3D SCANNING BASED ON COMPUTER VISION

By

Hansen Chen

Xiaobo Dong

Xingqian Xu

Final Report for ECE 445, Senior Design, Spring 2012

TA: Mustafa Mir

01 May 2012

Project No. 40

Abstract

We designed and built a 3D scanning device based on computer vision techniques. This 3D scanning device is able to automatically scan the object no larger than 6'x6'x6', and it is able to reconstruct the object's 3D model. Although we successfully reconstruct the shape of the object in cubic mesh, the 3d module does not appear smooth and nice. Further development of this device would allow us to represent the object in a triangle mesh form. This document details the design procedure and results of this project.

Contents

1. Introduction

1.1 Purpose

A 3D scanner is a device that analyzes a real-world object to collect data on its shape and appearance. Then the collected data can be used to reconstruct digital three-dimensional models. The reconstructed 3D model is useful for a wide variety of applications, especially for the online retailing industry and the gaming industry. There are plenty of technologies available now for digitally acquiring the shape of a 3D object. Most of the well-known 3D scanners are active scanners, which emit some kind of radiation and detect its reflection in order to probe an object. The active scanners are precise but expensive, because most active scanners are laser-based. Our project aims to make a passive 3D scanner with web-camera. The web-camera records the video of the object in 360 degrees, and then the video will be processed through computer vision algorithms to reconstruct the 3D model. The reconstructed 3D model will be in 3ds format.

1.2 Project Functions and Features

- Easy installation and user-friendly
- Do no need to buy expensive laser scanner
- Ability to reconstruct the 3D model with simple web-camera
- Can represent an art craft in more detail than 2D image
- Great for retailers to display merchandise online
- Rotate platform with speed control to maintain the object stable

1.3 Block Diagram

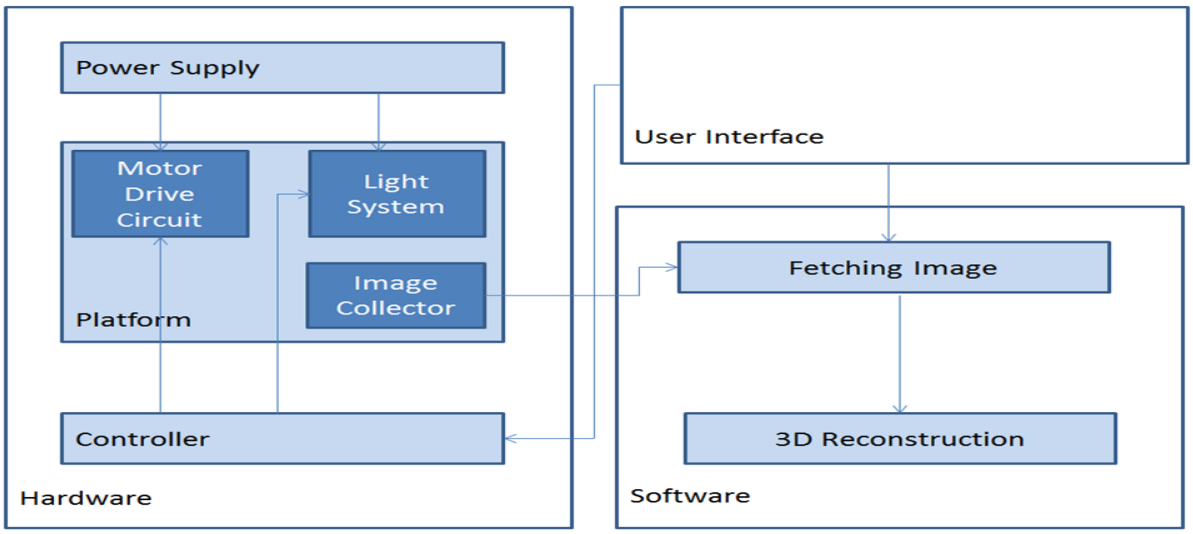

Figure 1: Block diagram

2 Design Procedure

2.1 Power Supply

The power supply was designed to supply +4V to the DC motor, +3V for the LED light and +5V for the web camera. The power sources we intend to use are AA batteries and power coming out of a USB cable that connects to a computer. To make the video processing easier, we want the platform rotates at constant 2RPM. Through some testing on the gear-head motor, we found that at 4V, the platform runs at 2RPM exactly. We considered using the USB power to supply the motor. However, the motor barely runs even when directly connect to the USB power. Thus we decided to use AA batteries to power the motor and the LEDs. In addition, AA batteries are cheap and easy to obtain, which also match the portability feature of our 3D scanner. The web camera is connected to a computer through a USB cable.

2.2 Light System

The light system must be able to create a constant white background for the scanning object. Thus its light must have enough illumination but also at the same time does not consume too much power since we only have 2 AA batteries. The LEDs also need to be cooled properly.

2.3 Motor Drive Circuit

The Motor drive circuit is a dc boost converter that boosts the 3V AA batteries voltage up to constant 4V to drive the motor. The circuit needs feedback feature because the battery voltage will drop with continues operation and the load would also change since the object being scanned on the platform will have different weight. The circuit must have high efficiency.

Originally, we plan to use the TPS 63000 dc converter chip from Texas Instrument. But the two sample chips we got never operate as expected until we found a design online that the power pad of the chip is connected to the ground. Then our chip worked for 10 seconds and dead. So we built our own boost converter on a breadboard. And at the same time, we ordered two more TPS 63000 chips from TI. As the boost converter we build worked well, we did not go back to use the TI chip. But we figured out why the chip never worked. After some careful examination on the chip and the adapter that the chip was soldered onto, we found that the grounding pins on the chip are not well connected to the adapter. If the pins are soldered properly to the adapter, the chip should work.

2.4 Image Collector

The Image Collector needs to record the video of the object. Our initial design was to record the video using commercial software. Even though the performance was pretty good, it was hard to fulfill our automation task. In order to have a good automation, we implemented record function in MATLAB.

2.5 Fetching image

The image fetching part requires a video of the scanning object as input. The video contains the 360 degrees projection view of the object in a dark surrounding with constant white light background. The fetching image part would generate 36 frames (10 degree each) of the object.

In our initial design, the fetching image part contains the following steps:

A. Segment each frame using intensity threshold method.
B. Find the angle for each frame
C. Output the 36 frames.

First, we tested our segmentation method on our initial platform design, but we realized that the background was not as perfect as we thought. There was lots of noise around upper boundary of the image after segmentation. Though we tried to improve the lighting condition, we found that some noises were inevitable, which could badly affect the reconstruction process. In order to reduce the image noise, we implemented a size filter to clean out small noises. Since most of the noises were small in area and separated from our object, a size filter could reduce these noises significantly.

Second, as our test went on, we realized that it was hard to figure out the angle for each frame in our initial design. With this goal in mind, we chose to add some markers below the platform. The reason why we put marker below the platform was to avoid affecting the object image, and at the same time we could calculate the angle by finding the location of the markers. With the help of the marker, we figured out a good method to calculate the angle for each frame by some geometry theorems.

Third, when we calculated the angle for each frame by the markers, we met one special case. When the platform rotated to some particular position, the markers were overlapped from camera view. Therefore, we could not figure out the position of the marker. In this case the angle was not correct. This kind of situation was inevitable, and we did not prefer it to appear in the first frame. Therefore, we needed to find a good frame as our first frame, otherwise the angle difference error would accumulate and the reconstruction result would be distorted.

Forth, after those modifications, we realized that the operation on all the frames in the video took a long time. In order to reduce the running time, we decided to do the operation for each frame in marker region only. In this case, we needed to find the boundary of the image correctly. After that, the algorithm in the program ran much faster.

Overall, the fetching part contains the following steps:

A. To find image boundary
B. To do segmentation on the marker region
C. To find a good first frame
D. To find the angle for each frame
E. To do segmentation on the selected frames
F. To clean the segmentation result

2.6 3D Reconstruction

3D reconstruction is the core part in this project. It is also the last step of this project. The 3D reconstruction has various parameters as input:

- Segmented image data set
- Number of images
- Resolution
- Camera distance
- Camera view angle

Its output is a .3ds file which contains the 3d model of the scanning object. The algorithm for the reconstruction can be separated in four steps: First step is to calculate object's possible space position and generate a 3D matrix in each frame. Such 3D matrix contains only 0 and 1. 0 stands for empty space position and 1 stands for occupied space position. Second step is to rotate the 3D matrix to its corresponding angle value. The angle value comes from the image angle value which is linear to the image index. Third step is to combine all 3D matrixes into one single 3D matrix. This final 3D matrix contains the exact space position of the scanning object. The rule of generating this last 3D matrix is: if a certain location is marked as 1 in every 3D matrix, the value of this location is marked as 1 in the final matrix. Otherwise it is marked as 0. Last step is to write a 3ds file containing the final matrix.

In our reviewed design, we find out that several inputs are not changeable. Camera distance and angle from the platform is not permitted to change because of the fixed position of angle detection markers. Though these parameters still can be modified in software, these values are fixed inside the program. The idea that let user input resolution parameters is also impractical. When running our program, resolution higher than 100x100x100 will result in the running time more than two minutes. Also Matlab has limitation on the element number in matrix variable. Therefore, our team decides to fix the resolution to 50x50x50. Furthermore, the number of images is also set to 36(10 degree each). Inputting image set larger than 36 would definitely be unnecessary, because higher angular resolution cannot be displayed in low resolution matrix. On the contrary, using fewer images may cause object model twisted.

Our approach in the reconstruction algorithm will be insensitive to concave surfaces. Applying different lighting conditions such as 30 or 15 degree strong light may help to detect concave surface according to the shadow. However, analyzing shadows will need higher level computer vision technique together with a more complex algorithm. Meanwhile, collecting data will take much more time. As one of the main objectives of this project is for family use, this algorithm will tolerate some level of inaccuracy and focus more on fast algorithm and friendly user interface.

2.7 User Interface

The purpose of our user interface is to transmit control signals and input parameters to our project hardware. In our first design, user interface is either software integrated or hardware integrated. Controlling signals sent to hardware include start and stop signals to light and motor. Input

parameters include camera ID number, model resolution and camera resolution. The signal transmitting routine is through USB 2.0 port. When software interface is applied, we decide to use Matlab GUI. Interface integrated on hardware will be switches and buttons.

In our reviewed design, we decide to provide user both software and hardware interface. Hardware interface is a switch on one side of the motor box. Such switch controls the power supply to turn on motor and LED lights. Software interface is generated by Matlab GUI. It has a start button for camera recording. Once it start recording, image data will be sent from camera to PC through USB port. Secondly, camera ID is the only Input parameter on our GUI. The reason why we discard other inputs is because we are using fixed camera and model resolution. Lastly, we add a "simple test" section on the user interface. This section links with four existed test case, from which user can run reconstruction program directly. This section is also very helpful for debugging.

3. Design Detail

3.1 Power Supply

2 AA batteries are used to supply electricity to the LED lights and the motor. The web camera is connected to a computer through a USB cable.

3.2 Light System

We found five LEDs that have already been mounted in a row on an aluminum plate from the lab inventory. The picture can be seen in diagram 1 in Appendix B. The five LEDs are connected in parallel to the batteries through a simple flip switch. The reason we did not connect the LEDs to the output of the boost converter circuit is because at 4V, the LEDs consumes 10 times more power compare to 3v. The actual power consumption can be seen in table 1 below. Besides, at 3V the LEDs can deliver enough illumination to create a constant white background for the video recording process.

Input (V)	LED Current (A)	Motor Current (A)	Total Current (A)
3	0.6	0.1	0.7
4	4.3	0.1	4.4

Table 1: Power consumption on LED and motor

3.3 Motor Drive Circuit

As mentioned in the design procedure, the TPS 63000 Texas Instrument DC converter chip did not work at first, so we built our own boost converter. The circuit diagram can be seen in diagram 2 in the appendix B. We used a Texas Instrument MSP430G2131 microcontroller to generate the PWM switching signal for the NTE 2980 logic level MOSFET. The microcontroller's input voltage is from the USB cable that connects the camera to the computer. We connect the 5V USB power though two zener diodes to V_{cc} of the microcontroller chip because we need a stable reference voltage so that we can compare it with the motor voltage for feedback purpose. Since for a boost converter, the output and input voltage has a relation as shown in equation 1 below.

$$V_{out} = V_{in} \times \frac{D}{1-D} \qquad (1)$$

For 3V input and 4V output, the duty ratio (D) should be 25%. We compare the actual voltage on the motor to the reference voltage which is the also the V_{cc} for the microcontroller, 3.6V, after two zener diode voltage drop of 1.4V. We programmed the microcontroller so that if the actual voltage is lower than 4V, the PWM switching duty ratio will be decreased; and if it is higher than 4V, the duty ratio will be increased until the motor voltage stays at 4V. We also have duty ratio boundary on the microcontroller so the boost voltage will not go sky high. The circuit performance waveform can be seen in diagram 2 in the appendix B. We can see that when the battery voltage is at normal 2.85V or low 1.56V, the motor voltage stays very close to 4V.

Also for open circuit protection, we have a 1.5 V ballast resistor at the output end.

3.4 Image Collector

The image Collector uses the Image Acquisition Toolbox software in MATLAB. We use an immediate trigger method. First we create a video object linked to the camera. With the immediate trigger method, the video object executes the trigger immediately after we start the video object running with the start command. In our design, one frame per trigger is enough. In order to record complete video, our camera needs to record one minute, which is approximately 1440 frames. Therefore, our parameter: trigger repeated, is set to 1440.

3.5 Fetching image

Our Fetching image part contains the following steps:

A. To find image boundary

Since the background image is stationary, the boundary position is stationary. In order to find the boundaries for all the frames, we just need to find the boundaries for the several random sample frames and get the average of them. First, we do the segmentation on several random sample frames. It would output a binary image. The image bottom boundary could be detected when the bottom boundary is between half height and full height. The criteria of the bottom boundary is the first row has black pixels more than one forths of the width. Another critical boundary value is the platform height. The platform could be detected when the number of black pixels are nearly equal to the width in the middle region. An example is shown in the figure 2 below.

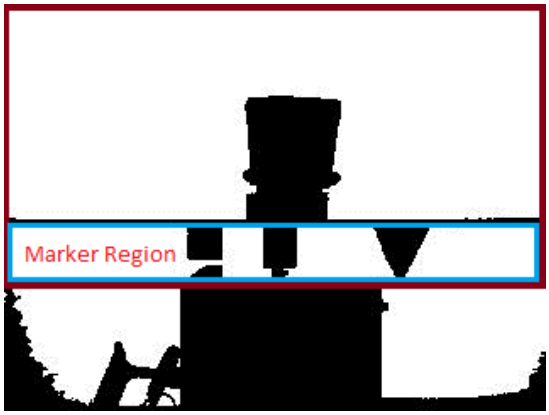

Figure 2: Labeled image boundary

B. To do segmentation on the marker region

Once we find the boundary of the image, we would know where the marker region is in each frame. We will fetch these pixels and do segmentation. The method used to do segmentation is the same as the method used in step E. The segmentation details can be seen in step E.

C. To find a good first frame

If the original first frame is like the figure 3 below, then it is hard to figure out the angle of the frame. In order to decrease the angle error, we can skip those frames, which guarantees our first frame has an accurate angle. In order to find a good frame, we linearly search the beginning frames until we find a good frame. By observation, we find that for a good frame, there are three components (cylinder marker, cone marker, and platform center) in the marker region. In order to accomplish this, we use a method called finding connected components in binary image, as shown in figure 4 below. We choose the row just below the platform and count its connected components number.

Figure 3: bad start frame example

Figure 4: finding connected component method

D. To find the angle for each frame

In order to find the angle for each frame, we need to find the position values (X, Y) of the markers. As we mentioned above, there are some situation that it is hard to find the position of the markers. We will treat those frames with angle of zero degree at this first, and once we have values for those good frames, we use linear approximation of those zero values.

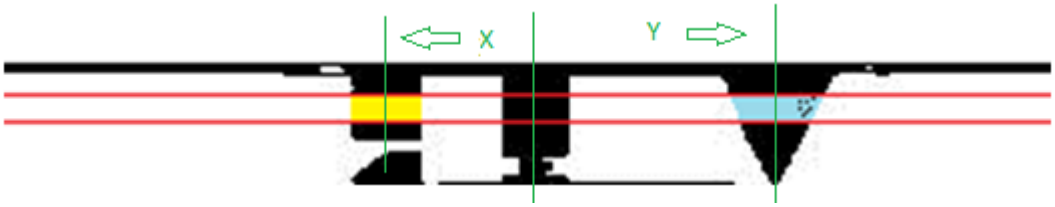

Figure 5: vertical view of platform

Once we have the position values of the markers for all the frames, we need to calculate the angle for each frame. In order to calculate the angle, we set up the coordinate for the geometry. The graph above (figure 5) is the vertical view of the platform. When we derive the relation between X, Y, and angle Θ, there are 4 different kinds of cases. However, the idea behind is the same. Let us take position 4 into consideration. The geometry can be seen in figure 5 in appendix B.

Using geometry, we can derive the following relations between X, Y and angle Θ as equation 2.

$$\frac{D}{-X} = \frac{-r * \sin(\Theta)}{X + r * \cos(\Theta)}; \quad \frac{D}{Y} = \frac{-r * \sin(\Theta)}{Y - r * \cos(\Theta)}$$

$$\frac{\mathrm{Y}}{\mathrm{X}} = \frac{\mathrm{r}*\cos(\Theta)-\mathrm{Y}}{\mathrm{X}+\mathrm{r}*\cos(\Theta)} \tag{2}$$

E. To do segmentation on the selected frames

We use threshold method to do segmentation. We use Otsu's method to choose the threshold value such that pixels possessing values greater than the threshold are assigned to one region whilst those that fall below the threshold are assigned to another (adjoint) region. Threshold creates a binary image b(x, y) from an intensity image I(x, y) according to the simple criterion as equation 3:

$$b(x,y) = 1 \; if \; I(x,y) > T \; or \; b(x,y) = 0 \; otherwise \tag{3}$$

(T is the threshold value)

The threshold selection is essentially based on a conceptual or actual consideration of the image histogram. We would like to separate pixel values belongs to the object from the background. In order to make the segmentation easy, our background is design to be constant (white color). Then, basically what we do for segmentation is to separate the background peak out from the histogram. The Otsu's method we use is based on a relatively straightforward analysis which finds that threshold which minimizes the within-class variance of the threshold black and white pixels. In other words, this approach selects the threshold which results in the tightest clustering of the two groups represented by the foreground and background pixels.

F. To clean the segmentation result

In order to reduce the noise of the object image, we implement a size filter to clean out those small noises. Basically the size filter is to reserve the maximum connected graph in the image and filter out the small noise in the image.

3.6 3D Reconstruction

3.6.1 Reconstruction Flow Chart

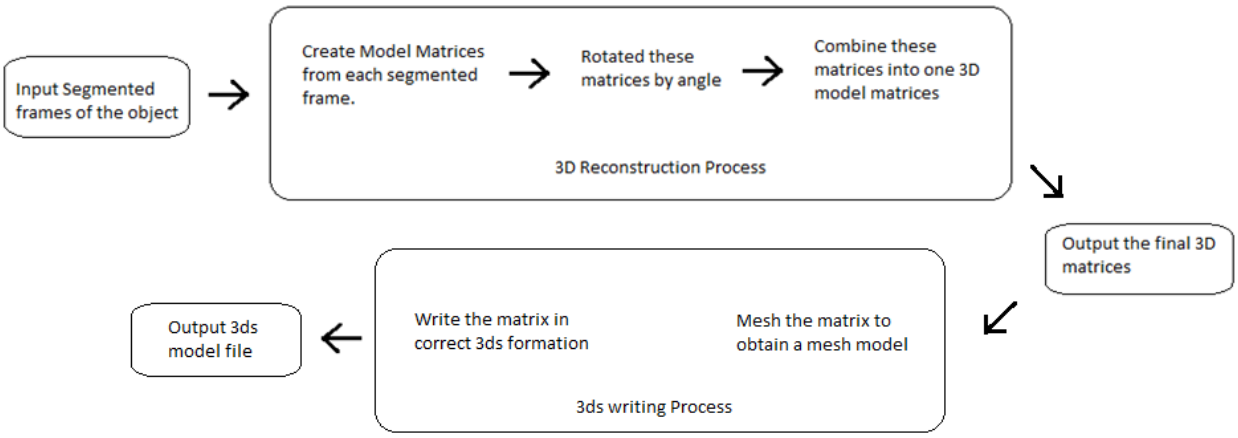

Figure 6: 3D reconstruction flow chart

3.6.2 Reconstruction Algorithm

$For\ i = 1{:}\,36\ frames$

$$Cube_{init}\{i\}(x,y,z) = I_{segment}\{i\}(x,y) \times \begin{pmatrix} \frac{Dis+z}{Dis} & 0 & 0 \\ 0 & \frac{Dis+z}{Dis} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Cube_{rotated}\{i\}(x,y,z) = Cube_{init}\{i\}(x,y,z) \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(10° \times i) & -\sin(10° \times i) \\ 0 & \sin(10° \times i) & \cos(10° \times i) \end{pmatrix}$$

$$Cube_{segment}\{i\}(x,y,z) = Threshold(Cube_{rotated}\{i\}(x,y,z))$$

end

$$Cube_{finalmodel} = \bigcap_{i=1}^{36} Cube_{segment}\{i\}$$

$Note{:}\ Operator \times represent\ affine\ transformation\ with\ tfm\ matrix\ (\because)$

Program running time calculation and estimation:

$Input\ parameter{:}\ n(image\ number), r^3(matrix\ resolution)$
$Inner\ loop\ with\ n{:}$
$First\ step{:}\ r \times (r^2 affine\ transformation) = O(r^3)$
$Second\ step{:}\ r \times (r^2 affine\ transformation) = O(r^3)$
$Thrid\ step{:}\ traverse\ the\ r^3 matrix = O(r^3)$
$Inner\ loop\ end$
$Outer\ loop{:}\ traverse\ all\ n \times r^3 matrix = O(n \times r^3)$

$Total\ reconstruction\ running\ time\ is\ n \times \big(O(r^3) + O(r^3) + O(r^3)\big) + O(nr^3) = O(nr^3)$

Prove that 36 angle images are sufficient for 50x50x50 matrix resolution:

$Angle\ differenct\ between\ images: \frac{2\pi}{36} = 0.1743$

$Estimate\ size\ of\ the\ scanning\ object: 50 \times 0.6(platform\ occupied\ rate) = 30$

$Estimate\ object\ radius: \frac{30}{2} = 15$

$Radius\ difference: 15 \times 0.1734 = 2.601$

$Pixel\ shift\ distance: \sqrt{2^2 + 2^2} = 2.82 \approx 2.601 \pm 1(estimate\ noise)$

More importantly, 10 degree is a convenience number to pick.

3.6.3 Meshing Algorithm

During the last step, we are able to obtain a matrix with 0 and 1. By listing up all these coordinates (x,y,z) where cube(x,y,z)=1. We obtain a plot cloud for 3d model. Therefore, our next step is to mesh the plot cloud. Not like meshing an arbitrary plot cloud, our case only need to consider very few conditions because our plot coordinates are all integers. Plots with distance 1 are adjacent plot and therefore should belong to one object.

$Switch\ case: Object\ layor(third\ dimension)$
$Case: top\ of\ the\ object$
    $Along\ second\ dimension:$
        $For\ each\ line\ segment(ai, bi)\ in\ this\ line$
            $For\ each\ line\ segment(cj, dj)\ in\ next\ line$
                $If\ they\ overlape\ Then\ triangle(ai, cj, bi)\ \&\ triangle(ai, dj, bi)$
            end
        end
    end
$Case: body\ of\ the\ object$
    $Find\ nearest\ point\ pair\ between\ \big(Boundary\big(layor(i)\big), Boundary\big(layor(i+1)\big)\big)$
    $If\ \{ai, bi, ci\ \ldots\}\ have\ pair\ up\ with\ k(i+1)$
        $Then\ triangle\big(ai, bi, k(i+1)\big)\ \&\ triangle\big(bi, ci, k(i+1)\big)\ \&\ \ldots$
    $Else\ If\ \{a(i+1), b(i+1), \ldots\} have\ pair\ up\ with\ pi$
        $Then\ triangle(a(i+1), b(i+1), pi)\ \&\ \ldots$
    $Else$
        $triangle\big(ai, bi, m(i+1)\big)\ \&\ triangle(m(i+1), n(i+1), bi)$
        $\{ai, m(i+1)\}\ and\ \{bi, n(i+1)\}\ are\ nearest\ pairs$
$Case: bottom\ of\ the\ object$
    $Reverse\ top\ operation$

Program running time calculation and estimation:

$Input\ parameter: n(image\ number), r^3(matrix\ resolution)$
$loop\ with\ third\ dimension:$
$First\ case:\ r \times r = O(r^2)$
$Second\ case: (r \times 2\pi)^2 = O(r^2)$
$Thrid\ case: r \times r = O(r^2)$
$loop\ end$
$Total\ meshing\ running\ time\ is\ r \times O(r^2) = O(r^3)$

Unfortunately, the meshing algorithm is not implemented in the project. Therefore, there will be no meshing algorithm verification in the Design Verification part.

3.7 User Interface

Software interface looks like:

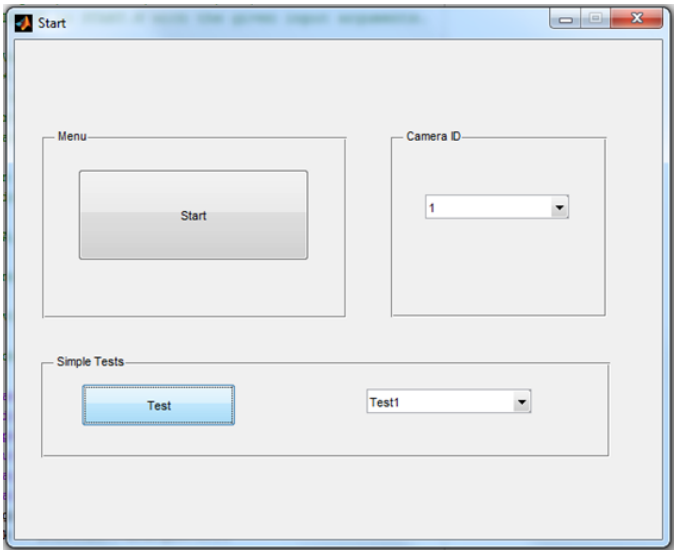

Figure 7: User interface

- Start button: Control the camera on the device and start 3D scanning.
- Camera ID: In case multiple cameras exist on computer, input camera ID is needed.
- Test: Run the existing data for reconstruction test. (totally 4 tests)

4. Design Verification

4.1 Power Supply

The two AA batteries can be seen on the circuit as shown in figure 6 in the appendix B.

4.2 Light System

A picture of the light system is attached in diagram 1 in appendix B.

4.3 Motor Drive Circuit

The actual circuit can be seen in figure 6 in the appendix B.

4.4 Image Collector

The camera can be seen in figure 1 in appendix B.

4.5 Fetching image

A. To find image boundary

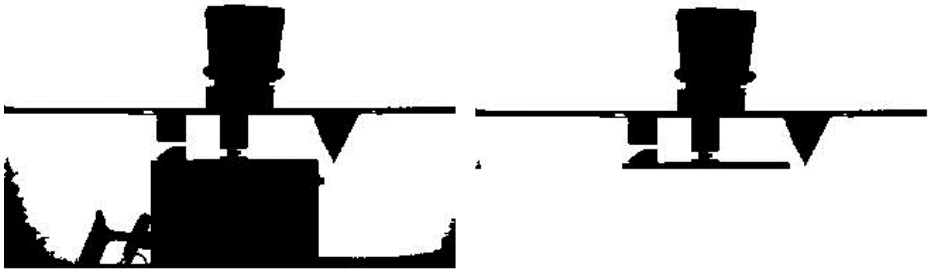

Figure 8: find image boundary

B. To do segmentation on the marker region

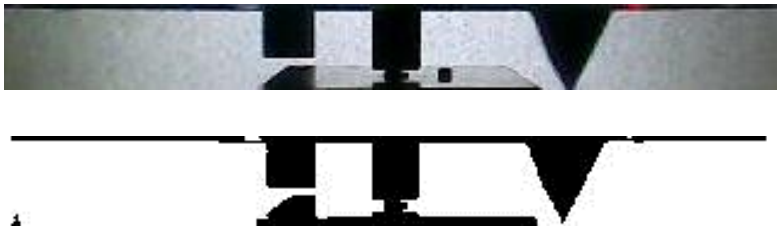

Figure 9: Segmentation on the marker region

C. To find a good first frame

Given the images as shown in figure 7 in appendix B, we can successfully find the first frame is frame NO.17

D. To find the angle for each frame and to do segmentation on the selected frames.

The 36 images as shown in figure 8 in appendix B are 36 different viewpoints of the scanning object.

E. To clean the segmentation result
The left one is the image without cleaning, and the right one is the image after cleaning

Figure 10: image cleaning

4.6 3D Reconstruction

4.6.1 Reconstruction algorithm step verification

- 3D matrix generated successfully from one projection image (cup case):

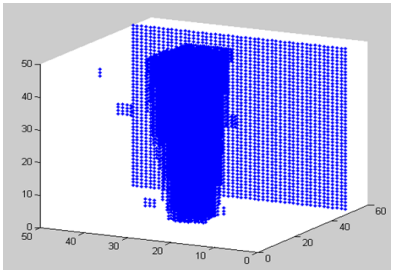

Figure 11: Successfully generated 3D matrix

- 3D matrix rotation success (cup case):

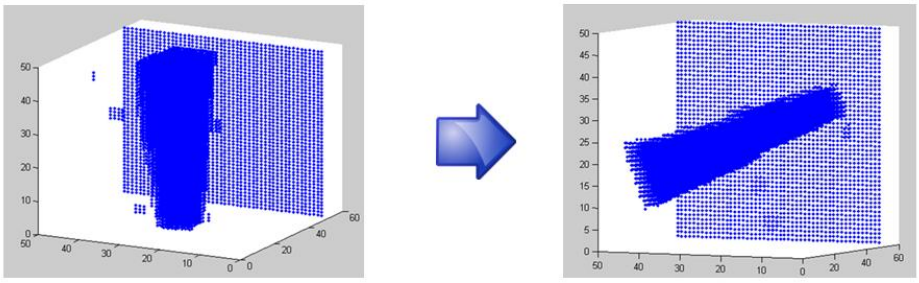

Figure 12: 3D matrix rotation

- 3D matrix combination success (cup case):

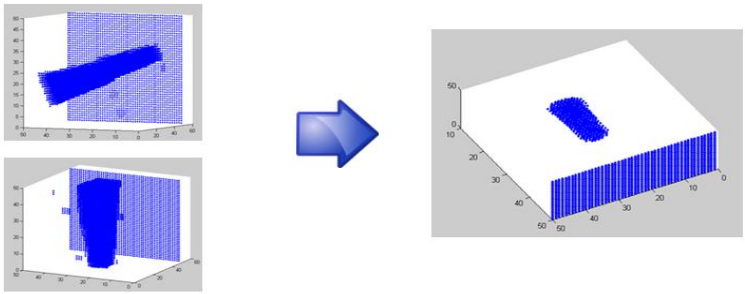

Figure 13: 3D matrix combination.

4.6.2 Ideal Case Test

3D Reconstruction Result from Ideal segmented images (vase):

Input View from each angle:

Figure 14: Ideal segmented Image

Horizontal Intersection views show that the model in 3D matrix successfully reconstructed:

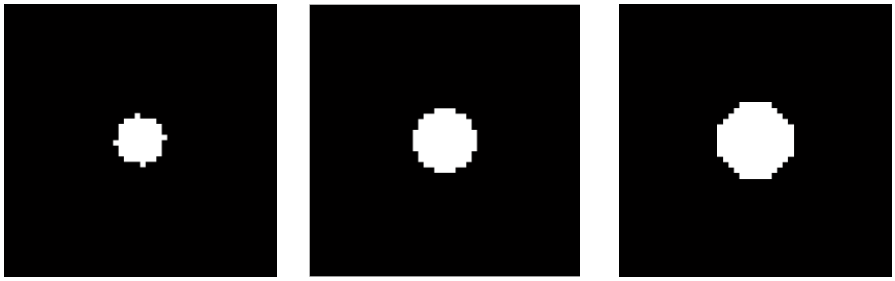

Figure 15: Horizontal intersection views

Vertical Intersection views show that the model in 3D matrix successfully reconstructed.:

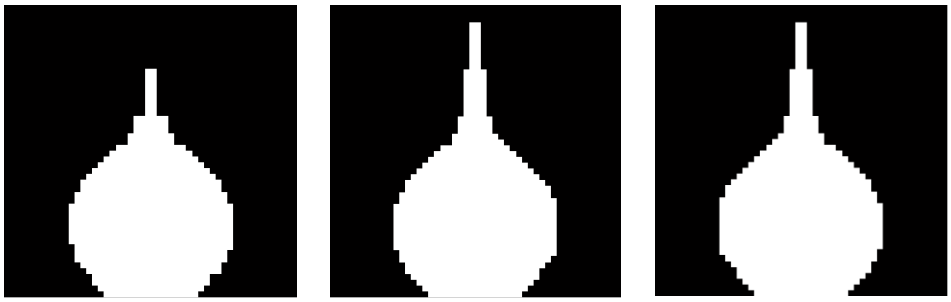

Figure 16: Vertical intersection views

4.6.3 3DS Formatting Test

3DS formatting success:

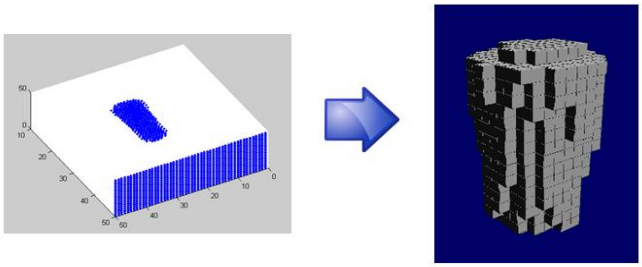

Figure 17: 3DS formatting.

4.6.4 Real Tests

Several reconstructed 3D models can be seen in figures 9 through 11 in appendix B. Figure 9 is non-symmetric object test (mouse). Figure 10 is non-symmetric object (SpongeBob). Figure 11 is non-symmetric object with concave surface (coffee cup made by Prof. Carney).

4.7 User Interface

Our user interface is working well because all other results are generated through it.

5. Costs

5.1 Parts

The total parts cost ended up being $46.69. Since most parts of our project are obtained from the power lab, so we did not pay for them, we just took them. But if we take these parts into our final budget, per unit retailing cost is $76.91. If parts are purchased in large quantity, we can lower per unit cost to $57.03. Detailed parts cost can be seen in table 1 in appendix B.

5.2 Labor

We had set our salary at $60/hour and estimated that we would spend around 200 Hours (16 Hours/Week for 12 Weeks) working on the project. This came out to be $12,000 per person. We both ended up with fewer hours than predicted (12Hours/Week for 12 Weeks), which reduced our research and development costs by almost 28%.

Name	Rate	Hours	*2.5
Hansen Chen	$60/hour	200	$30000
Xiaobo Dong	$60/hour	200	$30000
Xingqian Xu	$60/hour	200	$30000
		Estimated Labor Cost	$90000
		Actual Labor Cost	$64800
		Under Budget by	$25200

Table 2： Labor Costs

6. Conclusion

6.1 Accomplishments

We were very pleased with the final result of our project. We finished all of our module sections and combined everything together to get a working project. The 3D scanner can be powered with 2 AA batteries for 5 hours; the platform rotation speed stayed constant with low battery voltages or varying weight scanning object on the platform; and most importantly, it was able to scan objects like mouse and reconstruct its 3D model in 3ds file format.

6.2 Future Work

Overall, our project is functioning correctly. However, several works can be added in future:

Hardware:

- Replace the huge metal rail by small holding arms that can be conveniently folded up.
- Add a front light system to capture model color information.
- Add user control signal to start LED and motor through software user interface.

Software:

- Implement the meshing algorithm. Combine it with our main program.
- Apply fast algorithm using FFT. Optimize our back-light-projection algorithm to run fast.
- Use more advance algorithm to avoid the Matlab limitation in matrix variable.
- Implement color part in 3D model using front light video.

6.3 Ethical considerations

We agree to uphold the IEEE Code of Ethics, and will address any relevant ethical concerns about our project.

As with any engineering project, there are numerous ethical concerns to be considered throughout the course of this project. User safety and product reliance are our main concern for this project; we need to consider and avoid any electrical shock that may happen to the user since main part of the device is a metal platform. Since the device is targeted as a user-friendly scanning device, easy operation of the device must be achieved. In addition, all information obtained from outside to assist in completing this project will be given appropriate references and all outside help will be addressed at the end of this project as well.

References

[1] Y.Ma, S. Soatto, J. Kosecka, S.S.Sastry, An Invitation to 3-D Vision: From Images to Geometric Models.

[2] R. Ziegler, W. Matusik, H. Pfister, L. McMillan, 3D Reconstruction Using Labeled Image Regions.

[3] P. T. Krein. ECE 469 - Power Electronics Laboratory: Laboratory Information and Guide.

[4] TPS63000 Data Sheet. Texas Instruments Inc., Texas. [Online] Available: http://www.ti.com/corp/docs/landing/**tps63000**/index.htm

[5] Chris Solomon, Toby Breckon, Fundamentals of Digital Image Processing, Chapter 10 Image Segmentation P.264 to P. 274 Wiley-Blackwell,1988

[6] Wikipedia, Otsu's method, Retrieved March 13, 2012 http://en.wikipedia.org/wiki/Otsu%27s_method#cite_note-Otsu-1

Appendix A Requirement and Verification Table

Requirement	Verification	Verification status (Y or N)
<u>Power Unit</u> 1. The battery needs to provide a stable 1.5V DC power source. 2. Buck-boost converter must provide a stable voltage level for the Motor. (The voltage level will be given based on the actual duty ratio of the circuit) 3. The input voltage for the motor needs to regulate motor's speed at 2RPM.	1. The voltages across the battery and the Buck boost converter's output voltage are measured using oscilloscope.	Y
	2. To ensure the motor rotates at 2RPM, we will measure the voltage on the dc battery before installed it in the circuit. Then we will measure the voltage on the motor to check if the buck-boost converter functions as expected.	Y
	3. Moreover, we will change the parameter of the buck-boost converter (duty ratio) until the speed of the motor is at 2RPM.	Y
<u>Image Collector</u> 1. The video recorded by the camera must have right focus on the object. The video must also be clear, not blurred for segmentation purpose. 2. The object should occupy at least half of the frame.	The recorded video will be examined with several sampled frames. 1. Using Matlab to obtain several frames of the video. Then we will verify these sample frames by eyes.	Y
	2. The occupation of the object in the frame depends on the size of the object and also object's distance to camera.	Y
	3. We will change the distance to accommodate different size object.	Y
<u>Object Platform</u> The object Platform must rotate stably so that the object would not vibrate badly or slide on the platform. 1. It must rotate at a constant speed (2 RPM). 2. It cannot vibrate badly.	1. In order to test the speed of the Object Platform, we will mark one spot on the edge of the platform. Using a timer, we can observe how many rotations the platform has gone in 2 minutes, divided by two, we can verify whether its speed is 2RPM or not. We will no multiple tests (> 4 times) to ensure the result.	Y
	2. To check the vibration, we will test different objects on the platform to see if it stays stable when the platform rotates.	Y
<u>User Interface</u> 1. The scan button must function correctly. When pressing the button, correct "startup" data package will be sent to main controller.	1. By inserting a data package receiver between User Interface and main controller to see whether the correct data package is sent out.	Y

2. The resolution change buttons must function correctly. When pressing the resolution buttons, the resolution parameter in software input must be changed.	1. Monitor software parameters during debug mode, to see whether pressing the resolution button will make a change in the resolution parameter input to the software.	Y
<u>Main Controller</u> 1. Main Controller must correctly response to user input. 2. It must correctly send/acquire data packages to/from web camera. 3. It must correctly control the background LED light in order to obtain the right segmentation.	1. In order to test whether the main controller can correctly correspond with user input. We need to monitor output of the main controller to see whether a correct signal is sent. Inserting testing LED light will be an easy way to see a coming signal.	Y
	2. As our web camera is built outside the main controller. We can monitor the camera performance to see whether main controller is working correctly.	Y
	3. Background LED light can also be monitored by tester. When the platform is rotating at a constant speed, background LED should be lighted up by the main controller.	Y
<u>Data Processor</u> The data processor contains several parts. 1. Segmentation: It needs to process the input image and successfully segment the object out. 2. Given by the Segmentation image, it needs to reconstruct the image 3D model. We use 3D matrix to contain the information of 3D model. 3. It needs to represent the 3D model in 3Ds files.	1. The data processing result will be examined using Matlab. We will analyze the Matlab output figure and graph to check the result.	Y
	2. Threshold method is used in Segmentation part. Therefore, in order to achieve a good performance in segmentation process we need to check different threshold values.	Y
	3. Compare each segmented result with the original graph. If the pixel(x,y) on the original graph belongs to the object, then in the segmented image pixel(x,y) should has value 1. Otherwise, it has value 0.	Y
	4. Once the 3D matrix has been obtained. We will check the intersection graph horizontally and vertically. And compare these intersection graphs with the real object.	Y
	5. In this part, we need to find a method to represent the 3D model for the user. We will compare the .3Ds file with the object in shapes and colors.	Y

Appendix B Diagrams and waveforms

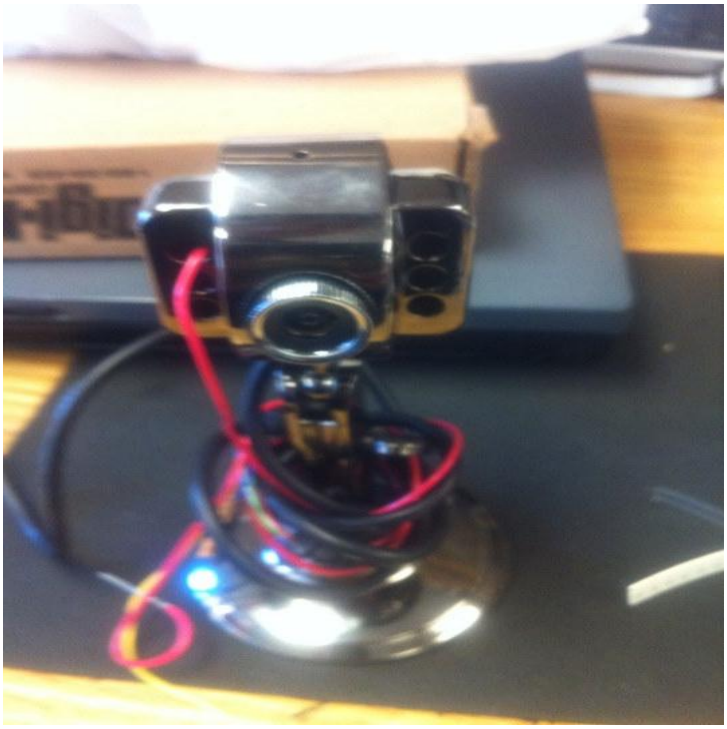

Figure 1: Web Camera

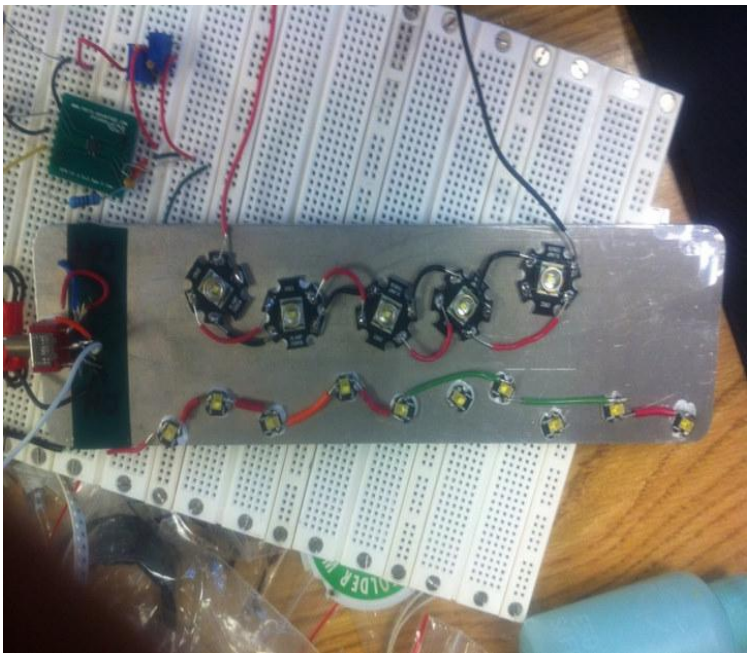

Figure 2: LEDs

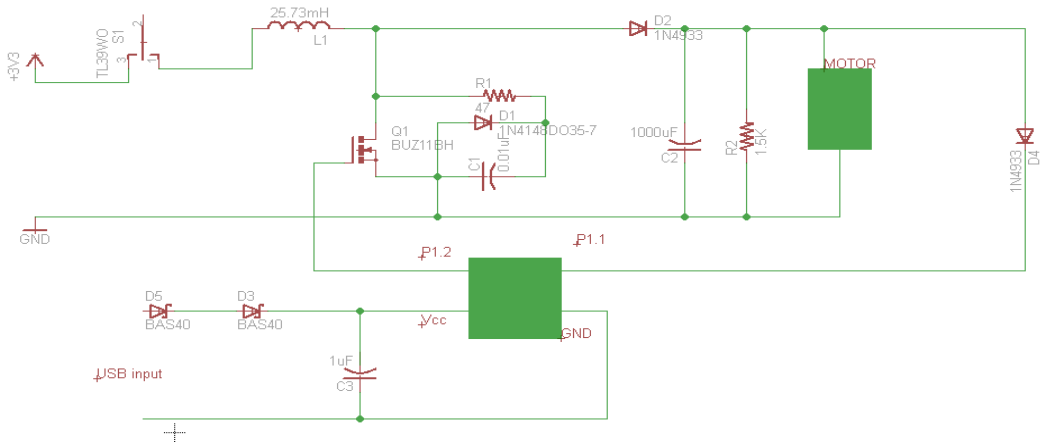

Diagram 3: motor drive circuit diagram

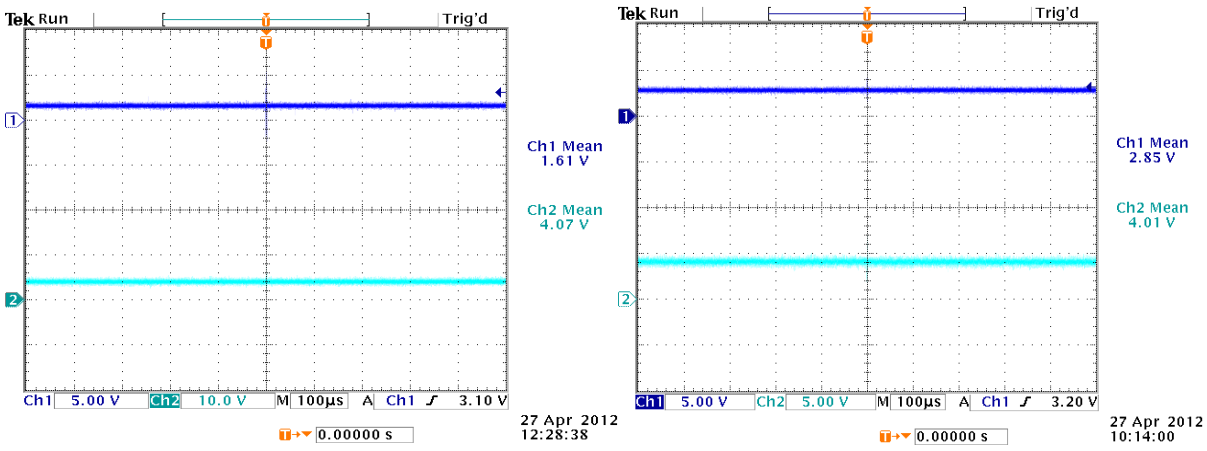

Diagram 4: motor drive circuit performance waveform

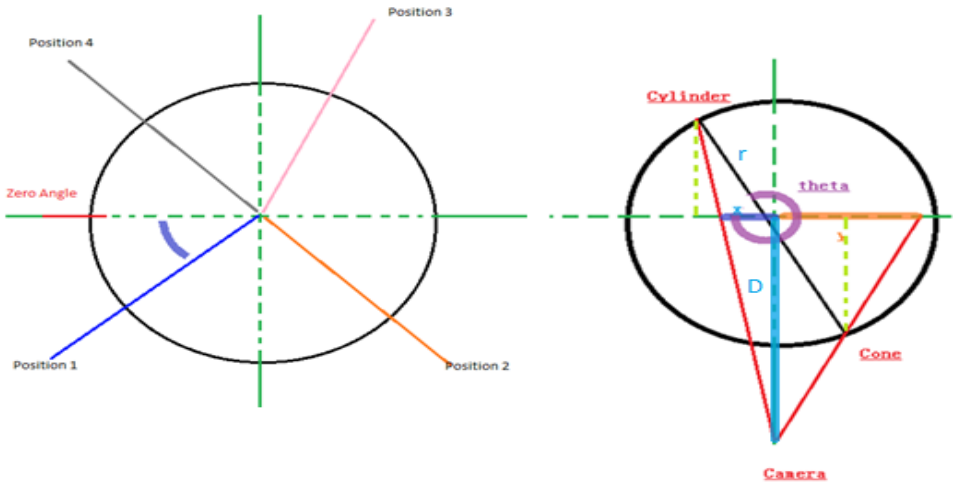

Figure 5: Angle geometry

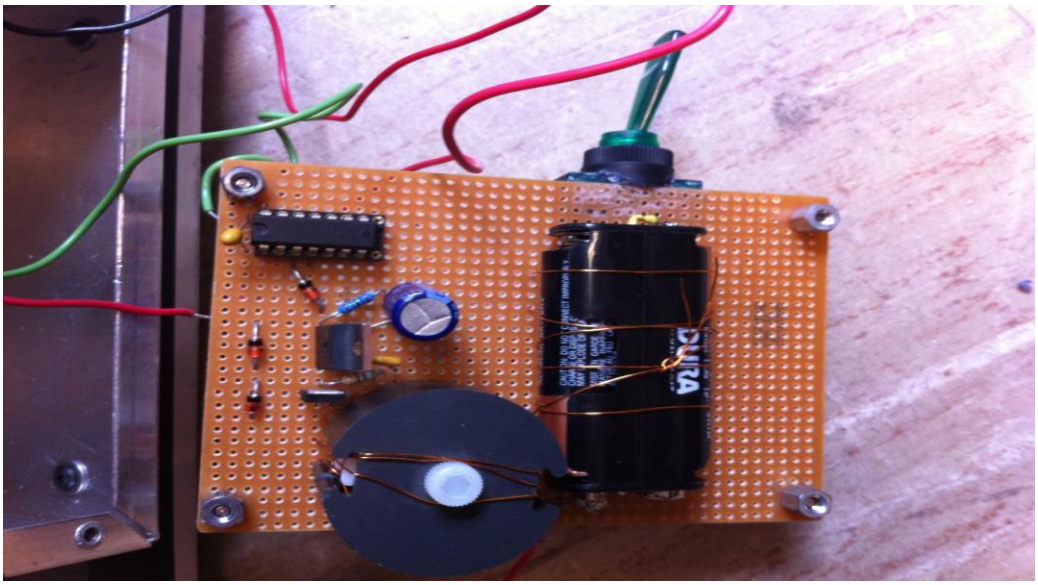

Figure 6: actual motor drive circuit

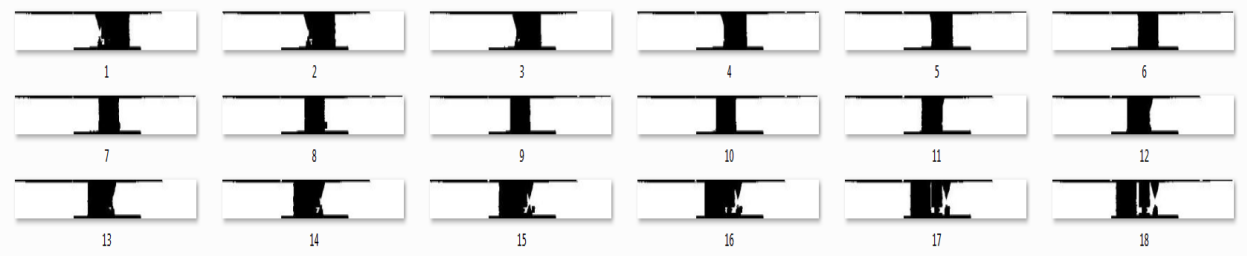

Figure 7: Finding start frame

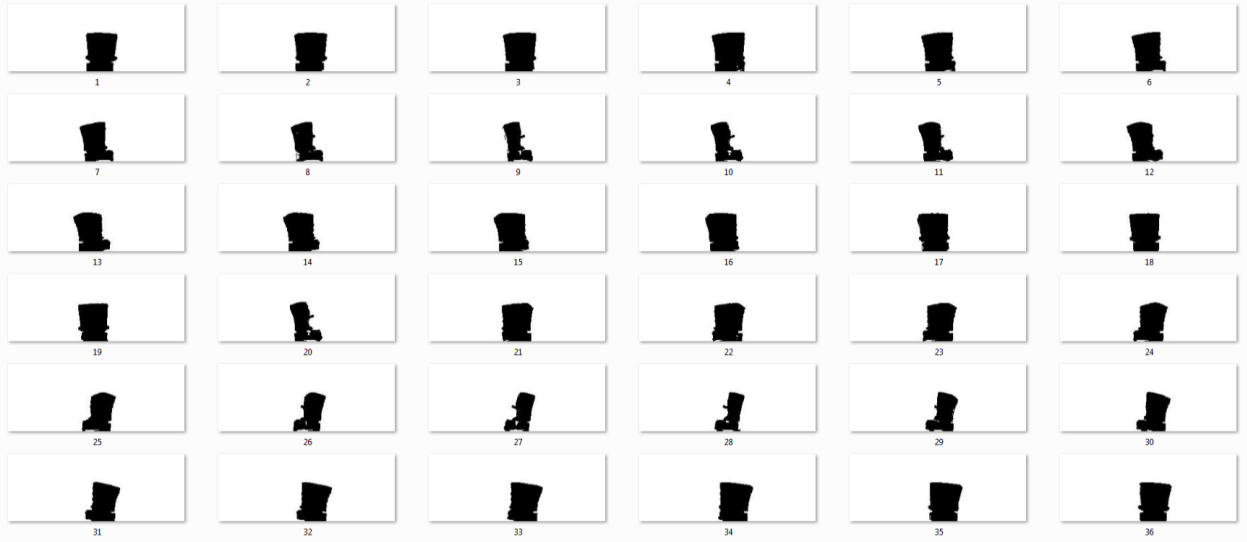

Figure 8: 36 viewpoints

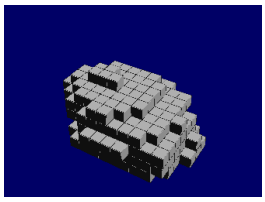

Figure 9: non-symmetric object (Mouse)

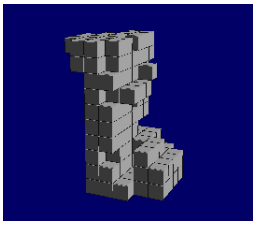

Figure 10: non-symmetric object (SpongeBob)

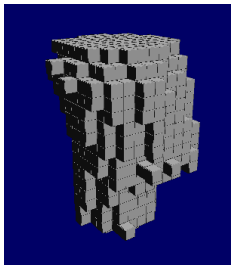

Figure 11: non-symmetric object with concave surface (cup)

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Metal Platform	ECE Machine Shop	40	30	30
DC motor	Igarashi Motors 2732	12.50	7.5	0
LED light	N/A	1.00	0.80	0
Camera	Logitech	12.50	11.50	12.5
Micro-controller	Texas Instrument	0.95	0.86	0
Light cover	N/A	3.00	1.50	0
Power MOSFET	2N6756	3.19	2.07	3.19
Coupled Inductor	ECE Power lab	2.77	2.00	0
1.5V AA battery	Duracell AA-CTx20	1.00	0.80	1.00

Table 1: Parts Costs