Project 4: Calibration and Augmented Reality
Arun Madhusudhanan

Project Description

The objective of this project is to develop a system that utilizes computer vision techniques and augmented reality technology to project virtual objects onto physical targets. The system employs a calibrated camera that uses the corners of the target as reference points for accurate tracking of the target's pose. The targets used in the system are both chessboards and Aruco boards. The cameras were calibrated using corners extracted from multiple images of a chessboard. Additionally, a separate program has been developed to explore the use of feature detection algorithms such as Harris corners in augmented reality. The laptop (ASUS TUF) webcam is used for the project.

Tasks Performed

Task 1: Detect and Extract Chessboard Corners

The corners of the chessboard were found using the function *findChessboardCorners*. Corner locations found were refined using *cornerSubPix* function and detected corners were drawn in the frame using *drawChessboardCorners.* Figure 1 shows an image of a chess board with corners detected.

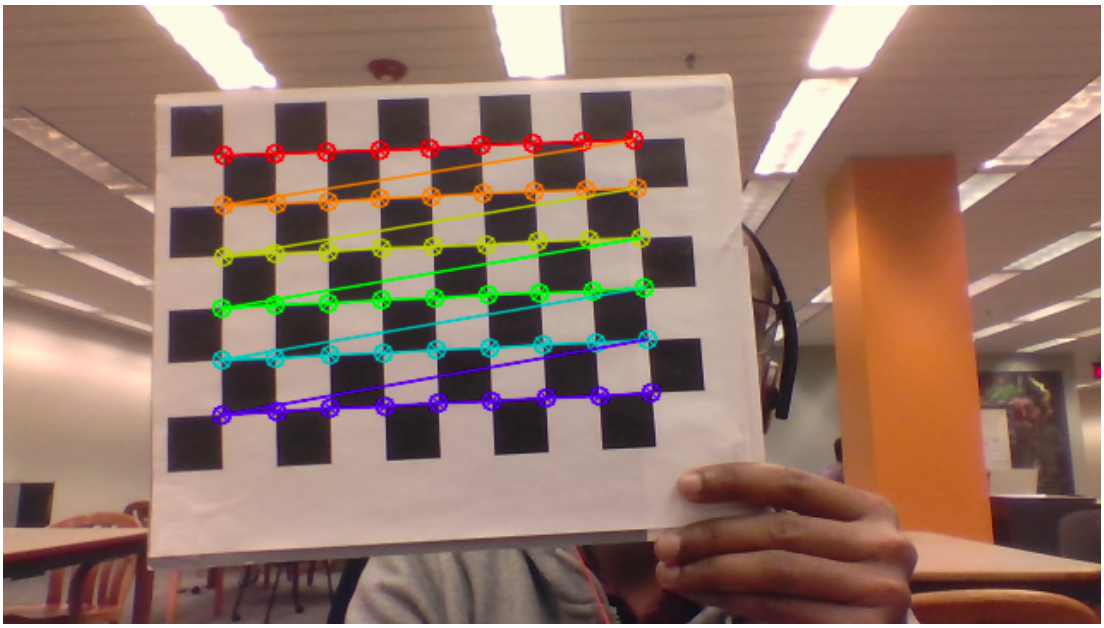

Figure 1: Detected chessboard corners

Task 2: Select Calibration Images

I used the frames captured from the live stream to calibrate the camera. I have added more than 15 images in the calibration set.The system allows the user to save the images used for

calibration by pressing ‘s’. The program displays the camera matrix and distortion coefficients in the terminal each time an image is added to the calibration set. First, the chessboard is identified and extracted a list of corners using the function findChessboardCorners. Next, we converted this corner list into a point list and utilized these vectors to calibrate the camera.

Figure 2 shows examples of a calibration image with chess board corners highlighted

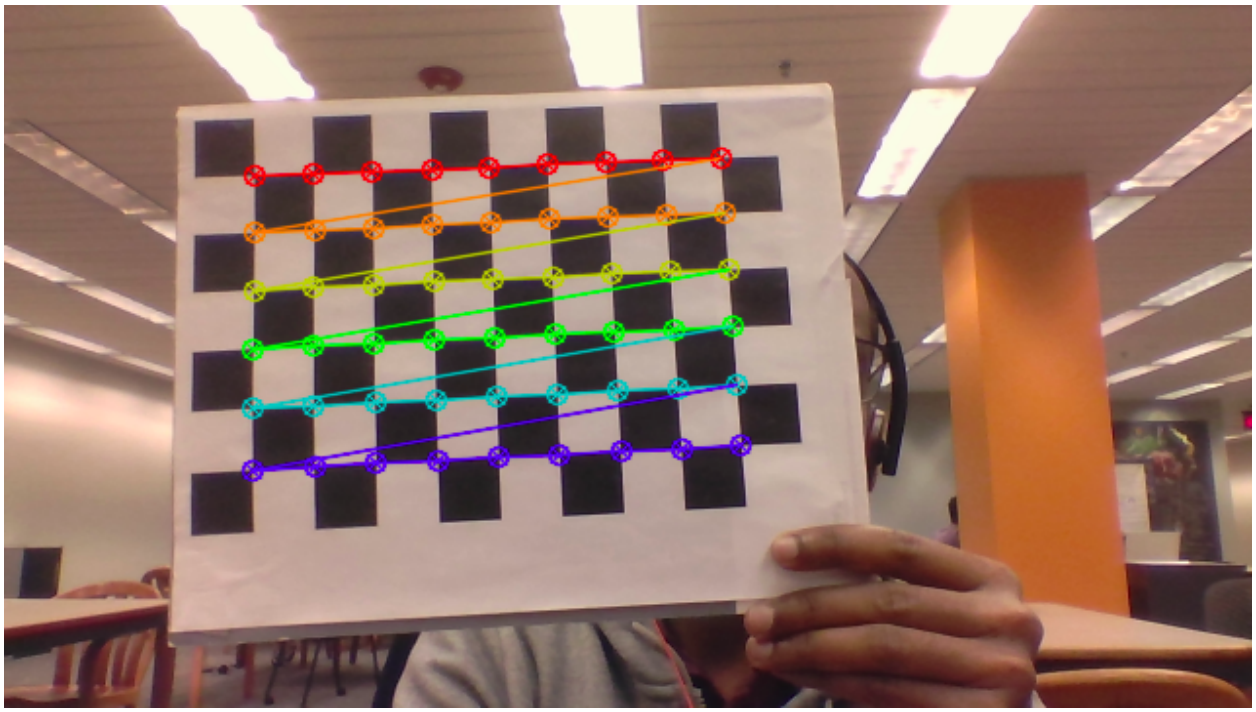

Figure 2: Example of calibration Image

Task3 : Calibrate Camera

The camera matrix was initialized using the parameters shown in equation 1 and distortion coefficients were all initialized as zero. The message shown in the command window while initializing is shown in figure 4.

[1, 0, frame.cols/2]
[0, 1, frame.rows/2] (eq : 1)
[0, 0, 1]

```
Initializing camera matrix and distortion coefficients
Camera matrix before calibration: [1, 0, 320;
 0, 1, 180;
 0, 0, 1]
Distortion Coefficients before calibration: [0, 0, 0, 0, 0]
```

Figure 3: Initial camera matrix and distortion coefficients displayed in terminal

The final calibration starts when the user presses ‘c’. There needs to be at least 5 images in the calibration set. The final camera matrix displayed in the terminal is shown in figure 5 and value are tabulated in figure 6 and 7 respectively

```
Final Re-projection Error: 0.212705
Final camera matrix: [427.2755281528624, 0, 304.4943299060169;
 0, 427.222476596039, 197.5708366009673;
 0, 0, 1]
Final distortion Coefficients: [0.01962484662485538, -0.1244731489173837, 0.004567230392922368, 0.001021840653133681, 0.08750626956723942]
```

Figure 4: FInal camera matrix and distortion coefficient displayed in terminal

Initial Camera Matrix			Final Camera Matrix		
1	0	320	427.27	0	304.49
0	1	180	0	427.22	197.57
0	0	1	0	0	1

Figure 5: Camera matrix

Distortion Coefficients	
Initial	Final
0	0.0196
0	-0.1244
0	0.0045
0	0.0011
0	0.0875

Figure 6: Distortion Coefficients

Final Re-projection error obtained is 0.212705, which is less. The camera matrix and distortion coefficients are stored in a calibration_data yaml file in the location specified by the user for later use.Both focal lengths are approximately the same (~427) and the image center is close to initial estimates.

Task4 : Calculate Current Position of the Camera

“arystem.cpp” will read the intrinsic camera parameters from the yaml file obtained in task 3. System detects a chessboard, grabs the corner locations and uses solvePnP to get the target’s pose. The rotation vector and translation vector were displayed in the terminal during the testing phase of the system.

```
rotation vector: [2.881326712205988;
 -0.1327294923692985;
 0.03243546717330444]
translation vector: [-7.79798350357778;
 -2.077456805792753;
 11.82866448639047]
```

Figure 7: Rotation vector and translation vector displayed in terminal

Task5 : Project Outside Corners or 3D axes

System uses projectPoints function to project the 3D points corresponding to four outside corners of the chessboard onto an image plane; it also displays the 3D axes at the origin of the chess board.

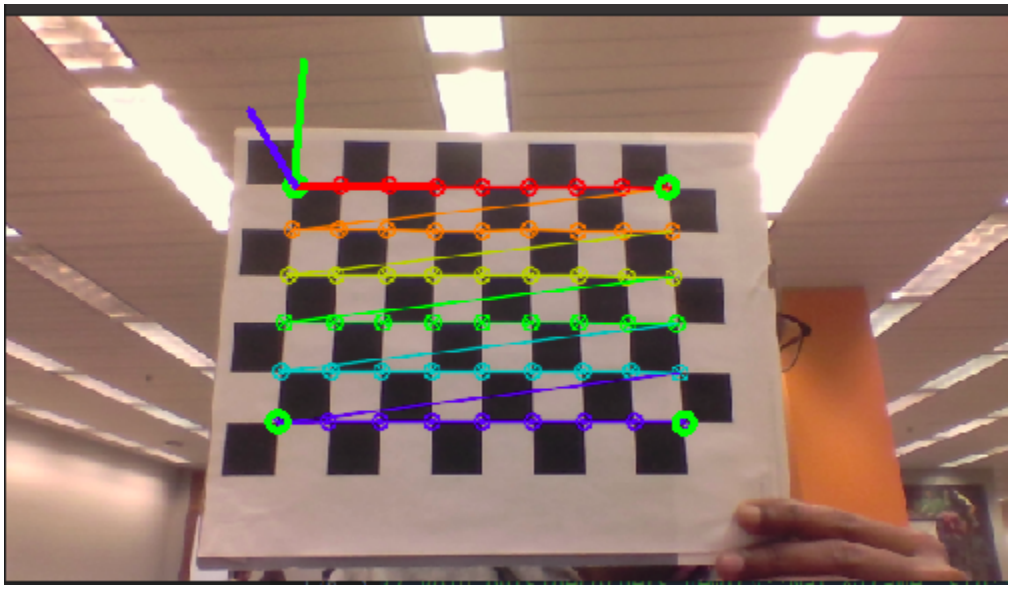

Figure 8: Projected corners and 3D axis

Task 6: Create a Virtual Object

I created virtual objects with the shape of a ‘Trapezoid’, ‘L’ and a ‘chair’. The ‘L’ shape is used since it is asymmetrical. It helps us to make sure that the object is always in the exact location when the camera moves around.

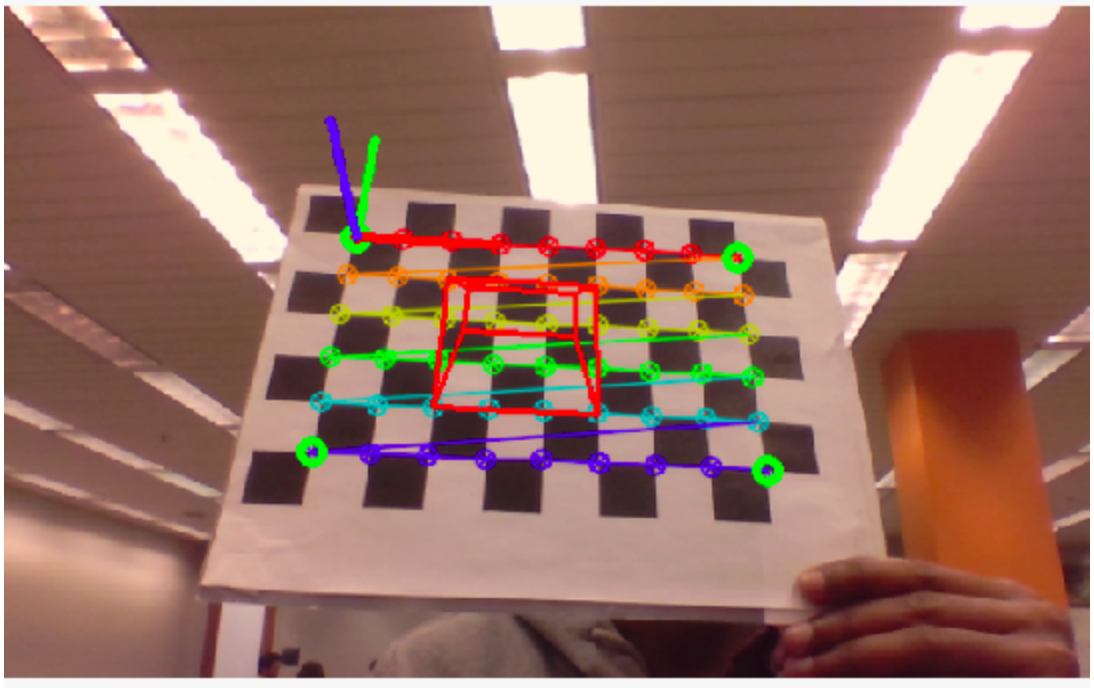

Figure 9: Trapezoid virtual object

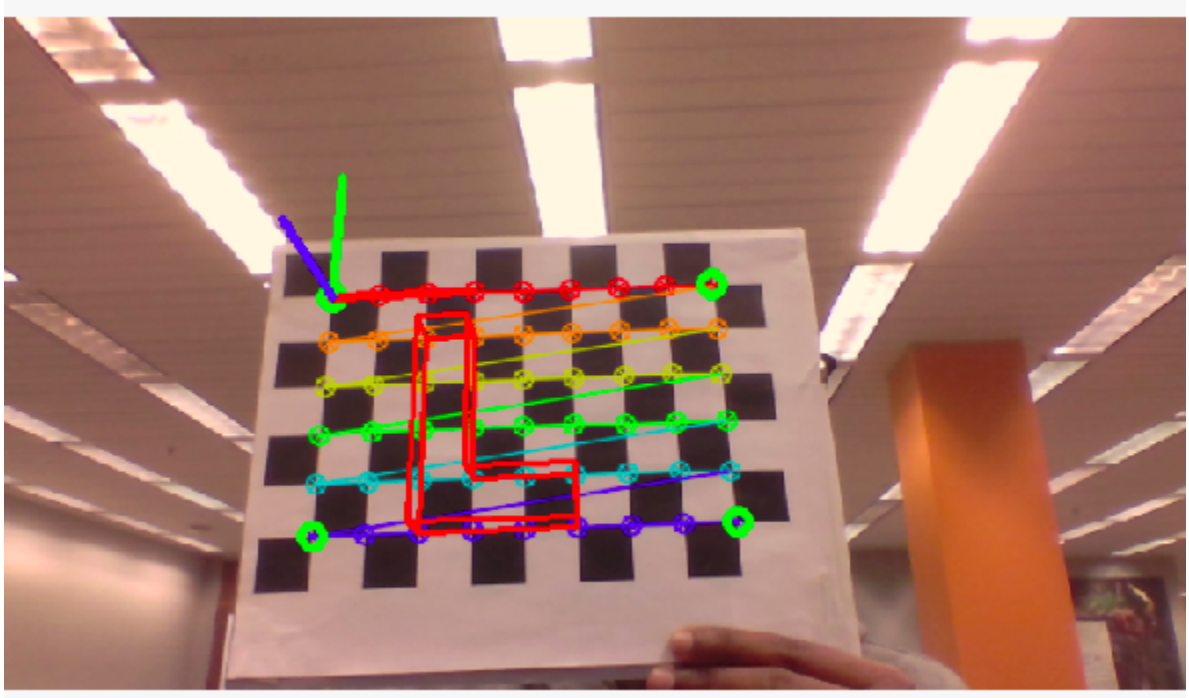

Figure 10: L shape virtual object

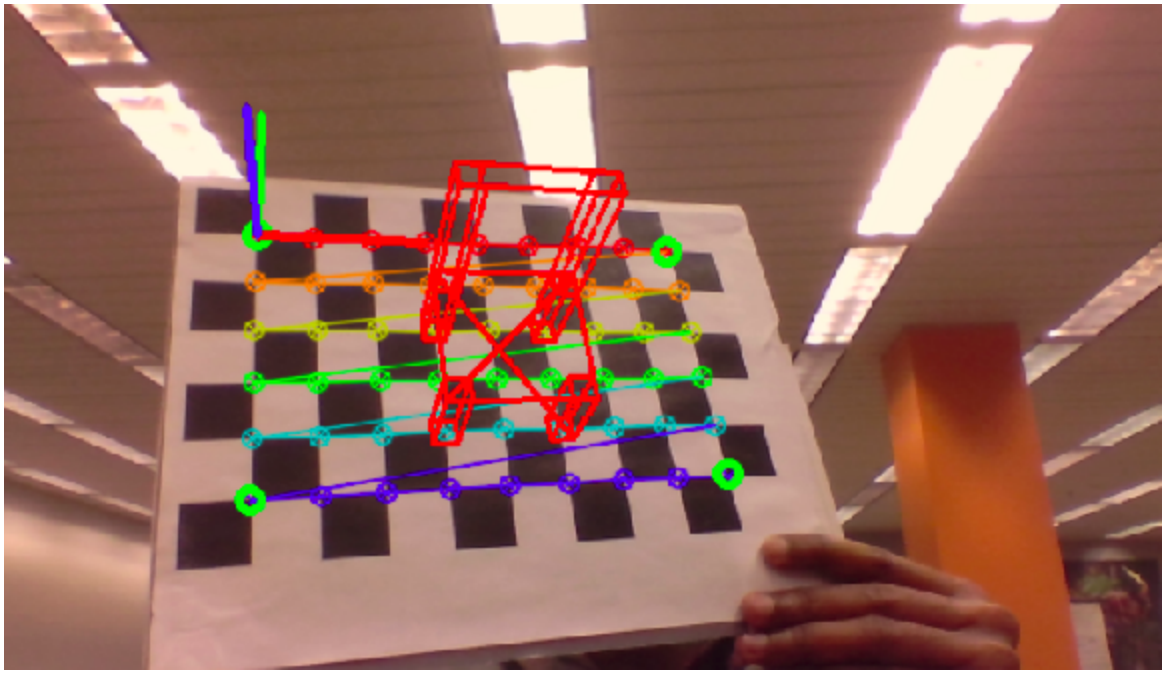

Figure 11: Chair virtual object

The video of system in action is available in the following links

Demo on chessboard:
https://northeastern-my.sharepoint.com/:v:/g/personal/madhusudhanan_a_northeastern_edu/EflOFViBgixFkfIbUyCAvTUB_N_z9WfwR4pDt3bURC-12Q?e=7JA4Nu

Demo on Aruco board:
https://northeastern-my.sharepoint.com/:v:/g/personal/madhusudhanan_a_northeastern_edu/EfqoRfUQp1JPsqU1N0OqvQcBVwsSccHDhIIm4AqSArOIWg?e=0vktpb

Task 7: Detect Robust Features
In this particular task, I was able to detect Harris corners on two different targets: a chessboard and a custom pattern.

Harris corners can be used to estimate the camera pose by matching the detected corners in the image with their corresponding 3D points in the real world scene. This involves using the correspondence between the 2D image coordinates and the 3D world coordinates, which is established through camera calibration.After we have established the correspondence between the 2D image coordinates and the 3D world coordinates, we can use it to estimate the camera pose. This involves solving for the position and orientation of the camera relative to the scene, using techniques such as perspective-n-point (PnP) algorithms.Once we have estimated the camera pose, we can use it to project virtual objects onto the image, in order to create an augmented reality experience. Harris corners can also be used for tracking the camera and objects in the scene, by detecting and matching distinctive features across multiple frames of

the video feed, which can help to maintain the alignment between the virtual objects and the real world scene.

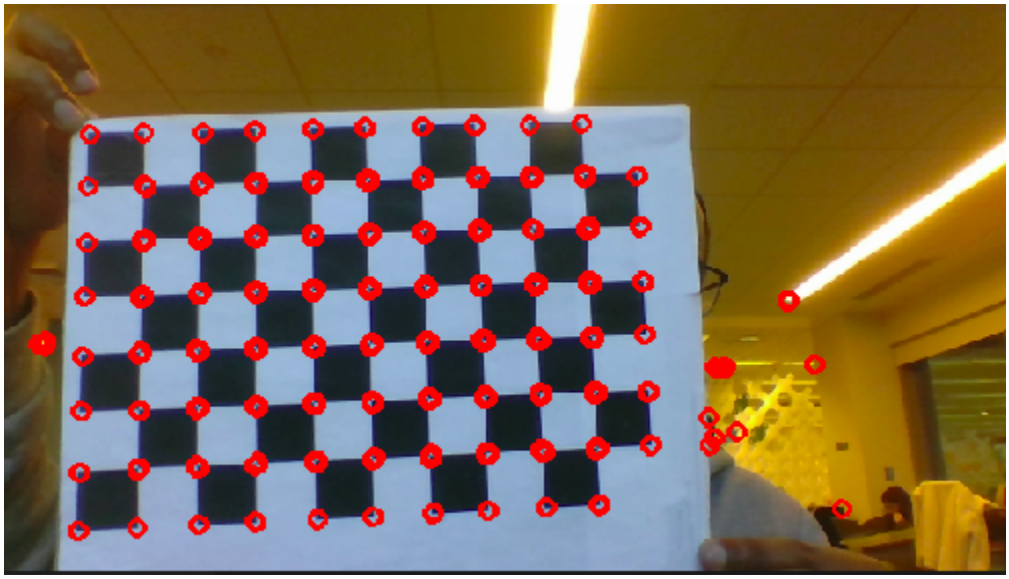

Figure 12: Harris Corners detected on chessboard

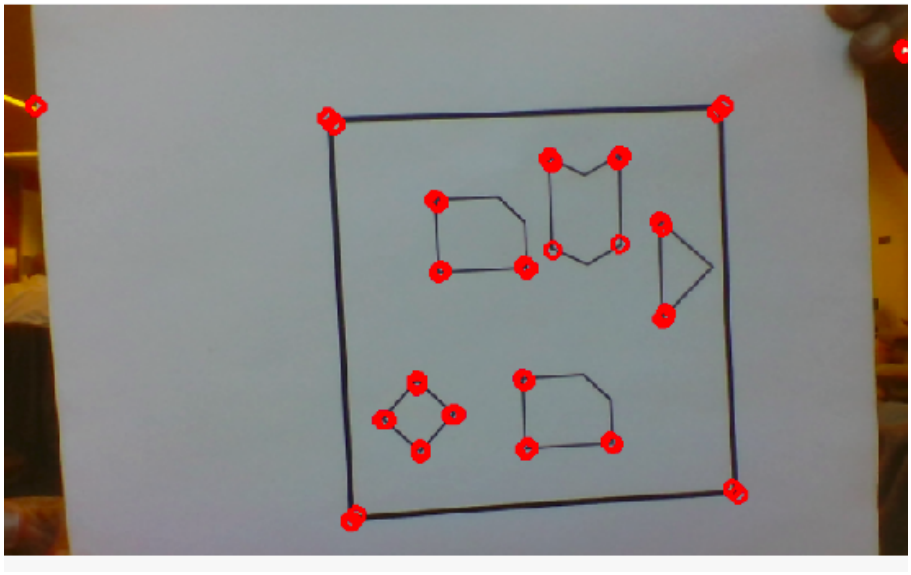

Figure 13: Harris Corners detected on pattern

Demo :
https://northeastern-my.sharepoint.com/:v:/g/personal/madhusudhanan_a_northeastern_edu/EW6N5WTBtqFPhqD__cjVfQEBTn-I0QKMzmGGihsPaadCPw?e=KwRbAe

Extensions:

1) Implementation of Aruco Boards as target

System can detect an Aruco Board as a target as shown in figure 14 and can project an 'L' shape virtual object on the Aruco Board target as shown in figure 15.

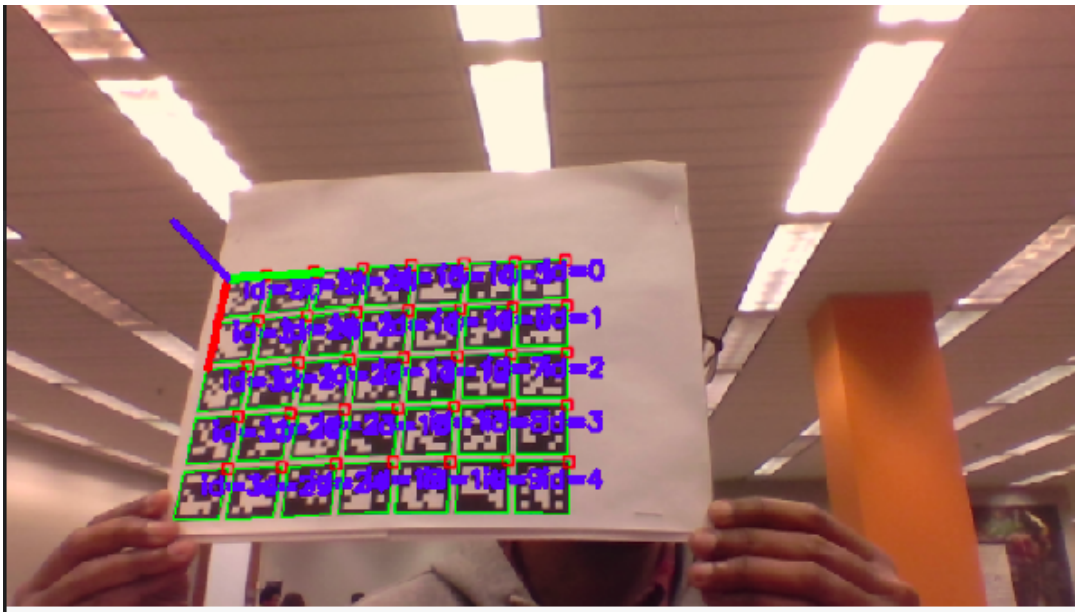

Figure 14: Aruco target is detected

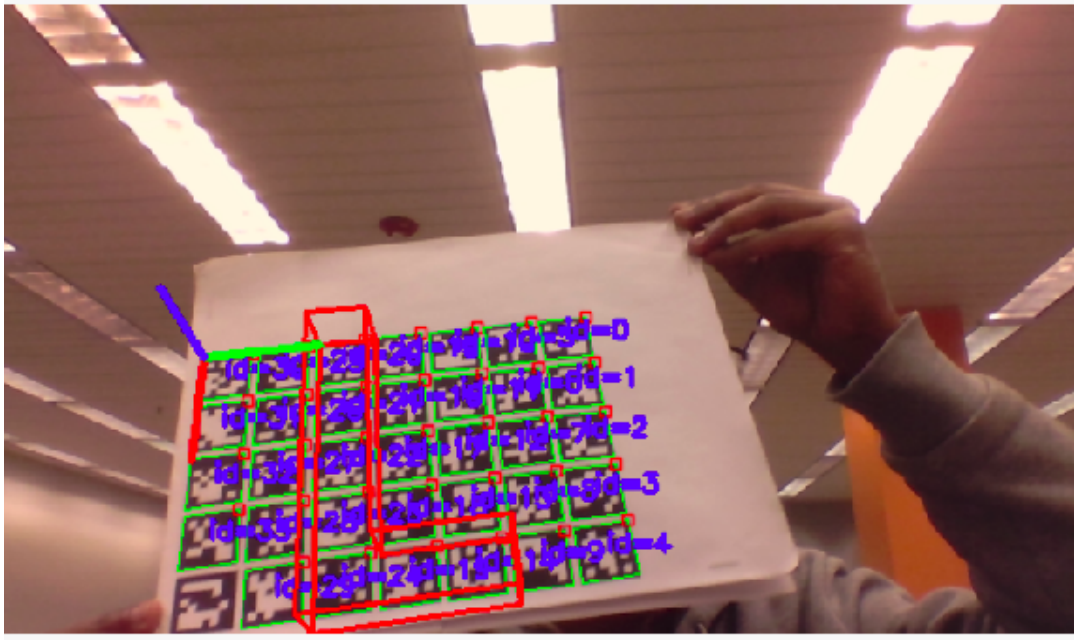

Figure 15: L shape virtual object on Aruco board

2) Detection of multiple targets at once

System can detect multiple targets (Chess board and Aruco board) at once as shown in figure 16.

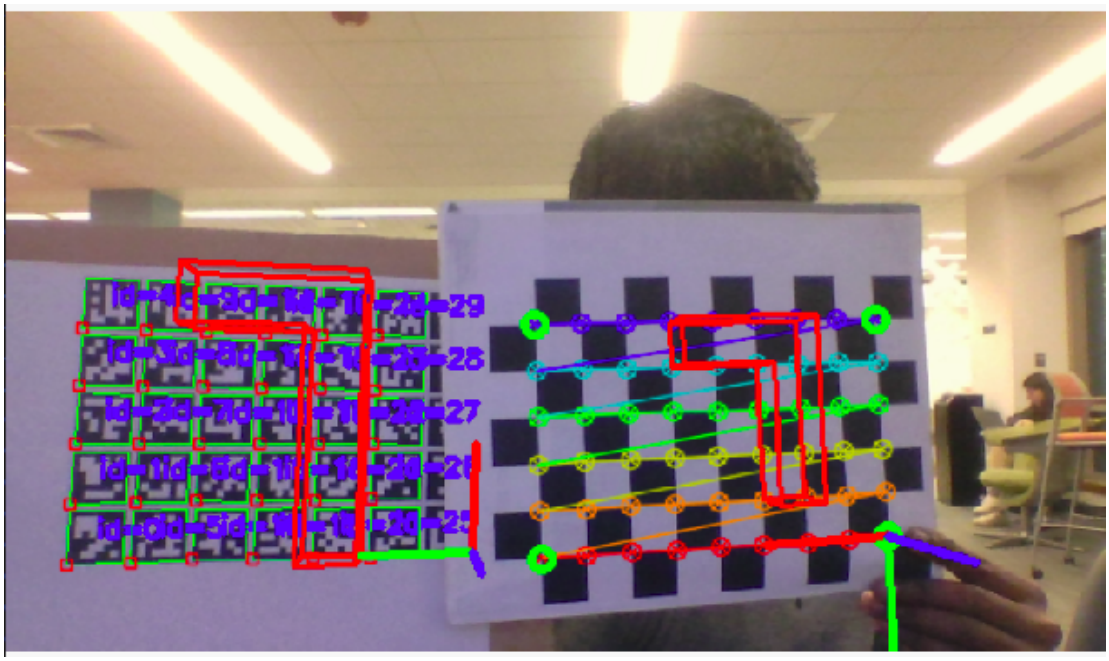

Figure 16: Multiple targets detected

3) Calibration of multiple devices

AverMedia webcam and a smartphone (Redmi Note 10 Pro) camera were also calibrated using the system.

Final Camera Matrix		
775.2715	0	485
0	779.85	241
0	0	1

Distortion Coefficients	
Initial	Final
0	0.1851
0	-0.6894
0	-0.0147
0	0.0105
0	0.7224

Figure 17: camera matrix and Distortion Coefficients of webcam

Final Camera Matrix		
640.97	0	471.73
0	641.43	279.043
0	0	1

Reprojection error : 0.832024

Distortion Coefficients	
Initial	Final
0	0.1408
0	-0.7743
0	-0.0059
0	-0.0051
0	1.4463

Figure 18: camera matrix and Distortion Coefficients of Smartphone

Reprojection error : 0.320392

Reprojection error of smartphones is less compared to that of the webcam , this might be due to the fact the smartphones are factory calibrated during the manufacturing process itself. For both webcam and smartphone, focal lengths in x and y are approximately the same and the image center is close to initial estimates.

4) The target doesn't look like a target any more.

An image provided by the user can be overlaid over the chessboard target if the user presses 'o' as shown in figure 17.

Figure 17: Image superimposed on target

Reflection

I got a deeper understanding about how to perform camera calibration and about augmented reality concepts. I understood about various in-built functions used for camera calibration and augmented reality in opencv. Learning about storing calibration data in a yaml file for future use was beneficial since many popular GitHub repositories use this method. It was fun to project virtual objects onto the target image. I got a deeper understanding about Harris corners and how it can be used for augmented reality tasks

Acknowledgments

1. Camera calibration using Aruco Boards
   https://docs.opencv.org/4.2.0/da/d13/tutorial_aruco_calibration.html
2. Superimpose images over target
   https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/
3. Camera calibration using Chess Boards
   https://docs.opencv.org/4.2.0/d4/d94/tutorial_camera_calibration.html
4. Harris Corners using OpenCV
   https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector.html