Project Report on

Scanner3D

Ekalavya Internship Program 2017

Submitted in partial fulfilment of the project

By

Anchal Singh, NIT Uttarakhand
Animesh Srivastava, NIT Hamirpur
Sagar Satapathy, NIT Rourkela
Soumya Sambit Rath, NIT Rourkela

Mentor: **Akshay Chipkar**
Project Manager: **Rajesh Kushalkar**
Program Director: **Avinash Awate**

Under the guidance of
Prof. D. B. Phatak

Acknowledgement

We would like to express our heartfelt gratitude towards **Prof. D.B. Phatak** for offering and allowing us to carry out this Project on **Scanner3D**. We express our special thanks to our project mentor **Mr. Akshay Chipkar**. We also thank **Mr. Rajesh Kushalkar** (Project Manager, IDL, IIT Bombay) for his help, without which we would not have been able to concentrate on this project. We would also like to express our sincere thanks to the entire IDL (Integrated Development Lab) team, who helped us with their knowledge and support. We also thank our parents without whose support, this entire venture would not have seen the light of success.

5th July 2017

Project Approval Certificate

Summer Internship 2017

Integrated Development Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

The project entitled Scanner3D submitted by Anchal Singh Panwar (NIT Uttarakhand), Animesh Srivastava (NIT Hamirpur), Sagar Satapathy (NIT Rourkela) and Soumya Sambit Rath (NIT Rourkela) is approved for Summer Internship 2017 program from 8th May 2017 to 5th July 2017, at Department of Computer Science and Engineering, IIT Bombay.

Dr. Deepak B. Phatak	**Mr. Avinash Awate**
Dept. of CSE, IITB	Project In charge

Declaration

We declare that this written submission represents our ideas in our own words and wherever external references have been used, we have adequately cited and referenced the sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not plagiarized or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also invoke penal action from the sources wherever proper references haven't been mentioned.

____

Anchal Singh Panwar
NIT Uttarakhand

____

Animesh Srivastava
NIT Hamirpur

____

Sagar Satapathy
NIT Rourkela

____

Soumya Sambit Rath
NIT Rourkela

Contents

Purpose

This project is about developing the hardware and the software for a 3D Scanner. The purpose of this project is to generate 3D image of an object using a 2D camera, which could be opened and edited with a Third Party software like Blender or Meshlab.

Two lasers have been deployed in the setup. At the centre, there is a platform where the object has to be kept for the purpose of scanning. The platform is rotated in steps by a stepper motor controlled by the software through a microcontroller.

A line laser is used to set the reference axis for the camera and the other line laser is projected on the object in such a way that the beam takes the outline of the edge of the object. The code extracts only the contours of the laser beam and saves the coordinates for all the points in that contour and another piece of code integrates all the points into a single 3D image. For the convenience of the user, a complete GUI application has been developed.

Although, there are many technologies present for 3D Scanning, but most of them use expensive hardware and software. Since the technologies used in the project are completely free and Open Source, the only cost of the project is due to its hardware. All efforts have been made to keep the project as affordable as possible and the User Interface as easy as possible.

Applications of the 3D Scanner:

This project is aimed at producing 3D images of objects. It will find its use in different domains of Science, Engineering and other allied fields. Some of the specific uses listed below require modifications and adaptations in order to be used in different applications with different requirements.

The 3D Scanner in its present form can be used for Educational, Scientific and Archaeological purposes.

- **Manufacturing:**
  Can be used for making the Die and Mould design or modification. It can be used for raw casting or forging inspection. It can also be used for positioning fixtures. The 3D data collected by the Scanner can be used by the CNC machines.

- **Automotive:**
  Can be used for the inspection of the product for flaws and defects. It can be used for the calibration of the tools used in the production line.

- **Archaeology:**
  It can be used for the scanning of the ancient remains, antiques and findings at the sites. It can also be used for cloning the antiques for further experimental studies.

- **Educational:**
  It can be used to optimize research related to Image Processing and Computer Vision. It can be used for 3D rendering studies. It can be used for the development of Virtual Reality and Augmented Reality.

- **Medical:**
  It can be used for making the 3D models of different intricate organs of the body. It can be used for making moulds for prosthetic limbs and other artificial organs.

Present technologies used in the field of 3D Scanning:

The concept of 3D scanning is not new. It has been in development since the 1990s. Some of the existing technologies are:

1. 3D Shape Scanning using Depth Camera (Time of Flight camera):

This Scanner developed by Yan Cui, IEEE member and his team implement a method for 3D object scanning by aligning depth scans that are taken from around an object with a Time-of-Flight (ToF) camera. Depth cameras have a variety of conceptual advantages over previously used sensor techniques for shape scanning.

They capture full frame depth at video rate and provide depth data with negligible latency as well as they do not need to subsequently scan scene points for a single depthmap (like a laser scanner).

Figure 1

2.LiDAR based 3D Scanner:

The LiDAR chips are produced on 300-millimeter wafers in commercial CMOS foundries. This technology is light weight, cheap, with a range of 5 miles and resolution of 3 cm at a 2 mile range. The devices are designed for robotics and autonomous vehicle applications. This sensor is being used to develop very accurate 3D scanners to be used in various fields of structured light or photogrammetry.

Figure 2

3. 3D Reconstruction using shadow carving:

This concept uses cast shadows to find concavities of an object. Given a conservative estimate of the volume occupied by an object, it is possible to identify and carve away regions of this volume that are inconsistent with the observed pattern of shadows. This technique is quite widely used for low precision scanning since it cannot estimate engravings and depths with much precision as a 3D scanner with laser setup does.

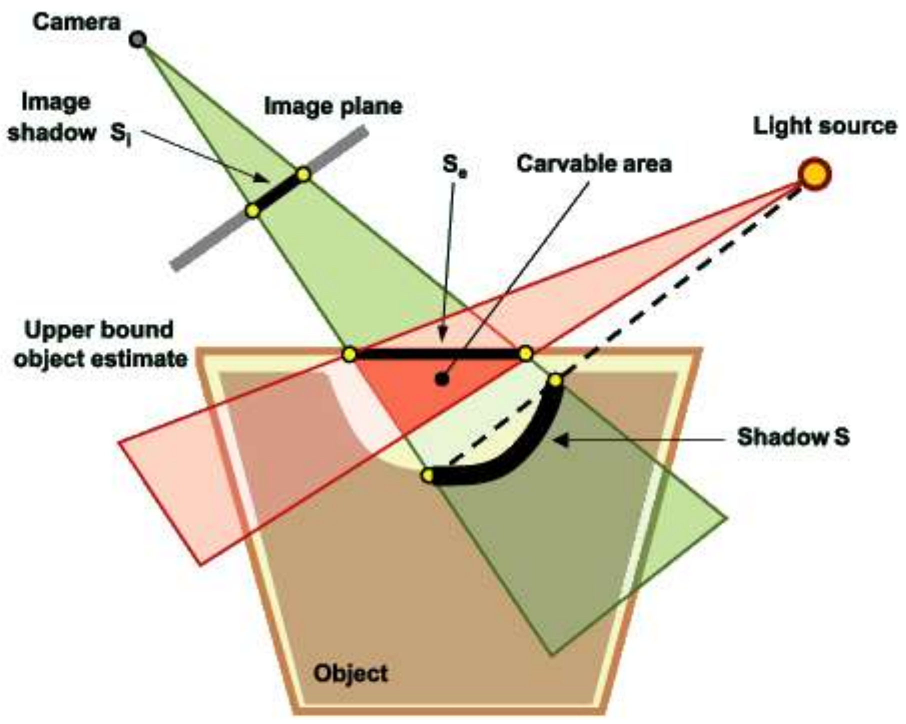


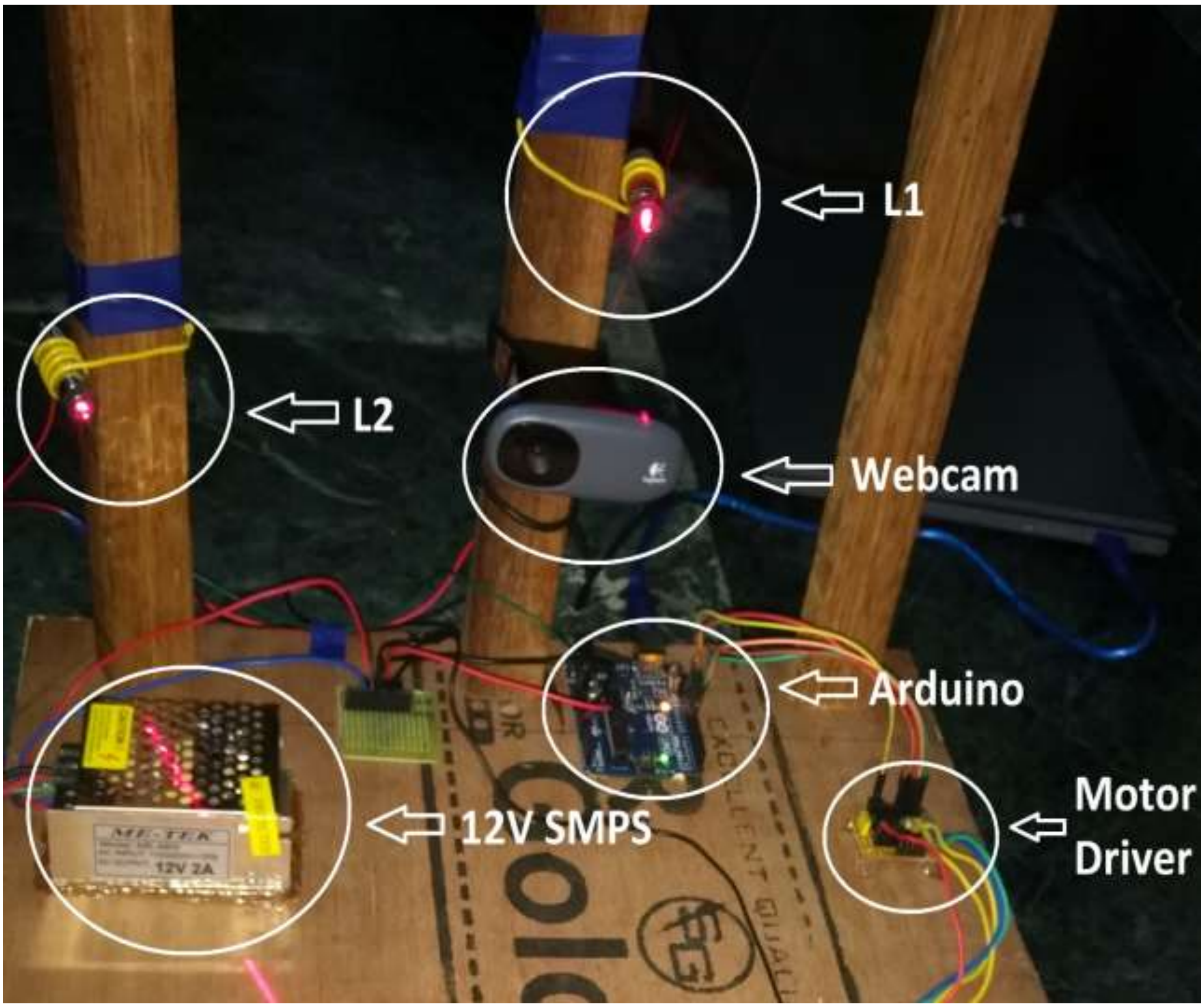
Figure 3

Methods

1. Image Capture:

The set up requires two laser modules to be oriented in such a way that one helps to determine the reference of the axis (L1) and the other laser (L2) allows the camera to capture edges of the object (to be scanned) being intensified by it(L2).

The laser(L2) is set at an angle of about 40-45 degrees with respect to L1. The laser (L2) throws a vertical beam of red light on the object and the camera detects it. Since the light falls on the object from an angle, it takes the outline of the object. This helps to extract the outer outline of the object from each frame. The setup is so designed to rotate a total of 360^o in 48 steps and capture 48 such frames in total. All the frames that are captured are in 2D format.



Laser and Camera Setup

2. Extraction of contours formed by the incident laser:

The frame detected by the image is then processed using OpenCV in order to extract only those contours (regions) which come under the incident Laser beam, leaving other parts of the frame totally black (R=0,G=0,B=0).

The frames are numbered from 0 to 47 and saved as '.png' files. But, the most important thing that we require from those processed frames is the coordinates of the points that make the contour. Hence, the coordinates are stored in '.txt' files numbered from 0 to 47.

3. Conversion of 2D coordinates to Spherical coordinates:

This conversion requires the values of Radius ('R'), Theta ('ϴ') and Phi ('ϕ'). It is so because calculation of spherical coordinates of a point of the object is easy since we have the value of ϕ from the rotation of the stepper motor.

The saved text files which contain the 2D coordinates of each frame are parsed to obtain X and Y coordinates of all the points associated with the contours and then used to calculate the values of (R,ϴ,ϕ) of all the points collected during the 360^0 rotation.

The details of the mathematical calculations associated with this conversion have been included in **Appendix B**.

4. Formation of 3D object file (.ply):

Although calculation of the coordinates in the spherical coordinate system is easy, but finally the 3D format file contains the coordinates of the points in the Cartesian coordinate system. Hence the calculated (R,ϴ,ϕ) coordinates are again converted to (X, Y, Z) coordinates.

The coordinates so obtained are used to write a 3D image file. Polygon File Format ('.ply') is used to store the final 3D image data. These files can be used in various 3D image visualizing applications like Blender, which is also an Open Source Software. Various other 3D files can be created, for instance .obj, .stl, etc.

5. Plotting the 3D image:

Either the final coordinates obtained can be used directly or the generated Polygon File Format(.ply) file can be used to display the 3D object. The function used for this purpose is Matplotlib Scatter plot (a plotting library of Python).The Matplotlib window allows user to pan, tilt and zoom the obtained 3D image.

Another application 3DLoader.py has also been developed which directly opens the created 3D file and displays it in a window. An available library 'Trimesh' was used to accomplish this.

Hardware :

1. Arduino UNO:

Microcontroller-ATmega328
Operating Voltage- 5V
Input Voltage (recommended)- 7-12V
Input Voltage (limits)- 6-20V
Digital I/O Pins- 14 (of which 6 provide PWM output)
Analog Input Pins- 6
DC Current per I/O Pin- 40 mA
DC Current for 3.3V Pin- 50 mA
Flash Memory- 32 KB of which 0.5 KB used by Bootloader

2. Laser Module(INT 2549):

Typical Output Power:3.0mW
Max Output Power: 5.0mW
Typical Working Current: 20mA
Maximum Working current: 25mA
Typical Working Voltage: 4.5V
Max Working Voltage: 8.0V
Wavelength: 650nm
Focused Dot Width: < 2mm spotting 3 meters away

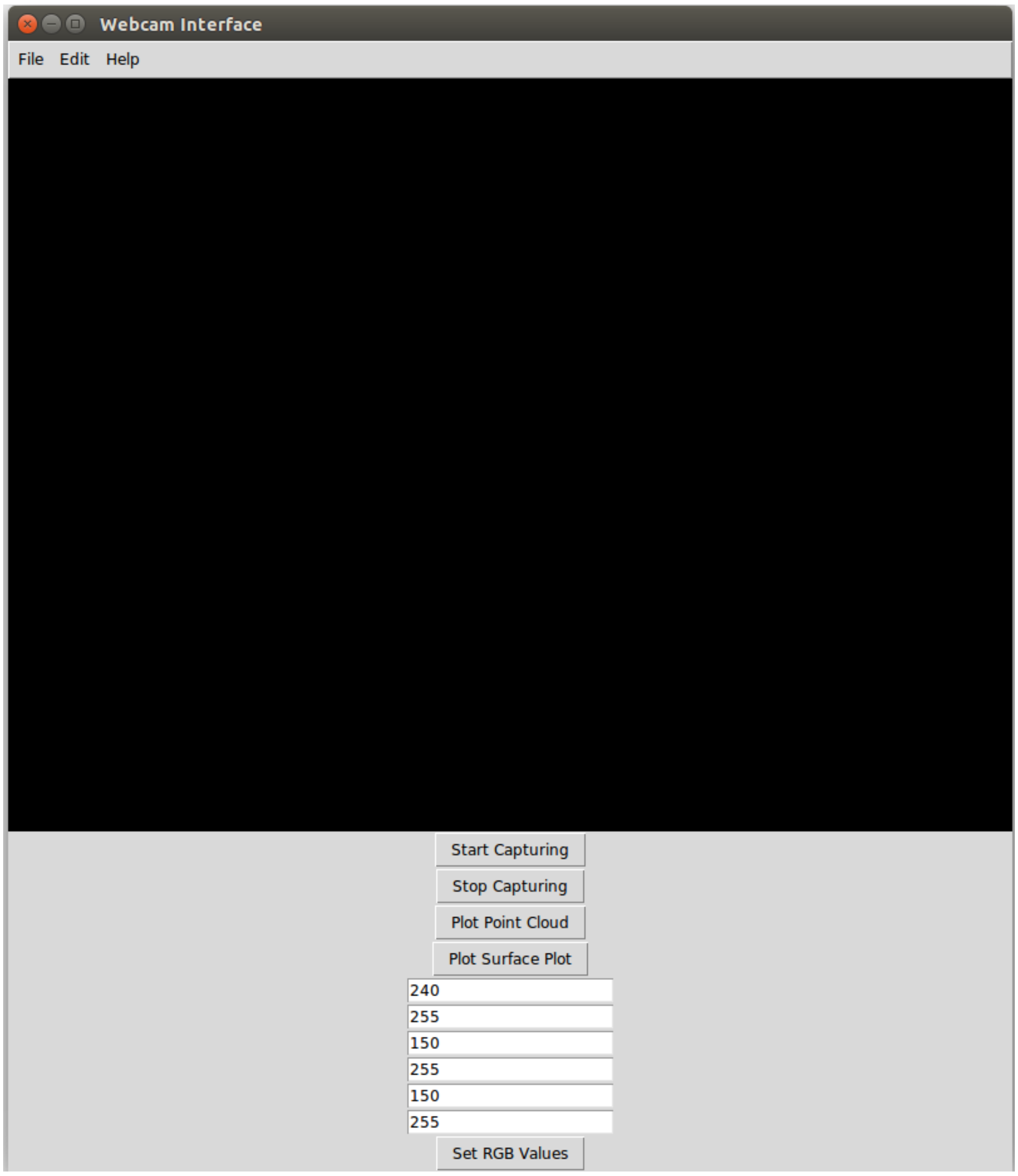
3. Stepper Motor (Generic Bipolar 6 wire):

Rated voltage : 12V DC
Number of Phase: 4
Stride Angle : 7.5^{o}
DC resistance : 6.6Ω±7%(25℃)

4. Current Amplifier IC (ULN2003)

7 Channel Darlington Array
Max Output Current per Channel: 500mA
Max Collector Output Voltage: 50V

Software (GUI interface):



Menus and Submenus:

1. File Menu:

Menu containing File submenus.

Submenus:

1. **Exit:**

   Used for Exiting from the Main Window

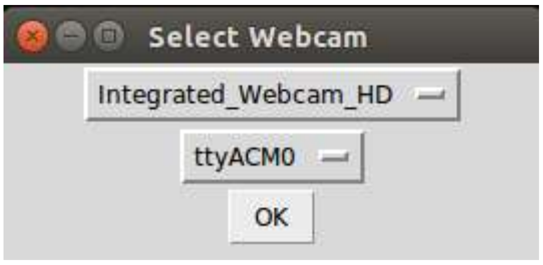
2. Edit Menu:

Menu containing Edit Submenus

Submenus:

1. **Settings:**

   Shows the details of Microcontrollers (Arduinoes) and Webcams connected to the system and allows user to choose from the available list. OK button is to be pressed after selecting the Arduino and Webcam from the Combo-Box.



3. Help Menu:

Menu containing Help Submenus

Submenus:

1. **About:**

This submenu creates a new window and shows the details about the Project and the team that created it.

Widgets in the Main Window:

1. Start Capturing:

This button when clicked, starts capturing the frames.The frames get saved within the same directory inside a directory with a system generated name based on the current date and time.

2. Stop Capturing:

This button has been added to stop all the processes in case any problem arises.

3. Show Scatter Plot:

This button when clicked will generate the final Scatter plot of the coordinates so obtained using Matplotlib’s Scatter function.

4. Show Surface Plot:

This button when clicked, will generate the Surface plot of the coordinates by using Delaunay Triangulation.

4. Set RGB values:

Set the RGB values by manually entering the values between 0-255 as per requirementand click on the “Set RGB values” button to apply the changes.

5. Progress bar:

The progress bar indicates the current progress in accordance with the rotation of the Stepper motor that rotates the turn table. The process can also be stopped/aborted in the middle by pressing the “Stop Capturing” button.

Libraries Used:

1. **Python 2.7**
   Python was chosen as the default language for writing all the programs throughout the project. The version used by the developers was 2.7.6

2. **PyOpenGL 3.1.0**
   PyOpenGL is the most common cross platform Python binding to OpenGL and related APIs. It has been used in the project for showing the 3D graphics in the Tkinter window.

3. **OpenCV (Python) 3.2.0**
   OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. Written in optimized C/C++, the library can take advantage of multi-core processing.
   This library was used for all the image processing done throughout the project. The capturing of the frames, extraction of contours, conversion of images to different formats, masking of images were done using the functions defined using this library.

4. **Numpy 1.13.0**
   NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.
   This library was extensively used throughout the project as the output format of the processed image at any step was in form of a numpy array. It was also used for calculation of final coordinates.

5. **Matplotlib 2.0.2**
   It was used for plotting the scatter plots and the surface plots.

6. **Pygame 1.9.1**
   This library was used to host the OpenGL window inside the Tkinter's frame in the main window.

7. **Tkinter**
   Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It was used for creating the windows and the widgets.

8. **Pyserial 2.6**
   This library was used for serial communications with the microcontroller from the code.

Benchmark testing of the Hardware and Software

Hardware

Sl No.	Parameters	Values
1	Steps taken by the Stepper Motor to complete one rotation	48
2	Stride Angle	7.5^o
3	Speed of the Stepper Motor	5 RPM
4	Supply Voltage Range	8-12V
5	Current (idle)	40-50 mA
6	Current (scanning)	800mA – 1A

Software

Sl No.	Parameters	Values (approx.)
1	Time taken for the main window to load*	3 sec
2	Time taken to complete a scan*	1 minute
3	Time taken to generate a Scatter Plot*	10 sec
4	Time taken to generate a Surface Plot*	10 sec
5	Total(average) RAM consumed in runtime	100-150 MB
6	Max CPU load during the total process*	92%
7	Size of the output file generated (.ply)	3-5 MB
8	Total Size of the .png files and the .txt files generated	5-10 MB

Responsiveness of various processes (On a scale of 1-10)

Sl No.	Parameters	Values (On a scale of 10)
1	Responsiveness of the Initial Window	7
2	Responsiveness of the mask based on the RGB range specified	9
3	Responsiveness to generate ply file	9
4	Responsiveness of the Matplotlib window showing the Scatter Plot	5
5	Responsiveness of the Matplotlib window showing the Surface Plot	5
5	Responsiveness of the Pygame Window which is used for capturing the image through the Webcam	8

*The above mentioned parameters were calculated on a computer with Intel Core i5-7200U processors having a clock speed of up to 2.8GHz. These values would vary on different processors.

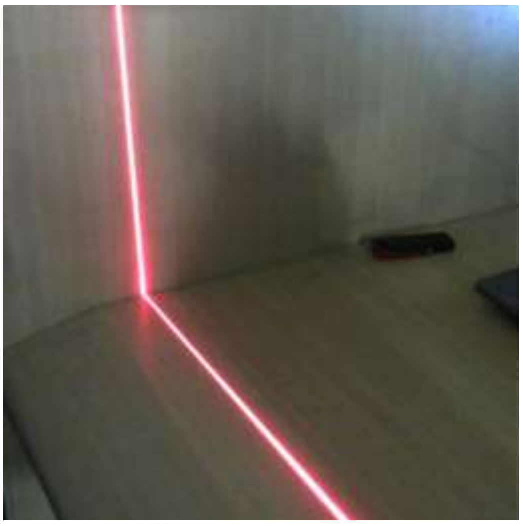
Results

1. Image Captured by the Camera:

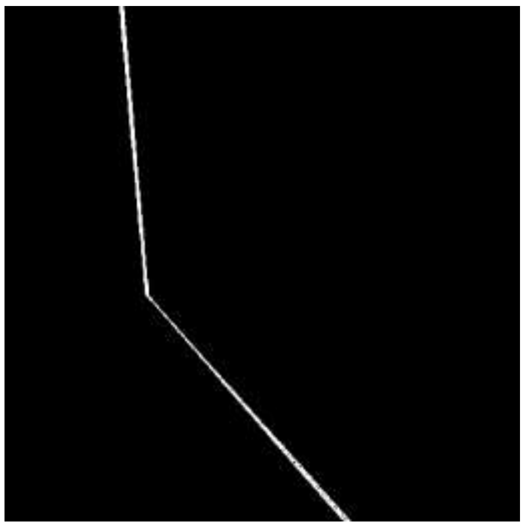
The image capture is being done with the OpenCV’s built in function cv2.videoCapture(). This function returns the image in form of a numpy array (n-dimensional array). This numpy array is then passed as argument to different function for further processing of the image.

2. Extraction of contours formed by the incident laser:

The numpy array generated in the above mentioned steps is applied with a mask. Only those pixels are masked out which within the range of RGB values specified by the user. For ease of visualisation and understanding, the pixels having some value in the mask are made all white and the rest of the useless pixels are made black. This is now stored in a .png format. A total of 48 .png files are generated at the end of scanning.



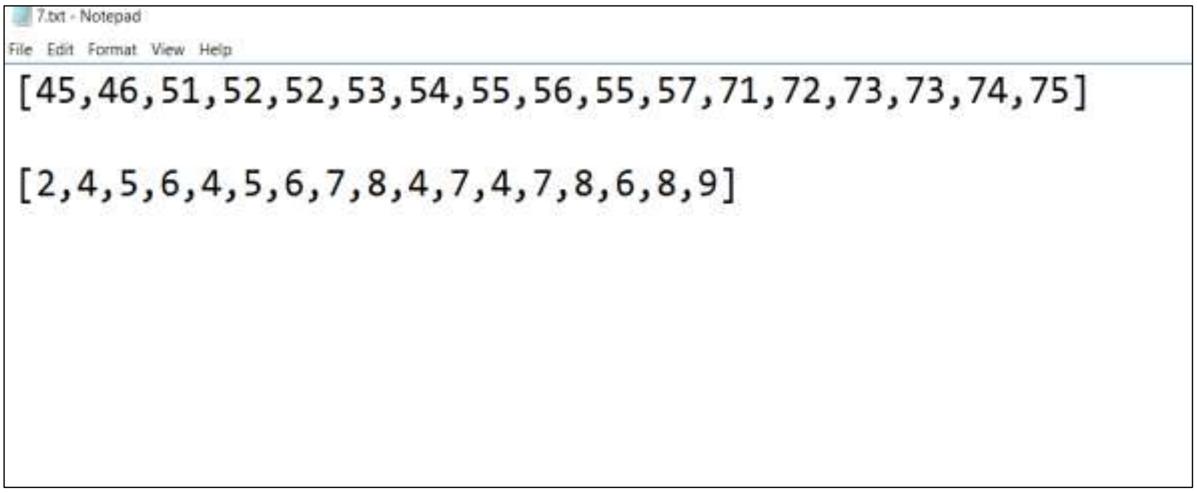
As captured by camera



After processing

3. Extraction of points from the contours:

The contours formed in the previous step contain the required 2D points. The Cartesian coordinates are stored in .txt files. A total of 48 .txt files are at the end of the scanning. These files contain the X coordinates followed by the Y coordinates.



```
[45,46,51,52,52,53,54,55,56,55,57,71,72,73,73,74,75]

[2,4,5,6,4,5,6,7,8,4,7,4,7,8,6,8,9]
```

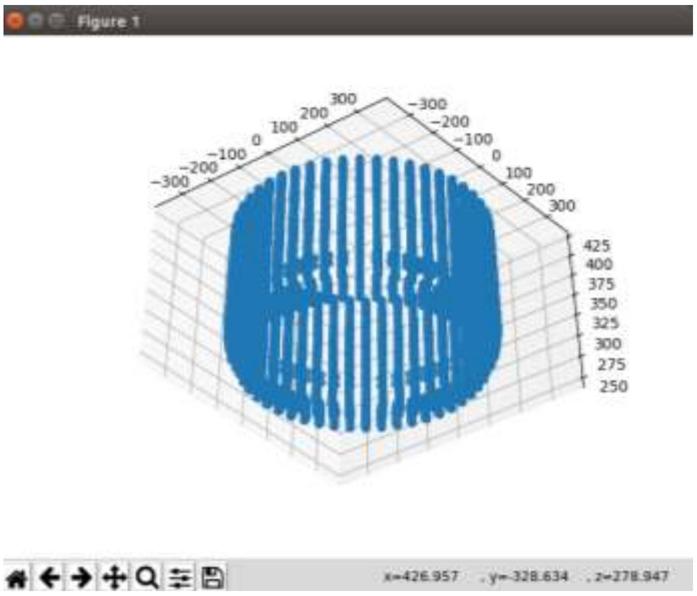
Sample .txt file

4. Calculation of spherical coordinates:

All the 48 .txt files are parsed and the 2D Cartesian coordinates are used to calculate the spherical coordinates of the final image. The final values of the R,Θ,ϕ are stored in three lists only during the runtime.

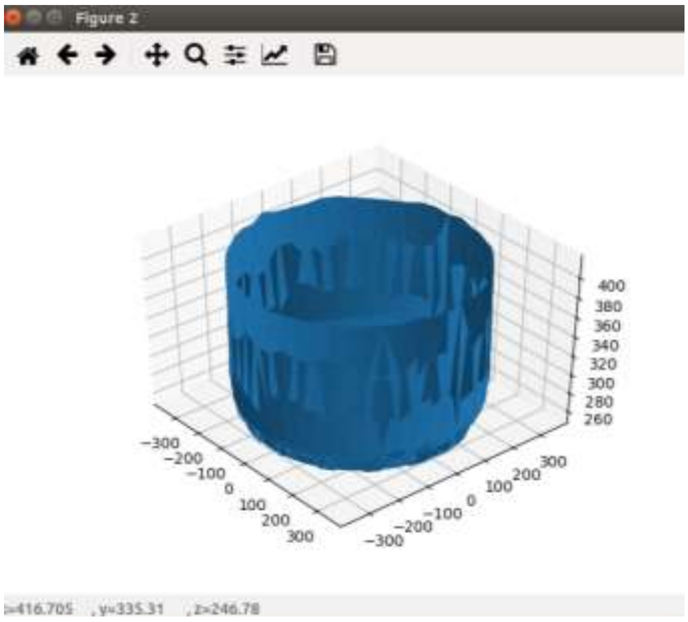
5. Plotting the point cloud:

The point cloud is plotted using the Scatter function of Matplotlib package. This opens a window showing the 3D graph of the point cloud in it. This figure can be stored to disk in a .png format.



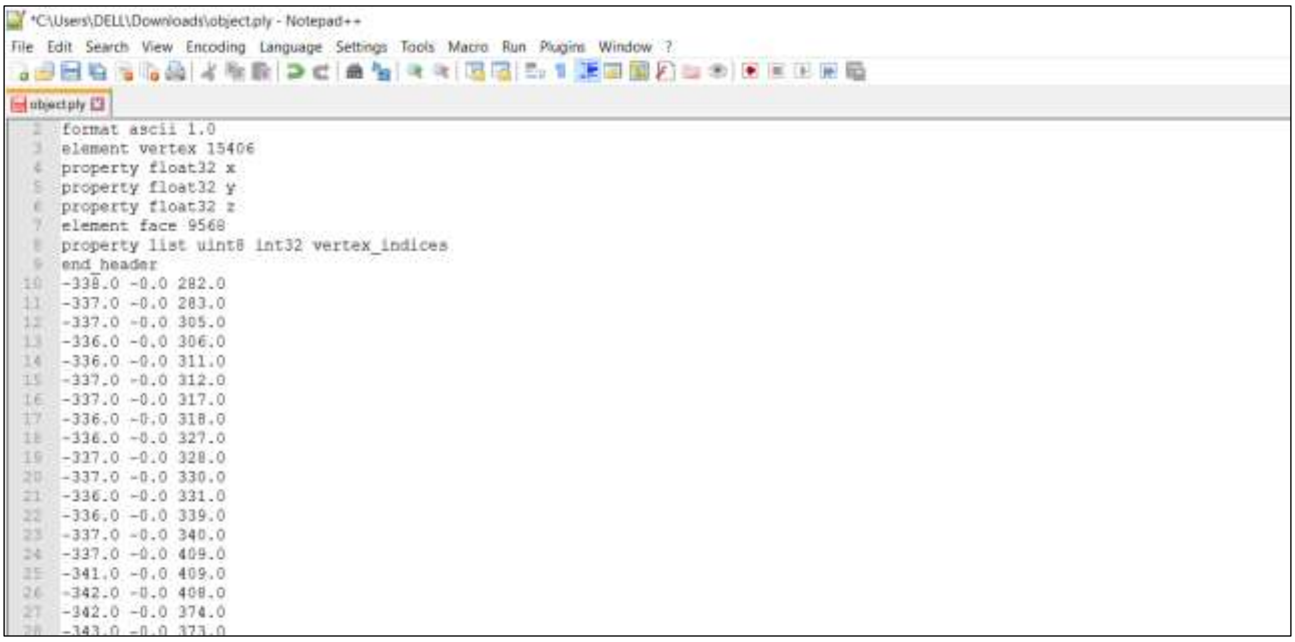
6. Plotting the surface plot:

The surface plot is plotted using the Trisurf function of the Matplotlib package. This opens a window showing the 3D graph of the scatter plot in it. This figure can also be stored in the disk in .png format.



7. Creating the final 3D format file:

The final file that contains all the information about the Cartesian coordinate is a .ply file (Polygon Format File). This gets generated when either the Scatter Plot or the Surface Plot is created.


*C:\Users\DELL\Downloads\object.ply - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
object.ply

```
format ascii 1.0
element vertex 15406
property float32 x
property float32 y
property float32 z
element face 9568
property list uint8 int32 vertex_indices
end_header
-338.0 -0.0 282.0
-337.0 -0.0 283.0
-337.0 -0.0 305.0
-336.0 -0.0 306.0
-336.0 -0.0 311.0
-337.0 -0.0 312.0
-337.0 -0.0 317.0
-336.0 -0.0 318.0
-336.0 -0.0 327.0
-337.0 -0.0 328.0
-337.0 -0.0 330.0
-336.0 -0.0 331.0
-336.0 -0.0 339.0
-337.0 -0.0 340.0
-337.0 -0.0 409.0
-341.0 -0.0 409.0
-342.0 -0.0 408.0
-342.0 -0.0 374.0
-343.0 -0.0 373.0
```

A Sample .ply file

Discussions

Present:

1. Interface:

The interface works very efficiently with minimal lag as far as the Main window is concerned. The subsequent windows work well too. The windows showing the Scatter plot and the Tri Surf plot face some lag due to a large number of points (Coordinates).

2. CPU utilization:

Current CPU utilization is in excess of 90% (as tested on an i5, 8GB RAM system). The .ply file generated is approximately 3-5 MB with the image files (.png) generated of size is approximately 10-20KB. The associated text files generated are approximately 50-100KB.

3. 3D Image formation:

The 3D imaging techniques used in the 3D Scanner are:

- Scatter Plot:

  A scatter plot (also called a scatter graph or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

- Tri Surf Plot:

  Tri-Surface plot is based on Surface triangulation, specifically Delaunay triangulation (refer Appendix E). This technique helps generate triangles using the coordinates passed as arguments.

4. Generation of faces:

Presently, there is no such algorithm for the generation of faces in the software developed.

5. Time lags in the setup:

Although the software developed for 3D scanning works fine. But further debugging is necessary to find out the avoidable delays that occur during the runtime.

6. 3D file format compatibility:

The software is generating a .ply file as of now. A ply file is one of the common 3D file formats and is being used widely. A lot of Open Source Software are capable of opening and manipulating .ply files.

7. Microcontroller:

The setup uses Arduino Uno for controlling the physical device i.e. the stepper motor.

8. Stepper Motor:

The setup uses a Universal motor (in the Unipolar mode) having7.5 degrees as the step angle and total steps of 48.

Scope:

1. Interface:

The interface can be made robust by introducing more efficiency in the present code. The number of points(Coordinates) generated can be reduced using suitable algorithms resulting in reduced or apparently negligible time lag in the Scatter plots and Tri Surf plots.

This is aimed to be achieved by:

- **Averaging the Coordinates of similar points:**

  A proposed technique to reduce the number of coordinates generated finally is by taking the average value of many points which occupy a very small region. This may be achieved by dividing the entire figure into small cubes of a quantum dimension and then taking the center point of the cube (nearly average) as the only point instead of considering all the points inside the cube.

- **Discarding the points within a given range:**

  Similar to the previous technique, a region containing unnecessarily a large number of points can be avoided.

2. CPU utilization:

The faces of the ply file can be generated with the help of Delaunay triangulation or a similar technique which can be used by trimesh() function of matplotlib for displaying the 3D image. After the faces are created for the ply file, it would be fit for use with Blender and other 3D image loading software.

3. 3D file format compatibility:

The software can also be programmed to generate any 3D file viz. .obj, .stl , etc.

4. Microcontroller:

Presently, all the capturing is being done by an external computer attached to the setup. If the code becomes a bit more efficient and requires less physical memory and processing load during the runtime, then the entire system can be implemented in an embedded computer (like Raspberry Pi or Beaglebone Black).

The microcontroller can also be changed according to the specifications. An MCU with fewer physical pins can also be used.

5. Stepper Motor:

The Stepper motor specifications can be varied as per requirement. Stepper motor with smaller step angle can be used which will increase the number of frames capture, hence increasing the accuracy of the 3D image captured. But again this would increase the number of points taken into consideration.

Further scope of improvement for the project:

1) **Development of a better algorithm for capturing 90^o corners:**
   The present code is not capable of detecting pointy edges and sharp 90^o corners. However it is not clear whether the resolution of the camera used in the scanning process will also have a major role in capturing it. A better algorithm should be developed for the same. For better results, two cameras positioned at different angles and heights can also be used.

2) **Reduction in the number of points captured inside the contour:**
   As of now, a lot of points are considered inside the contour. Due to this, finally a lot of unnecessary points get piled up and it consumes a lot of processing power as well as memory during the runtime. Hence the complexity of the process increases.
   If the number of points representing the 3D figure are somehow reduced by using a reduction algorithm, the load on the processor and the memory would be less.

3) **Construction of faces:**
   As of now, the software only finds out the 3D vertices (point cloud) of the object. But for construction of a solid figure, faces need to be created from the point cloud. Faces are formed by taking three points in a combination.

4) **Improvement of the GUI platform:**
   The developed GUI platform is simple and not very interactive. Further work needs to be done in improving it.

5) **Platform Independence:**
   The software developed currently can work with Linux only with all the dependencies installed. It is essential to develop the Windows, Android and iOS versions of the software.

Conclusion

The project statement was to develop a 3D Scanner using openCV and laser setup within a span of 8 weeks as one of the FOSSEE projects under the Ekalavya Summer Internship (2017) programme. The 3D Scanner has been designed, with much scope left for improvement and research. The purpose of the scanner is to scan real life models that can be used for various purposes, typically in Education, Healthcare and Archaeology. The design of the scanner has been kept quite simple and efficient. The use of Open Source technologies (both hardware and software) has enabled the project to flourish as an outstanding mark in the Open Source community in general and Open Source Hardware (OSH) community in particular.

The Scanner is developed using Open Source technologies with ample focus given on cost reduction contrary to the traditional 3D Scanners available in the industries. The approximate cost is around $23 (amounting to INR 1600). The scanner has been designed to be user friendly in its interface, portable, effective and cost efficient as a whole. The technical competency required for the modification of the setup has been kept low, allowing many electronics and technology savvy population to make changes in the codes so as to improve the performance and efficiency of the scanner.

As of now, the scanner is able to scan symmetric cylindrical bodies with agreeable accuracy and generate the surface plots and scatter plots. A lot more ideas were also proposed, forcing changes in the scratch-level codes. Due to lack of time, the team was unable to work out on these propositions. But there is always a scope for improvement and the team would like to work on the same, even after the internship ends.

Technologies used

References:

Alexander Mordvintsev ,Abid K
OpenCV-Python Tutorials Documentation, Jun 09, 2017
https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf

John W. Shipman
Tkinter 8.5 reference: a GUI for Python, 2013
http://infohost.nmt.edu/tcc/help/pubs/tkinter/tkinter.pdf

NumPy User Guide Release 1.11.0
https://docs.scipy.org/doc/numpy-1.11.0/numpy-user-1.11.0.pdf

stackoverflow
https://stackoverflow.com/

LiDAR based 3D Scanner
http://ieeexplore.ieee.org/document/7268968/?reload=true

ToF based 3D Scanner
http://www.spar3d.com/news/related-new-technologies/time-of-flight-vs-phase-based-laser-scanners-right-tool-for-the-job/

Sample .ply files
https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html

Matplotlib plotting tutorial
https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Delaunay Triangulation using Python
https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.Delaunay.html

Surface Triangulation
https://en.wikipedia.org/wiki/Delaunay_triangulation

Arduino related stuffs
https://www.arduino.cc/

Appendix A

Extraction of Red Component from the captured image

The extraction of the red component from the image is a crucial part of the project. The laser beam falls on the object and takes the outline of surface of the object.

The image captured by the camera is an RGB mesh. For each point, there is a corresponding RGB value.

Using OpenCV's built in functions, this goal was achieved.

Inside *Root.py*, this part of the code does the task:

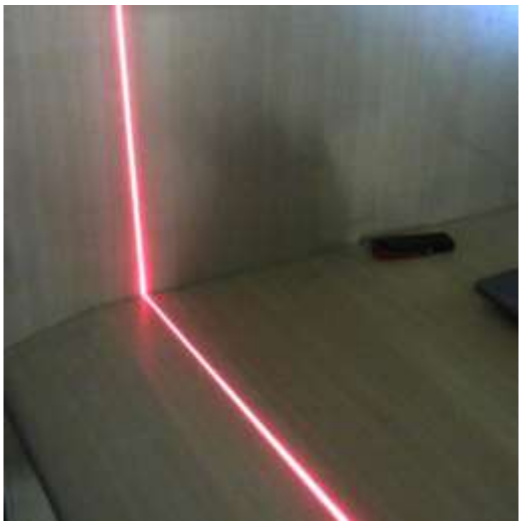
```
self.lower=numpy.array([self.bluelow,self.greenlow,self.redlow])
self.upper=numpy.array([self.blueup,self.greenup,self.redup])
self.mask=cv2.inRange(self.frame, self.lower, self.upper)
```

self.lower is a numpy array containing the lower values of RGB
self.upper is a numpy array containing the upper values of RGB
By default, self.lower is ([240,150,150]) and self.upper is ([255,255,255])
These values can also be changed from the main window of the software as per the requirements.
A mask is created that contains only those points whose RGB values fall in the range specified. This mask is created using cv2.inRange() function.

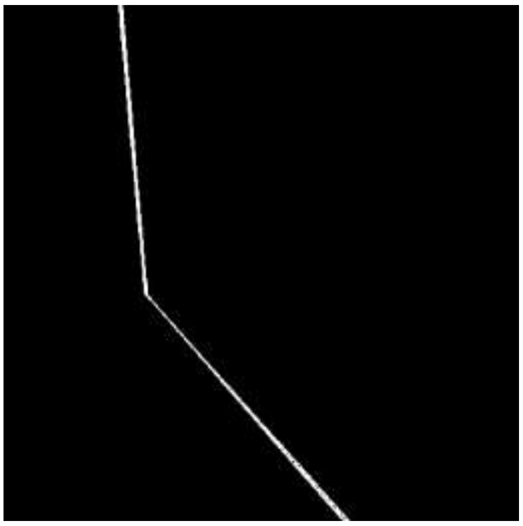
```
self.output_img[np.where(self.mask==0)] =0
self.output_img[np.where(self.mask>100)] =255
```

Then to make the image more usable and less space consuming, it is converted into grayscale and those regions where the contours lie are mode White (255 in Grayscale). Those regions which don't contain any contours are made Black (0 in Grayscale).

As a result, the final output is something like this:



As captured by camera 1



After processing

Appendix B

Calculation of the points for the 3D point cloud

The camera captures only the 2D image. Hence from each frame, only the X and the Y coordinates of the required points can be extracted. But the final image has to be a 3D object. For producing a 3D object, we require another dimension. It is a mammoth task to calculate the depth of the point using a normal Camera. LiDAR based cameras are generally used for achieving the same.

An effective solution to that was to calculate the coordinates of the point using spherical coordinates. It is easy to calculate the coordinates in the spherical coordinate system as we can calculate the R,Θ of a point in the frame from its X,Y values and the ϕ value from the frame count.

$$\boldsymbol{R} = \sqrt{\boldsymbol{x^2} + \boldsymbol{y^2}}$$

$$\boldsymbol{\Theta} = \boldsymbol{tan}^{\text{-}\boldsymbol{1}}(\boldsymbol{y}/\boldsymbol{x})$$

The stepper has been programmed to take 48 steps in total to cover 360^0. The final frame count is 48. Hence the stepper covers an angle of 7.5^0 in each step. From this data, the ϕ value can be calculated.

Suppose the frame count is 3, then ϕ=$\boldsymbol{3} * \boldsymbol{7.5}^0$= 22.5^0

Similarly, If the frame count is 12, ϕ=90^0 and if frame count is 42, ϕ= 315^0

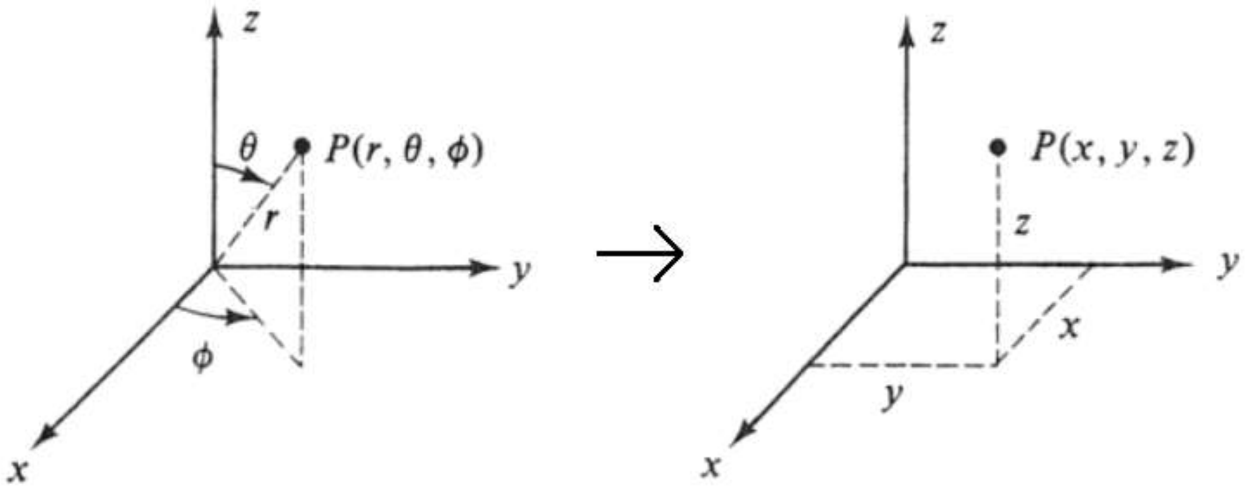


Figure 4

Appendix C

Controlling the Stepper Motor

Since the coordinates of the points is calculated using the spherical coordinate system, the precise rotation of the stepper is very essential for the precise calculation of ϕ. For this, a stepper motor was used to precisely rotate the object in steps.

A generic unipolar stepper motor is used for this purpose. Arduino UNO is used to control the stepper motor. It is programmed for serial communications with the main computer.

It is programmed to make the stepper motor rotate a step when it receives serial input of ‘1’ and after completing the step, it would send ‘.’ .

This two way system of communication was implemented to make sure that no steps get missed during the runtime.

A pre-built library ‘Stepper.h’ was used with the Arduino IDE for the same.

Designing the current amplifier for the stepper motor

Arduino UNO is programmed to decide the logic or the sequence in which the four pins will be HIGH or LOW. It cannot supply the required amount of current. So, either a motor driver is used or a self-designed current amplifier can also be used.

ULN2003, current driver IC is used for this purpose.

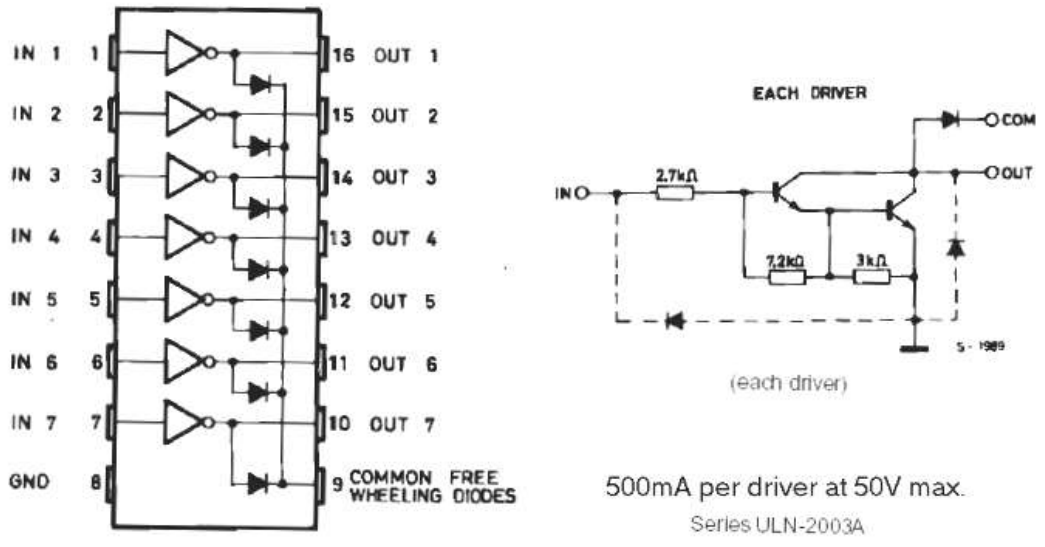
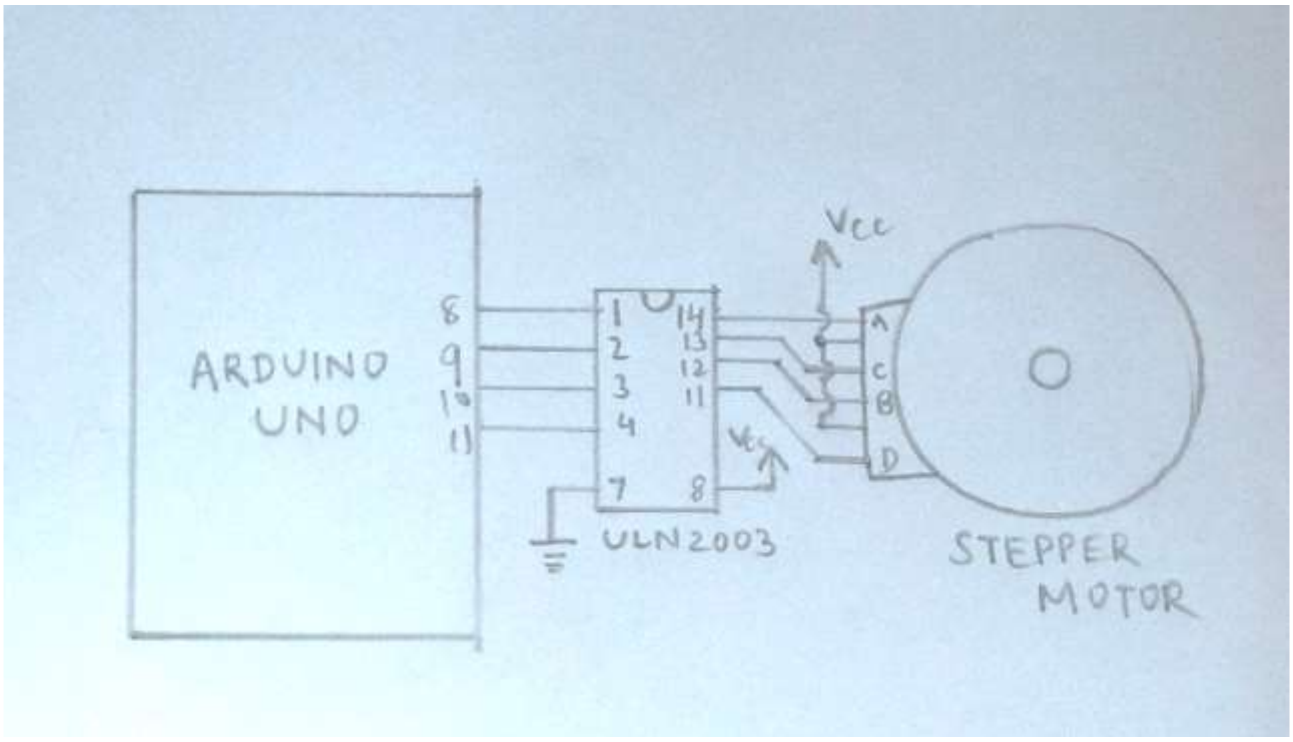


Figure 5

Circuit diagram for wiring the stepper motor on breadboard



Hardware Schematic

Appendix D

Delaunay Triangulation:

Surface Triangulation:

Triangulation of surface means:

- a net of triangles, which covers a given surface partly or totally, or
- The procedure of generating the points and triangles of such a net of triangles.

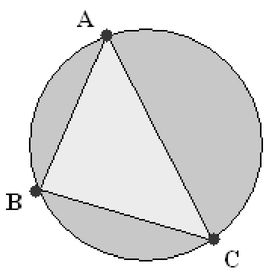


Figure 6

The circumcircle of a triangle is the circle that passes through all of the triangle's vertices (A, B, C).

Methods of Triangulation:

1. Dividing the 3D region into cubes:
   This method divides the 3D region of consideration into cubes and determines the intersections of the surface with the edges of the cubes in order to get polygons on the surface, which thereafter have to be triangulated (cutting cube method).
2. Marching method:
   In this method, the triangulation starts with a triangulated hexagon at a starting point. This hexagon is then surrounded by new triangles, following given rules, until the surface of consideration is triangulated. If the surface consists of several components, the algorithm has to be started several times using suitable starting points

Delaunay Triangulation:

Delaunay Triangulation is a computational method for triangulation of "P" points in a plane. A Delaunay triangulation for a set P of points in a plane is a triangulation DT (P) such that no point in P is inside the circumcircle of any triangle in DT (P).A Delaunay triangulation in the plane with circumcircles is shown in the figure.

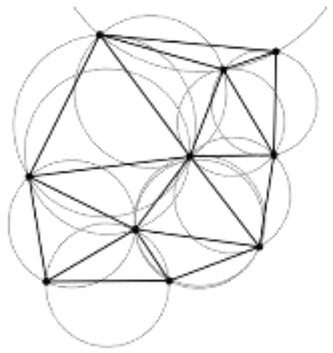


Figure 7

Algorithms:

1. Flip Algorithms
2. Incremental
3. Divide and Conquer
4. Sweep Hull

Applications in Image Processing:

Advantage:

The advantage of using Delaunay triangulation is that it maximizes the minimum angles of the triangles. In this way, the triangles tend toward equiangularity, which avoids having triangles that are very long and thin. Therefore, the resulting triangulation looks geometrically balanced. Aside from equiangularity, Delaunay triangulation is particularly non-restrictive. Thus, it is ideal for interpolation algorithms, which attempt to avoid introducing distortions.

Delaunay Triangulation in Python:

classscipy.spatial.Delaunay is a class defined in scipy package's spatial module that is used for Delaunay Triangulation.

Sample code:

```
>>>points = np.array([[0, 0], [0, 1.1], [1, 0], [1, 1]])
>>>fromscipy.spatial import Delaunay
>>>tri = Delaunay(points)
>>> import matplotlib.pyplot as plt
>>>plt.triplot(points[:,0], points[:,1], tri.simplices.copy())
>>>plt.plot(points[:,0], points[:,1], 'o')
>>>plt.show()
>>>tri.simplices

array([[3, 2, 0],
    [3, 1, 0]], dtype=int32)
>>>points[tri.simplices]

array([[[ 1. ,  1. ],
[ 1. ,  0. ],
[ 0. ,  0. ]],
    [[ 1. ,  1. ],
[ 0. ,  1.1],

[ 0. ,  0. ]]])
```

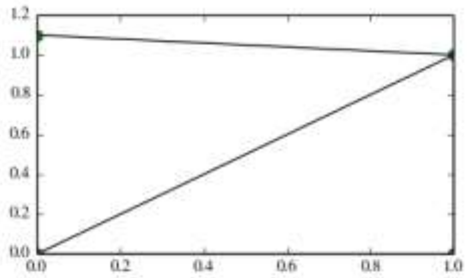


Figure 8

Here, the tri.simplices returns a numpy array of integers indicating the indices of points forming the simplices in the triangulation. The points generated are oriented counter clockwise by default.Point indices and coordinates for the two triangles forming the triangulation are shown in the figure.

Appendix E

Plotting of the Scatter Plot and the Surface Plot

Finally after the completion of the calculation of all the points, the next step is to plot those. Although it is easy to plot the point cloud or the Scatter plot, but it is a mammoth task to plot the surface plot. But the use of the available library makes it easy to plot both of these.

For Scatter Plots:

The scatter() function is used to plot the point cloud. X, Y, Z are three lists that are given as arguments to this function. The three lists contain the Cartesian coordinates of the points in the point cloud.

Code:

```
fig=plt.figure()
ax=fig.add_subplot(111, projection='3d')
SavePLY.SavePLY(X,Y,Z)
ax.scatter(X, Y, Z)
plt.show()
```

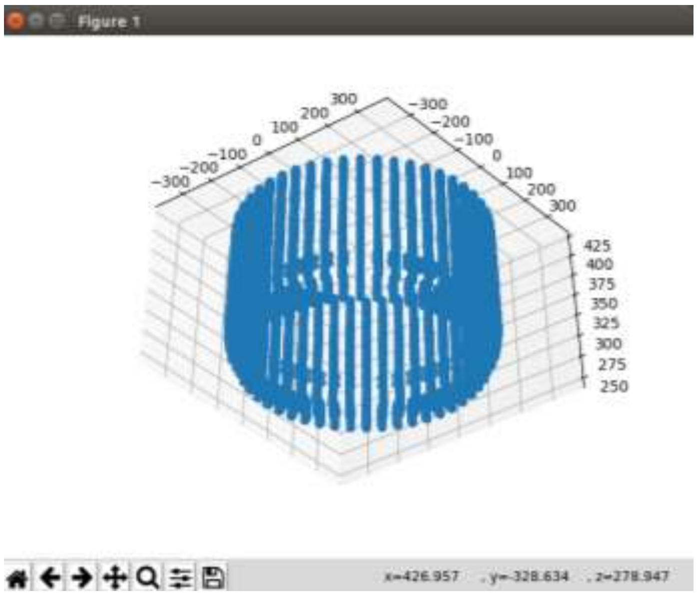


Figure 9

For Surface Plots:

It is a difficult task to generate the surface plots from scratch as it is difficult to arrange the index of the points in an order to make faces. But the task becomes simplified with Matplotlib'strisurf function.

Code:

```
fig=plt.figure()
ax=fig.add_subplot(111, projection='3d')
SavePLY.SavePLY(X,Y,Z)
ax.scatter(X, Y, Z)
plt.show()
```

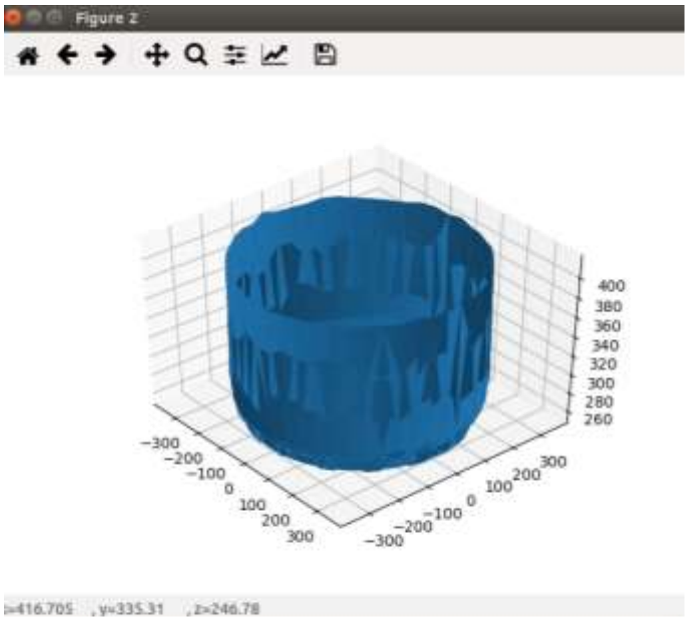


Figure 10

Appendix F

Generation of .PLY (3D) Files

PLY is a computer file format known as the Polygon File Format or the Stanford Triangle Format. It was principally designed to store three-dimensional data from 3D scanners. The data storage format supports a relatively simple description of a single object as a list of nominally flat polygons. A variety of properties can be stored, including color and transparency, surface normals, texture coordinates and data confidence values. The format permits one to have different properties for the front and back of a polygon. There are two versions of the file format, one in ASCII, the other in binary.

File format

Files are organized as a header, that specifies the elements of a mesh and their types, followed by the list of elements itself. The elements are usually vertices and faces, but may include other entities such as edges, samples of range maps, and triangle strips.

The header of both ASCII and binary files is ASCII text. Only the numerical data that follows the header is different between the two versions.

In computer programming, the term magic number has multiple meanings. It could refer to one or more of the following:

The header always starts with a "magic number". Magic number can be

1. A constant numerical or text value used to identify a file format or protocol; for files, see List of file signatures.

2. Distinctive unique values that are unlikely to be mistaken for other meanings (e.g., Globally Unique Identifiers).

3. Unique values with unexplained meaning or multiple occurrences which could (preferably) be replaced with named constants.

After the ply **header**, the description of the file type is made by anyone of the following-

format ascii 1.0
format binary_little_endian 1.0
format binary_big_endian 1.0

The **'element'** keyword introduces a description of how some particular data element is stored and the quantity. Hence, in a file where there are 12 vertices, each represented as a floating point (X,Y,Z) triple, one would expect to see:

element vertex 12

The no. of vertex can be easily found out by the length of X co-ordinate or Y co-ordinate or Z co-ordinate.

property float x
property float y
property float z

Other 'property' lines might indicate that colours or other data items are stored at each vertex and indicate the data type of that information. Regarding the data type there are two variants, depending on the source of the ply file. The type can be specified with one of char, uchar ,short, ushort, int, uint ,float, double, or one of int8 ,uint8, int16, uint16, int32, uint32, float32, float64.

For an object with ten polygonal faces, one might see:

element face 10
property list ucharintvertex_indices

The word 'list' indicates that the data is a list of values, the first of which is the number of entries in the list (represented as an 'uchar' in this case). In this example each list entry is represented as an 'int'. At the end of the header, there must always be the line:

end_header

After the end of the header, the values of the vertices and then the values of the faces are specified.