UNDERGRADUATE REPORT

Extracting a 3D Model Using Stereo Vision and a Structured Light Source

by Jason Hammer
Advisor: S.K. Gupta, Edward Yi-tzer Lin

UG 2001-2

3D Modeling

Extracting a 3D Model Using Stereo Vision and a Structured Light Source

Summer 2001 REU Project
By Jason Hammer
August 3, 2001

Advisors
Dr. S.K. Gupta
Dr. Edward Lin

Table of Contents

1 Introduction

1.1 Motivation

Depth recovery has a number of different applications in robotics, manufacturing, and entertainment. The focus for this project was the manufacturing aspect. Specifically, there are a lot of different computerized machining tools that, given the correct input, will cut the parts one needs without the user ever controlling the movement of the bit. This allows for very precise machining. Sometimes it is desirable to know where everything is inside the machine while it is operating or at least some of the key parts. One use for this is as a safety check, in case the machine malfunctions the computer will see when something inside is out of place and can automatically shut down the operation so that nothing gets damaged. Since the machines that do this type of manufacturing are quite expensive, this would be a very useful tool.

1.2 The Idea

If light is projected onto an object, it gets warped around the curves and indentations of the shape. The goal of this experiment was to project a known pattern onto an object and take pictures of what the pattern looked like on the object. Then use the resulting distortion to determine the shape and size of the object.

There are three main techniques for depth recovery right now. There are laser-based techniques, stereo vision techniques that are based on edge detection, and there is the Rainbow Camera, which projects a pattern onto an object and uses the pattern to recover the shape of the object. However, these methods have some setbacks. Some methods are slow, some are not very accurate, they can be expensive, and laser-based techniques may be hazardous to people's eyes. The method that was used for this project is similar to that of the Rainbow Camera, but it is felt that since a structured light source was used instead of a random display we may have the potential to obtain better accuracy than the Rainbow Camera.

Laser-based techniques are really good at determining the distance to a point. The problem is that this can take a long time because a large sample of points must be used to capture the entire shape of an object. In this experiment I tried to show that this technique of using a projected pattern has the potential of being faster than laser-based techniques while still keeping a comparable accuracy level. Lasers can get far more accurate results than using a camera, but if sub-millimeter accuracy could be obtained using this technique then it would be accurate enough for these manufacturing safety purposes. In addition it could be developed further in order to come up with a fast and accurate visual 3D modeling tool.

1.3 Considerations

There are many things that need to be taken into consideration when attempting to process the images taken. For instance, the color quality of the images may not be very clear. In other words, what a person sees as bright green might appear to be dark green to the camera, or green with a combination of red and blue components. These effects can make it difficult to determine what color each pixel really should be, and makes it

necessary to choose a color scheme that is going to be easily distinguishable. There are also issues that are affected by the focus of the camera, lighting conditions, and distortion caused by the camera lens. Each of these issues is discussed in detail later.

2 Overview of Current Techniques

3D modeling is a subject that has been explored for many years and there are three main techniques that have been successful. They are laser scanning, stereo vision using edge detection, and the Rainbow camera which is based on stereo vision, but uses a random light display for pixel correlation rather than edge detection.

2.1 Laser

Laser scanning is a very accurate way to build a 3D model. It works by using a process called Laser Scanning Confocal Microscopy (LSCM). LSCM shoots a laser beam to a beam splitter, which sends part of the beam to a photodetector and the other half goes to an objective lens and is then focused onto the object. The laser beam is then reflected off of the surface of the object and the objective lens directs the beam back up to the photodetector. The two beams that are directed toward the photo detector are then out of phase by some angle. Using the speed of light and the phase shift between the two beams a computer program is able to calculate how much farther the second beam traveled and thus how far away the object is. This process is shown in figure 1.

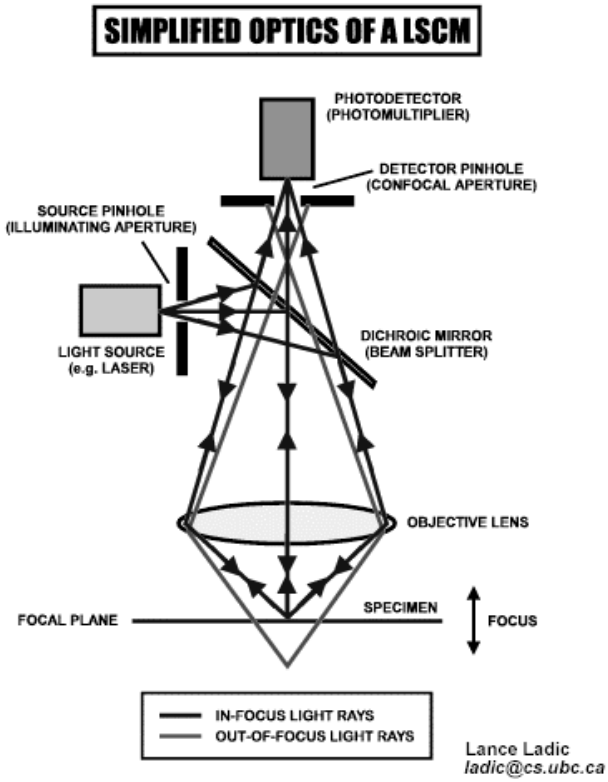

Figure 1: Cycle of a laser beam in a LSCM

The problem with this method is that it is slow. The laser has to scan the object and determine the location of each point on that object one at a time. Of course, the operator does not have to move the apparatus along each point in order to capture the whole image. This would be time consuming and inaccurate. One alternative technique uses servomotors to control the laser, which scans each row one at a time until the whole object has been viewed. There are also other ways of scanning the object to speed up this process but it cannot be done in real time. Another problem with this method is that it is an eye hazard. Lasers are very dangerous if they are directed toward someone's eye.

2.2 Stereo Vision – edge detection

Stereo vision has the possibility to work faster than laser scanning. Even though the images still need to be traversed one pixel at a time all of the data is collected at once. This gives it an advantage over laser scanning. The main focus in developing this process uses edge detection technology. The problem with edge detection is that it is inaccurate. Shadows are one source of error when looking for the edge of an object especially if the edge of the object is dark. The computer does not know where the shadow starts and the object ends. Another problem is that the computer is looking for different light intensities to determine where the edges are. This means that color variations within a region of solid color can cause problems with point correlation between the two different images. This can happen when the light on the object is not uniform, for example if the object is curved the light gets dimmer as you look toward the sides of the object. Once the pictures have been taken and the pixels have been correlated triangulation is used to determine the depth of each point. An example of the output of a stereo vision algorithm based on intensities is shown in figure 2.

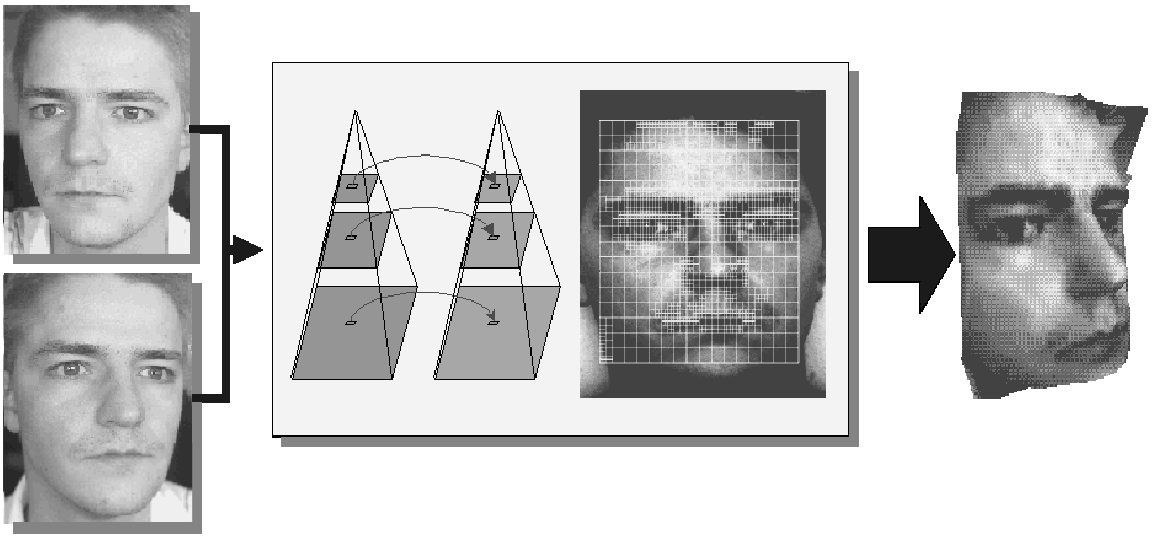

Figure 2: Output of an intensity based stereo vision algorithm

2.3 Rainbow Camera

The Rainbow Camera is a product developed by Genex Technologies Inc. What the camera does is project a pattern onto an object then uses the different colors of the pattern to aid in point correspondence. The pattern is generated by shining a white light onto a prism, which then splits the light into colors resulting in a light array that looks just like a rainbow. The rainbow camera and some of the images that it recovered are shown below.

Figure 3: Picture of the Rainbow Camera

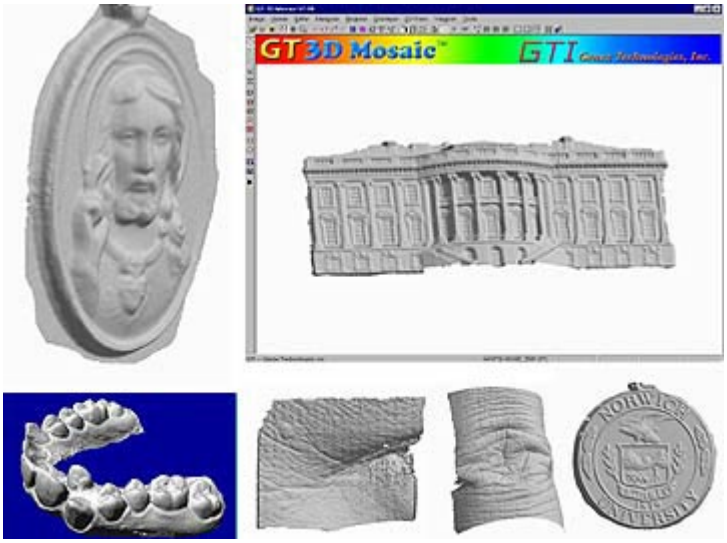

Figure 4: Some 3D images that were recovered by the Rainbow 25™

The images shown in figure 4 are recovered by the Rainbow 25™, which is designed for small objects. Genex Technologies also has some other cameras that can be used to model larger objects such as human heads or torsos. The pattern that is projected by the Rainbow Camera is random, which means that they do not know where the colors should be ideally. This is where this experiment differs from the Rainbow Camera. Since I am using a structured light source I know what order the colors should appear, as well as approximately how many pixels each color feature should occupy in the images.

I also have a finer resolution of color features which should make it easier to sample more points in the images.

3 The experiment

3.1 Prepared Approach

The approach that we wanted to take was to project a known pattern onto an object and use the shifts in the pattern to recover the shape of the object. The technology for this experiment is based on stereo vision, which means that points from two images must be correlated. For this method the color features in the pattern correlate the points. The whole process can be broken up into six main steps and they are:

3.1.1 Calibration Image Acquisition

The first thing that had to be done was to take the camera calibration pictures. These would be used later to calibrate the cameras. Calibrating the cameras gives their location and rotation, these parameters are known as the extrinsic parameters of the cameras. The cameras in this experiment were calibrated by placing eight dots onto a white background, four of these dots were placed directly onto the backdrop and the other four were raised away from the back. This allowed for more accurate depth perception. Then a coordinate system was arbitrarily assigned with its origin at the lower left corner of the backdrop, this is called the World Coordinate System (WCS). Then each of the dots were numbered and their locations in the WCS were determined. A white light was projected onto the pattern and pictures of this scene were taken from two cameras, one on each side of the projector. Then the backdrop was moved to various angles and distances from the cameras and pictures were taken at each of these locations. Once these pictures were taken it was important not to move the cameras at all, because any movement would require them to be recalibrated.

3.1.2 Data Image Acquisition

Next, the calibration backdrop was replaced with a plain white background and was placed perpendicular to the direction of the projector. An object was placed in front of the board and a pattern consisting of red, blue, green, and white squares was projected onto it. Once all of the data was collected we were able to start processing the images.

3.1.3 Image Processing

Processing the images was broken up into two steps. First, color correction was necessary because the color quality and resolution of the cameras was not very fine. Then each square from the projection pattern as seen by the cameras was classified by the row and column that it came from in the original pattern. The original pattern then became the link between the two images, so it was known which set of pixels in the right camera corresponded to which set of pixels in the left camera. At this point the only thing left to do before calculating the 3D locations was to calibrate the cameras.

3.1.4 Camera Calibration

Calibrating the cameras gives the extrinsic parameters with respect to the WCS. This process was also broken into two steps. First, find the eight dots in the images, number them in the order that was assigned previously, and store the pixel location of the center of each dot. Then, matrices consisting of the 3D locations of the dots that were measured earlier and the 2D pixel locations of the dots were supplied to a MATLAB program dots along with some intrinsic parameters of the camera. The output was the extrinsic parameters.

3.1.5 3D Point Recovery

The following is a list of the equations used for the depth recovery and a description of the variables.

- dx and dy are the CCD dimensions
- m and n are the pixel dimensions in the x and y directions respectively
- px and py are the pixel locations on the CCD
- u, v, and w are the coordinates in the camera coordinate frame
- f is the focal length of the camera lens
- x, y, and z are the 3D locations in the WCS

Equation 1.1 converts the from the camera coordinate system (u,v,w) to the pixel coordinate system and equation 1.2 is equation 1.1 expressed symbolically.

$$\begin{bmatrix} w*p_x \\ w*p_y \\ w \end{bmatrix} = \begin{bmatrix} -\frac{f*n}{dx} & 0 & n/2 \\ 0 & \frac{f*m}{dy} & m/2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (1.1)$$

$$P = A*U \quad (1.2)$$

The output from the MATLAB camera calibration program gives the matrix for converting between the WCS and the camera coordinate system. This matrix is called R. Combining these two steps gives a conversion between the WCS and the pixel coordinate system.

$$M = A*R \quad (2)$$

The conversion between the coordinate systems is then given by equation 3.

$$P = M * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3}$$

Multiplying these matrices together gives 3 equations, one for each variable in P. If the first two equations are divided by the third and then the x, y, and z terms are collected on one side, the following system of equations results. Notice each camera results in only two equations and that is why two cameras are needed for depth recovery.

$$\begin{bmatrix} p_x^l * m_{31}^l - m_{11}^l & p_x^l * m_{32}^l - m_{12}^l & p_x^l * m_{33}^l - m_{13}^l \\ p_y^l * m_{31}^l - m_{21}^l & p_y^l * m_{32}^l - m_{22}^l & p_y^l * m_{33}^l - m_{23}^l \\ p_x^r * m_{31}^r - m_{11}^r & p_x^r * m_{32}^r - m_{12}^r & p_x^r *_{33}^r - m_{13}^r \\ p_y^r * m_{31}^r - m_{21}^r & p_y^r * m_{32}^r - m_{22}^r & p_y^r * m_{33}^r - m_{23}^r \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} m_{14}^l - p_x^l * m_{34}^l \\ m_{24}^l - p_y^l * m_{34}^l \\ m_{14}^r - p_x^r * m_{34}^r \\ m_{24}^r - p_y^r * m_{34}^r \end{bmatrix} \tag{3.1}$$

$$G * H = I \tag{3.2}$$

Since G is not a square matrix, equation 3.2 must be solved for H by finding the pseudo-inverse of G.

3.1.6 Displaying the Model

Once the 3D points were collected they were sent to a program called Power Crust, which connected all of the dots creating a set of polygons. The output of this program was then used by an open GL viewer, which displayed the data that we extracted from the images. This would give visual verification that the process was working as it was supposed to.

3.2 Equipment

The equipment used for this experiment consisted of

- 1 - LCD projector
- 2 - laptop computers
- 2 - QuickCam Pro™ digital cameras
- 3 - poster boards
- 2 - plastic feet with notches cut for holding the poster board up
- 2 - (measurements) plastic cubes
- 2 - (measurements) plastic cubes
- 8 - black dots (5mm radius)

- 2 - boxes used to elevate the cameras
- paper used to fold into shapes.

3.3 Setup

First I created a projection pattern using MS Power Point. The pattern contained 84 columns and 72 rows. The first row contained a 4 color pattern: green, red, blue, and white; this pattern continued to the end of the row. The second row had the same pattern, except it started with blue. This two row pattern continued down to the bottom row. Then I connected one camera to each laptop and connected the LCD projector to the laptop that contained the projection pattern.

12 lines were drawn on one of the boards, there were 3 sets of four lines. One set had straight lines perpendicular to the direction of the projector. The lines of the second set started at the same point as those in the first but were at an angle of 15.017 ° with the first set. The lines of the third set ended at the same point as those in the first but were at an angle of –15.017 ° with them. Figure 5 shows the orientation of the lines.

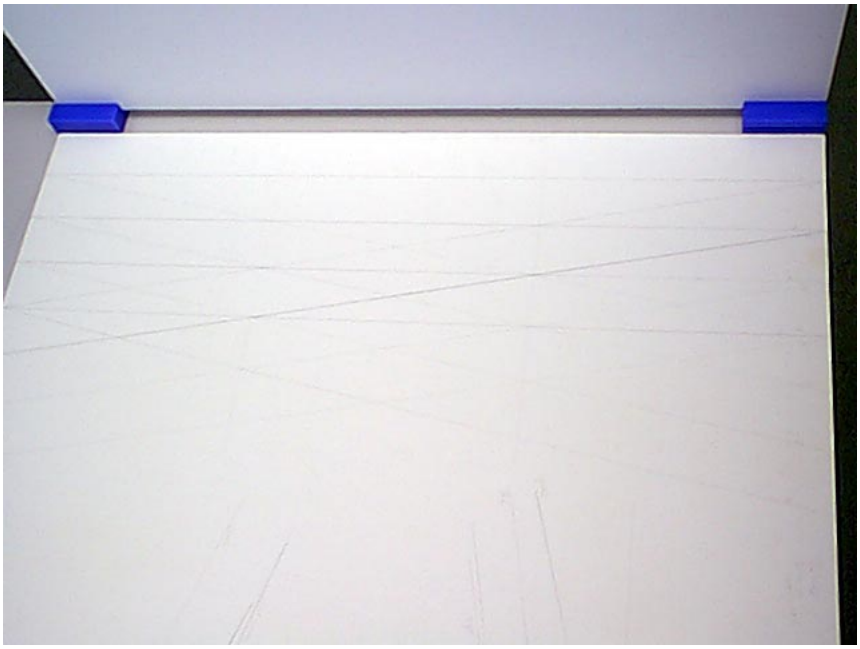

Figure 5: Layout of the lines that the calibration background was aligned with

On the next board I drew eight x's where the calibration dots were going to be placed and measured their x, y, and z values in millimeters with the origin at the lower left corner of the board. The x's were placed in a pseudorandom order, in the sense that four were placed on the left side of the area that the objects were going to be placed and the other four were on the right. Then I covered the cubes with white paper and placed a calibration dot in the center of the topside of each cube. The centers of the cubes were placed on the x's on the right side and the remaining four dots were placed on the x's on the left. Figure 6 shows this setup.

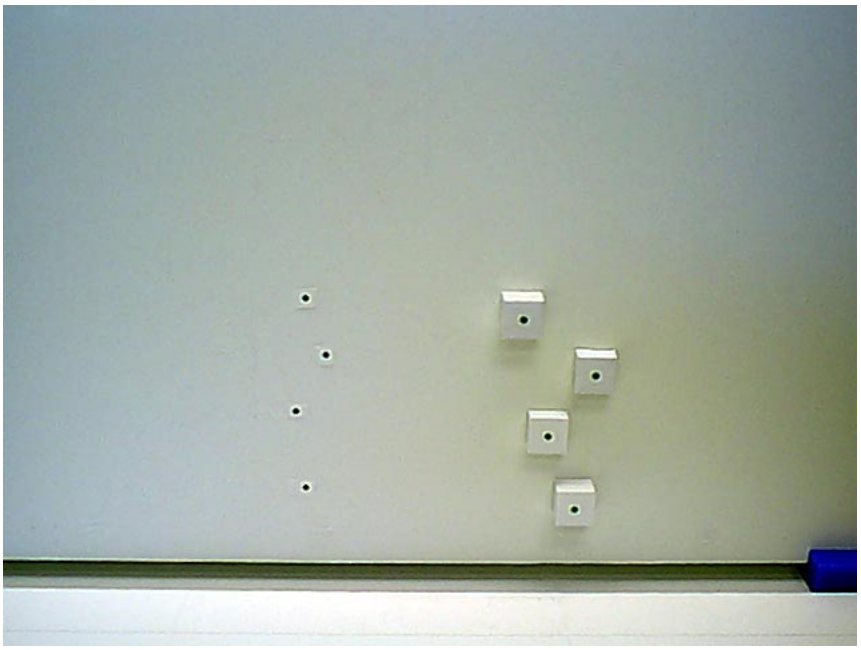

Figure 6: Orientation of the calibration dots

Then I folded some shapes out of paper. The shapes that I used had only straight edges, so that I didn't have to deal with curves and minimized the problem of features being hidden from the cameras. These shapes are to prove the concept and get something that works, later on it would be more feasible to make cones, cylinders, and other non-linear objects to test the procedure.

The plastic feet were to be used the hold up the board with the calibration dots and the remaining board. The boards can be interchanged as needed by slipping them in and out of the grooves in the plastic. A picture of the complete setup is shown in figure 7.

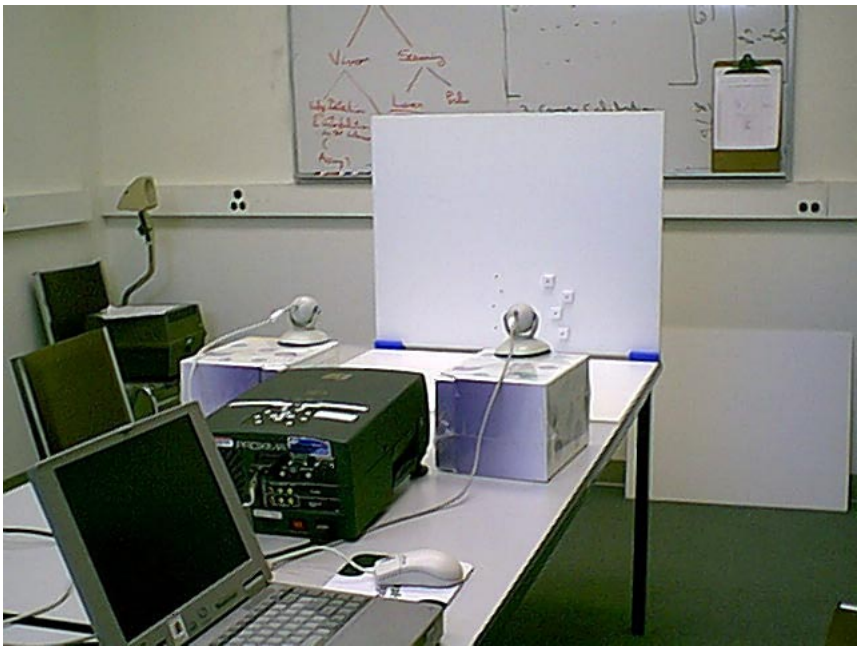

Figure 7: Layout of the experimental setup

4 Results

4.1 Successes

I was successful in every step of the process by varying degrees. Most of the attention for this project went into image acquisition and processing. This is because I was starting from scratch and the results from these two steps needed to be somewhat accurate before I could even attempt to move on. There were a number of unexpected problems encountered throughout these processes that needed to be resolved. Some of these problems include:

1. Poor color quality
2. Low resolution
3. Shadows
4. Viewing the object at an angle caused horizontal lines to appear at an angle

Image distortion caused by the curvature of the lens was another problem that was not taken into consideration for this experiment. Since this phenomenon usually occurs more around the edges of the image, I kept the objects of interest near the center of the image to minimize the error. This is a problem that many people have researched and there are solutions readily available on the Web. The successfulness of the different steps will now be discussed in more detail.

4.1.1 Image Acquisition

Capturing the images was one of the more successful steps in this experiment, but it still could have been better. Improvements could be made on this step by putting more time into a lighting scheme that produces optimal contrast in the images. In addition, the cameras could have been set to a higher resolution. In this experiment the image sizes were 320x240 but the cameras were capable of capturing images at 640x480. Also there are many cameras available which will give a higher resolution as well as better color quality. These improvements would drastically improve the quality of the final 3D model, since each of the remaining steps is affected by the image quality.

4.1.2 Image Processing

The image processing could have also been better. The variables for the color correction method in my program were all hard coded. This means that the program would only work properly for the images taken with the same cameras, under the same lighting conditions, and with the same image capture options (color intensities, contrast, brightness, etc…). This process will definitely want to be automated somehow and the way that seems the most feasible is the following. Allow for user input to determine what colors to look for. Letting the user enlarge an image and select all of the different pixels that he wants to be classified as one color could do this. The user could then repeat the process for each of the colors and also specify the order of the pattern. The program could then use the RGB intensities of the pixels selected and the known order of the pattern to statistically determine what color each pixel is in the rest of the images.

The algorithm used for correcting for the shadows works pretty good for images with straight vertical edges, but would not work at all if the edges were at an angle. A description of the algorithm is given in appendix 7.1.2. Essentially what it does is determines the feature number of the square that is right next to the shadow on the top row of the object. Then it assumes that all of the other features right next to the shadow came from the same column.

Once the color has been corrected and the shadows have been taken into account the feature classification is pretty solid. The features were numbered by the column and row that they came from in the original object. The only mistakes that it makes are cascaded from errors in the color and shadow corrections. A detailed description of this process is shown in appendix 7.1.2.

4.1.3 Camera Calibration

The intrinsic parameters of the camera were needed for the calibration process. This information is generally not given in the camera's users manual so they have to be determined by contacting the company. Unfortunately, Logitech would not dispense this information because apparently it is confidential. In order to work around this problem I estimated what the values were based on information that I could gather from other digital cameras. The output of the calibration data may not be entirely accurate according to the defined WCS, but some of the error might be alleviated through the process of recovering the 3D locations. This is because the intrinsic parameters are also used in these equations and they should cancel each other out as long as the values stay consistent. More research would need to be done to verify this theory and due to time constraints it was not done in time to be included in this report. It is also probable that these parameters could be found by further research on the Internet or even by taking apart one of the cameras and looking for part labels inside. In general, camera calibration is the weakest step in the process at this point and attention should probably be given to this area first for further research.

4.1.4 3D Point Recovery and Display

Recovering the world coordinates is pretty good. The output from this process on the other hand is not as accurate as I had hoped. The recovered 3D model does slightly resemble the original object, but the edges are very rough. The equations used for the depth recovery could not be improved upon very much since their derivations are based on the geometry of the camera and conversions between different coordinate systems. If all of the above improvements were made, better results should be obtained.

4.2 Accuracy

Up to this point in the research the accuracy is unknown. In order to calculate the accuracy steps first need to be taken to examine the data and determine whether the recovered 3D points makes sense according to the known WCS. Then the size of the recovered object must be determined and compared to the actual size of the original object. Once this is done the error can be determined. Unfortunately eight weeks was not enough time to be able to get to these debugging issues and this will need to be looked into by someone else who is continuing this research project.

4.3 Speed

The original idea for this project was to have a streaming video feed from two cameras fed to a program that would recover a 3D model of the images in real time. Right now the program is able to process the images in about 5-6 seconds, and once the calibration is complete the 3D point recovery takes about another 5 seconds seconds. This is obviously too slow to be considered real time and on top of that it takes Power Crust quite a bit longer do generate a 3D model from the point cloud. The Power Crust step could be skipped if a program were going to analyze the raw data. There are some problems with this analysis, the accuracy of the procedure needs to be determined because if the output is not valid data it may turn out that it will take even longer to extract accurate points. Another problem is that the color features used in this experiment were somewhat large which limits the resolution that is possible for the 3D model. If smaller squares were used in the pattern, the image processing would probably take even longer yet. This procedure is however faster than some laser techniques that can take minutes to render an image.

5 Future Research

The underlying framework for the 3D image recovery is now setup but more work needs to be done in order to refine the final output. The first thing that needs to be done is to analyze the output from the camera calibration code and determine how accurate it is. If the results are not suitable, more effort should be put into obtaining the intrinsic parameters of the camera or if necessary a new program should be used for the calibration. Then more work needs to be done for the shadow correction so that more interesting objects could be examined. Another source of error is Power Crust because of the way it generates a viewable image file. There are other more powerful programs, which do more interpolation and optimizing. This will give a better visual image of the 3D model that was created, since the point cloud is not very dense. These are the main parts of the process that need work right now. Once they are finished the code should be modified so that the hard coded variables are either assigned automatically or by user input. For example, the corners of the projection pattern and range of RGB intensities to be classified as a certain color are hard coded right now. After these steps are complete all parts of the process could be tweaked in order to get better and/or faster results.

6 Conclusion

In conclusion, there were a lot of interesting problems that needed to be solved for this experiment. I believe that the choice of colors for the projection pattern is the best possible combination because the differences in the RGB intensities are maximized. The color quality, resolution, and focus of the cameras could be improved by simply using a higher quality camera. There is not a whole lot more that can be done to improve the quality of the images from the QuickCam Pro™. The environment lighting that was used

for taking the pictures was not monitored very closely and is something that may be looked into for further improvements but it is not expected to make a large difference especially if higher quality cameras were used. Processing the images was full of challenges most of which were solved with some confidence. Once this is done, all of the data is captured and the only thing left to do is perform some calculations on the values. Camera calibration code is available on the web, I chose to use the MATLAB code because it seemed to be the easiest to implement. However, better results may be obtained by looking for another package. The actual implementation of the MATLAB toolbox can be found in [6] and this can be compared with other resources available. Recovering the 3D coordinates after everything else is done turns out to be a very simple calculation.

The patent document for the Rainbow Camera [**5**] discusses other methods by which to display a pattern onto the object. One of these ways is using an LCD projector to project a pattern onto the object for point correspondence. This is exactly the same idea that was being looked into for this experiment but as far as I can tell Genex Technologies does not actually use this procedure for any of their products. It would be interesting to visit their site and talk with someone about what they do and how they solved similar problems that were encountered during this experiment.

7 Appendix

7.1 Code algorithms

7.1.1 Color Correction and Noise Reduction

Color correction was done by manually finding out the RGB intensities for different shades of red, green, blue, and white. This was accomplished by zooming in on a picture and sampling pixels of a certain color and writing down their color components. After about 5 to 10 pixels have been looked at for each color, different ranges of the intensities were classified as the different colors. These ranges can be thought of as volumes in the color cube shown in figure 8.

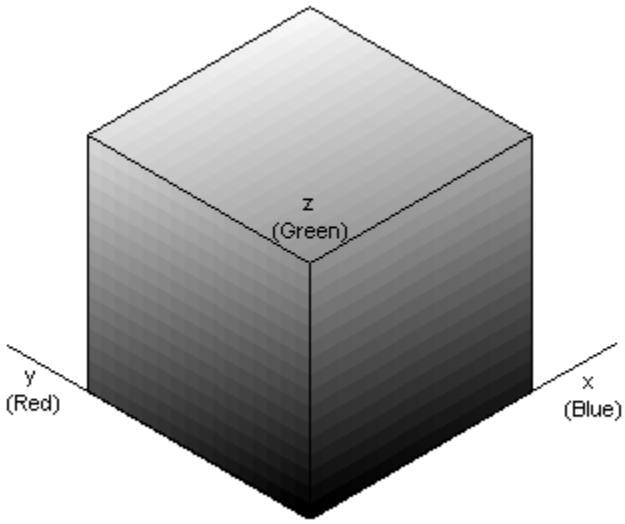

Figure 8: Color Cube

Note: Make sure that none of the regions overlap.

Noise reduction was done by locating the four corners of the projected pattern and making everything outside of the pattern yellow. Yellow does not have any significance it was chosen because it was not a color used in the projection pattern. The four corners of the pattern at this point are hard coded.

The program flow is as follows:

1. Determine color of unambiguous pixels and store the absolute values in a modified data array. Ambiguous pixels are set to color "UNKNOWN".
    a. Background
    b. Shadow
    c. Pattern features
2. Second run through image is through the modified data array to fill in all of the unknowns.
    a. Use the known pattern to determine what each pixel color should be. The pattern does not appear as straight lines so there are a number of different cases that can result in a certain color. They are:
        i. Is pixel on the edge of the pattern
        ii. Is pixel next to a shadow
        iii. If not i and not ii then determine based on the color of the pixels above and to the left of the unknown

7.1.2 Feature Classification and Shadow Correction

Feature classification and shadow correction were done simultaneously. The program starts in the upper left corner of the projected pattern in the modified data image and continues left to right, top to bottom. The following process makes the classification:

1. Feature (0,0) ➔ (row, column) gets color of pixel in upper left corner of pattern
2. Check the pixel to the right
    a. If same color → Add pixel → go to 2
    b. If different color, go to 3
3. Check if this is the edge of the shadow – If color is black and y is the same row as the top left of the shape. This is the shadow – go to 3a. Otherwise go to 4.
   NOTE: Top left corner of the shape is hard coded. Code should be added to automate this
    a. Go down 3 pixels
    b. Continue moving right until the pixel color is not black.
    c. Move up to the first pixel below the shadow.
    d. Check its color
        i. If same as the feature color right before the shadow was found.
            1. Clear all pixels in this feature – Don't care about features that are not on the shape
            2. Add this pixel as the first pixel in the feature.
        ii. Else If color is different → Create a new shape in the same row but next column
        iii. Whatever column this pixel was assigned to is now the furthest left feature that is going to be defined. In other words, when the end of the row is reached, the start of the next row will be in this column.
        iv. Go to 2.
4. Check the pixel diagonally down to the left
    a. If same color → Add pixel → go to 2
    b. If different color, go to 5
5. Check pixel directly below
    a. If same color → Add pixel → go to 2
    b. If different color, go to 6
6. Check the pixel diagonally down to the right
    a. If same color → Add pixel → go to 2
    b. If different color, go to 7
7. Find pixel that is the top left of the next feature
    a. Move to the top right pixel of this feature.
    b. Check pixel that is diagonally up to the right
        i. If color of next feature → Create new shape in the same row but next column → go to 2
        ii. If different color, go to 7.c
    c. Check pixel that is to the right

    i. If color of next feature → Create new shape in the same row but next column → go to 2
    ii. If different color, go to 7.d
d. Check pixel that is diagonally down to the right for the same color
    i. If color of next feature → Create new shape in the same row but next column → go to 2
    ii. If different color, go to 7.e
e. Move to the right most pixel in the second row.
f. Check pixel that is diagonally up to the right
    i. If color of next feature → Create new shape in the same row but next column → go to 2
    ii. If different color, go to 7.g
g. Check pixel that is to the right
    i. If color of next feature → Create new shape in the same row but next column → go to 2
    ii. If different color, go to 7.h
h. Check pixel that is diagonally down to the right for the same color
    i. If color of next feature → Create new shape in the same row but next column → go to 2
    ii. If different color, go to 7.i
i. Move to next row by moving to the pixel in the lower left corner of the first feature in this row. The first feature will not be column 0 once the shadow is found, it will then be what ever column the first feature after the shadow was assigned to.
j. Check the pixel diagonally down to the left
    i. If color of next feature → Create new feature in the next row and the "left column"→ go to 2
    ii. else, go to 7.k
k. Check pixel directly below
    i. If color of next feature → Create new feature in the next row and the "left column" → go to 2
    ii. else, go to 7.l
l. Check the pixel diagonally down to the right
    i. If color of next feature → Create new feature in the next row and the "left column" → go to 2
    ii. else, go to 7.m
m. Once it gets here, the bottom row has been completed and the feature classification is done.

References

[1] Applied Research Associates NZ Ltd.
http://www.aranz.co.mz

[2] Banerjee, Prashant and Zetu, Dan. *Virtual Manufacturing.* John Wiley & Sons Inc., 2001.

[3] Cyra Technologies: Cyrax 3D Laser Scanner
http://www.cyra.com

[4] Geng, Jason. “Color ranging method for high speed low-cost three dimensional surface profile measurement”. U.S. Patent, 1997.

[5] Geng, Jason. “High speed three dimensional imaging method”. U.S. Patent, 1997.

[6] Heikkilä, J. "Geometric Camera Calibration Using Circular Control Points". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 10, Oct 2000.

[7] Jones, Richard. *Introduction to MFC Programming with Visual C++*. Prentice Hall PTR, 2000.

[8] Genex Technologies Inc.
http://www.genextech.com/home.html

[9] Robert, Luc and Deriche, Rachid. “Dense Depth Recovery From Stereo Images: a Minimization and Regularization Approach”.
http://www-sop.inria.fr/robotvis/personnel/der/Demo/Stereo/stereo.html

[10] Robert, Luc and Deriche, Rachid. “Dense Depth Map Reconstruction: A Minimization and Regularization Approach which Preserves Discontinuities”
http://www-sop.inria.fr/robotvis/personnel/der/Demo/Stereo/stereo.html