Homework 4
Calibration and Augmented Reality
Cs 5330: Pattern Recognition & Computer Vision

Author: *Rishabh Singh & Hrithik Ketanbhai Kanoje*

Instructor: *Prof. Bruce Maxwell*
Department: Computer Science

Contents

1 Summary

This project is about learning how to calibrate a camera and then use the calibration to generate virtual objects in a scene. After getting calibration parameters System will be able to identify a target and then position a virtual item in the scene next to the target so that it moves and orients itself appropriately in response to camera or target motion.

1.1 TASK 1 : Detect and Extract Chessboard Corners

In this task we have used opencv's built-in function "**findChessboardCorners()**" ,"**cornerSubPix()**" and drawChessboardCorners()"**drawChessboardCorners()**" to detect and extract chessboard corners. Refer to Fig.1

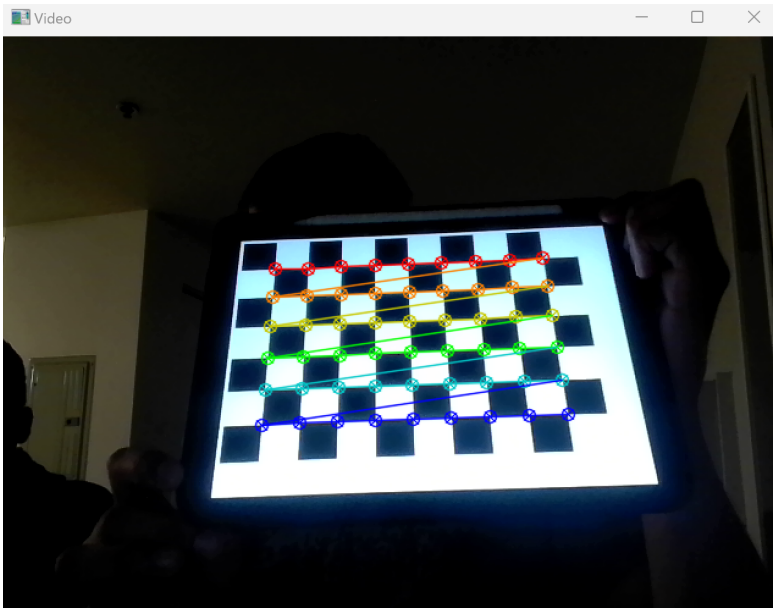

Figure 1: Task 1: Detected corners

1.2 TASK 2 : Select Calibration Images

when the user presses 's' on keyboard it takes the current image as image for calibration, user need to perform this 5 times or more ans the program will store corner locations with respect to 3D world points which will be taken as input in order to calibrate camera. Refer to Fig.2

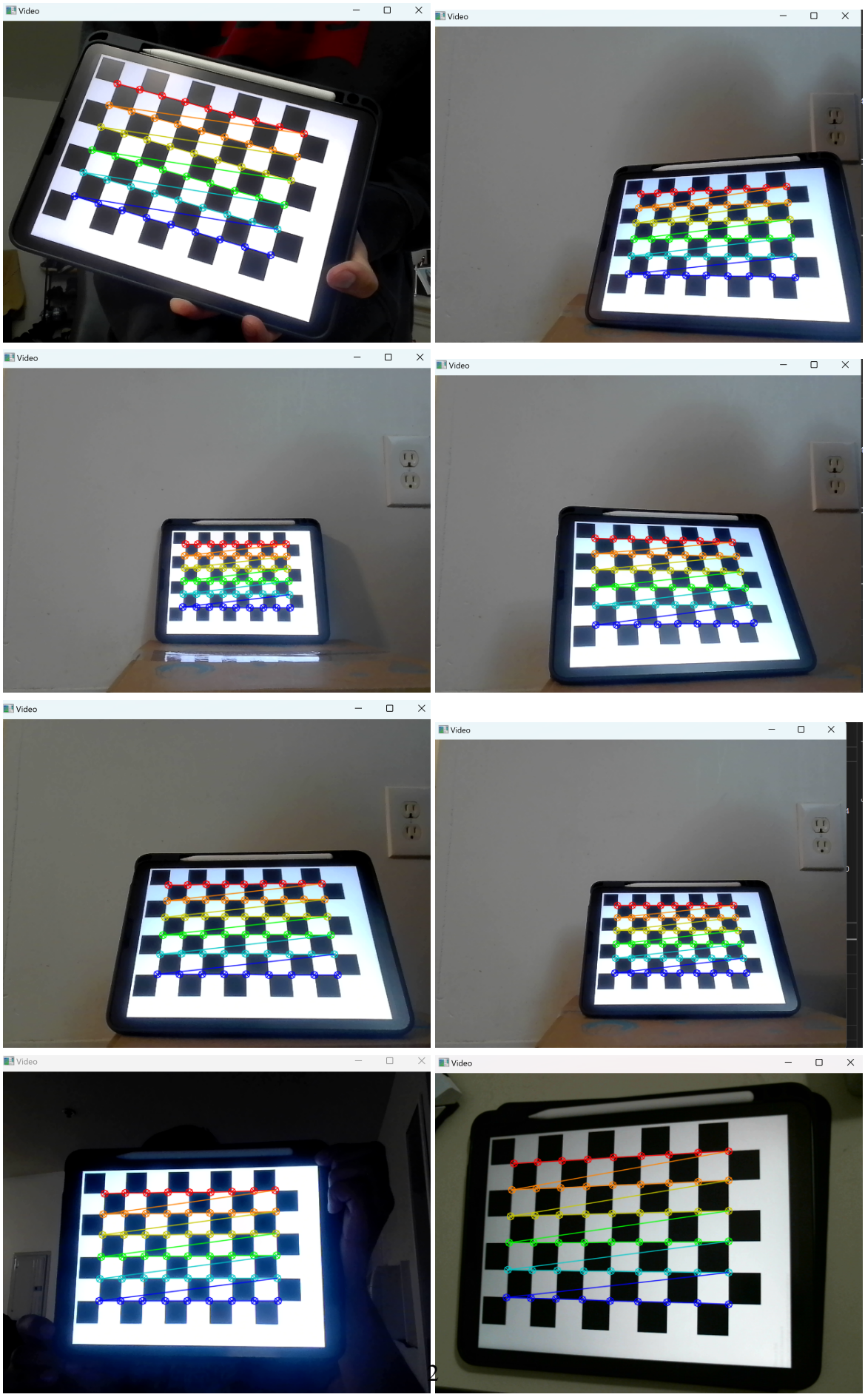

Figure 2: Task 2: Selected Images for calibration

1.3 TASK 3 : Calibrate the Camera

after selecting a minimum 5 number of images or more than that user needs to press ‘c’ in order to perform calibration. For calibration we are using opencv’s built-in function ”**calibrateCamera()**” Refer to Fig.3

```
7 frame collected, running a calibration:
initial camera matrix:
1       0       320
0       1       240
0       0       1
camera_matrix:
543.177 0       319.573
0       543.177 239.698
0       0       1
distortion_coefficients:
0.115348        -0.448995       -0.00263548     -0.00281418     0.473387
re-projection error: 0.163768
save camera matrix and distortion coefficients to csv file.
```

Figure 3: Task 3: Camera Matrix, Distortion coefficient and Re-projection error

1.4 TASK 4 : Calculate Current position of the Camera

Obtaining camera calibration data from a file, followed by starting a video loop. The application tries to identify a chessboard pattern in each frame of the video loop. In the event that the pattern is discovered, the program records the positions of the corners and applies **”solvePNP()”** to determine the pose of the board, which takes into account its rotation and translation. Refer to Fig.4

```
translation matrix
-6.80398
-0.679069
15.5054
rotation matrix
2.67662
-0.163139
0.12676
translation matrix
-7.23069
-0.540975
15.9067
rotation matrix
2.61159
-0.158109
0.0718168
translation matrix
-7.88197
-0.228788
16.652
rotation matrix
2.58078
-0.166997
0.0750626
translation matrix
-8.29767
0.00960283
```

Figure 4: Task 4: Rotation and translation Matrix

1.5 TASK 5 : Project Outside Corners or 3D

posture calculated in the previous phase to produce a collection of 3D points that represent the chessboard’s four outer corners in the world coordinate system.Then, as the chessboard or camera moves, it employs the **”projectPoints()”** function to project these 3D points onto the picture plane in real-time.This enables us to see where the chessboard’s corners are located in the camera’s field of vision and to track its movement and orientation in relation to the camera.

In order to view the board's rotation and translation in real time, 3D axes are also added to the board's origin. Refer to Fig.5

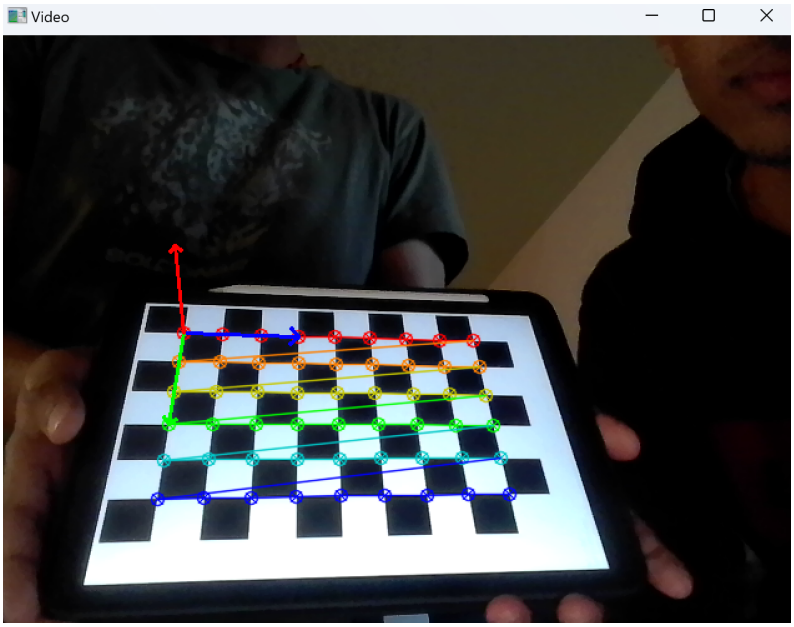

Figure 5: Task 5: 3D axes

1.6 TASK 6 : Create a Virtual Object

In this task, we have used teapot object from online resources and when chessboard corners are found it reads parameters from CSV file and Displays the virtual object on chessboard Refer to Fig.6

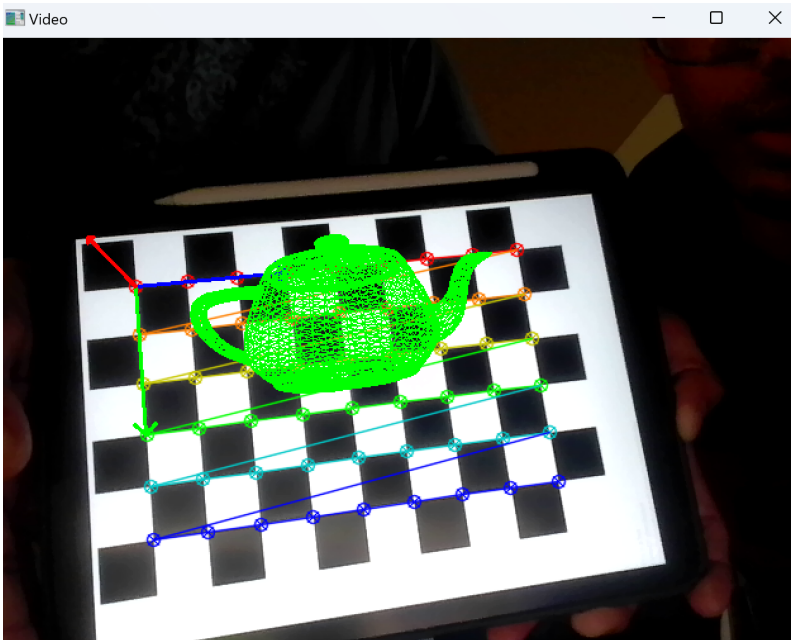

Figure 6: Task 6: Virtual object

1.7 TASK 7 : Detect Robust Features

For this task, we have written separate program for detecting Harris corners. For this task we have used opencv's in-built function **"cornerHarris()"**. The Harris Corner algorithm is useful for locating a panel in an image to create a 3D object rendering. Similar to the **"findChessBoardCorners()"** function, Harris Corner identifies the prominent corners in the image. However, it is essential to ensure that the corners are stable and in a specific order. Unlike a chessboard, where we know the number of points per row and column beforehand, it is challenging to describe the points for a custom target. The Harris algorithm does not guarantee the exact number of points and only provides the locations of the corners without any specific order. Without an order, it is challenging to map the pixel-level 2D coordinates with the real-world 3D coordinates for calibration. Refer to Fig.7

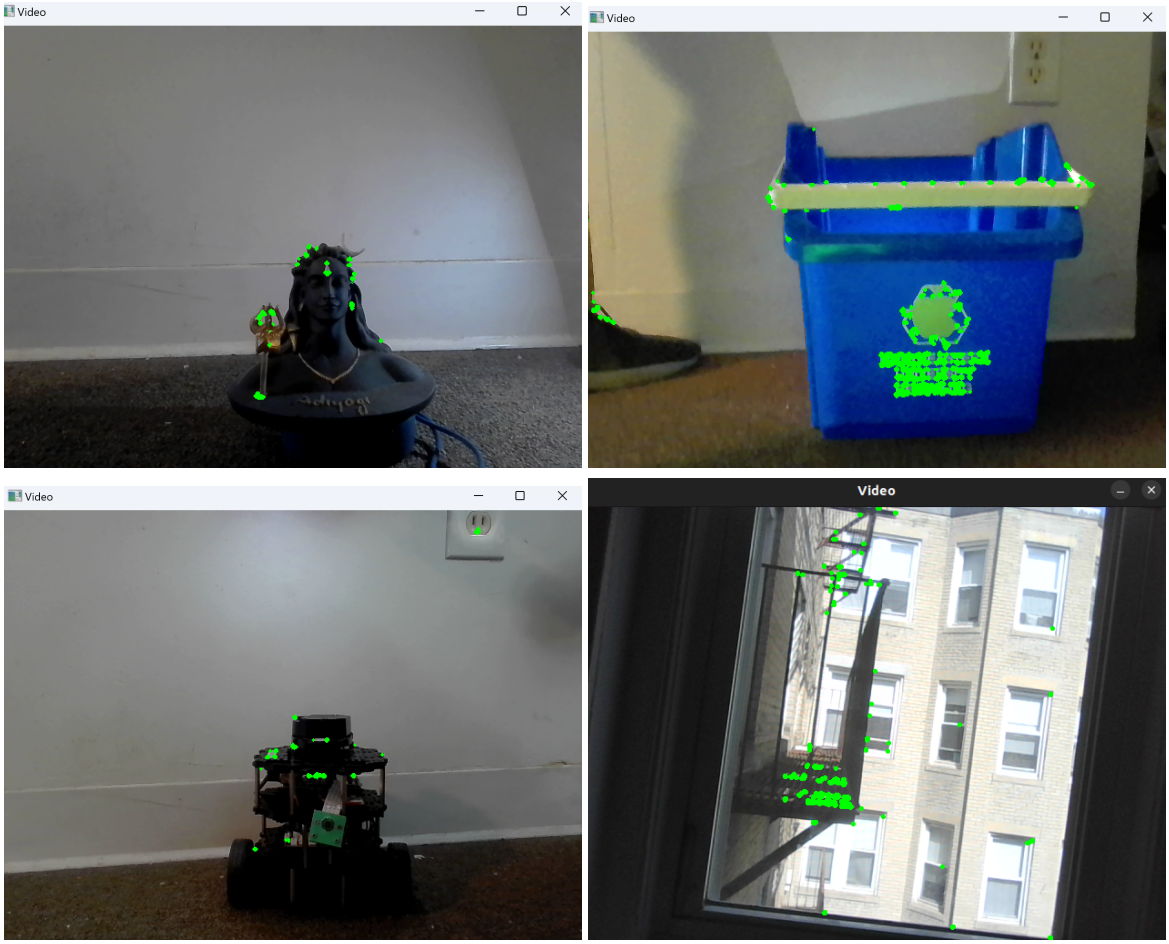

Figure 7: Task 7: Harris corners

1.8 TASK 8 : Extension

1.8.1 Extension 1 - Get your AR system working with a 3D object placed on checkerboard. Place an AR object somewhere in the world relative to the object.

I looked for a number of rendering-related open-source object files.These object files are quite basic; they only have two fields, "v" and "f," where "v" denotes a vertex represented by a 3D

coordinate and ”f” denotes a face made up of the index of these vertices.The reasoning in this case is to read the vertices and faces, then draw a number of lines to connect the appropriate vertices on each face.There is one file in my submission: a teapot Refer to Fig.8

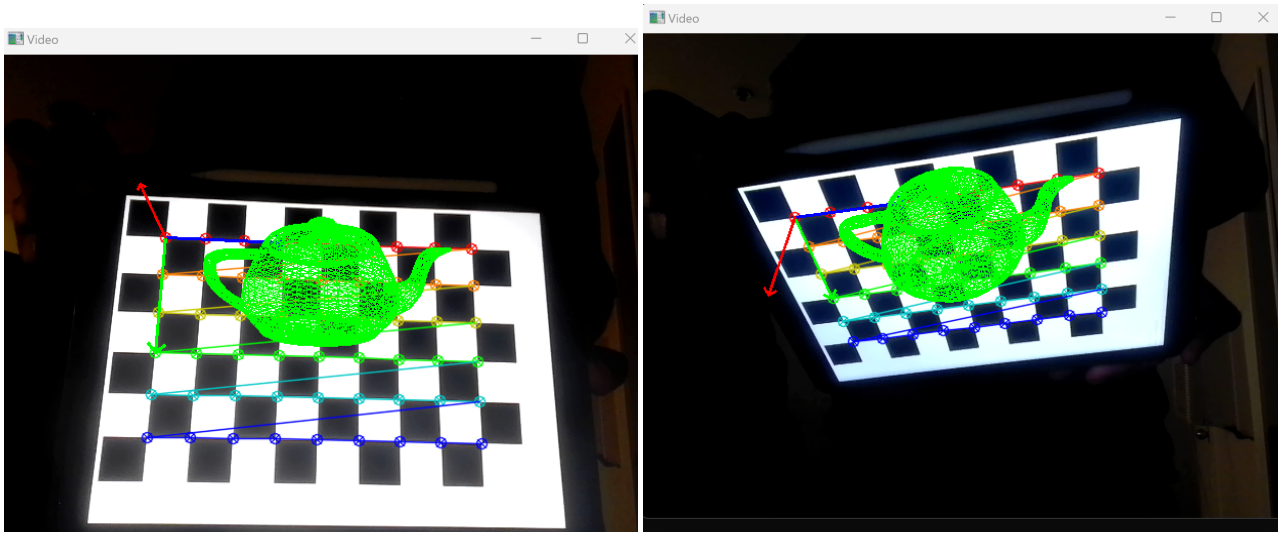

Figure 8: Extension 1: Teapot on chessboard

1.8.2 Extension 2 - Performed camera calibration on 3 different camera and comparing the result.

For this task, We performed camera calibration on 3 different cameras and comparing results of all devices. From the table it can be seen that re-projection error depends on number or images used for calibration, the environment and lighting condition and quality of image quality of camera.

Table 1: Comparison of Camera Calibration

Camera	Number Of Images	Reprojection Error
Dell Laptop in-built Webcam	7	0.163768
MSI Laptop in-built Webcam	11	0.15143
Logitech USB Webcam	10	0.131255

2 Learning Outcomes

- Learned about camera calibration
- Learned about Harris corner detection
- Learned about various OpenCV commands such as findChessboardCorners, drawChessboardCorners, calibrateCamera, solvePNP, projectPoints, cornerHarris and many more

3 Project Demo Videos

This link contains 2 folder naming Augmented reality and Harris. To see the demo videos of project please visit this link Homework-4 Video or go to the next url: https://drive.google.com/drive/folders/1Fl5zsKpFPa6sQMd1JSOk-HGvyS_igf2V?usp=sharing

References

[1] calibratecamera. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d.

[2] cameracalibration. https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html.

[3] cornerharris. https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html.

[4] cornersubpix. https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga354e0d7c86d0d9da75de9b9701a9a87e.

[5] drawchessboardcorners. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga6a10b0bb120c4907e5eabbcd22319022.

[6] findchessboardcorners. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a.

[7] projectpoints. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga1019495a2c8d1743ed5cc23fa0daff8c.

[8] solvepnp. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d.