Project 4: Calibration and Augmented Reality

Student: Yunyi Chi

Description:

In this project, I gained a valuable understanding of camera calibration and how to use it to create Augmented Reality experiences by generating virtual objects in an image. Using a chessboard as my example, I extracted the coordinates of all the internal corners of the chessboard in multiple images, and then used this data with world coordinates to solve for intrinsic parameters (camera_matrix, distortion_coefficients) and extrinsic parameters (translation, rotation) to complete the camera calibration process. Using the real-time video and the intrinsic parameters obtained, I estimated the real-time board's pose (rotation and translation) and created virtual objects by mapping world coordinates to image coordinates. To further enhance our understanding, we explored the Harris corners feature and wrote a separate program that shows the location of the features in the image in a video stream.

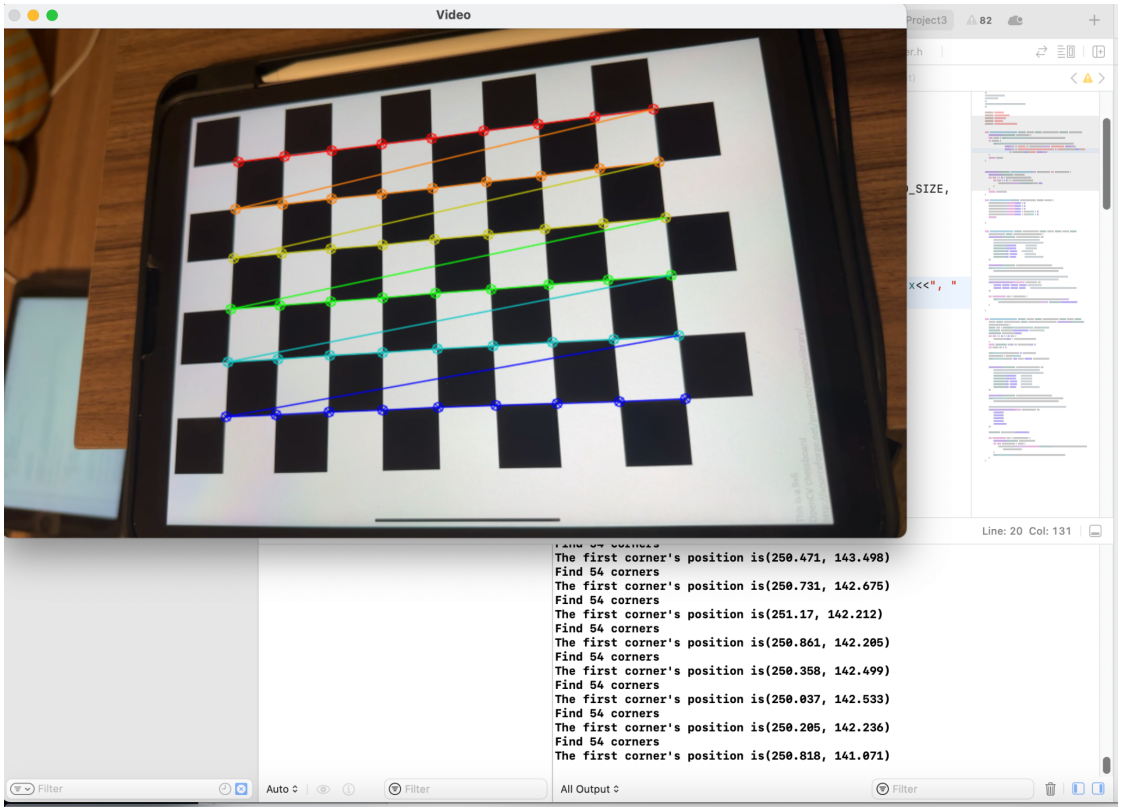
Additionally, we completed three extensions as part of this assignment:

1. Tested various cameras and compared the calibration results to evaluate the quality of each one.
2. Enabled the system to use static images or pre-captured video sequences with targets, demonstrating the insertion of virtual objects into the scenes.
3. Not only did we add virtual objects, but we also transformed the targets to make them appear less like targets.

Video: https://drive.google.com/file/d/1f5fawBZeVEkltX86ZWcb5sZAegXFcu6f/view?usp=share_link

Task 1: Detect and Extract Chessboard Corners

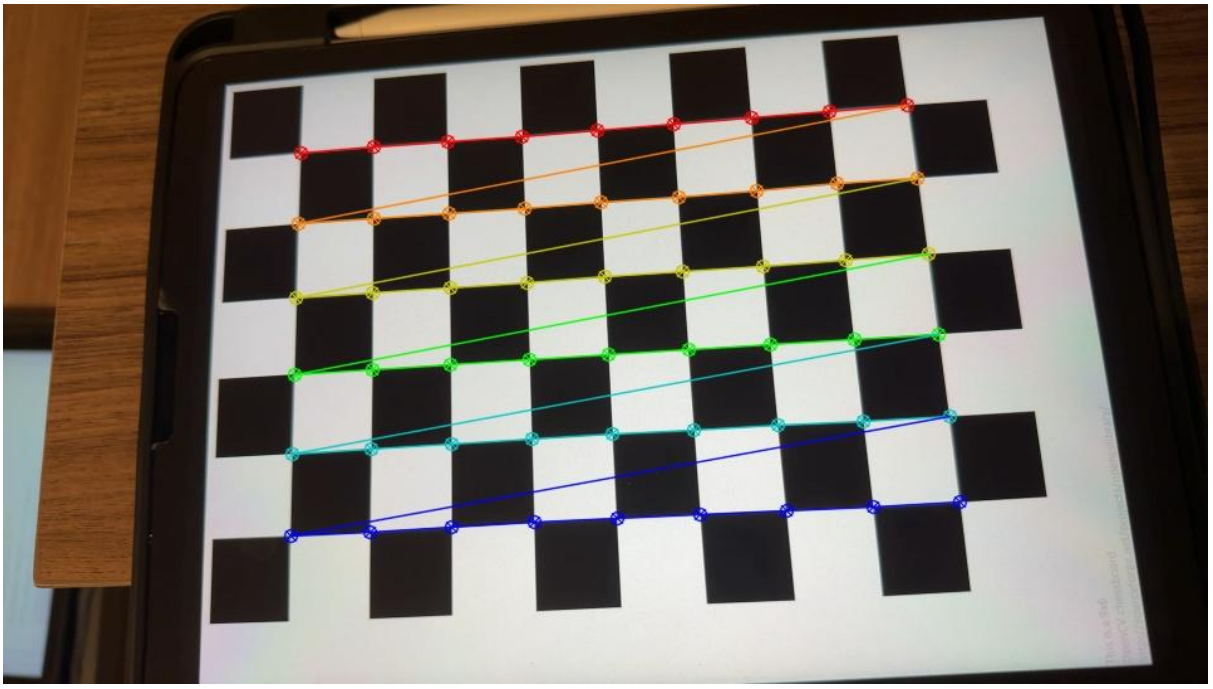
For this task, I use the functions findChessboardCorners() and drawChessboardCorners() in calibration documentation for OpenCV, it draws the corners in the chessboard, which were marked with the colorful circles, I also print out the number of corners it finds and the coordinates of the first corner, you can see that from the console in the image.



Chessboard Corners (console shows the number of corner and first corner's coordinates)

Task 2: Select Calibration Images

For this part, if user use 's', then it will save point_set and corner_corner to the point_list and corner_list. The calibration image with chessboard corners highlighted is shown below.



a calibration image with chessboard corners highlighted

Task 3: Calibrate the Camera

For this task, the camera matrix and distortion coefficients before and after the calibration, along with the final re-projection error is shown below.

```
Before calibration:
Camera Matrix:
[1, 0, 480;
 0, 1, 270;
 0, 0, 1]
Distortion Coefficients:
[0;
 0;
 0;
 0;
 0]

After calibrating the Camera
Camera Matrix:
[652.9505118426697, 0, 482.09293722128 74;
 0, 653.9602951297006, 276.4604712068307;
 0, 0, 1]
Distortion Coefficients:
[0.2086763772317427;
 -0.9428520175053567;
 -3.662744235875026e-05;
 -0.0004109849778706283;
 1.106951971678929]
Reprojection Error: 0.187208
```

matrix and distortion coefficients before and after the calibration, final re-projection error

Task 4: Calculate Current Position of the Camera

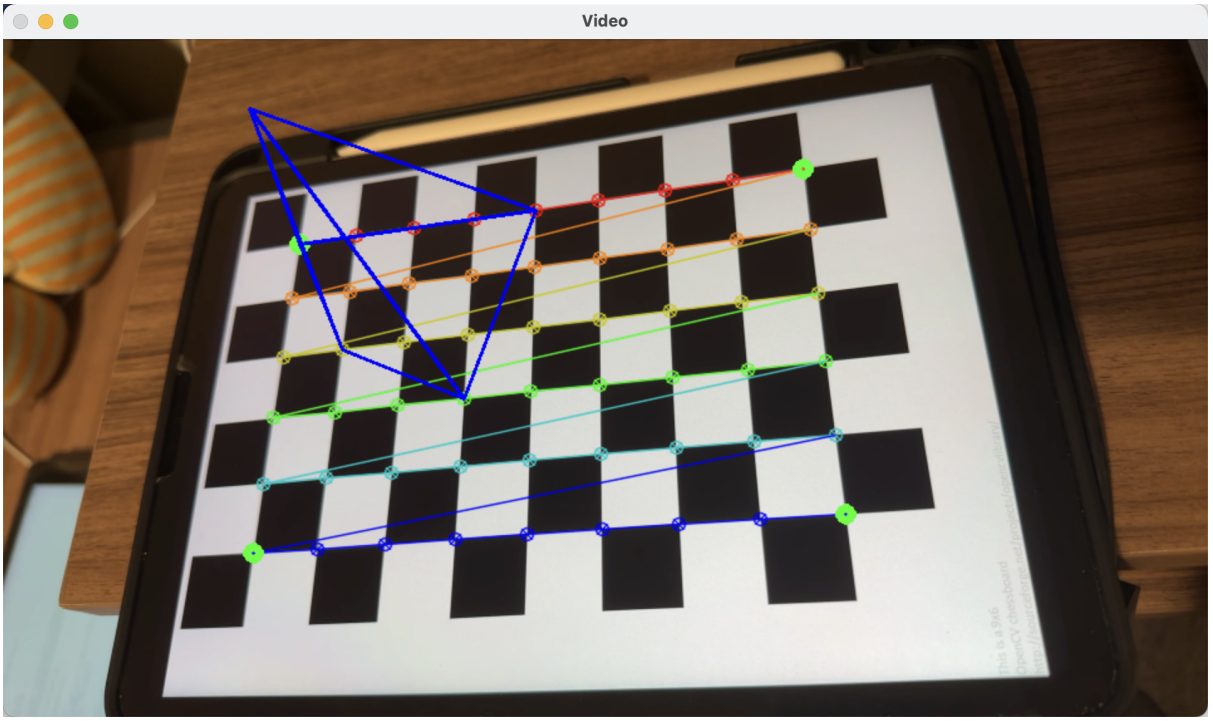
For this part, the rotation and translation data in real time is shown below.

```
Find 54 corners
The first corner's position is(189.128, 118.198)
Rotation: [2.967765023892879;
 -0.2146838686645173;
 -0.2789428065438399]
Translation: [-5.71687292215373;
 -3.013509074261457;
 12.67182068014515]
Find 54 corners
The first corner's position is(197.299, 109.721)
Rotation: [2.965005013117023;
 -0.20127885435428;
 -0.2800828287355052]
Translation: [-5.558661303669276;
 -3.196377536745053;
 12.71889100995149]
Find 54 corners
The first corner's position is(201.61, 102.617)
Rotation: [2.966732846268852;
 -0.1911197836727246;
 -0.2764769352918107]
Translation: [-5.475695737874252;
 -3.33028823281549;
 12.7177696731068]
```

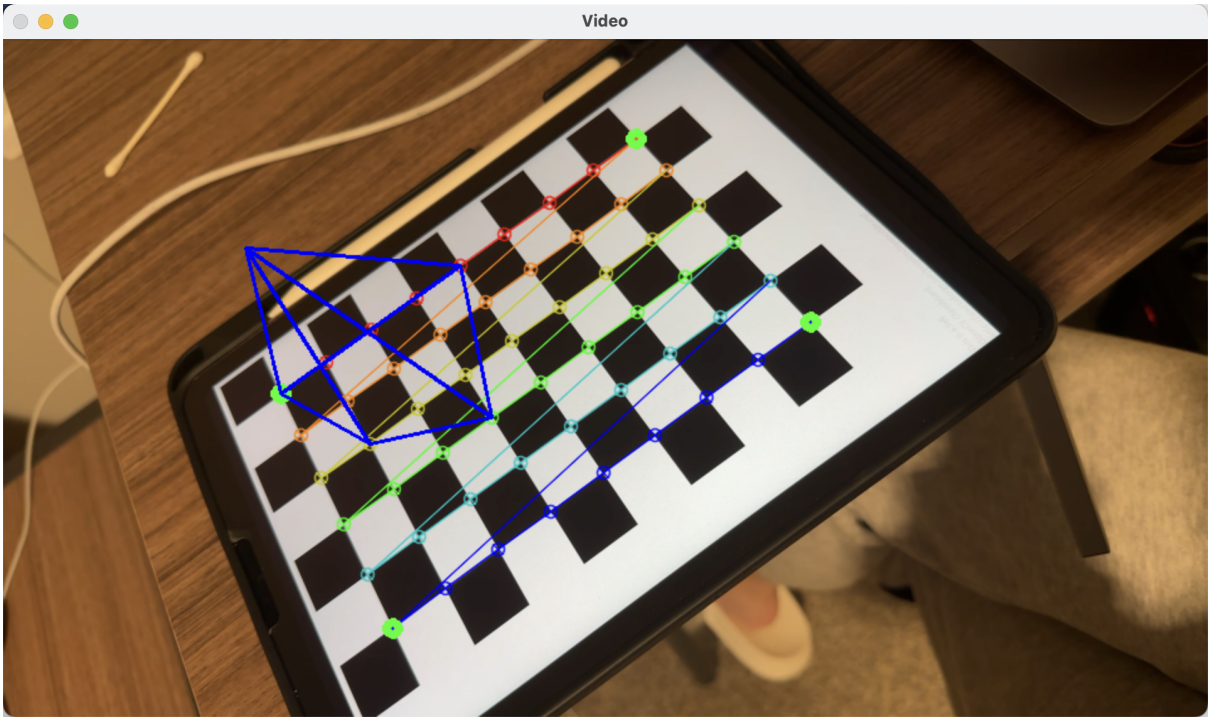
rotation and translation data in real time

Task 5: Project Outside Corners or 3D Axes

For this task, the outside Corners is marked in green color.



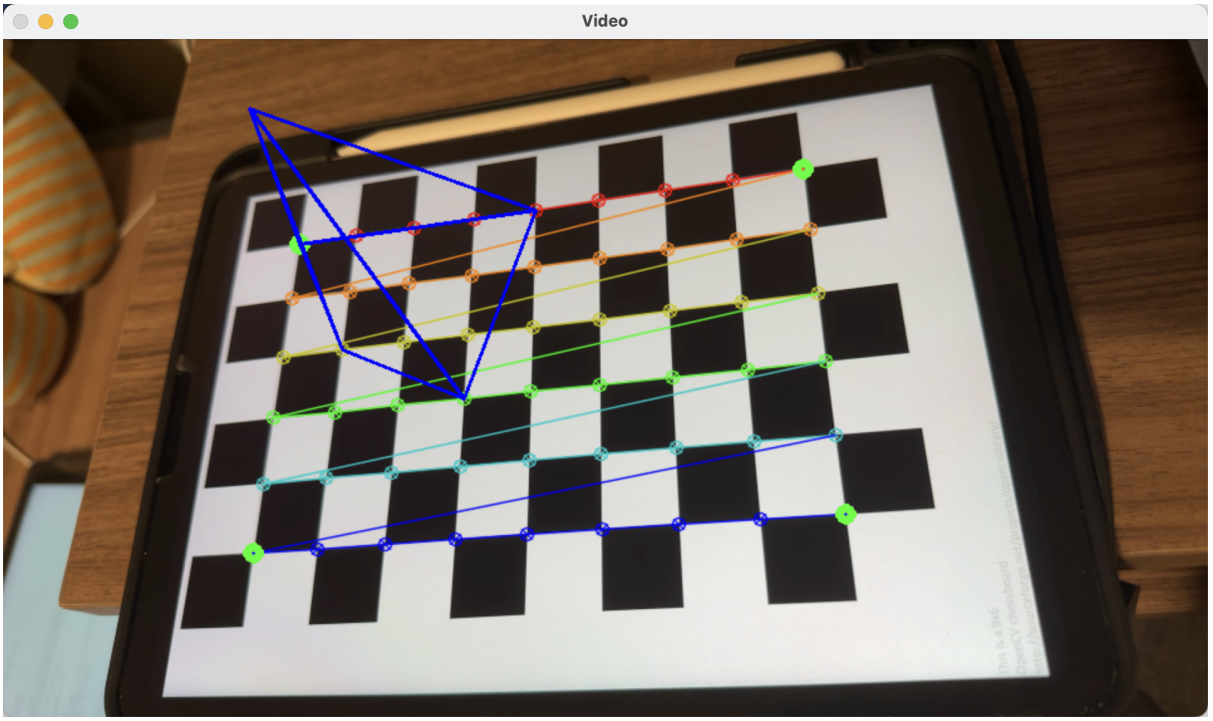
project outside corners in image



project outside corners in image

Task 6: Create a Virtual Object

For this task, an asymmetrical virtual object is shown below.

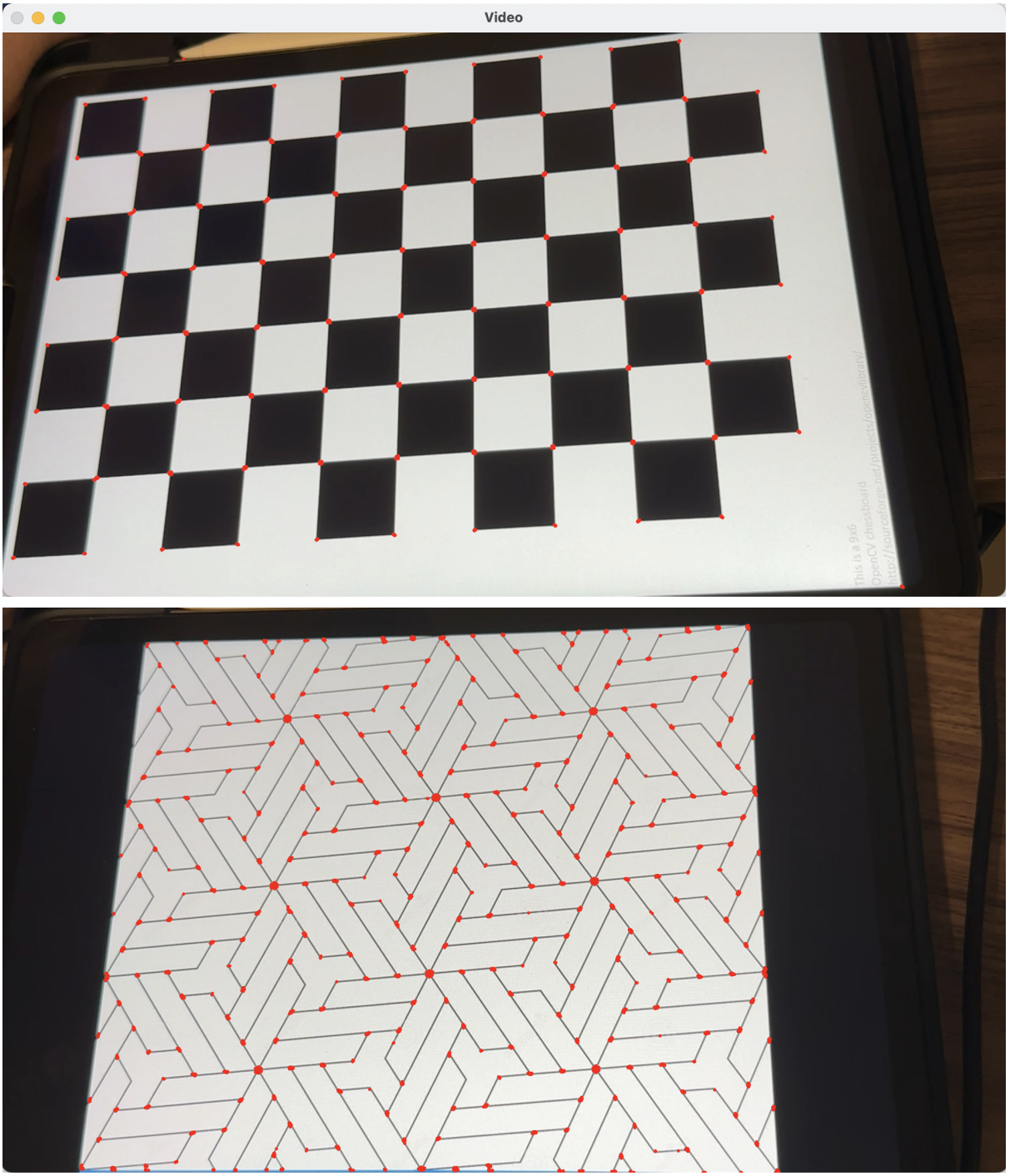


Task 7: Detect Robust Features

For this task, we utilized the Harris Corners Detector. The images demonstrate successful corner detection in a star image as well as on a chessboard.

Similar to detecting corners on a chessboard, before using the Harris feature points to implement augmented reality, we must first establish the correlation between 2D coordinates in the image and 3D coordinates in the real world. Once we can project the points in the image onto a 3D coordinate system, we can calibrate the camera and determine its position for future use.

Additionally, the Harris Corners Detector was able to detect more corners (even within letters) compared to the chessboard, which suggests it may be useful for detecting more detailed features.



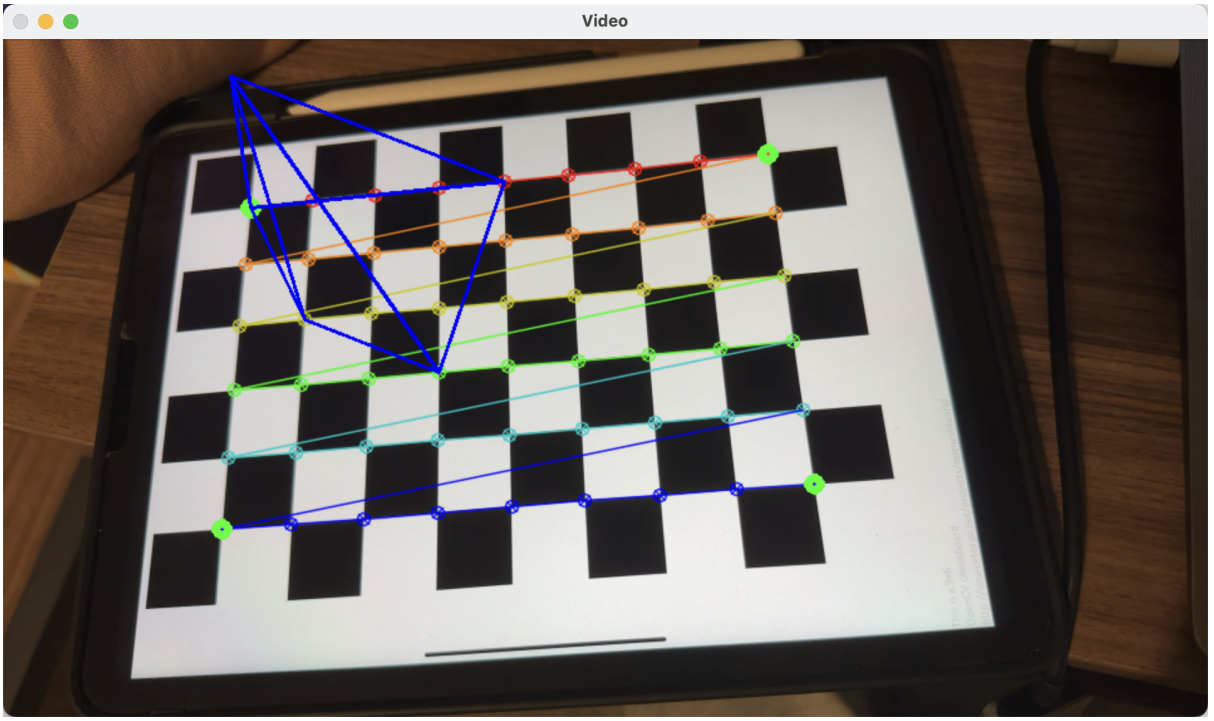
Extensions (3 extensions)

- **Test out several different cameras and compare the calibrations and quality of the results.**

- **Enable your system to use static images or pre-captured video sequences with targets and demonstrate inserting virtual objects into the scenes.**
- **Not only add a virtual object, but also do something to the target to make it not look like a target any more.**

1. Test out several different cameras and compare the calibrations and quality of the results.

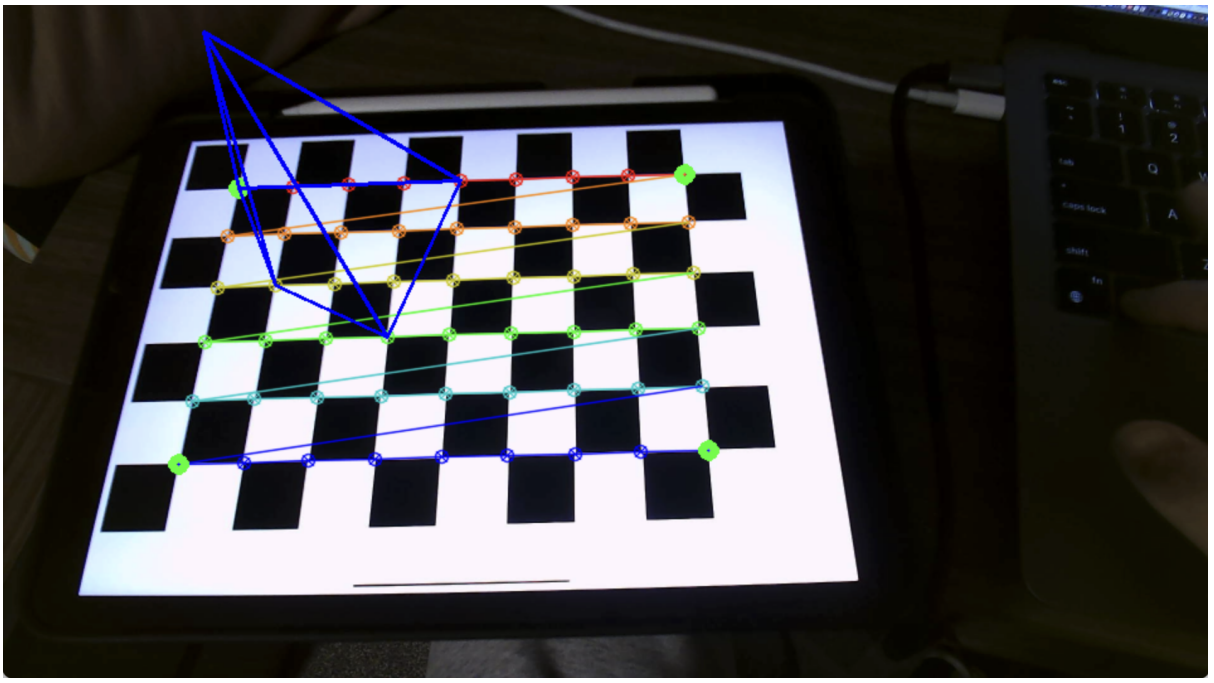
   In this extension, we use different cameras to test the results, one is iphone 14 pro, one is lenovo 300 FHD webcam. The different results are shown below.



```
Before calibration:
Camera Matrix:
[1, 0, 480;
 0, 1, 270;
 0, 0, 1]
Distortion Coefficients:
[0;
 0;
 0;
 0;
 0]

After calibrating the Camera
Camera Matrix:
[627.2683151991857, 0, 486.1695507792749;
 0, 625.905853141967, 278.3134313682236;
 0, 0, 1]
Distortion Coefficients:
[0.1261784577266839;
 -0.06073138858206889;
 0.0007863548998661905;
 0.0004826334352263 94;
 -1.224392495168904]
Reprojection Error: 0.16891
```

camera 1(iphone 14pro)



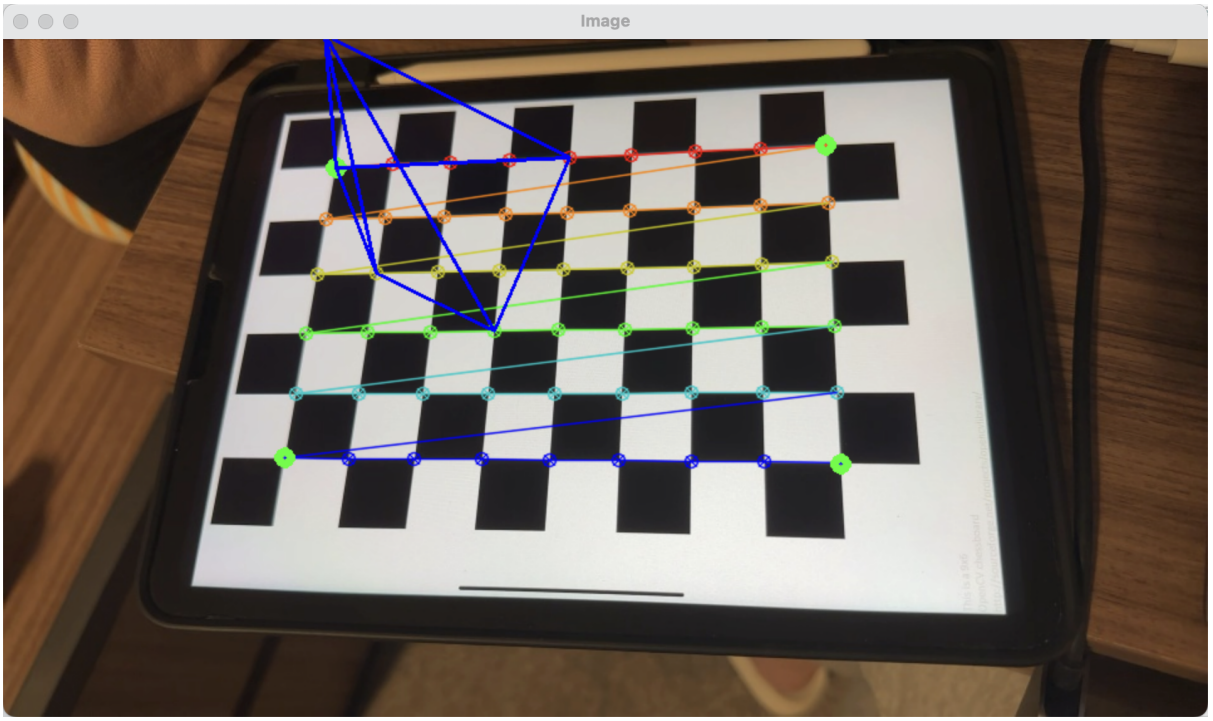
```
Before calibration:
Camera Matrix:
[1, 0, 480;
 0, 1, 270;
 0, 0, 1]
Distortion Coefficients:
[0;
 0;
 0;
 0;
 0]

After calibrating the Camera
Camera Matrix:
[562.2539002008152, 0, 456.6171192185751;
 0, 570.1188164935827, 296.7546824427125;
 0, 0, 1]
Distortion Coefficients:
[0.01320570156019551;
 -0.2537385177144483;
 -0.001467487096660233;
 0.0001048151105164663;
 0.4614899975009002]
Reprojection Error: 0.27359
```

lenovo 300 FHD webcam

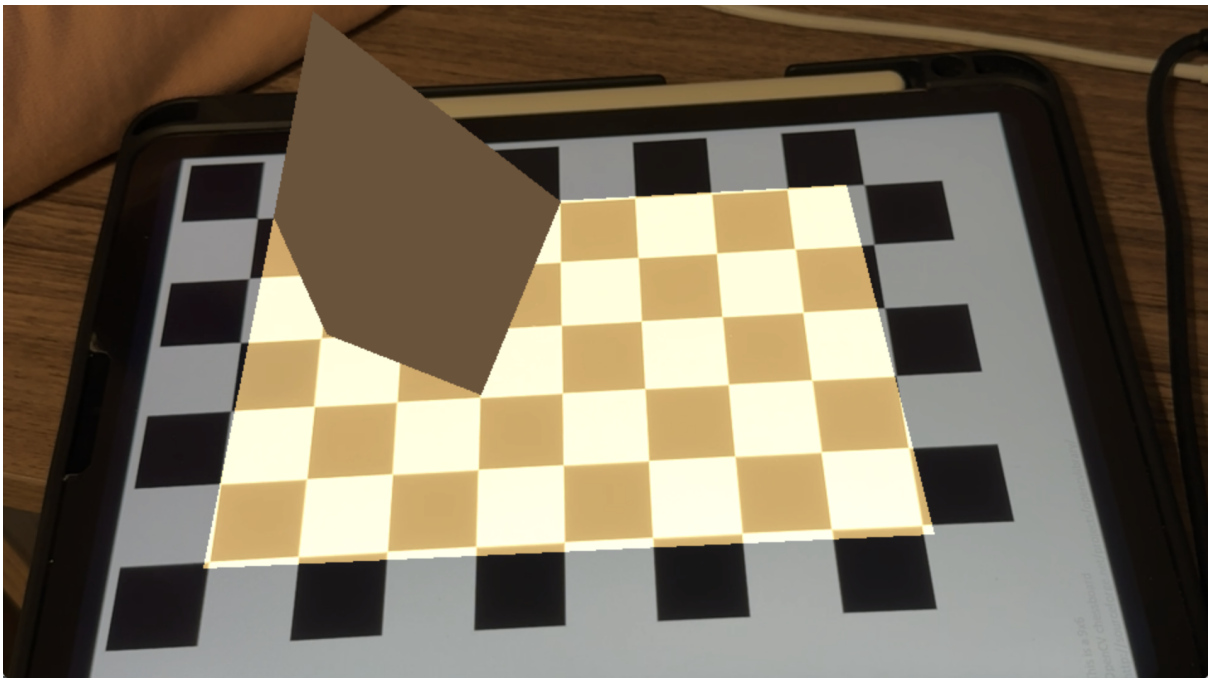
2. Enable your system to use static images or pre-captured video sequences with targets and demonstrate inserting virtual objects into the scenes.

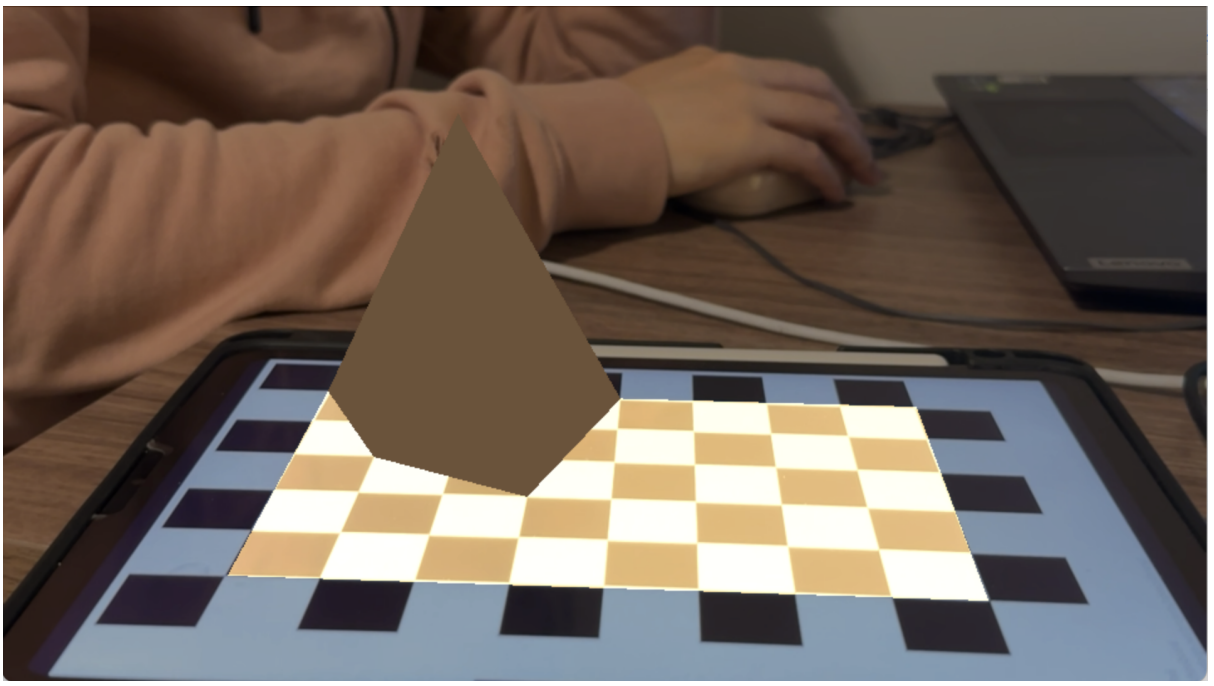
   We can use calibrated data to make Augmented Reality operation in an image, you can watch from the video.



3. Not only add a virtual object, but also do something to the target to make it not look like a target any more.

   in this extension, we colored the virtual object and the board to make it more like a cartoon world





What I learned

From this project, I learn the following things

1. I learned how to make Calibration and Augmented Reality.
2. I learned how to make a colored virtual object
3. I learned how to detect Robust Features like Harris corners or SURF features

Acknowledgement

Maxwell's lecture notes, by *Dr. Bruce A. Maxwell*

Computer Vision: Algorithms and Applications, 2nd ed, by *Richard Szeliski*

OpenCV Tutorials: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html