Term Project 2

3D Reconstruction using COLMAP

CSE586 Spring 2020

Team Members:

Soumitra Mehrotra (sxm6256)

Umar Farooq Mohammad (ubm5020)

Shreyas Hervatte Santosh (sjh6186)

Vikramaditya Poddar (vbp5094)

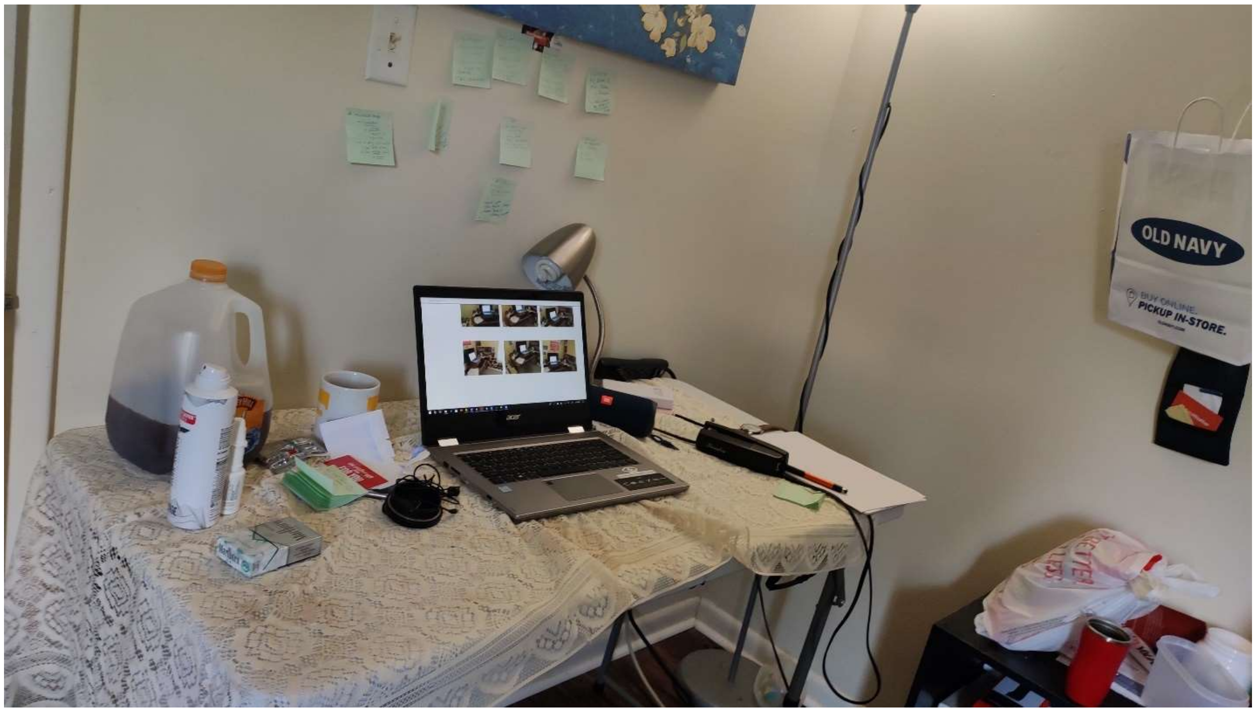
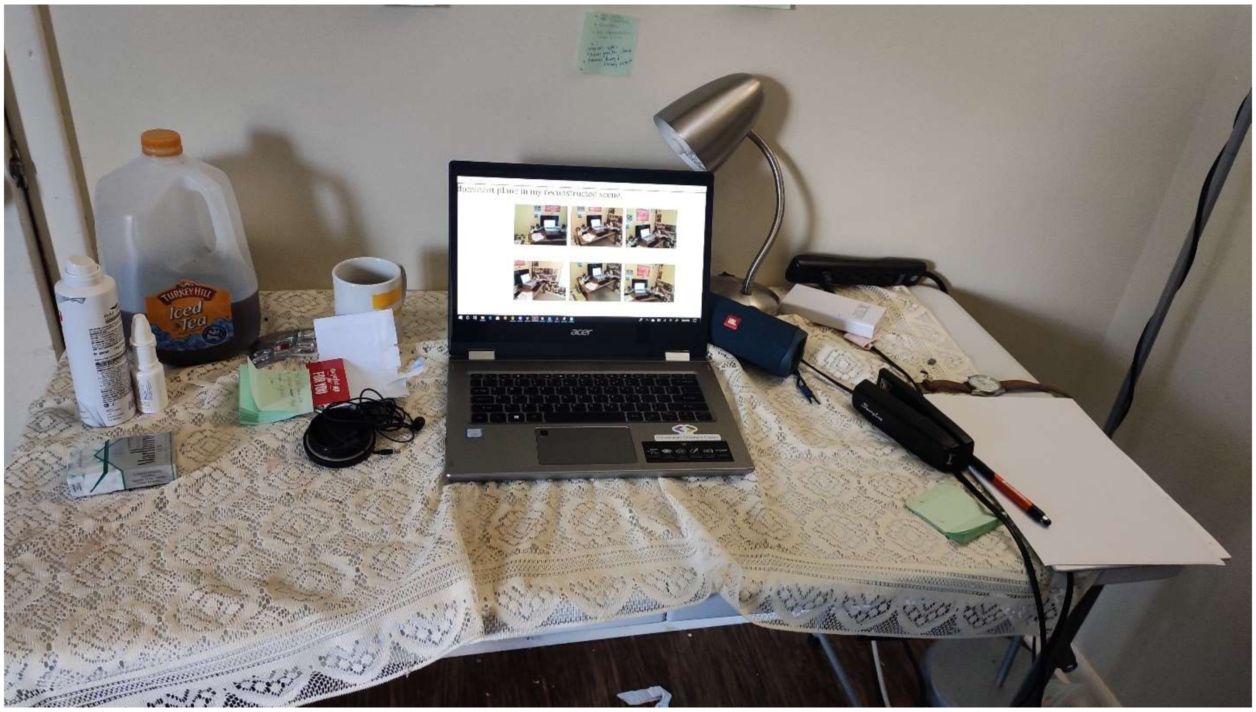
Implementation details

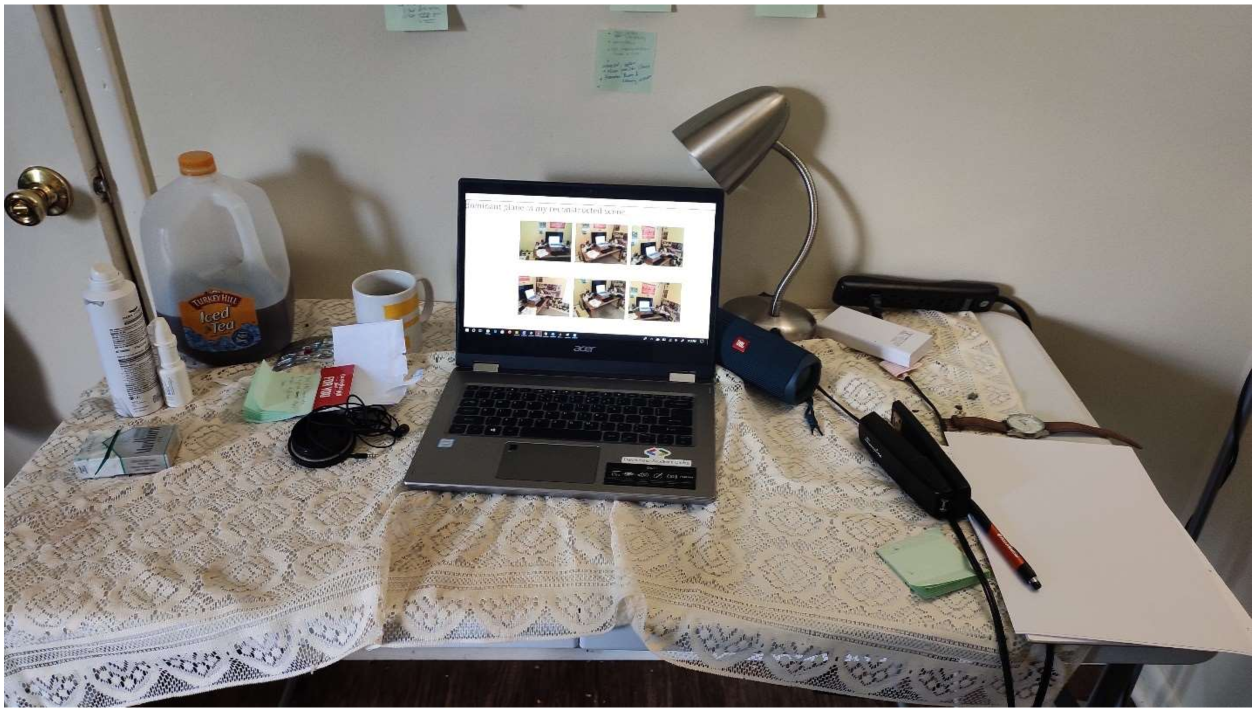
Step 1:

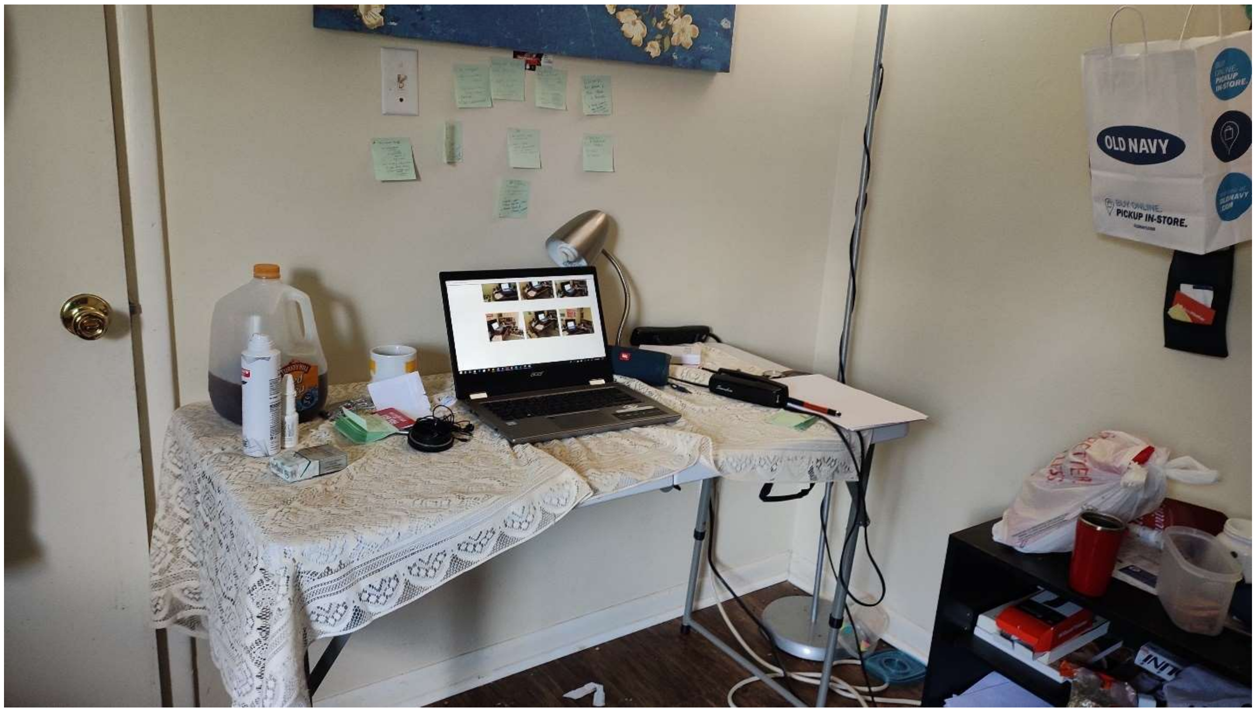
The initial plan for this step was to record a video of the target plane and then extract the frames of the video so that there is significant redundancy between images. The code for this was done in Python and the function looks like follows

```
import cv2
  # Function to extract frames
def FrameCapture(input_path, out_path):
      # input_path to video file
      vidObj = cv2.VideoCapture(input_path)
      # Used as counter variable
      count = 0
      # checks whether frames were extracted
      success = 1
      while success:
            # vidObj object calls read
                  success, image = vidObj.read()
            # Saves the frames with frame-count
                  cv2.imwrite(out_path+"frame%d.jpg" % count, image)
                  count += 1
```

However, we later realized that the performance was significantly better when we used fewer deliberately photographed scenes rather than using video frames. So, we subsequently chose to use fewer images. Few of the photographs we used are shown below. The table is intended to be the dominant plane. A tablecloth with a pattern was used for better corner detection.

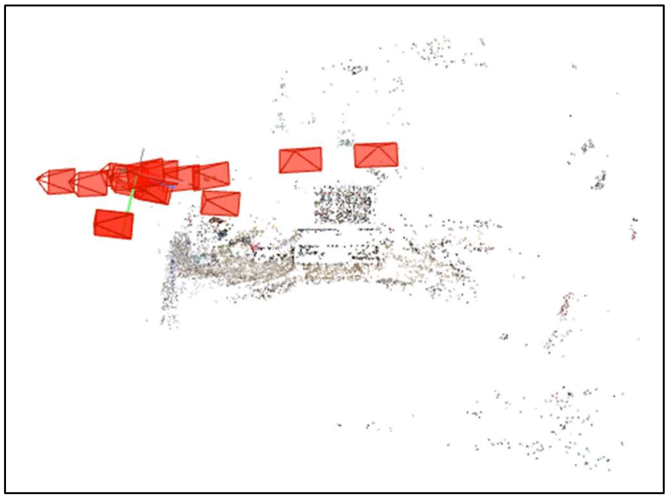




Step 2:

While attempting to run COLMAP on Windows machines by building by source, we came across several compatibility issues with all the dependencies. This was persistent across all our systems and other students who were using Windows PC informed us that they had similar issues. We therefore had to resort to running a pre-built GUI version of COLMAP through the bat file found at https://demuc.de/colmap/.

We then proceeded with the other sub-steps to generate the 3D model of the scene and exported it as a text file. The final 3D model looks like follows:



Step 3:

This is a straightforward step, we extracted the X, Y, Z coordinates from the 3D point cloud generated by COLMAP. We preferred exporting the point cloud as .PLY file instead of .txt file as it was easy to load and work with. The code for this looks as follows:

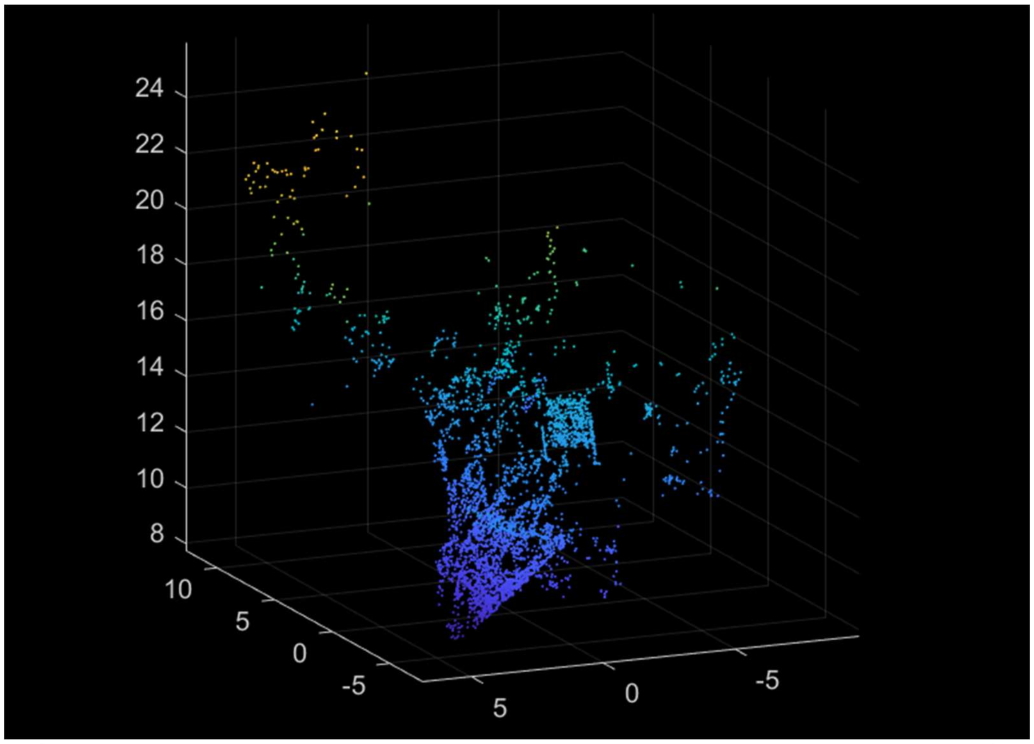
```
pcread('points3D.ply')
```

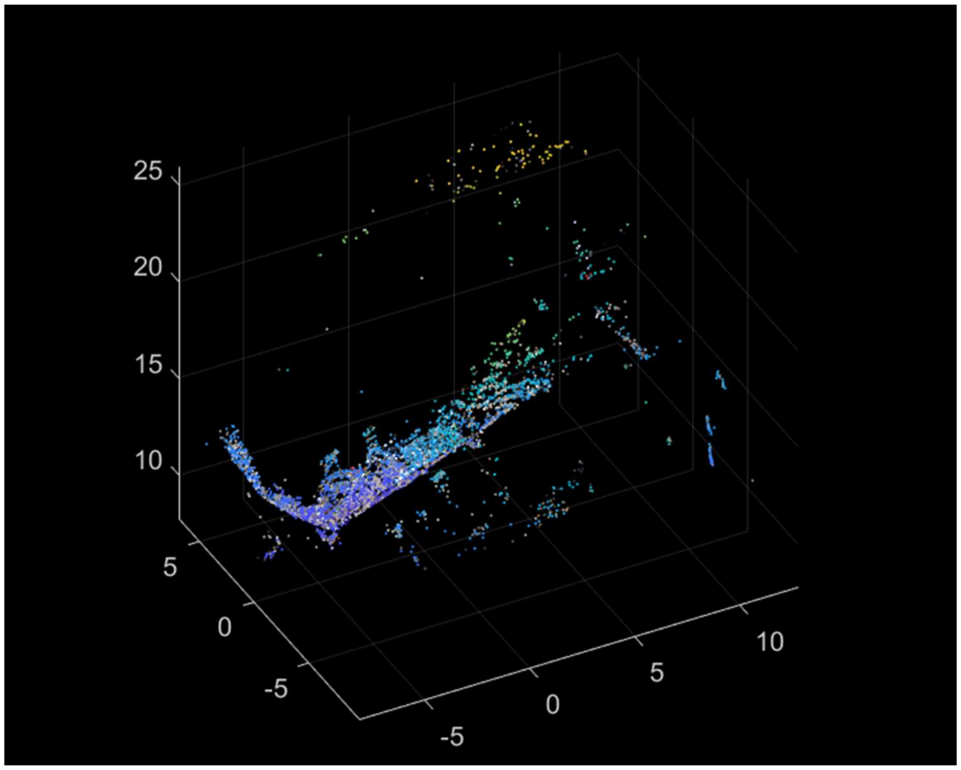
Step 4:

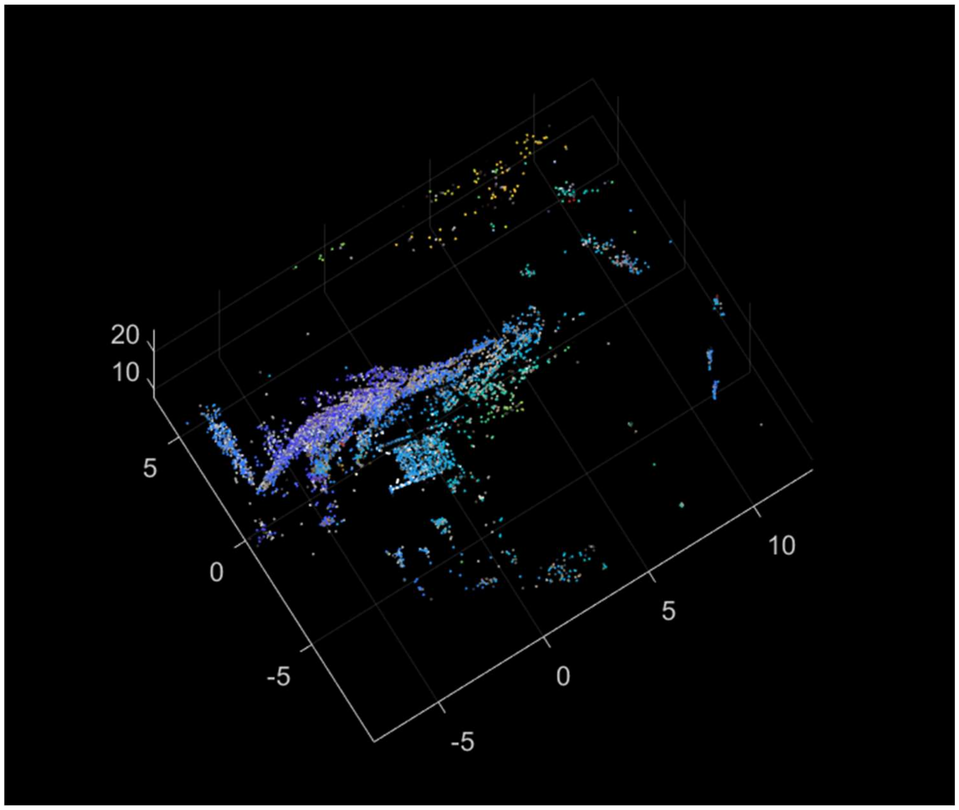
The code files we used for implementing RANSAC is given below:

Least_Squared_error.m Get_BestFit_Plane.m Visualize_BestPlane_Features.m

Upon successful implementation, the inlier points we obtained are plotted below:

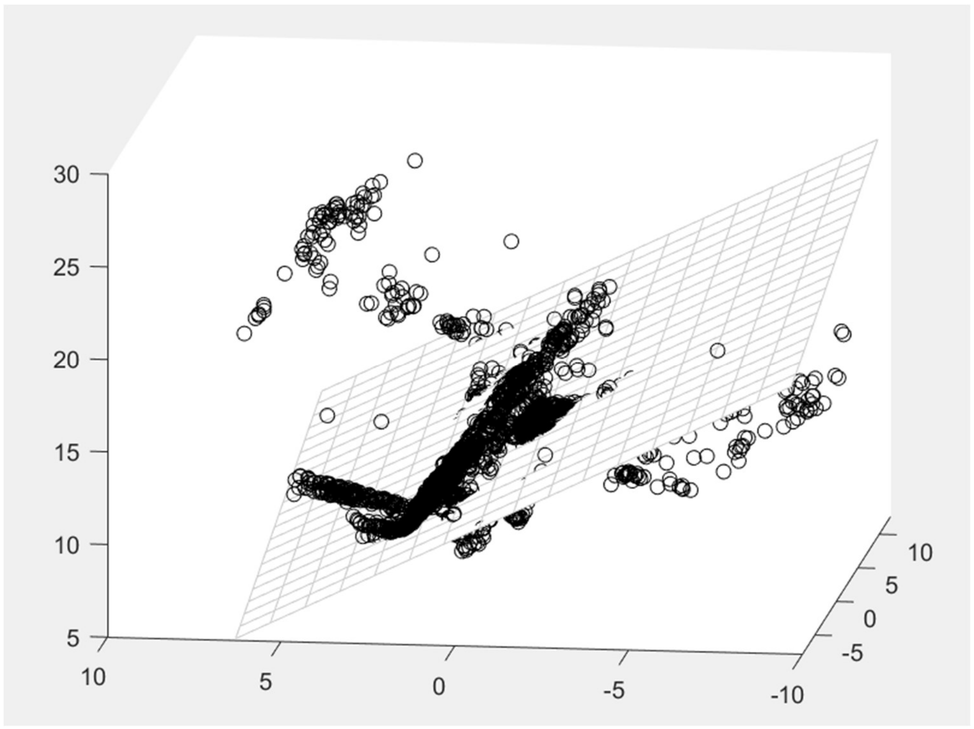

24
22
20
18
16
14
12
10
8
10
5
0
-5
5
0
-5

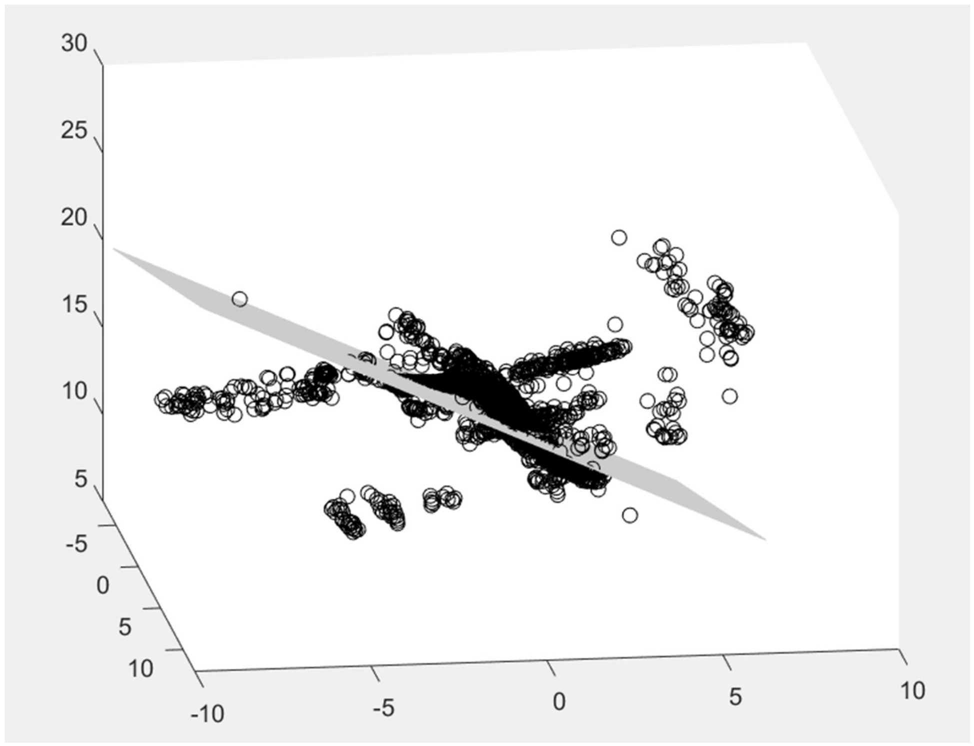

25
20
15
10
5
0
-5
-5
0
5
10

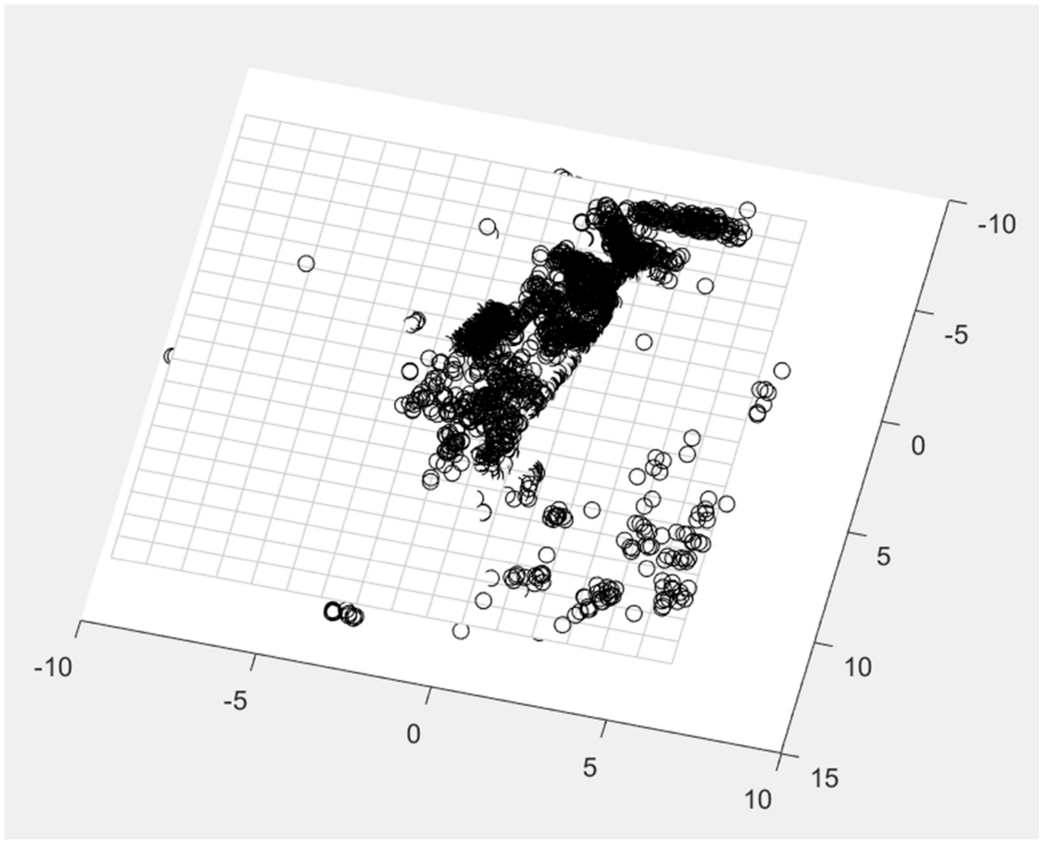

20
10
5
0
-5
-5
0
5
10

Step 5:

The dominant plane obtained from the code above looks like follows. This corresponds to the table in the picture.







Step 6:

To perform the calculations for rotation and translation matrix, we used to following theory (Ref: https://math.stackexchange.com/questions/1167717/transform-a-plane-to-the-xy-plane):

The equation of the plane is $ax+by+cz+d=0$. The translation vector from the local coordinate system to the world coordinates would be

$$\vec{t} : (x, y, z) \rightarrow (x, y, z - d/c)$$

The rotation vector would be

$$\begin{pmatrix} \cos\theta + u_1^2(1-\cos\theta) & u_1u_2(1-\cos\theta) & +u_2\sin\theta \\ u_1u_2(1-\cos\theta) & \cos\theta + u_2^2(1-\cos\theta) & -u_1\sin\theta \\ -u_2\sin\theta & u_1\sin\theta & \cos\theta \end{pmatrix}$$

Where

$$cos\theta = \frac{c}{\sqrt{a^2+b^2+c^2}} \quad u_1 = \frac{b}{\sqrt{a^2+b^2+c^2}}$$

$$sin\theta = \sqrt{\frac{a^2+b^2}{a^2+b^2+c^2}} \quad u_2 = -\frac{a}{\sqrt{a^2+b^2+c^2}}$$

The code for this part is available in the following files:

GenerateRotationMatrix.m Translate_Cloudpoints_2D.m

Step 7:

To create our 3D object, a cube in our case, the code looks as follows:

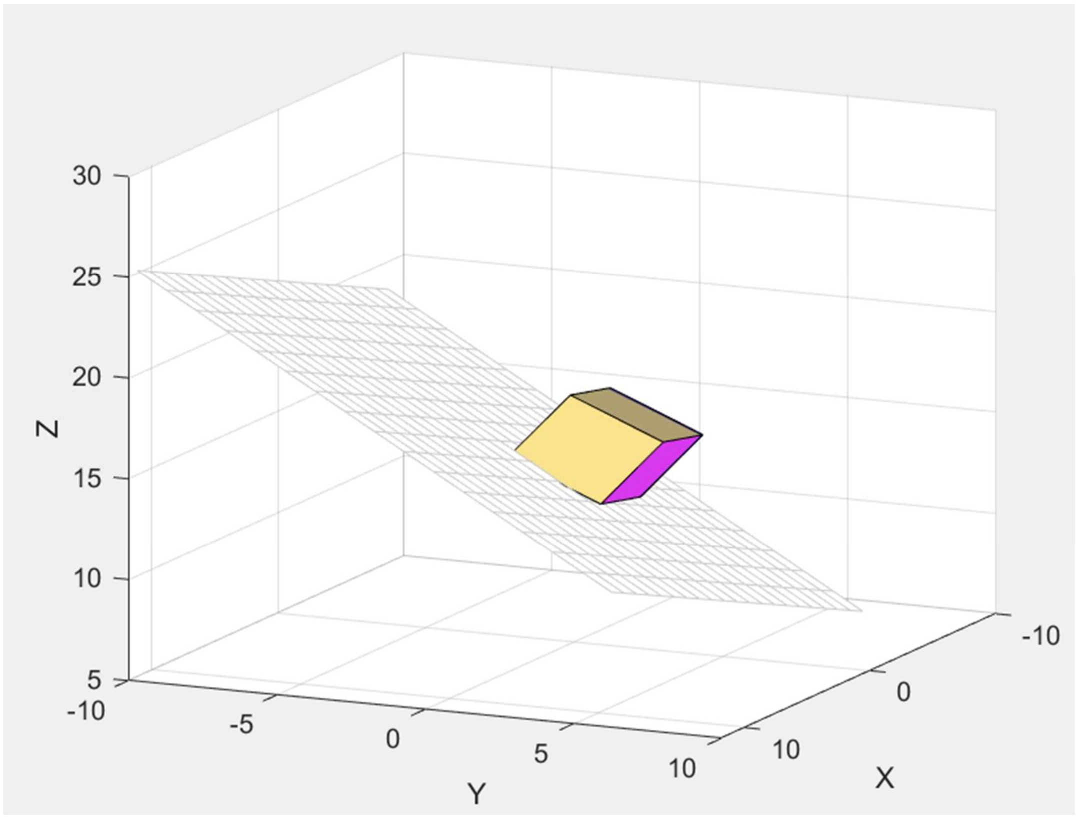
```
function poly_rectangle(p1, p2, p3, p4)
% The points must be in the correct sequence.
% The coordinates must consider x, y and z-axes.
x = [p1(1) p2(1) p3(1) p4(1)];
y = [p1(2) p2(2) p3(2) p4(2)];
z = [p1(3) p2(3) p3(3) p4(3)];
fill3(x, y, z, rand(size(p1)));

hold on
end
```

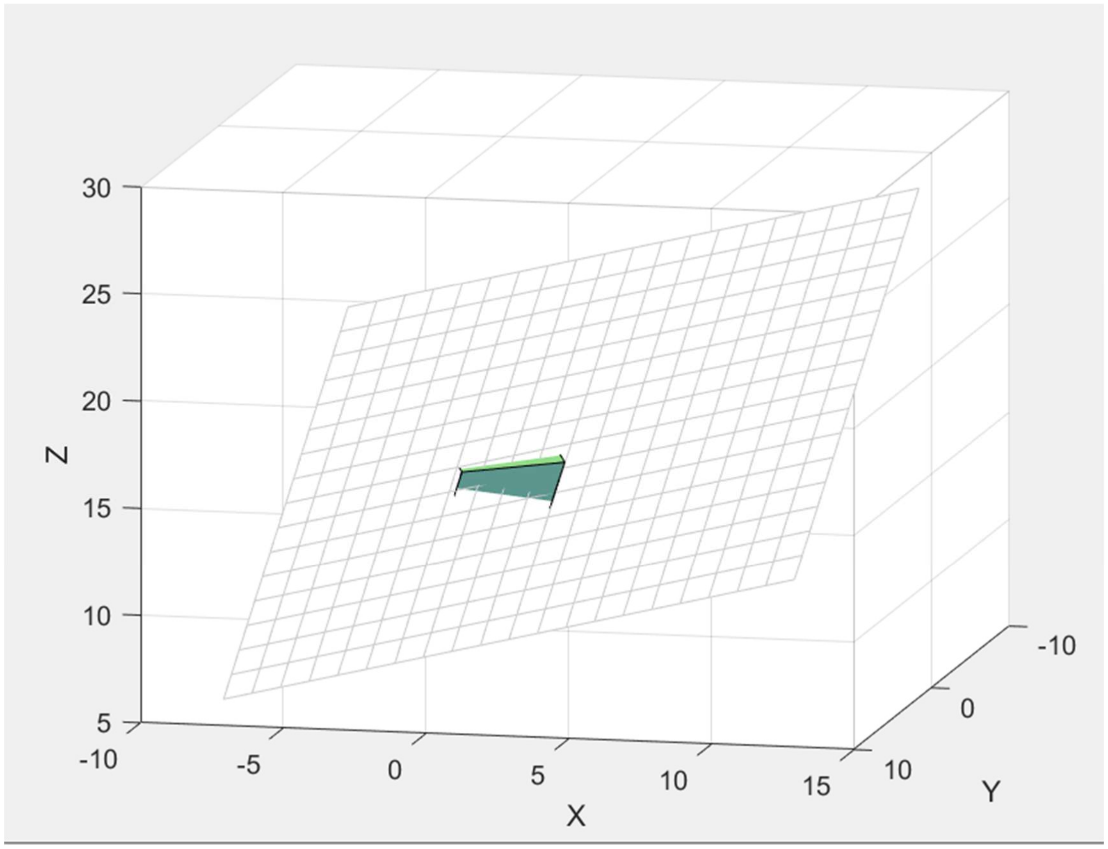
The above code creates a single side of the cube so, we call this function for 6 times with different coordinates In order to achieve a cube object in 3D plane. The snippet for this task is below:

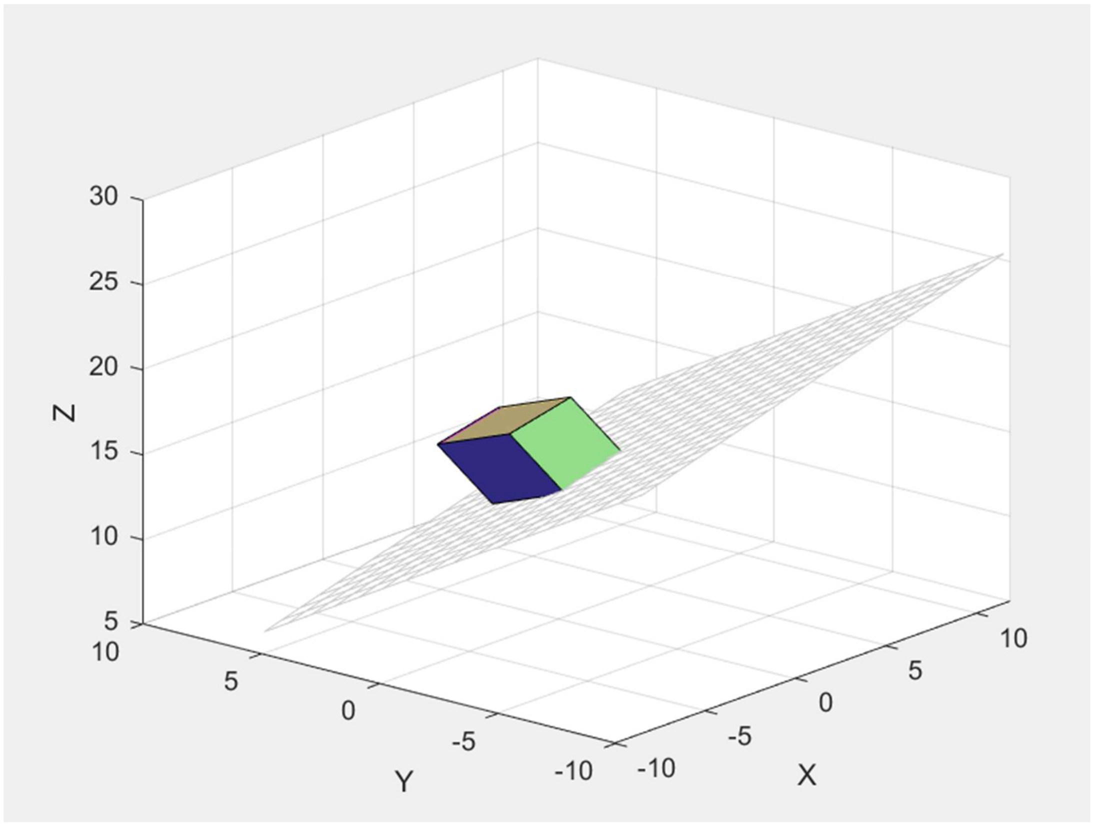
```
poly_rectangle(TranslatedObject(1,:),TranslatedObject(2,:),TranslatedObject(3,:),TranslatedObject(4,:))
poly_rectangle(TranslatedObject(5,:),TranslatedObject(6,:),TranslatedObject(7,:),TranslatedObject(8,:))
poly_rectangle(TranslatedObject(1,:),TranslatedObject(2,:),TranslatedObject(6,:),TranslatedObject(5,:))
poly_rectangle(TranslatedObject(2,:),TranslatedObject(3,:),TranslatedObject(7,:),TranslatedObject(6,:))
poly_rectangle(TranslatedObject(3,:),TranslatedObject(4,:),TranslatedObject(8,:),TranslatedObject(7,:))
poly_rectangle(TranslatedObject(4,:),TranslatedObject(1,:),TranslatedObject(5,:),TranslatedObject(8,:))
```

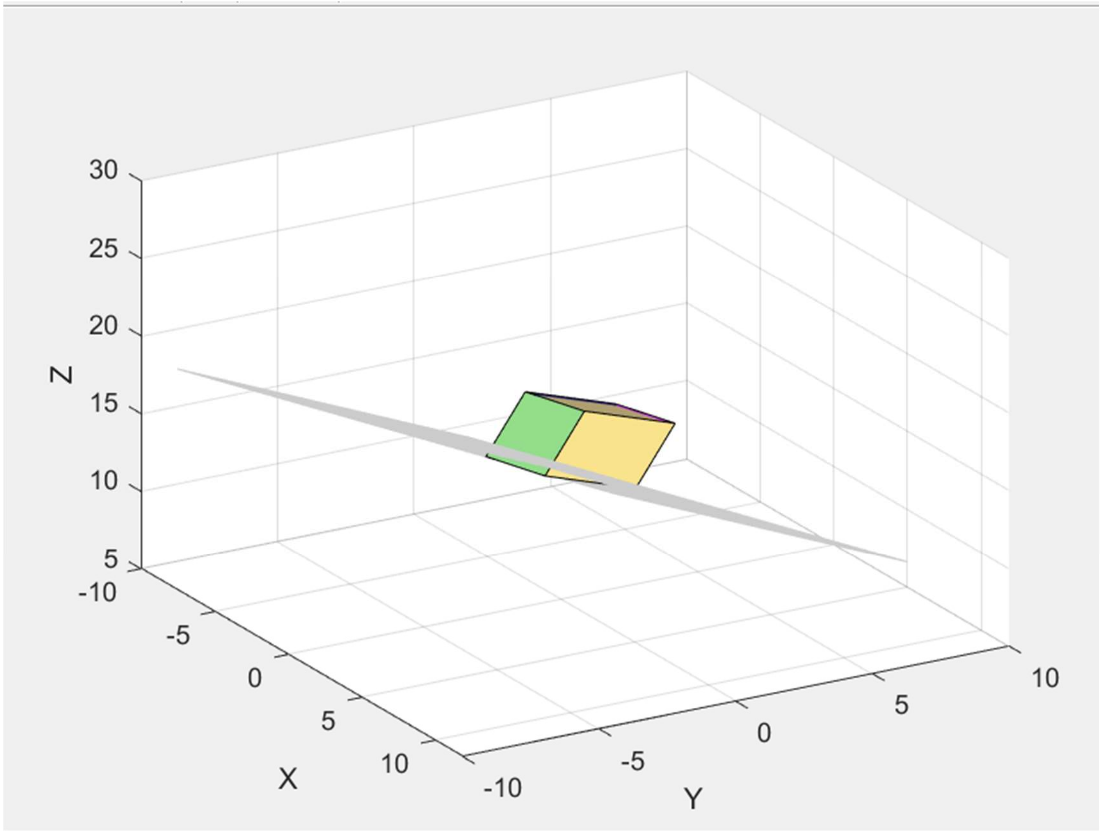
Where the parameter “TranslatedObject” contains all the 8 coordinates of the required cube in the 3D plane coordinate system X, Y , Z.

The resultant cube looks like follows:


Z
Y
X
30
25
20
15
10
5
-10
-5
0
5
10
-10
0
10


Z
X
Y
30
25
20
15
10
5
-10
-5
0
5
10
15
-10
0
10


Z
30
25
20
15
10
5
Y
10
5
0
-5
-10
X
-10
-5
0
5
10


Z
30
25
20
15
10
5
X
-10
-5
0
5
10
Y
-10
-5
0
5
10

Step 8:

The code for reading the parameters from the text file is available in the following file:

camera_parameters.
m

The code takes reads the internal camera parameters (SIMPLE_RADIAL model) and the external parameters of each image by reading the cameras.txt and images.txt file line by line. The function uses quat2rotm MATLAB function to convert quaternion to rotation matrix.

Step 9:

The code linked above read internal and external parameters including Rotation matrix, Translation Matrix, world coordinates (X, Y, Z) and A matrix to give the coordinates of the projection point in pixels (u,v). The equation is given by –

$$s\ m' = A[R|t]M'$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

For SIMPLE_RADIAL camera configuration $f_x = f_y$.

$c_x, c_y = principal\ point$

Step 10:

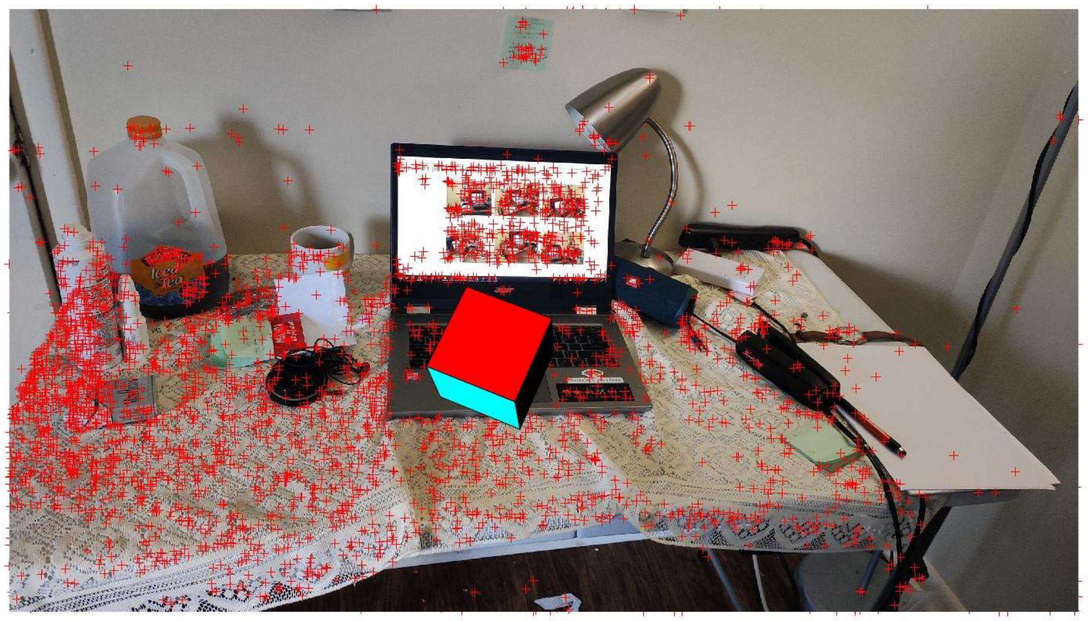
The code for overlaying the object pixels on the input image is in this file :

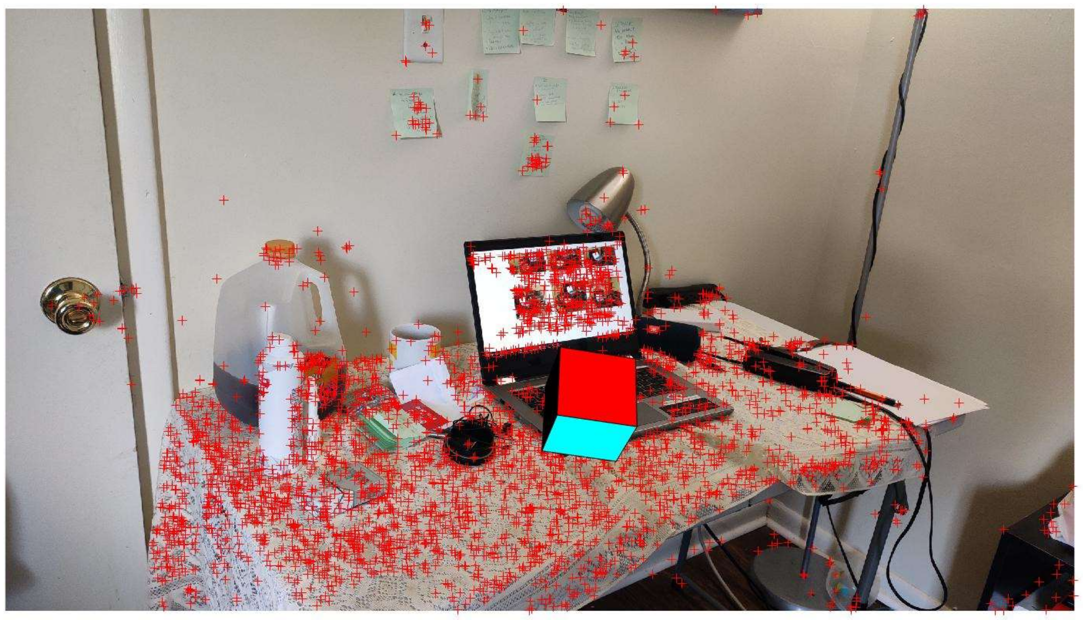
VisualizeObjectonImage.m

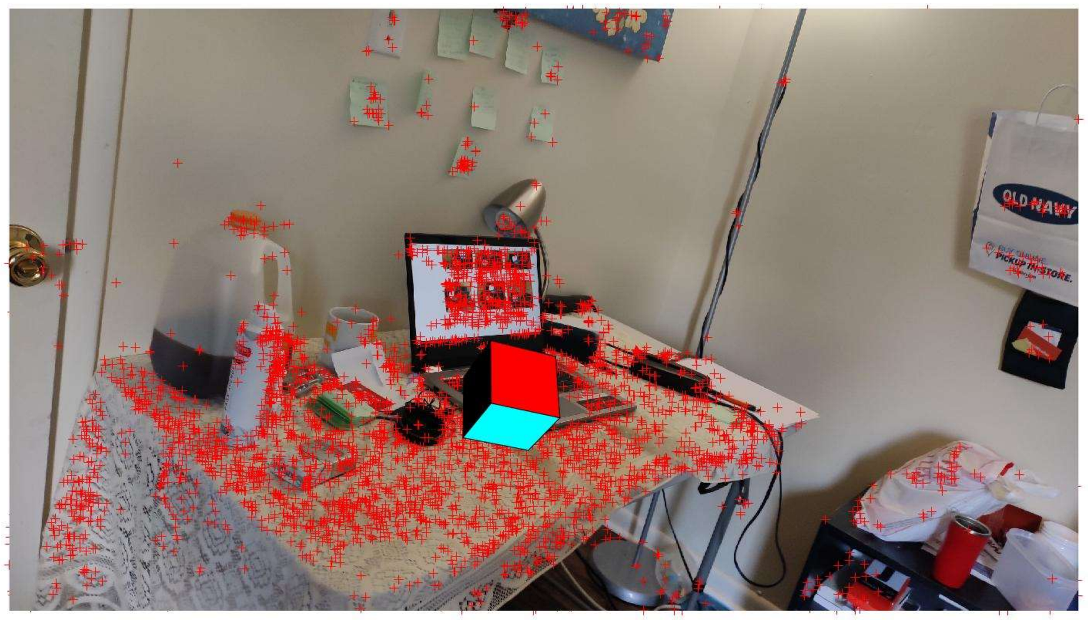
The code loops in each image iteratively and projects the cube/box obtained in earlier steps on to the image.

Results

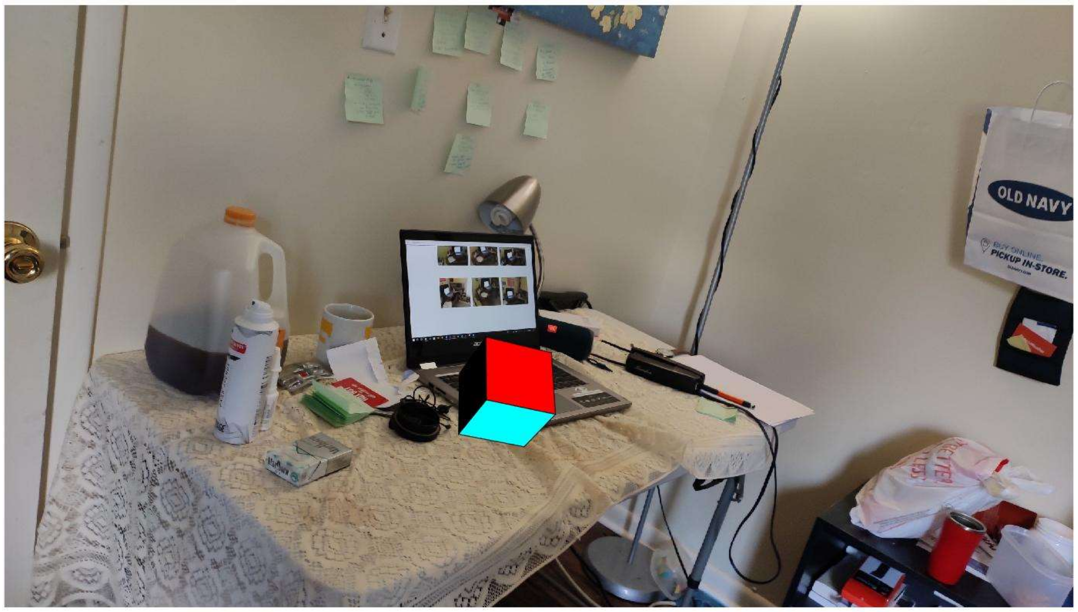
Output Scene with 3D scene cloud points and custom opaque object

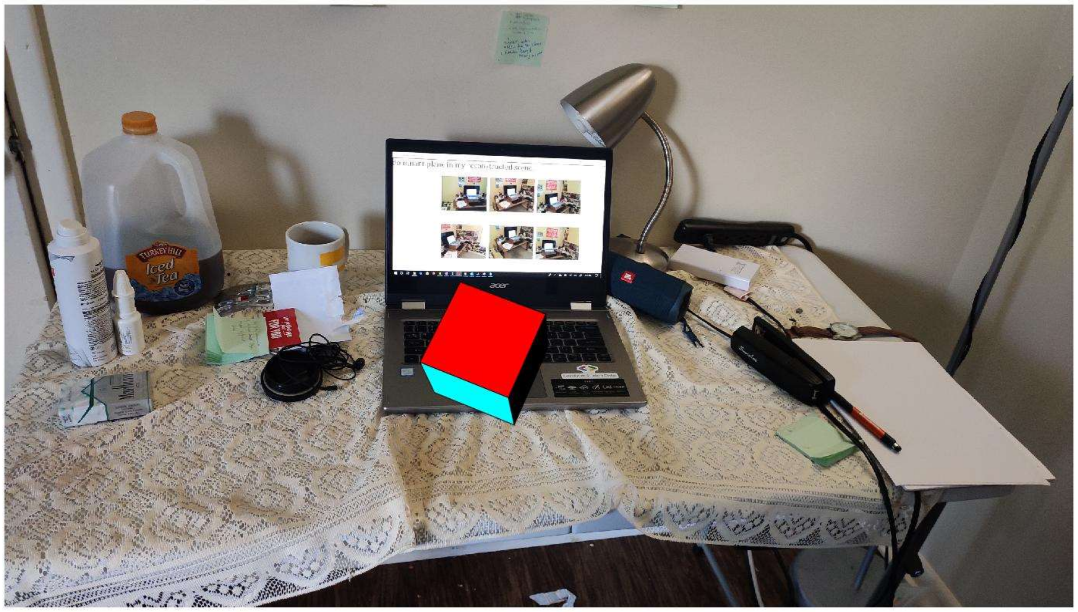


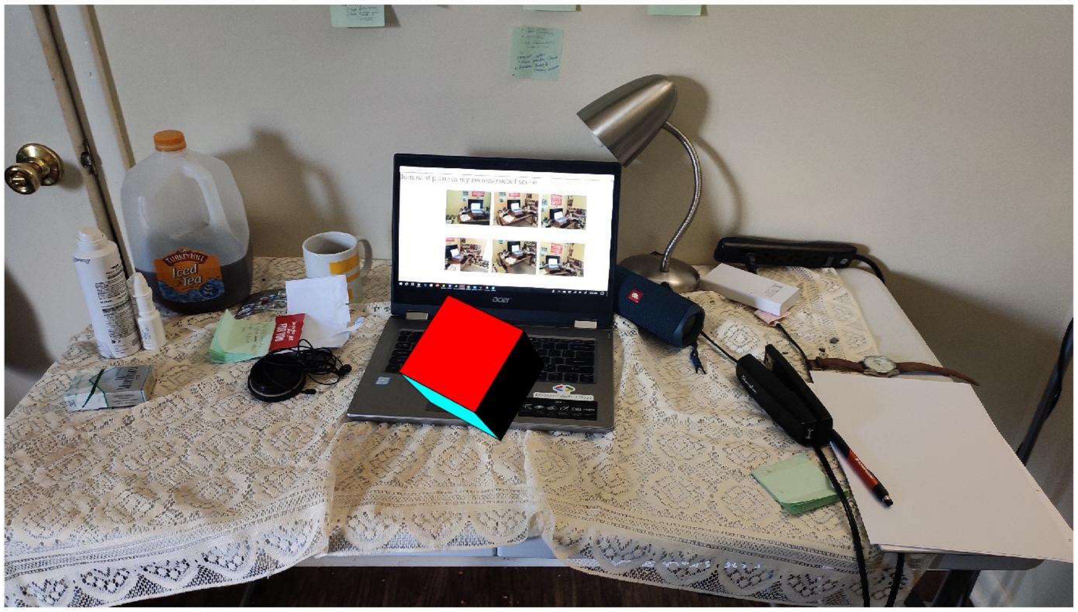


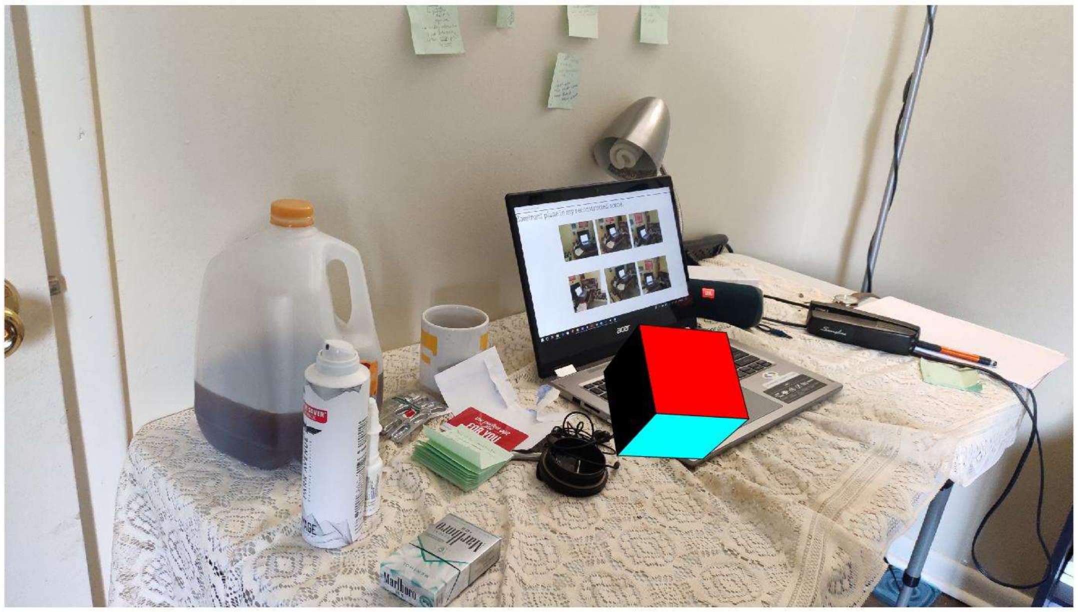

OLD NAVY
PICKUP IN STORE.

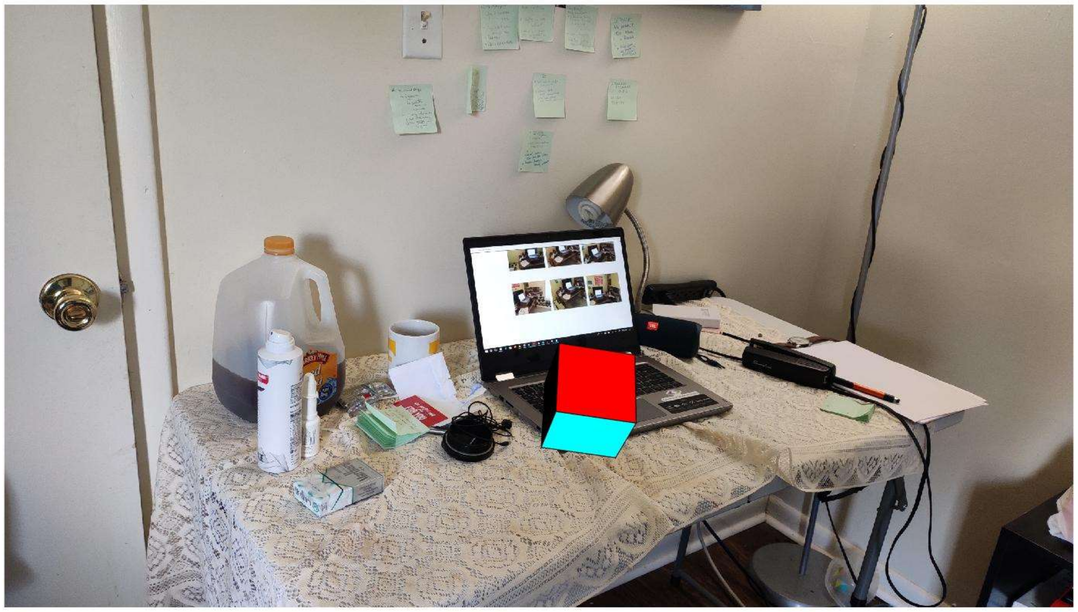
Final output scene with custom opaque object





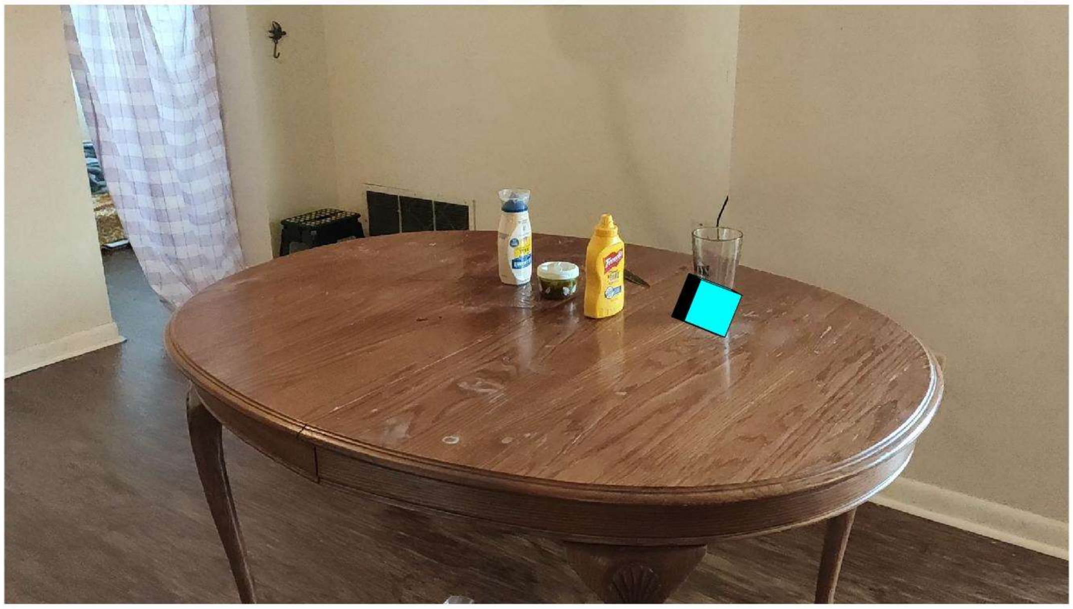


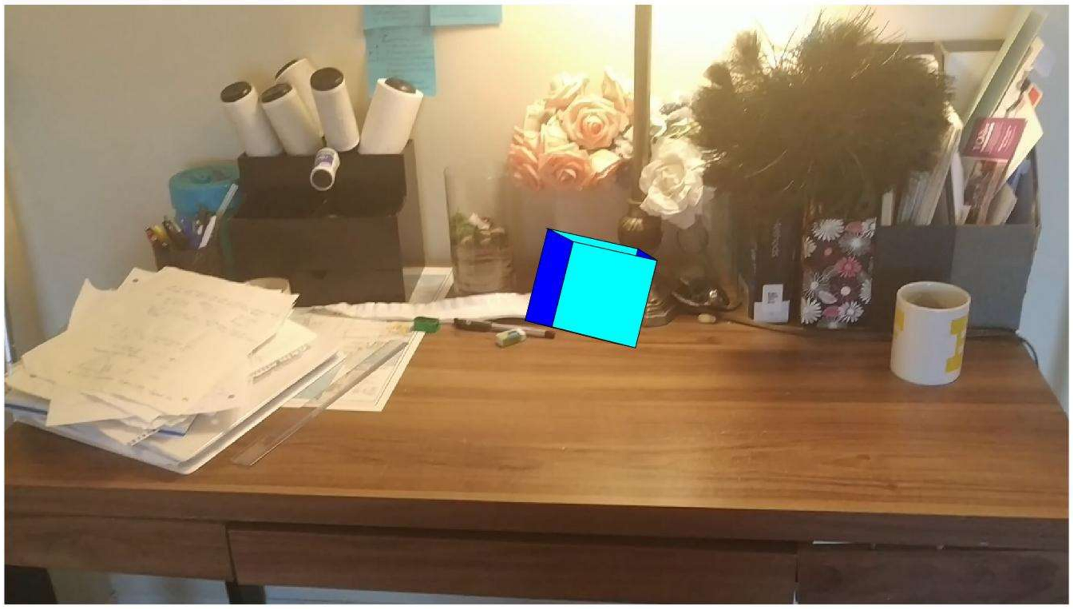




Extra Experiments-

We performed on experiment on few different setups but it did not give satisfactory results. For example





Individual Contribution

Soumitra Mehrotra (sxm6256) – Database collection and COLMAP training, Coding for Best fit plane, Code Camera Parameters Extraction, Part of Reporting and presentation.

Umar Farooq Mohammad (ubm5020) – Creating Virtual object and Its transformation for Images, Projecting the virtual object and features plotting, Part of Reporting and presentation.

Shreyas Hervatte Santosh (sjh6186) – Database collection and COLMAP training, Converting the 3D plane objects to 2D using camera parameters to be projected on output images, Report and Presentation

Vikramaditya Poddar (vbp5094) – Code Camera Parameters Extraction, Different plots for visualization of the 3D features & planes generated, Part of Reporting and presentation.

Note: The above mentioned are just the major responsibilities for project distributed among team, however the entire team have collectively worked with equal contribution on all the tasks and readings required for the completion of the Project.