CMPE 538 HW1

Büşra Oğuzoğlu

1 Selecting a Reference Picture:

As a reference picture, I used the picture of a box because it has a known geometry and it is easy to place a coordinate system based on the boxes edges and vertices. Also since I know the dimensions of this box, I thought I can make accurate calculations. It's also easier to select proper points on a figure like this.

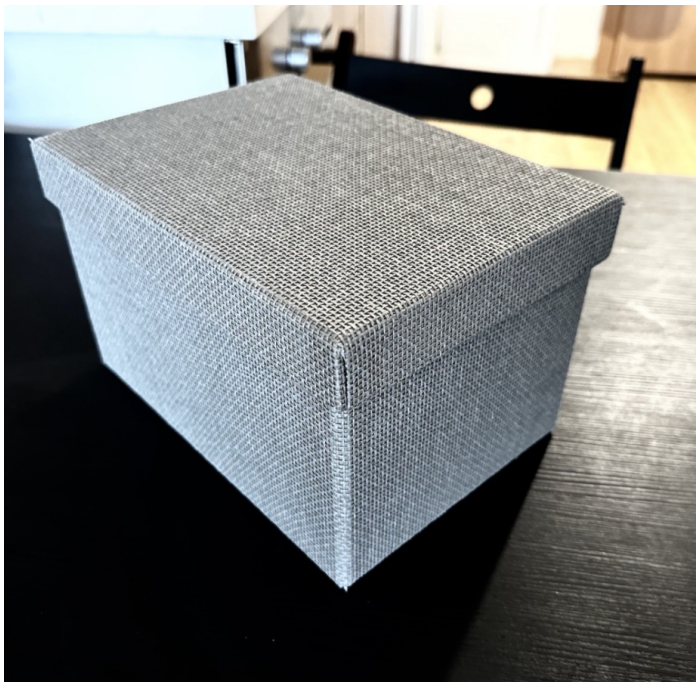

Figure 1: Selected picture

2 Coordinate and Point Selection in 2D Plane:

As the second step I assigned a 3D coordinate system on the picture. I choose the origin as the corner of the box and assigned the coordinates accordingly. The box is not exactly 2 to 1 but it is close, I choose the coordinates that way to make the calculations a bit easier.

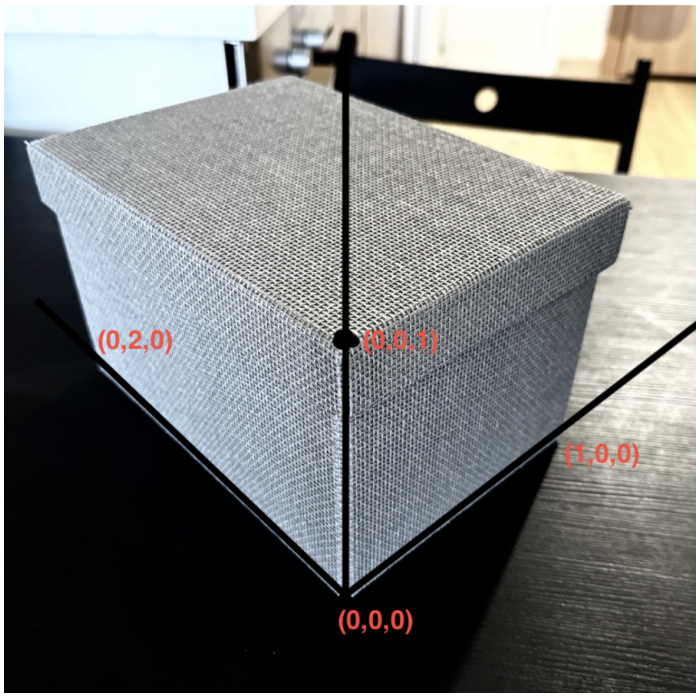

Figure 2: Coordinate Selection

After assigning the coordinates I proceed with the point selection. I used a Python script to get the coordinates on the given picture. Order was important since as the next step 3D coordinates were assigned to the corresponding 2D points. After that camera matrix was solved.

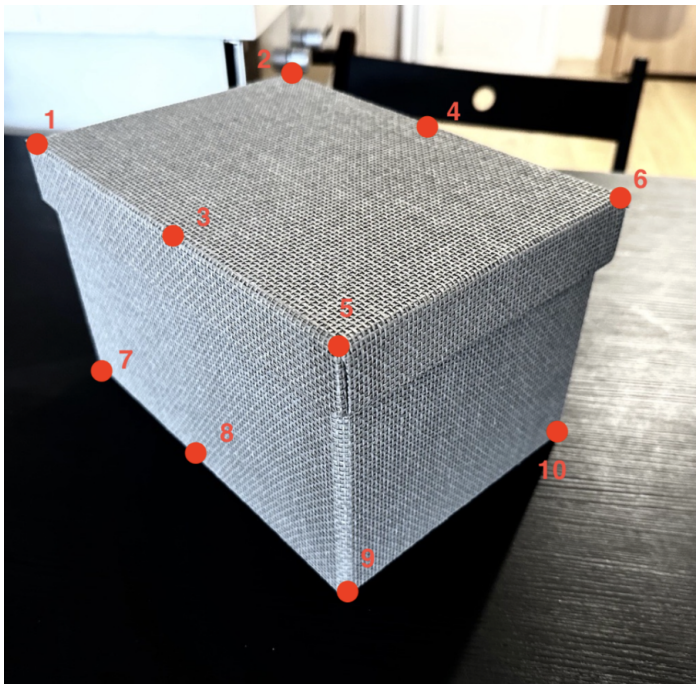

Figure 3: Point Selection

Selected 2D Points: (48, 201), (426, 103), (273, 346), (619, 177), (501, 501), (908, 284), (153, 531), (284, 649), (503, 851), (811, 623)

3D Projection of Selected 2D Points:

3D Correspondences: (0,2,1), (1,2,1), (0,1,1), (1,1,1), (0,0,1), (1,0,1), (0,2,0), (0,1,0), (0,0,0), (1,0,0)

3 Solving Camera Matrix:

The code to solve the camera matrix using the 2D and 3D points are written as below:

```
# Forming the Q matrix
    Q = []
    for i in range(len(points_3D)):
        X, Y, Z = points_3D[i]
        x, y = points_2D[i]
        Q.append([-X, -Y, -Z, -1, 0, 0, 0, 0, x*X, x*Y, x*Z, x])
        Q.append([0, 0, 0, 0, -X, -Y, -Z, -1, y*X, y*Y, y*Z, y])

    Q = np.array(Q)

    # Solve for m using SVD
    U, S, Vt = np.linalg.svd(Q)
    m = Vt[-1] # The last row of V^T

    # Reshape m to form the camera matrix M
```

```
M = m.reshape(3, 4)
```

Camera Matrix M: [[3.87717541e-01 -1.26202726e-01 -1.07524522e-01 4.12996759e-01] [-8.05243545e-02 -5.86509525e-02 -3.95517173e-01 6.96607075e-01] [1.67556048e-04 1.38799879e-04 -2.26992068e-04 8.23855159e-04]]

4 Testing the Camera Matrix:

In order to test the camera matrix, at first I tried to place a cube, then a sphere. Results are down below:

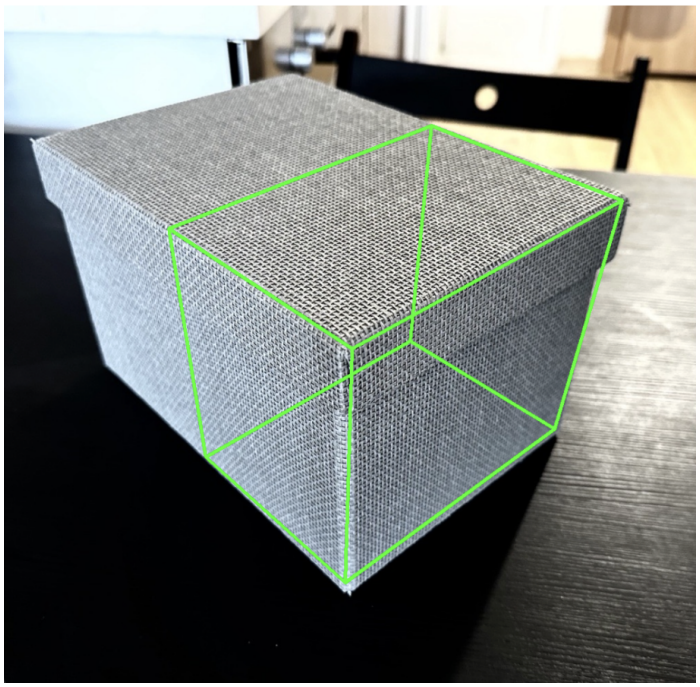

Figure 4: Placing Cube

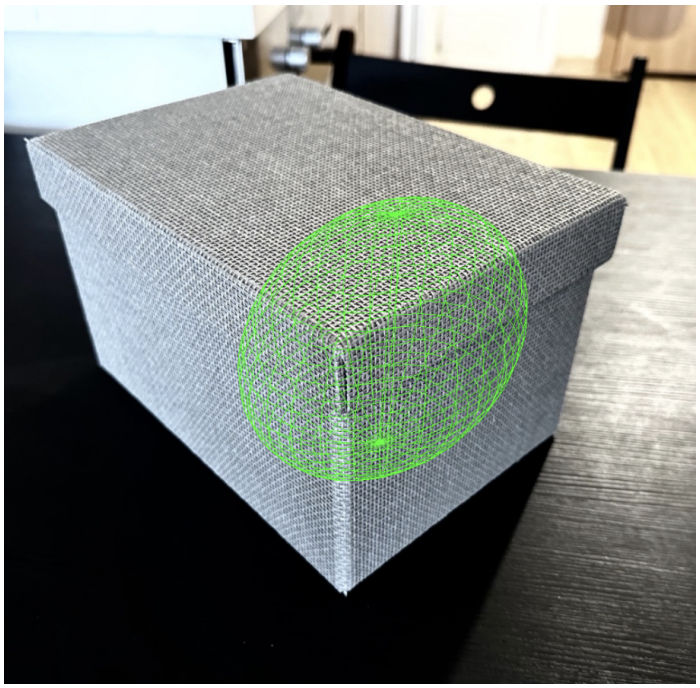

Figure 5: Placing Ball