Generation of Visual Hull Models of 3D Objects

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 method for generating visual hull models of 3D objects from digital images. The method can use one or more cameras in different setups to acquire various views of an object. From the generated images, 2D objects are segmented out giving the objects silhouettes. The silhouettes are projected into 3D space to create an estimate of the upper bound to the structure of the object (visual hull of the object). Methods to display the output are also provided.

1 Introduction

In computer vision systems it is often necessary to determine the three dimensional (3D) structure of an object. The information contained in the structure can be used for many things, such as classification and/or recognition of objects. An application where this could be of some benefit is in a shape classification system.
There are currently a number of different automated techniques available that could accomplish this task. The technique to be investigated in this paper involves the computer processing of digital images to generate the visual hull. Multiple images are generated from cameras positioned around an object in an accurately calibrated system. Calibration is discussed briefly in this paper.

1.1 The Visual Hull

The visual hull is computed by back-projecting silhouettes observed from multiple viewpoints in a process known as volume intersection. The visual hull is an upper bound estimate to the 3D structure of an object.

2 Cameras

Because cameras play such a significant role in this project, it is necessary to become familiar with the various characteristics of these digital devices. The pinhole camera model is briefly described below.

2.1 The Pinhole Camera Model

All cameras perform a perspective transformation from some 3D space (the real world) to a 2D point in the retinal plane (image plane). In the camera model, three different coordinate systems are defined: the camera coordinate system, the image plane coordinate system and the world coordinate system.
In the camera's coordinate system, the camera is positioned at the origin (0,0,0). The distance from the camera centre to the image plane is the focal length (f). The focal length is the distance from the camera center along the optical axis to the point where it intersects the image plane at the principal point (or image centre) [5].
Most images use pixels as their unit of measurement, but this is dependent on the technology of the camera. The image origin (0,0) is generally positioned in the top left corner of the image . To map a point from the image reference frame to the cameras, the units need to be converted to match that of the camera's. A coordinate translation, and sometimes a scaling, is then used to map a point from one reference frame to the other. This can be more clearly seen in figure 1 [1].

The world coordinate system is discussed in a later section.

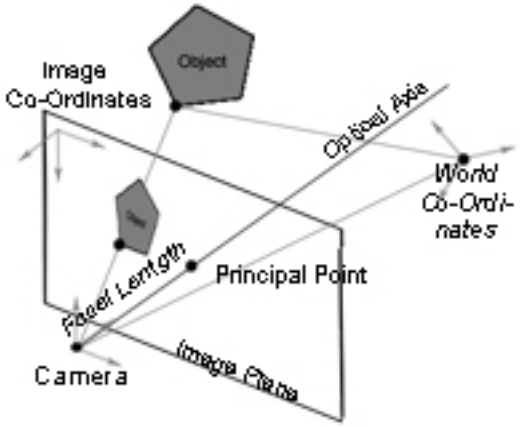

Figure 1: Pinhole Camera Model

2.2 Camera Systems

Multiple image views can be obtained from a variety of different camera setups. The more views of the object available, the more accurate the visual hull will be. A few different ideas for accomplishing this follow.

2.2.1 Multiple-Cameras:

As the name implies, more than one camera is used in this setup. (See figure 2)
This is the simplest method to implement and also the most expensive as it involves using more than one camera. Here, the cameras are placed in different positions relative to the object.

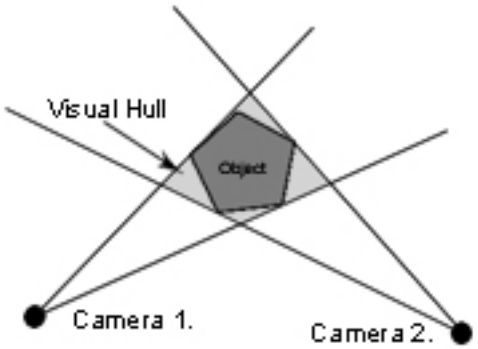

Figure 2: Multiple Camera Setup

2.2.2 Mirror Setup:

The acquisition of images using a mirror setup reduces the number of expensive cameras required in the implementation, while still providing the multiple views of the object. This could increase the amount of software processing. An example of this setup is displayed in figure 3.

An advantage of this method is that in order to improve the estimated upper bound of the object's shape, more mirrors are required, and not more cameras. A possible negative aspect is that more mirrors could also increase the processing requirements in the software, as it would have the effect of increasing the number of virtual cameras.

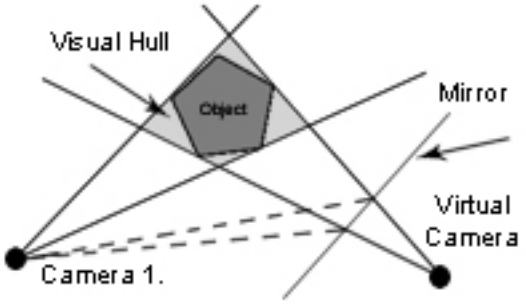

Figure 3: Mirror Setup

2.2.3 Rotation of Object or Camera:

A rotating camera setup is not always practical for most real time systems, but could be more useful in many other cases. This setup can produce more accurate results depending on the number of rotational increments used. The accuracy on the visual hull estimate increases with the number of views.

3 Calibration

In this system, the inputs are not limited to the images alone. A detailed knowledge of the camera's poses and positioning relative to each other is also a necessity. To obtain this information, our system needs to be accurately calibrated. System calibration can be done in many ways, the more commonly used method involving a calibration object.

3.1 Calibration Object

A calibration object can be of any size or proportion (preferably matched to the camera's field of view), so long as its dimensions are accurately known. As an example, a 20-sided calibration object is shown in figure 4. This object was constructed from cardboard. Each face is triangular. The faces of this object were marked using a binary labelling system.

Figure 4: Calibration Object

The near exact dimensions were calculated from many (in excess of 20) image views of the calibration object. The images, the number of sides on the object, and a description of the binary pattern were fed into a program which had as its output the exact dimensions of the input object [2].

3.2 System Calibration

Once the calibration object has been placed into the camera environment, calibration can begin. In initial experiments, two cameras were used (See figure 2. for a description). The images obtained from this environment, along with the known accurate structure of the calibration object, were then passed to a different program which returned information on the relative pose, positioning and focal lengths of the two cameras. The most important output returned by this program is the Rigid Body Transforms which enable the movement between each cameras reference frame to a world coordinate system that includes both the camera's [3].

3.3 World Co-Ordinate System

Once the desired camera setup has been implemented, a detailed knowledge of the environment is required. The important information being the relative pose and positioning of the different cameras with respect to the object and each other. The origin of the world coordinate system in this case is arbitrary. The world reference frame includes all cameras as well as the object. To move from a camera's reference frame to the world reference frame, a rigid body transform is required.

3.3.1 Rigid Body Transform

A matrix where the upper 2×2 matrix is orthogonal represents a rigid body transform. It preserves both lengths and angles of shapes. Note that an arbitrary sequence of rotations and translations is also of this form i.e. it is also a rigid body transform.

4 Image Processing

The previous sections describe the necessary procedure required before an object can be placed into the system. What follows are the tools used to complete the task.
In an experiment, a few points in Matlab were plotted (using plot3). A number of varying posses of an object with points marked on it were generated (see example in figure 5). The position of each point in the image plane was calculated manually (using ginput). These points were translated from the image plane into the camera's reference frame, and then the rigid body transform was applied to move them into the world reference frame.

4.1 Image Segmentation

An image is broken up into foreground (or target) and background. This procedure is known as segmentation. Some methods available to accomplish this include "Watershed" and "Region Growing" [4] segmentation. For our purposes, the target is our object and everything else is background. Although accurate segmentation is an important part of this project, it is not discussed in this paper.

4.2 Back Projection

Once the cameras and the points on the different image planes have been moved into the world co-ordinate system, all that is required is for the points to be projected to where they intersect each other. This is the the position in

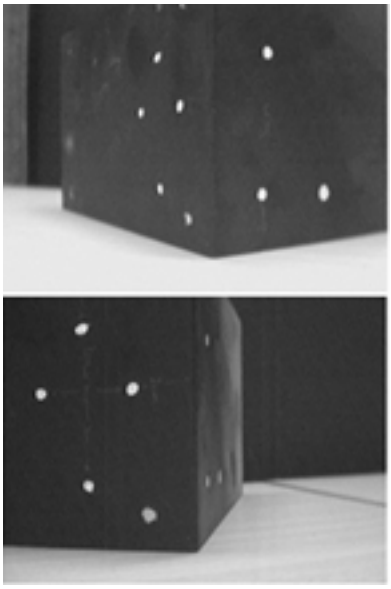

Figure 5: Two views of object points

the world where the points lie. Unfortunately, due to noise and human errors, the lines do not (and never will) actually intersect. Therefore, the point that is exactly between where they are closest is selected.

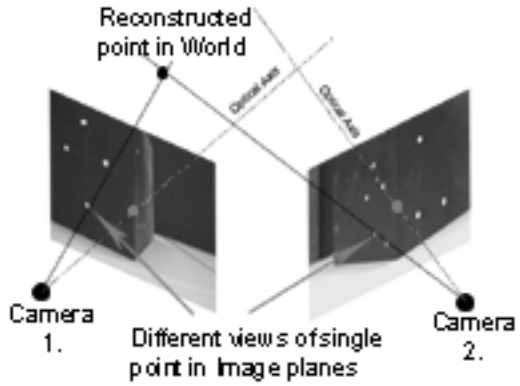

Figure 6: Image points projected into 3D space

Matlab provides tools to represent points in a 3D space (plot3 and scatter3). Standard matrix geometry can be used to project lines, and then compute the region where two projected lines are closest.

4.3 Representing an Object in 3D

There exist many different ways to represent a 3D object on a computer. Matlab is one program that contains different tools for displaying 3D objects. Another option is to use Virtual Reality Modelling Language (VRML) as it is more portable and was specifically designed for this purpose. VRML viewers can be downloaded from the internet, and plug into standard internet browsers. The viewer used to implement the example for this paper is by Cortona Graphics. A Matlab script was used to generate the ".wrl" file which was used by the VRML viewer.
The method implemented in tests used a number of voxels to construct the visual hull.

4.3.1 Voxels

Voxel is short for "volume element", and can be represented by a cube.
There are two ways to go about this. Either split the whole 3D region into some predefined number of voxels and then move through them, testing each element to see if it is part of the target or not.
The other, more efficient, method starts with an initial bounding box and splits it into 8 smaller cubes. Each subsection is tested to see if it is part of the target. The following steps are then followed: If all corners project to inside target, mark voxel as being part of target. If none of the corners are included in the target, leave it unmarked. If some of the corners are found to be part of the target, subdivide the smaller box and start the algorithm again. This is recomputed to some predefined limit. An example of this can be seen in figure 7. The algorithm is $\sum_{n=0}^{l} 8^n$.

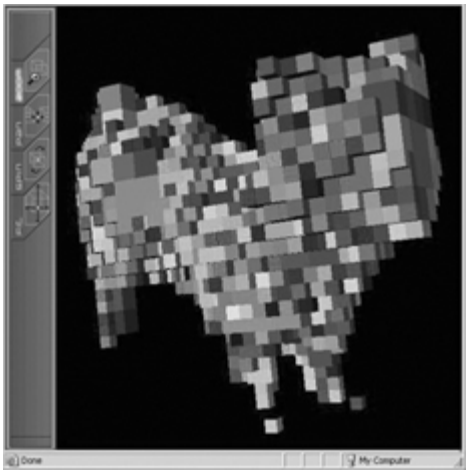

Figure 7: Reconstructed 3D Cat in VRML

This algorithm can become quite complicated to code, but uses a simple and efficient concept. It

does have a shortfall in that some parts of an object might fall within a voxel, but do not touch any corners. A snake like object is one example of this kind of phenomena. The algorithm would need some extra mechanisms to move beyond this obstacle.

5 Conclusion

A description of the ideas and methods involved with generating a 3D model of a 3D object is given in the sections above. This model can be generated from multiple views of the object using one or more cameras in an accurately calibrated system. System calibration was briefly discussed. Computations were implemented in Matlab. The output was displayed using Cortona Graphic's VRML viewer.

6 Future Research

The following areas require further research and study:

6.1 Mirror Camera Setup

To investigate feasibility of using the mirror setup described in a previous section.

6.2 Shape Analysis of Visual Hulls

Images can be classified and sorted according to their shape, and would require some sort of learning mechanism.

6.3 Texture Mapping

Mapping of the texture information from the images to the final visual hull models. This is made easier if the model is represented by many voxels.

6.4 Mex VRML generation

The software used to compute and generate the Visual Hull ran too slowly in Matlab due to the many "for" loops. The accuracy of the voxel method could be improved, but would need to be redone using a "mex" wrapper to speed up the algorithm.

Acknowledgements

I would like to thank TSS Technology, De Beers for all of their support.

References

[1] Ricardo Chavarriaga and Raquel Urtasun, *Camera calibration*, Tech. report, Graduate School in Computer Science, 2001.

[2] 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.

[3] David A. Forsyth and Jean Ponce, *Computer vision - a modern approach*, Alan Apt, 2003.

[4] Phillip Milne, *Pixel based segmentation using region growing techniques*, November 2001, Undergrad Thesis.

[5] E. Trucco and A. Verri, *Introductory techniques for 3-d computer vision*, 1998.