual hull. As the number of subdivisions increases, so does the amount of processing time require to compute the model.
Using a voxel structure to represent an object has certain flaws that can be overcome provided that the system has a basic knowledge on the shape of the object being modeled. When objects are snake-like or have sharp peaks, they appear discontinuous. This flaw can be overcome by making the voxels smaller and hence increasing the resolution of the visual hull.

3.1 The Octree Structure

An octree is a tree data structure [1] that can be used to represent the voxel based visual hull model. This involves specifying an initial voxel, known as the *universe cube*. The universe cube is split into 8 suboctants. Each suboctant is iteratively subdivided until some predefined limit is reached. This idea is more clearly illustrated in figure 4.

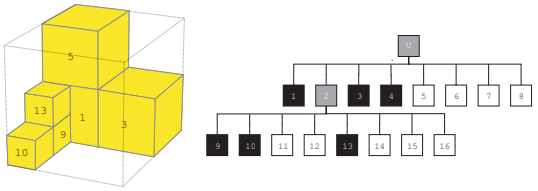

Figure 4: A simple volume represented by an octree and the corresponding tree structure with nodes [**?**].

Figure 5 illustrates how the accuracy of the visual hull can be improved by increasing the number of subdivisions. The first visual hull was computed with 2 subdivision levels, the second with 3 and the third with 4. The larger the number of subdivisions, the more computationally expensive the algorithm becomes.

4 Surface Approximation

Using a number of voxels to represent a visual hull is a simple concept, and easily implemented. The large number of voxels used to make up the visual hull causes other systems using these models to become sluggish. A solution to this problem is to apply the *marching cubes* algorithm to all surface voxels of the visual hull.
The marching cubes algorithm approximates a polygonal surface that passes through the surface

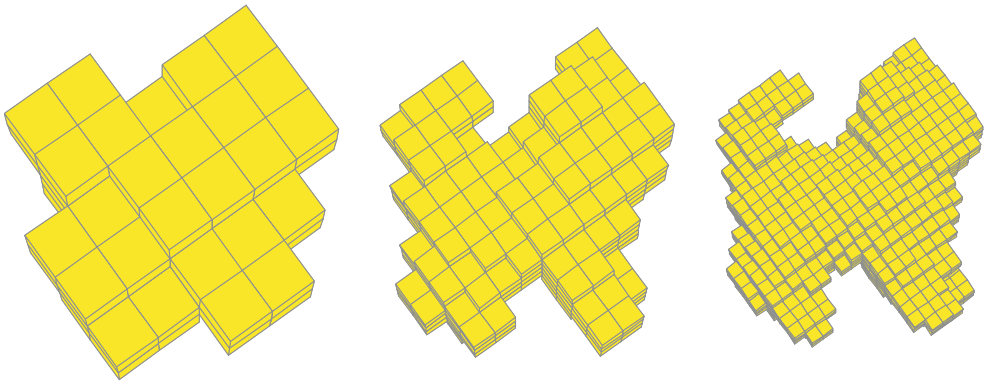

Figure 5: The 3D voxel based visual hull model of the imaged cat at 3 different subdivision levels.

voxel. It also results in a smoothing of the voxel corners along the surface of the visual hull [5].

4.1 Marching Cubes

The marching cubes algorithm was first used to visualise mathematical equations. It assigns a surface to a voxel depending on the voxels corner values and how they are arranged. Surfaces are made up of a number of triangular patches. The three vertices of the triangle are ordered using the *right hand rule*. This means that they are ordered anticlockwise about the surface normal. The marching cubes idea can be more easily illustrated by its 2D equivalent *marching squares*.

4.1.1 Marching Squares

Figure 6 shows how a line is matched to a square, depending on its corner configuration. Solid black circles indicate that a corner is part of the target, while the absence of a circle indicates that a corner is background.
Four images of the same silhouette view that have been placed inside a *universe square* are depicted in figure 7. Each universe cube has been subdivided 3 times. Black indicates the computed 2D visual hull while grey is the colour of the actual visual hull. Each image is described below:

A The silhouette image view inside a universe square.

B The 2D implementation of a voxel based visual hull.

C The marching squares visual hull model.

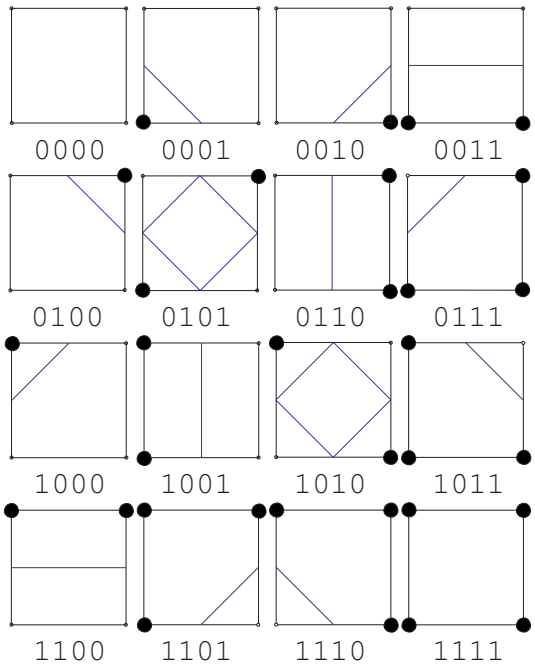

Figure 6: Marching squares.

D The marching squares model is refined using a binary search.

The concept of a binary search, shown in figure 7.D, is used in the final implementation of the marching cubes algorithm and is discussed in more details in a later section.

4.1.2 Cube Categories

The corner values of the voxel determine the type of surface that passes through the voxel. Of the 256 corner combinations, there are 15 unique categories

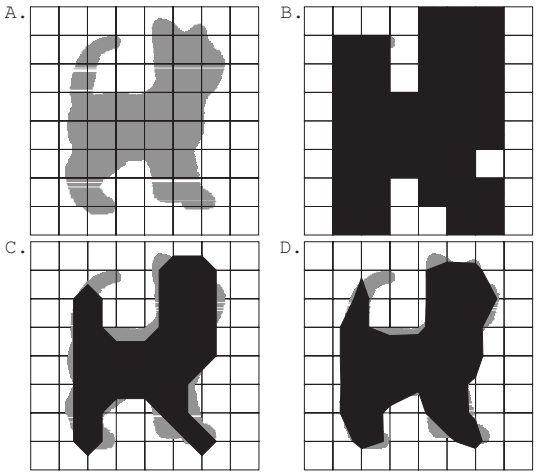

Figure 7: Marching squares sample.

that voxels can be classified into [3]. Each different category has a different surface which passes through it (see figure 8).

4.1.3 Ambiguities

Using the 15 surfaces illustrated in figure 8 alone results in various ambiguities [6]. These are easily visible in the resultant visual hull in the form of holes [2]. Figure 9 shows why these ambiguities occur, and also shows how they can be overcome. Using the ideas displayed in this figure, a new table of 33 surfaces can be created. (see figure 10). Using this table solves the problems that arise in ambiguous cases. Figure 11 shows the effects of applying the marching cubes algorithm to the voxel based visual hull models in figure 5. These models are smoother than the voxel based versions from figure 5. The patches are coloured according to their voxel corner category. The improvement on the voxel model is quite evident in the middle marching cube visual hull model.

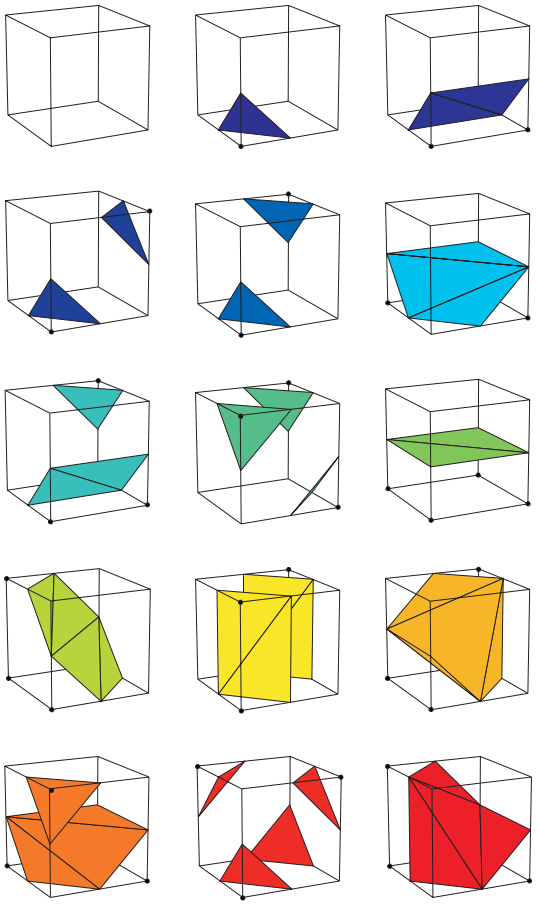

Figure 8: The 15 marching cube surfaces [7].

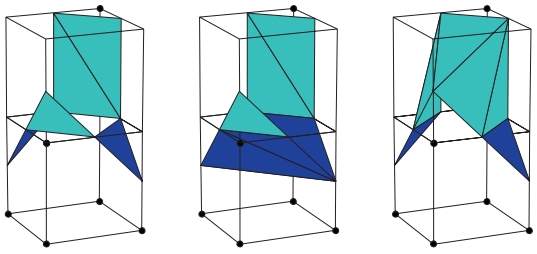

Figure 9: One of the ambiguous marching cube surfaces.

4.1.4 Refining the Surface

The accuracy of the visual hull model can be improved by applying a binary search to the vertices of each triangular patch making up the hull surface. This requires that each vertex in the assigned surface be projected back into the silhouette images and tested. The vertices are then moved along the cube edges in a direction determined by the result of their projection test. Figure 7D. shows the effect that a single iteration of this search has on a 2D model.

The result of applying this binary search to the marching cubes visual hull from figure 11 can be seen in figure 12. The effects are more noticeable in the first visual hull model. The higher the number of voxel subdivisions, the less noticeable the effects of this search become.

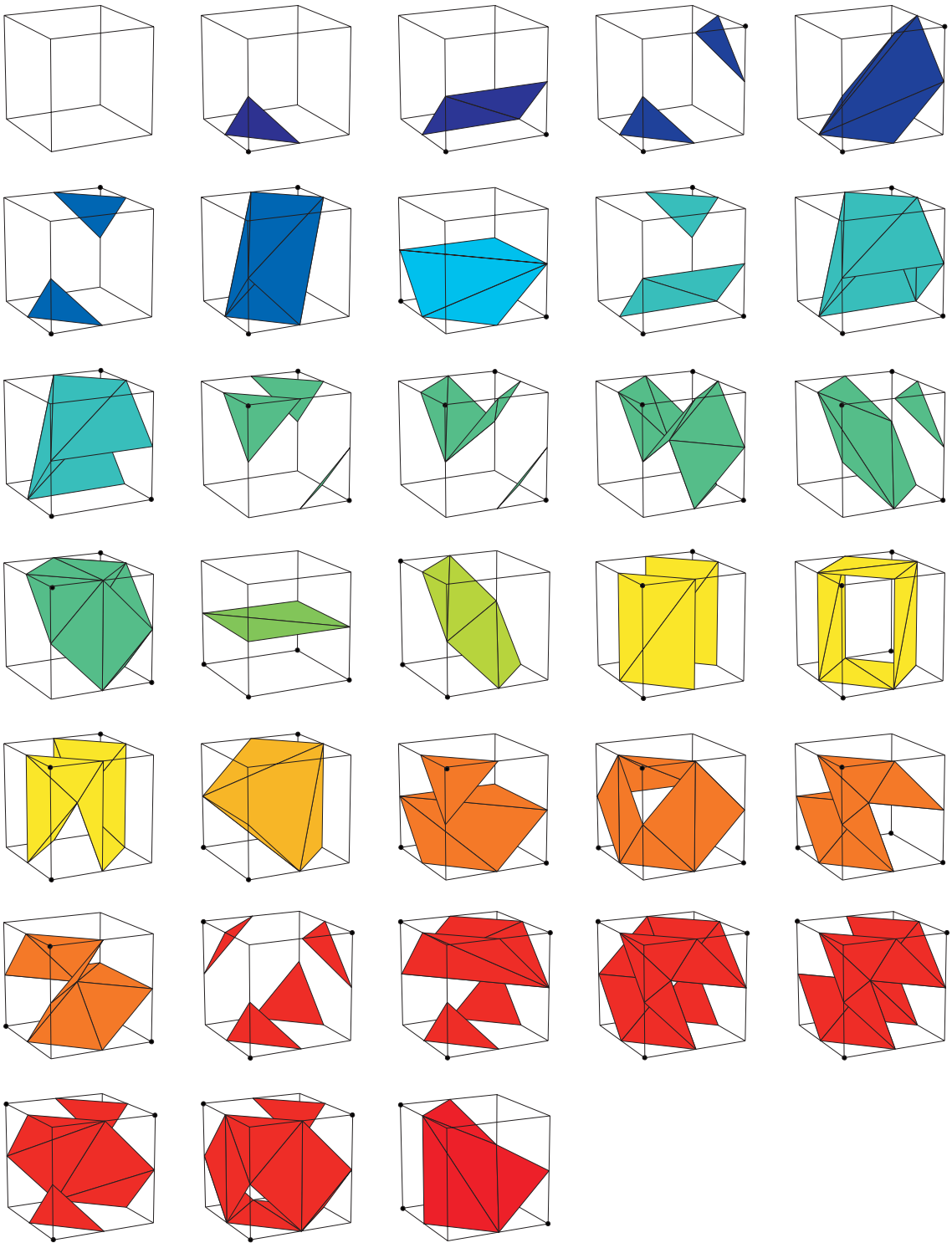

Figure 10: The 33 marching cube surfaces[6]

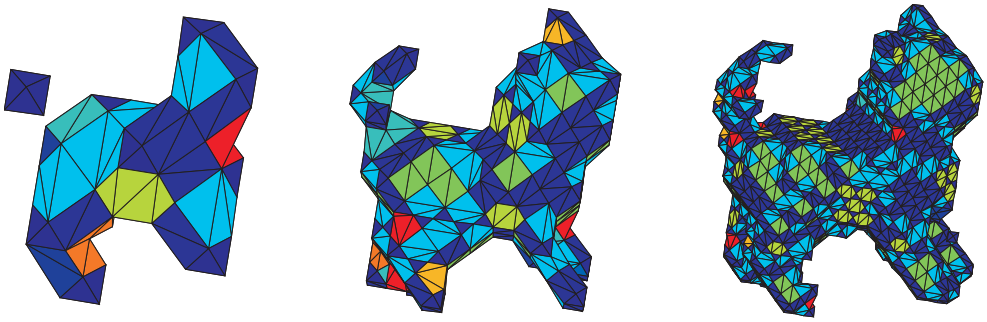

Figure 11: The 3D marching cube visual hull model of the imaged cat at 3 different subdivision levels.

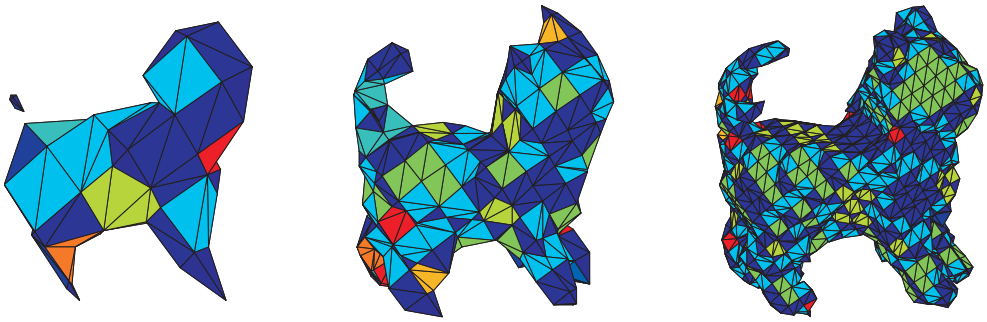

Figure 12: The 3D marching cube visual hull model of the imaged cat at 3 different subdivision levels, with 4 iterations through a binary search.

5 Conclusion

Marching cubes reduces the amount of time required by a computer to manipulate a visual hull. It produces more accurate models with smoother surface than the voxel based method (comparing figure 5 with figures 11 and 12). The binary search element in the algorithm improves the visual hull accuracy on voxel models with low subdivision levels, but its effects are less noticeable at higher levels.

6 Acknowledgements

I would like to thank the De Beers Group Technical Support for their financial support and input.

References

[1] Narendra Ahuka and Jack Veenstra, *Generating octrees from object silhouettes in orthographic views*, Pattern Analysis and Machine Intelligence **11** (1989), no. 2, 137–149.

[2] Ken Brodlie and Jason Wood, *Computer graphics*, Recent Advances in Volume Visualization (2001).

[3] R. Scateni C. Montani and R. Scopigno, *Discretized marching cubes*, Internet Paper.

[4] Keith Forbes, Anthon Voigt, and Ndimi Bodika, *An inexpensive, automatic and accurate camera calibration method*, Proceedings of the Thirteenth Annual South African Workshop on Pattern Recognition, PRASA, 2002.

[5] William E. Lorensen and Harvey E. Cline, *Computer graphics*, Marching Cubes: A High Resolution 3D Surface Reconstruction **21** (1987), no. 4, 163–166.

[6] Antonio Wilson Vieira Thomas Lewiner, Helio Lopes and Geovan Tavares, *Efficient implementation of marching cubes' cases with topological guarantees*, Ph.D. thesis, Pontifical Catholic University, INRIA, 2004.

[7] Kwan-Yee Kenneth Wong, *Structure and motion from silhouettes*, Ph.D. thesis, University of Cambridge, 2001.