Project 4: Calibration and Augmented Reality

CS 5330: Pattern Recognition and Computer Vision

Arjun Rajeev Warrier

Abstract

In project 4, the goal is about learning how to calibrate a camera and then use the calibration to generate virtual objects in a scene. The end result was a program capable of detecting a target to create and project 3D objects that could move and orient itself relative to the motion of the camera and the target.

Experiment

The functionalities that were programmed as part of this project has been described below. This is a brief overview with short explanations and images that demonstrate the effect of these codes.

1. Detect and Extract Chessboard Corners

A function was programmed using OpenCV functions to scan a video frame while detecting and extracting the chessboard corners from it. These corners were then drawn on the chessboard virtually for validation. Also prints out the number of corners detected.

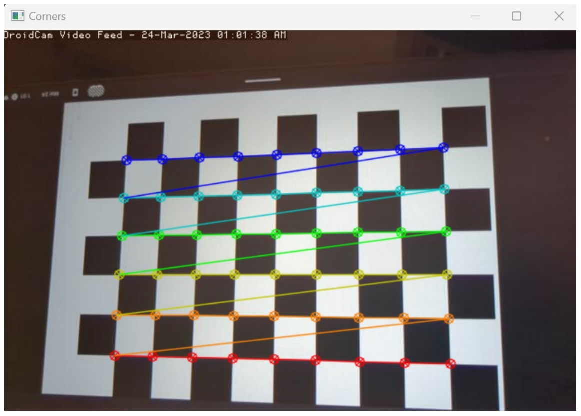
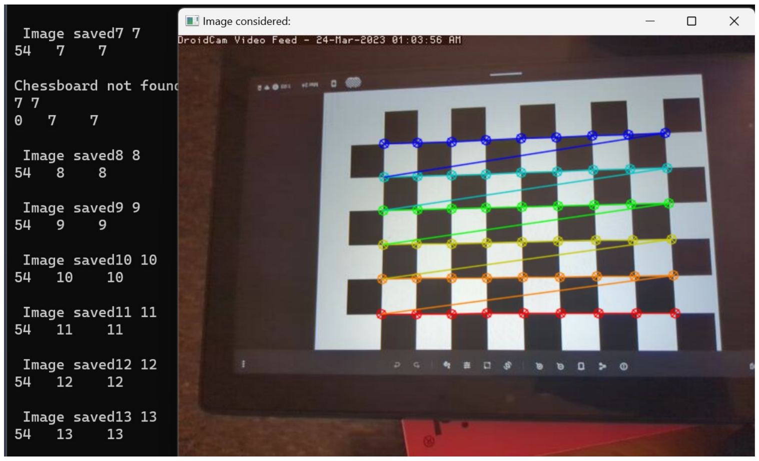


Figure 1: Chessboard corners drawn.

2. Select Calibration Images

A function that lets the user specify that a particular image should be used for the calibration and save the corner locations and the corresponding 3D world points. The user can continue pressing 'S' to store calibration images which are saved to the source folder as long as there is a chessboard detected in it. Below image is a saved calibration image.

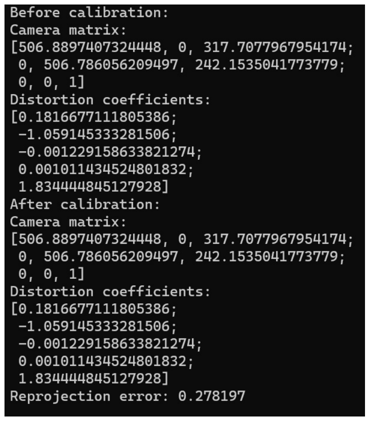


```
 Image saved7 7
54   7    7

Chessboard not found
7 7
0   7    7

 Image saved8 8
54   8    8

 Image saved9 9
54   9    9

 Image saved10 10
54   10    10

 Image saved11 11
54   11    11

 Image saved12 12
54   12    12

 Image saved13 13
54   13    13
```

Figure 2: A calibration image.

3. Calibrate the Camera

If the user selected at least 5 calibration images, then he can successfully use the calibration function that computes the intrinsic properties of the camera being used and stores them to an external xml file in the source folder. These intrinsic properties will later be used for projection onto 2D spaces.



```
Before calibration:
Camera matrix:
[506.8897407324448, 0, 317.7077967954174;
 0, 506.786056209497, 242.1535041773779;
 0, 0, 1]
Distortion coefficients:
[0.1816677111805386;
 -1.059145333281506;
 -0.001229158633821274;
 0.001011434524801832;
 1.834444845127928]
After calibration:
Camera matrix:
[506.8897407324448, 0, 317.7077967954174;
 0, 506.786056209497, 242.1535041773779;
 0, 0, 1]
Distortion coefficients:
[0.1816677111805386;
 -1.059145333281506;
 -0.001229158633821274;
 0.001011434524801832;
 1.834444845127928]
Reprojection error: 0.278197
```

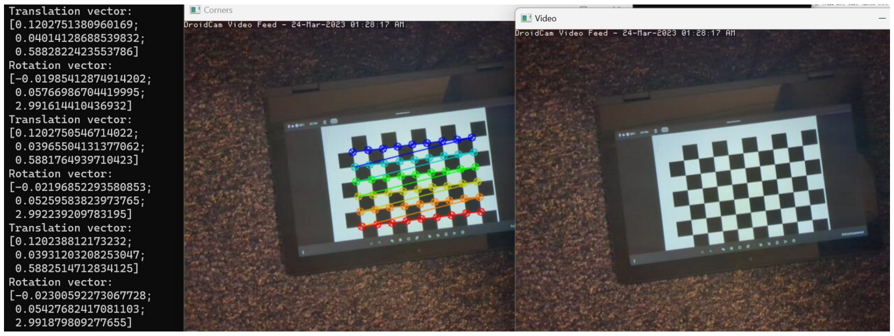
Figure 3: Re projection error after a successful calibration function usage.

4. Calculate Current Position of the Camera

The solvePNP function is then utilised to get the board's pose (rotation and translation) using the camera calibration parameters from the intrinsic.xml stored earlier.

5. Project Outside Corners or 3D Axes

Three functions were written here. Pressing 3 will let the user see the borders on the chessboard



```
Translation vector:
[0.1202751380960169;
 0.04014128688539832;
 0.5882822423553786]
Rotation vector:
[-0.01985412874914202;
 0.05766986704419995;
 2.991614410436932]
Translation vector:
[0.1202750546714022;
 0.03965504131377062;
 0.5881764939710423]
Rotation vector:
[-0.02196852293580853;
 0.05259583823973765;
 2.992239209783195]
Translation vector:
[0.120238812173232;
 0.03931203208253047;
 0.5882514712834125]
Rotation vector:
[-0.02300592273067728;
 0.05427682417081103;
 2.991879809277655]
```

Corners
DroidCam Video Feed - 24-Mar-2023 01:28:17 AM
Video
DroidCam Video Feed - 24-Mar-2023 01:28:17 AM

Figure 4: Rotation and Translation vectors being printed out.

with the corners marked. Pressing 4 will show the 3D axes on the chessboard. Pressing 5 will show circles around the 4 vertices of the chessboard.

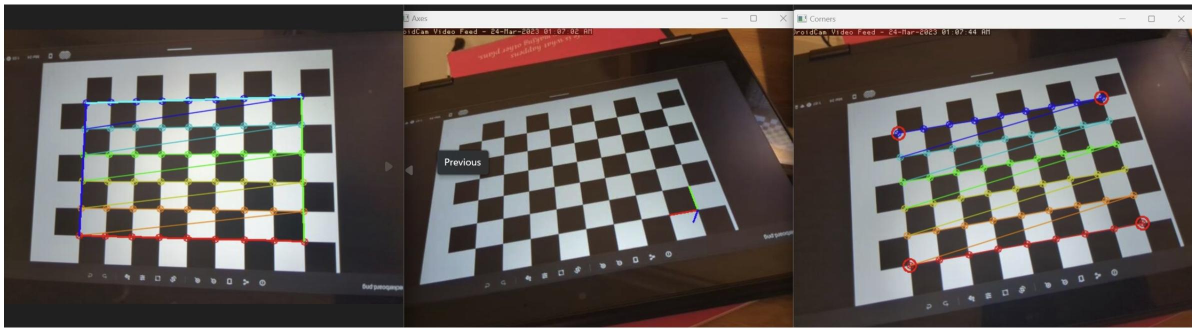

Axes
DroidCam Video Feed - 24-Mar-2023 01:07:02 AM
Previous
Corners
DroidCam Video Feed - 24-Mar-2023 01:07:44 AM

Figure 5: The outputs of the three functions.Border(left), 3D axes (center) and vertices(right).

6. Create a Virtual Object

A 3D square pyramid with the edges coloured differently was projected onto the center of the chessboard. The edges were coloured differently to make it easier to see the rotation of the object when rotating the camera. Video attached with code files.

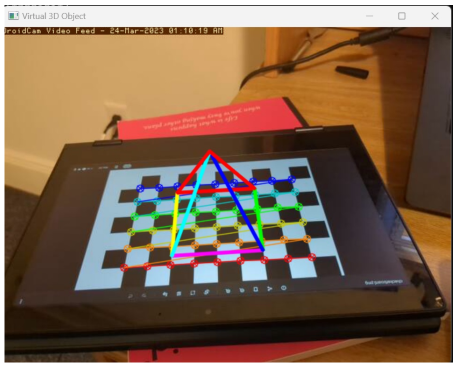


Figure 6: 3D square pyramid of different edge colours.

7. Detect Robust Features

A different program was written to exhibit the detection of Harris Corners. This implementation proved to be faster than what was see earlier. A video was included to demonstrate its working.

Making a pattern like a QR code or other similar AprilTag kind of designs, could help in identifying spaces to project 3D virtual objects onto. Place such a design onto the ground and project a pokemon on it. Now, if someon comes looking at it with a Pokemon GO camera, the smartphone could show a Pikachu where the earlier trained tag features show up. This is just an example I quickly thought up of.

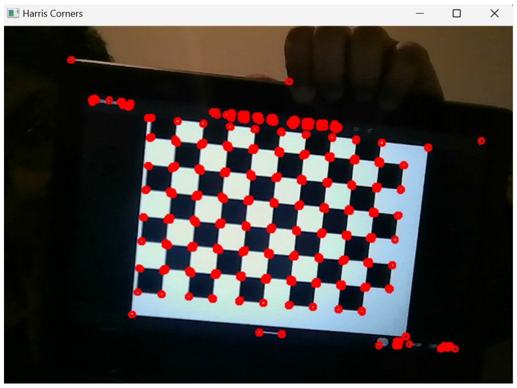


Figure 7: Harris corner detection.

8. Extensions

As part of the extension, in order to explore placement and scaling relative to the target, with the utilisation of corners acquired from the OpenCV functions; I implemented two functions. One function accurately places a cube on the first square of the chessboard. This can help in identifying the top corner of the chessboard as detected by the system. The second function implements a colour varying barrier/shield that masks the chessboard completely to give the disguise of an actual object placed on a surface. This has applications in masking or placeholding. Place this chessboard anywhere and projecting a 3D object onto it, will hel in identifying its appearence and size in a room.(An idea that is utilizable in real life) For example, to see how a cupboard would look in a positin in a room. PLace the chessboard on the ground and use this masking idea.

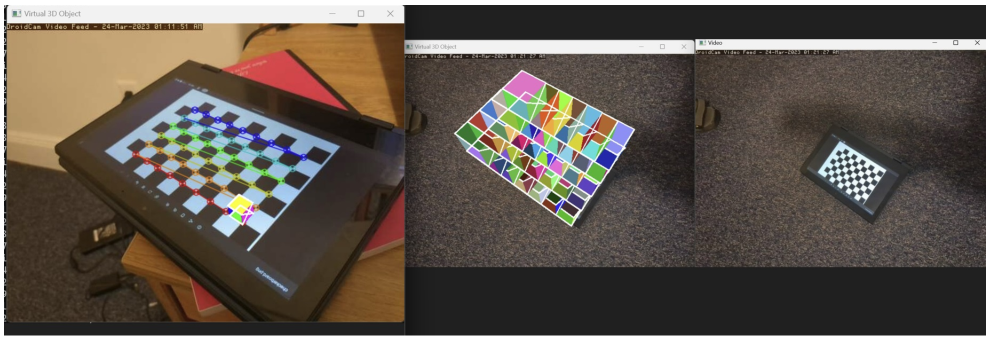


Figure 8: First square cube(Left). Masking(Right)

Reflections

Some key takeaways for this project were:

- Projecting 3D virtual objects onto the chessboard gives an idea of how masking and all those instagram filters actually work. My extension is definately something I will work on more to finally implement something similar to Amazons AR feature of trying out how an object would look in a room.

- Changing the appearence of the target virtually was also something explored through the extensions.

- This project gives an idea of how to start thinking and exploring augmented reality and virtual 3D objects. This of course also helps in giving an idea about how virtual reality goggles could also function.

References

[1] https://docs.opencv.org/

[2] Liu, Y., Liu, S., Cao, Y. and Wang, Z. (2016), Automatic chessboard corner detection method. IET Image Processing, 10: 16-23. https://doi.org/10.1049/iet-ipr.2015.0126

[3] https://stackoverflow.com/questions/55046025/chessboard-camera-calibration-using-opencv