

Project 4: Camera Calibration and Augmented Reality

Introduction:

This project is about learning how to calibrate a camera and then use the calibration to generate virtual objects in a scene. The end result should be a program that can detect a target and then place a virtual object in the scene relative to the target that moves and orients itself correctly given motion of the camera or target.

Learnings and reflections:

I was real fun working on this project. In this project, we learner to detect pattern, and use multiple images from different angles of same scene to calibrate the camera. Once the camera is calibrated, we projected 3D object relative to detected pattern into 2D image. We learned to generate rotation and translation matrices to keep the 3D object orientation intact relative to checkerboard even when camera is moved to different angle. We tested the method using two different cameras (phone camera better picture quality and laptop web camera). Our observation is, with laptop camera, corners are detected more often, and movement of 3D object is smooth, but it has larger reprojection error than phone camera. With phone camera, due to overhead of computation due to image size, detection and projection is not very smooth.

Extensions:

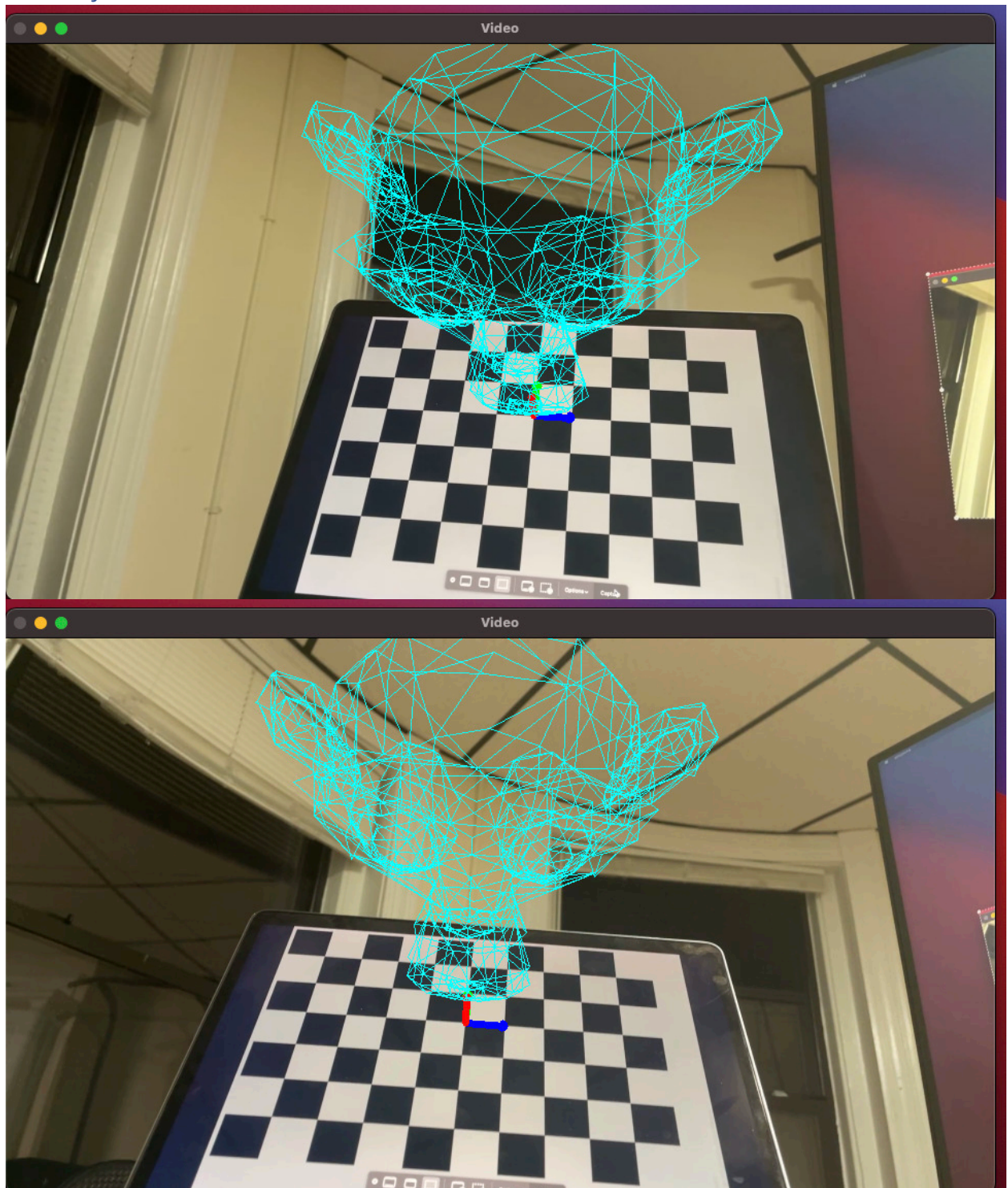
1. Ability to load and project 3D wavefront objects.
Instead of building simple 3D shapes using lines, we are using wavefront .obj file which had vertex coordinates of an object in 3D world and triangle face information stored. We parse this .obj file to create a vector of coordinates and store vertex index from face information which is used to draw lines to build the triangular face which ultimately build the 3D object.
2. Comparison of two different camera performance.

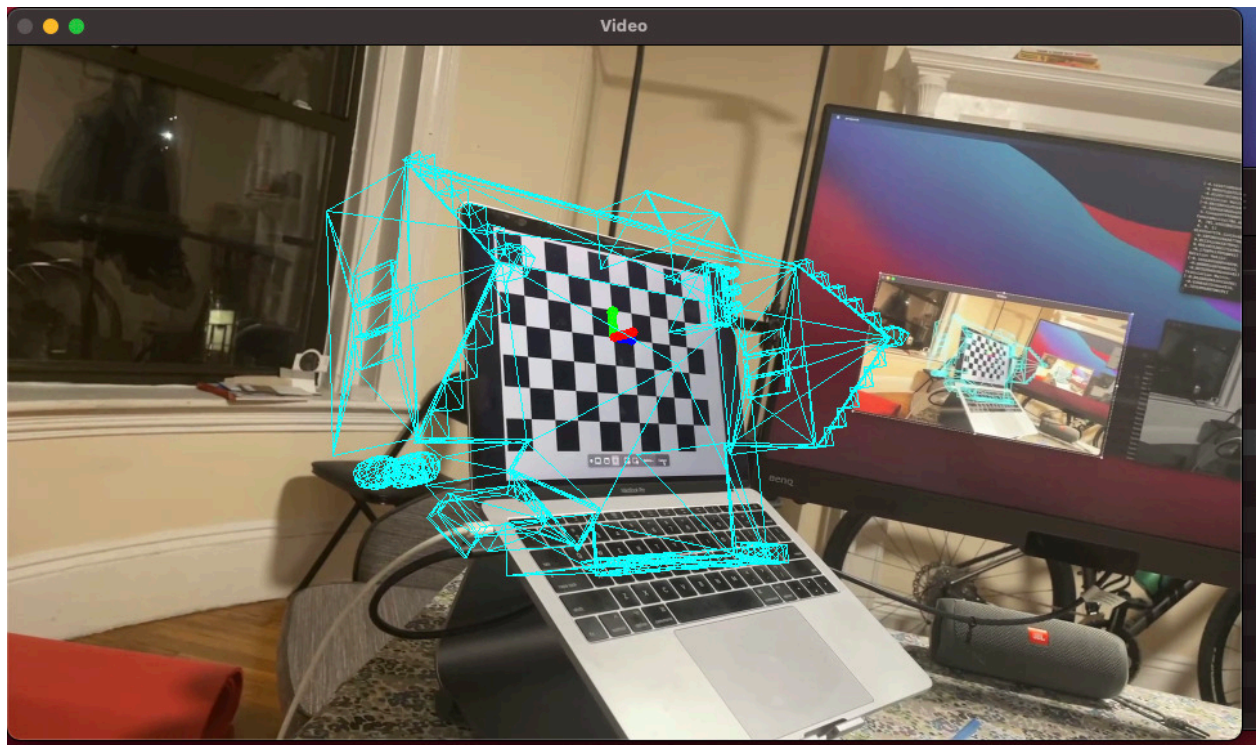
Demo Video:

Camera 1: <https://youtu.be/ne9HN1IRLOQ>

Camera 2: <https://youtu.be/-EO-rg1nDMk>

3D Object construction



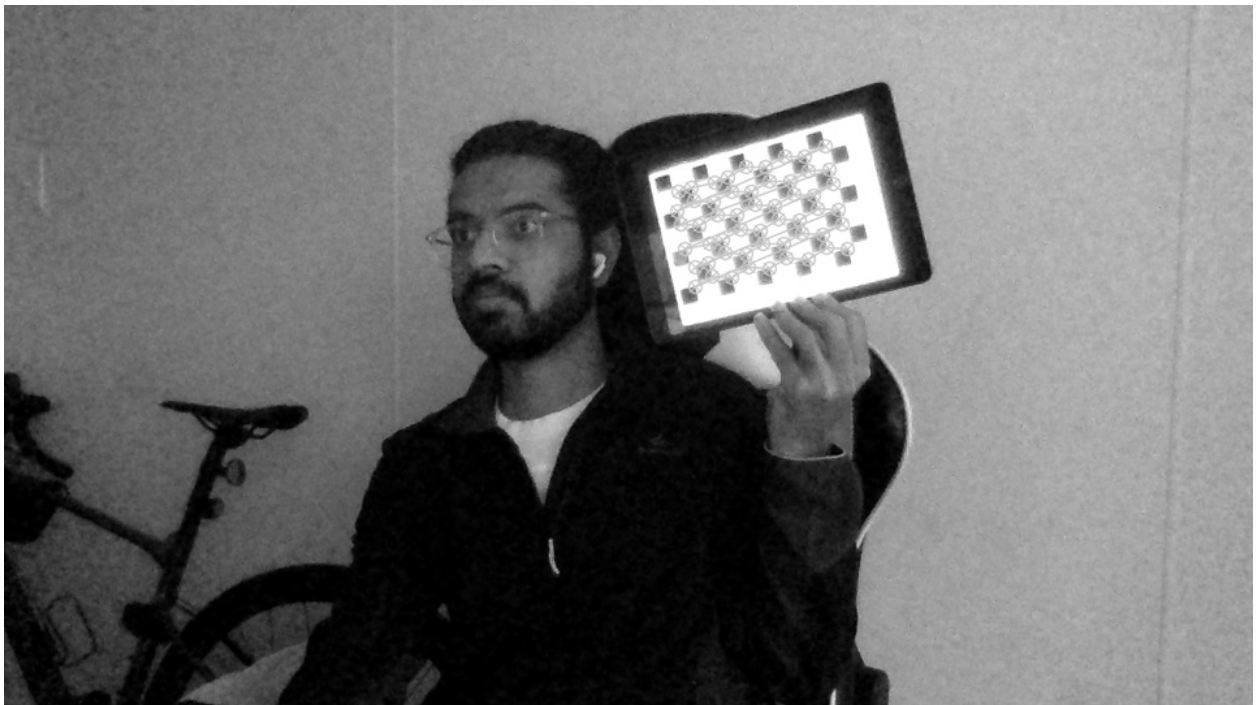


Calibration Image example:

Camera 1:

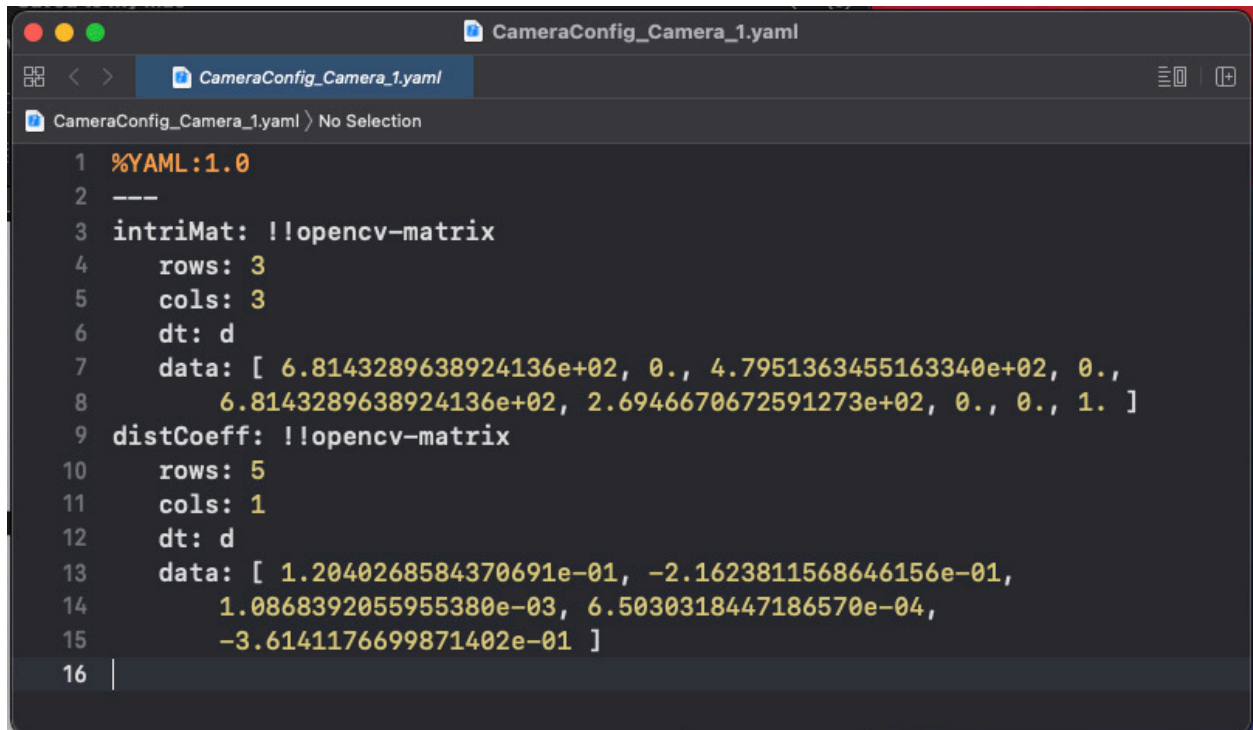


Camera 2:



Intrinsic Parameters stored:

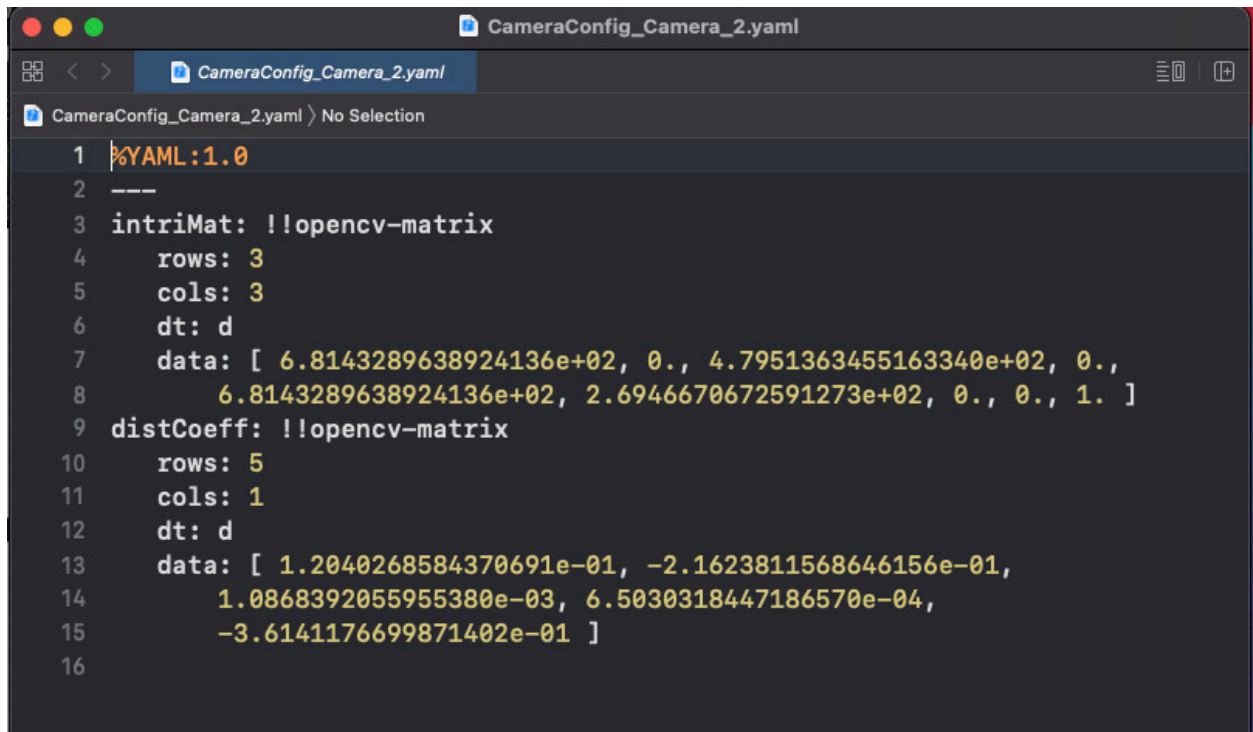
Camera 1:



The screenshot shows a code editor window titled "CameraConfig_Camera_1.yaml". The editor contains the following YAML content:

```
1 %YAML:1.0
2 ---
3 intriMat: !!opencv-matrix
4   rows: 3
5   cols: 3
6   dt: d
7   data: [ 6.8143289638924136e+02, 0., 4.7951363455163340e+02, 0.,
8           6.8143289638924136e+02, 2.6946670672591273e+02, 0., 0., 1. ]
9 distCoeff: !!opencv-matrix
10  rows: 5
11  cols: 1
12  dt: d
13  data: [ 1.2040268584370691e-01, -2.1623811568646156e-01,
14          1.0868392055955380e-03, 6.5030318447186570e-04,
15          -3.6141176699871402e-01 ]
16
```

Camera 2:



The screenshot shows a code editor window titled "CameraConfig_Camera_2.yaml". The editor contains the following YAML content:

```
1 %YAML:1.0
2 ---
3 intriMat: !!opencv-matrix
4   rows: 3
5   cols: 3
6   dt: d
7   data: [ 6.8143289638924136e+02, 0., 4.7951363455163340e+02, 0.,
8           6.8143289638924136e+02, 2.6946670672591273e+02, 0., 0., 1. ]
9 distCoeff: !!opencv-matrix
10  rows: 5
11  cols: 1
12  dt: d
13  data: [ 1.2040268584370691e-01, -2.1623811568646156e-01,
14          1.0868392055955380e-03, 6.5030318447186570e-04,
15          -3.6141176699871402e-01 ]
16
```

Reprojection Error Estimate:

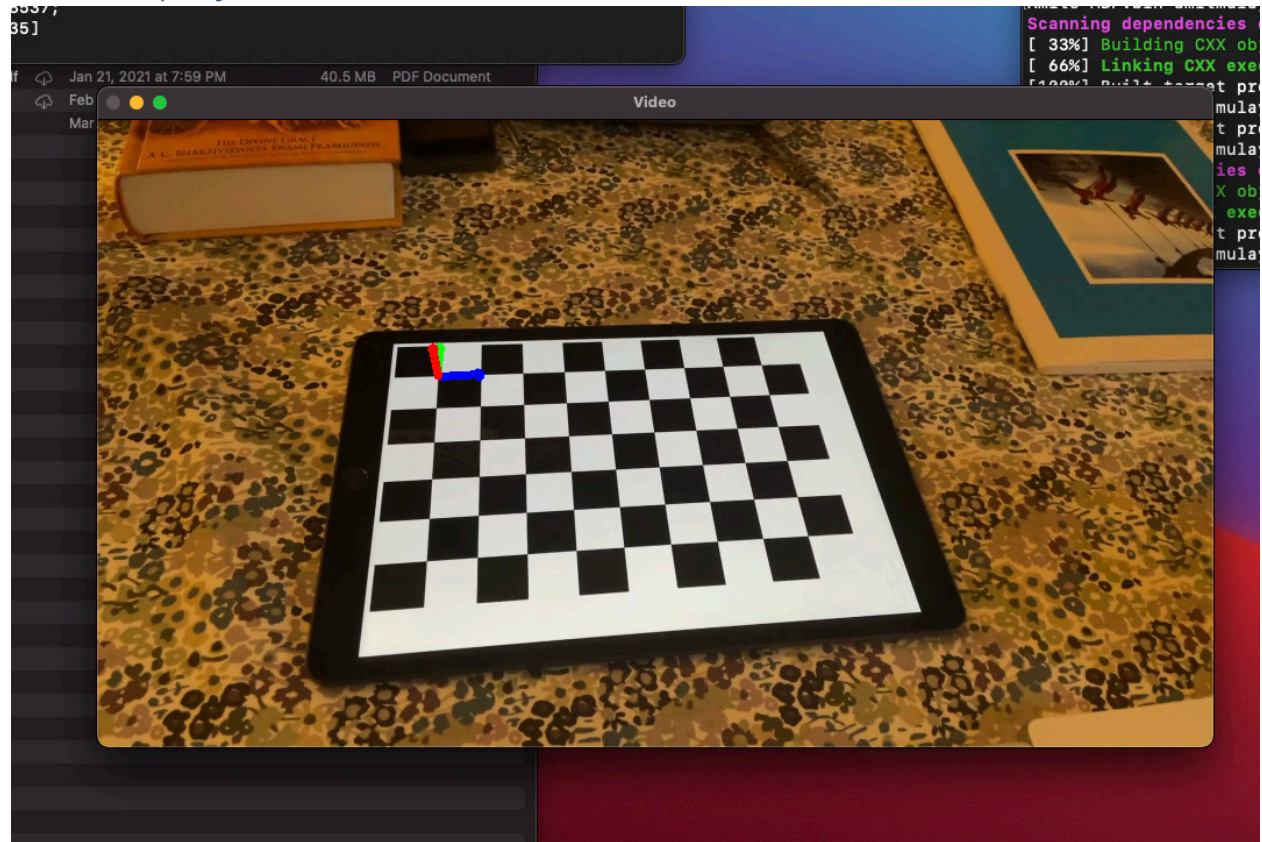
Camera 1:

Reprojection error: 0.113141

Camera 2:

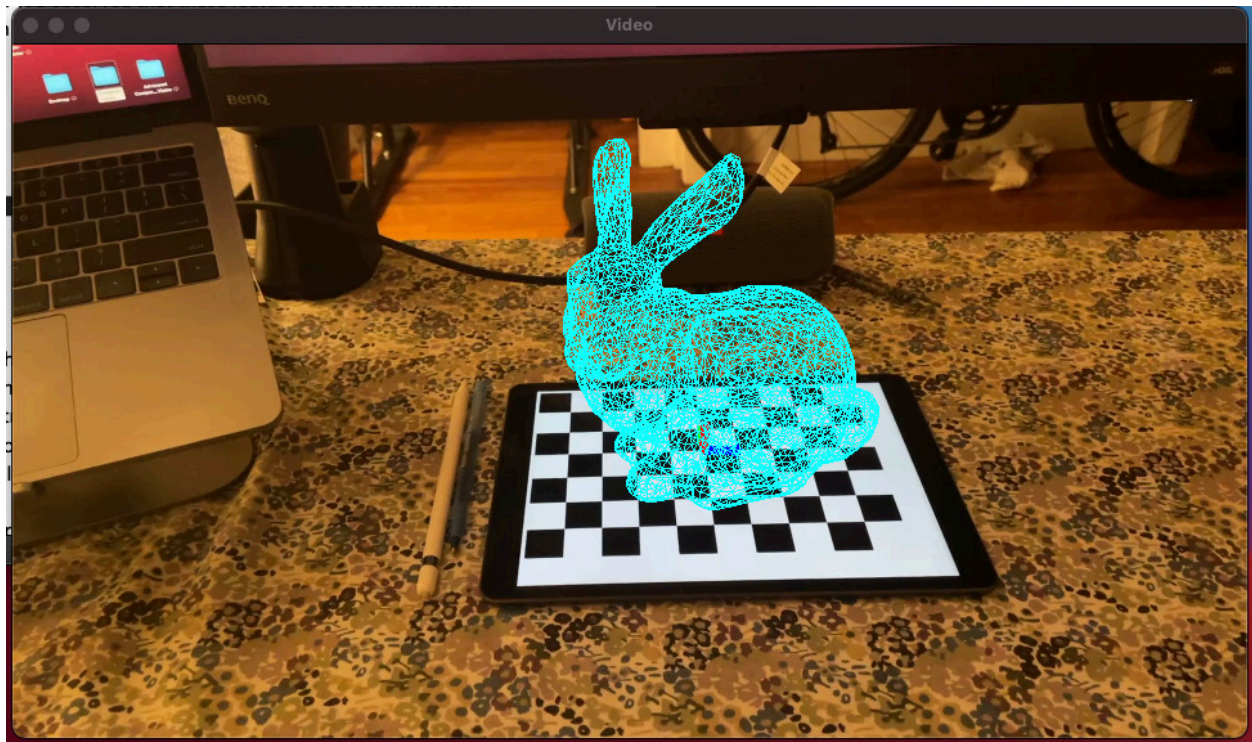
Reprojection error: 1.32226

3D Axes projection:

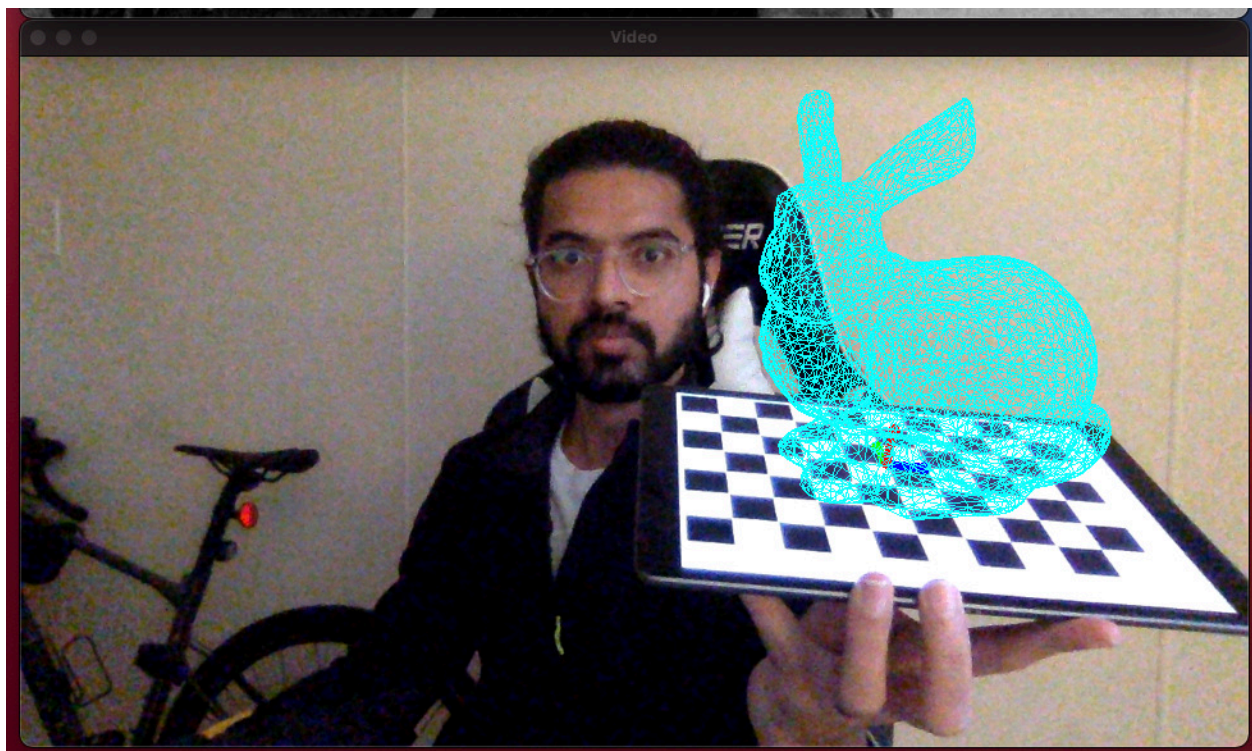


Virtual Object:

Camera 1:

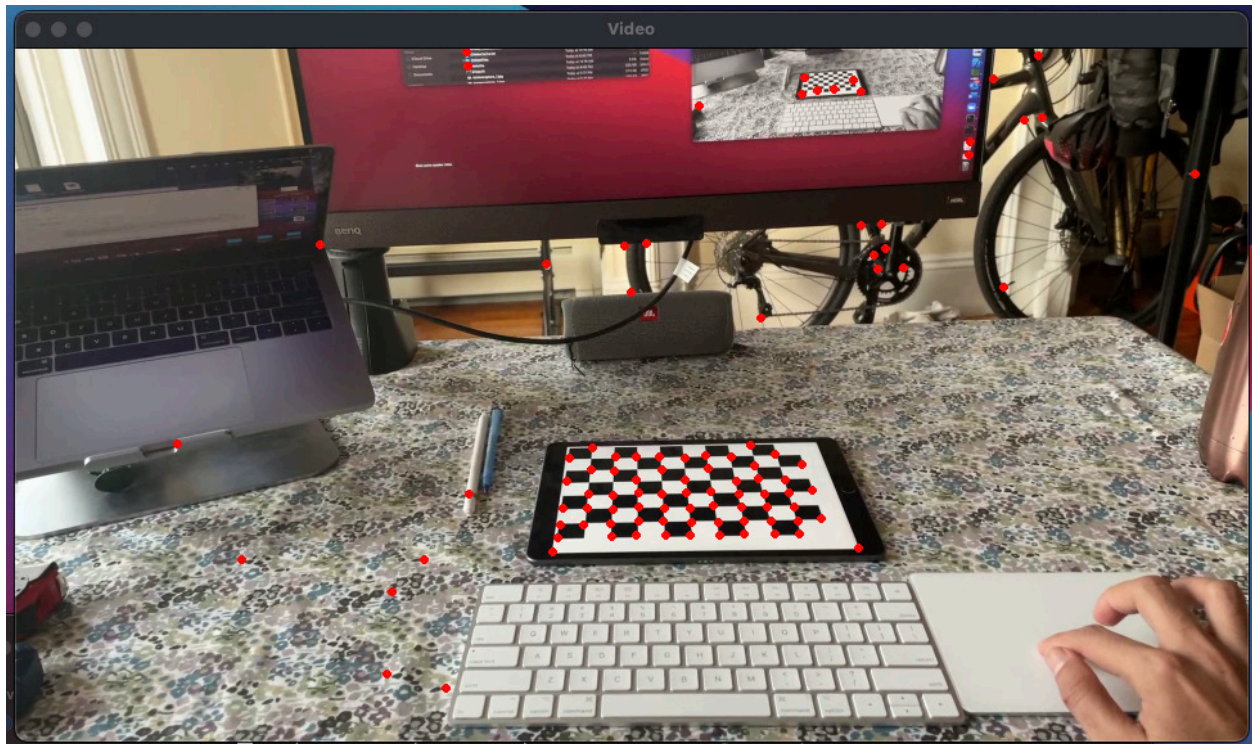


Camera 2:



Detect Additional Features (Harris Corners):

With robust features detection like detecting Harris corners, we can build an AR world based on these detected corners coordinates and we are no longer dependent on some user defined pattern like checkerboard. After we have these corners, we can use these to calibrate the camera and learn all the parameters and then use those to build the objects in the scene. To make the world robust, we can also find features of the areas surrounded by the corners in the scene. This will help us detect and identify the same corners in future frames when we move the camera.



References:

https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html