


Project 4: Calibration and Augmented Reality

Overview

The ultimate goal of the project is to render a virtual object onto a target. It begins with learning how to calibrate, I have set a threshold of 5 images, once the 5 images are collected, it prints ready to calibrate on the current frame. With the collected set of image coordinates and corresponding real-world coordinates, we calibrate the camera matrix and the distortion coefficients. I then create a 3D virtual object and then draw it on the 2D plane.

The implementation supports two targets: chessboard and circles grid.

Button	Operation
's'	Save the photos and corners to calibrate the camera working with
'q'	Quit the Program
'n'	Go back to the initial state of detecting chessboard
'c'	If the count is less than the threshold(5), print cannot calibrate Else if the count of images is more than 4, it calibrates the camera and saves the values in a CSV file
'p'	Draws the 4 corners and the axes and calculates the position of the camera(rotation and translation)
'd'	Alternate between drawing the cuboid 3D object.
'C'	To load the calibration from the CSV file where it is stored.
'b'	Removes the background and the target only shows the 3D object
'v'	Loads a hardcoded video and then detects the chessboard pattern and adds the virtual object to the video and saves it
'i'	Loads a hardcoded image and detects chessboard pattern and if present inserts the virtual object to the image
'h'	Harris Corner detection (featureDetection.cpp)

Task	Comment	Result
Task 1: Detect and Extract Chessboard Corners	It tries to detect chessboard from the live stream and then draws chessboard corners when it finds it. It also prints the top leftmost corner location and the number of corners detected from the chessboard.	

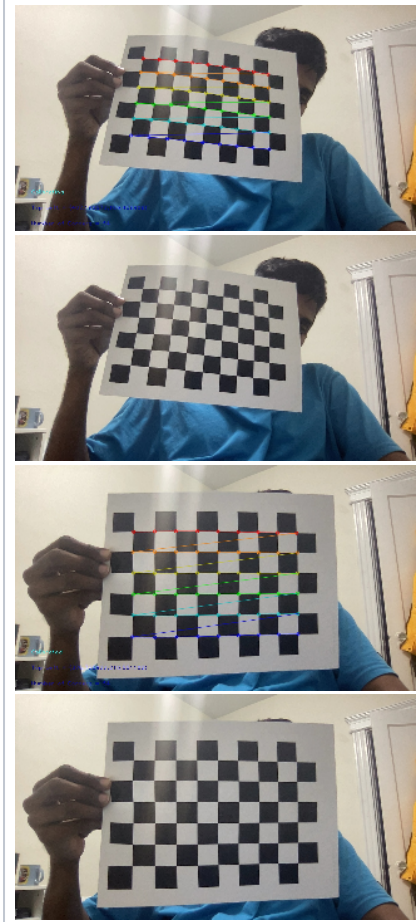
Task 2: Select Calibration Images

When the user clicks the 's' button, if there is a chessboard detected in the live stream then the corner points are collected, stored for calibration, and the image is stored in the local drive.

Have two sets of images in the result where the initial set is the image normal input picked from the live video stream and it prints the top left corner and the number of corners in the image.

Along with this, the calibration stage is also present, the second set of images have calibrated written with respect to the current scenario, and the first set has calibrating specifying that it needs still more images to calibrate.

During this phase, a text is printed saying the system is **not ready to calibrate** with the present data, till it reaches threshold images. Once it reaches it is updated saying **ready for calibration**. Once the calibration key is clicked it is updated saying **calibrated**.



Task 3: Calibrate the Camera

When the user clicks 'c', after collecting 5 image frames at least. It takes the image coordinates and the real-world coordinates and calculates the camera matrix, distance coefficients, and projection error.

Focal Length: 948.034 same for x and y

Error Estimate : 0.207107

C_x: 638.1

C_y: 362.47

Mat Camera Matrix After calibration

948.034	0	638.1
0	948.034	362.47
0	0	1

Mat Distortion Coefficient After calibration

-0.201851
1.93843
-0.00228692
-0.00219352
-5.58871

RMS error reported by calibrate camera: 0.207107

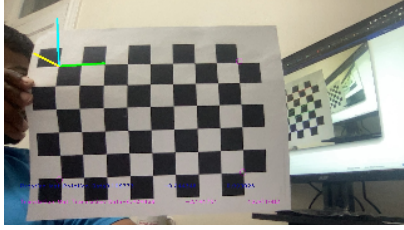
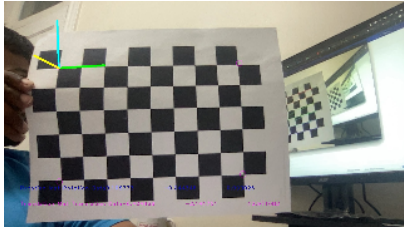
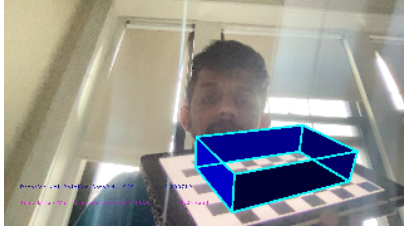
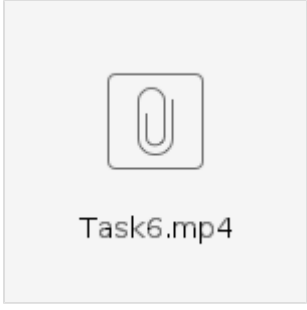

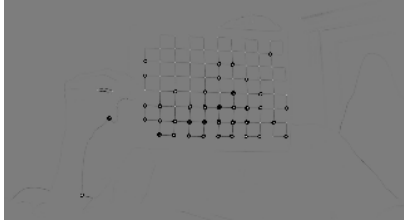
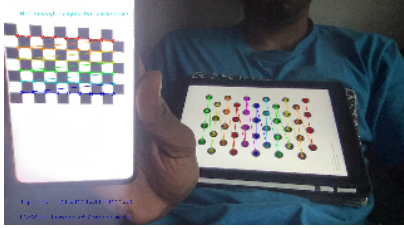
CSV file stored with the camera used so that we can fetch the properties later based on the camera, as part of the extension.

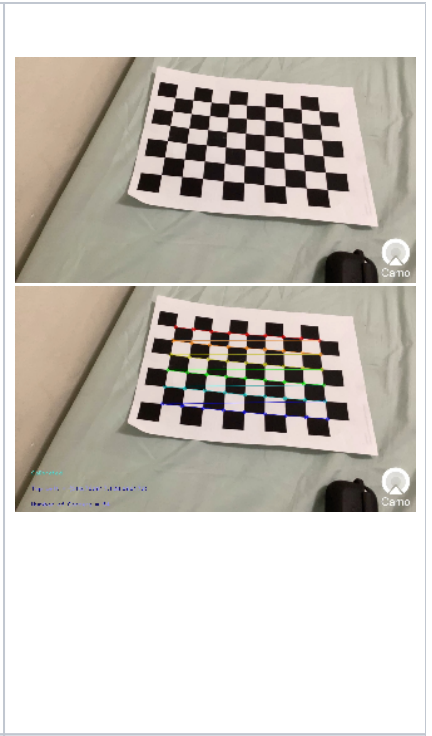
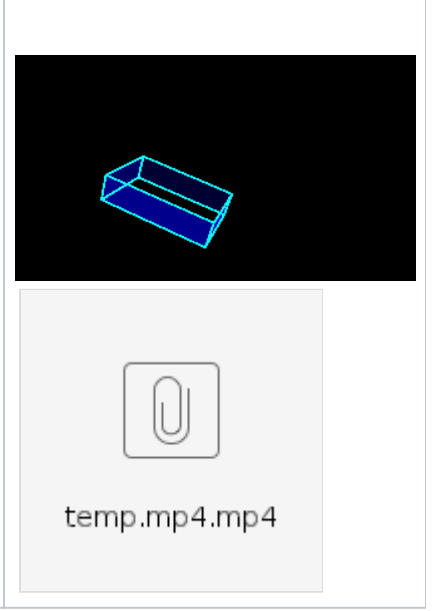
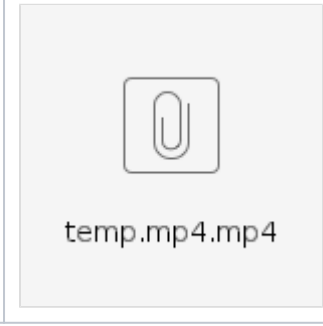
LaptopCamera,
948.03411865,948.03411865,638.09979248
,362.46972656,

-0.20185092,1.93842673,-0.00228692,
-0.00219352,-5.58871317

IPad,
1239.59362793,1239.59362793,988.383728
03,391.06466675,

-0.40376547,1.08666122,
-0.02504740,0.04703028,-1.03776157

Task 4: Calculate Current Position of the Camera	<p>When the user enters 'C' it reads the camera calibration from the CSV file based on the camera used.</p> <p>The rotation and translation are updated in the stream in real-time based on the position of the chessboard.</p>	
Task 5: Project Outside Corners or 3D Axes	<p>Projecting the outside corner with 0 as the z dimension value and then drawing the 3D axes.</p> <p>Pink circles are the projected outside corners and then the lines yellow, blue, green are the axes.</p>	
Task 6: Create a Virtual Object	<p>Created a cuboid, trying to fit in a cuboid of the size of the chessboard pattern.</p> <p>Added color to the sides of the cuboid, in the extensions showing this part of the video as well.</p>	 
Task 7: Detect Robust Features	<ul style="list-style-type: none"> • blockSize - It is the size of the neighborhood considered for corner detection. • ksize - Aperture parameter of the Sobel derivative used. • k - Harris detector free parameter in the equation. <p>The ksize parameter determines the size of the Sobel kernel (3x3, 5x5, etc..). As the size increases, more pixels are part of each convolution process and the edges will get more blurry</p> <p>The k parameter lets you influence in this step, trading off precision and recall. So with a bigger k, you will get fewer false corners but you will also miss more real corners (high precision), with a smaller k you will get a lot more corners, so you will miss fewer true corners, but get a lot of false ones (high recall).</p>	 
Extension 1: Get your system working with multiple targets in the scene.	<p>The system detects an asymmetrical grid of 4*11 along with a 9*6 chessboard.</p>	

<p>Extension 2: Test out several different cameras and compare the calibrations and quality of the results.</p>	<p>When the user clicks 'c', after collecting 5 image frames at least. It takes the image coordinates and the real-world coordinates and calculates the camera matrix, distance coefficients, and projection error.</p> <p>Focal Length: 1239.60 same for x and y</p> <p>Error Estimate: 1.3222</p> <p>C_x: 638.1</p> <p>C_y: 362.47</p> <p>Mat Camera Matrix After calibration</p> <table border="1"> <tr> <td>1239.60</td><td>0</td><td>988.3837</td></tr> <tr> <td>0</td><td>1239.60</td><td>391.06466</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> </table> <p>Mat Distortion Coefficient After calibration -0.40376547,1.08666122,-0.02504740,0.04703028,-1.03776157</p> <p>RMS error reported by calibrate camera: 1.3222</p> <p>The Error Estimate reported by the Ipad is comparatively more wrt the laptop camera.</p>	1239.60	0	988.3837	0	1239.60	391.06466	0	0	1	
1239.60	0	988.3837									
0	1239.60	391.06466									
0	0	1									
<p>Extension 3: Enable your system to use static images or pre-captured video sequences with targets and demonstrate inserting virtual objects into the scenes.</p>	<p>When the user clicks the button 'i', it opens the image and detects and draws a cuboid box if the camera calibration is done.</p>										
<p>Extension 4: Remove the target and only show the virtual object</p>	<p>When the user clicks button 'r' it changes the chessboard pattern to black between all the corners, so that the target does not look like a target anymore.</p> <p>This can be done just with the points inside the detected chess grid as well, but this looked better so did this instead.</p>	 <div data-bbox="1073 1619 1393 1942">  <p>temp.mp4.mp4</p> </div>									

Acknowledgment

- OpenCV documentation
- <https://stackoverflow.com/questions/54720646/what-does-ksize-and-k-mean-in-cornerharris>
- <https://anothertechs.com/programming/cpp/opencv-corner-harris-cpp/>
- <https://datahacker.rs/opencv-harris-corner-detector-part2/>
- <https://www.geeksforgeeks.org/draw-a-filled-polygon-using-the-opencv-function-fillpoly/>
- <https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/>

Reflection

The project was pretty much straightforward for the initial part, where it is mainly understanding the concept and calling the respective functions with proper values. It starts getting interesting when drawing the Axes and the 3D object, where I have applied little creativity to make the lines look like a cuboid box with play coloring. As part of the extension, the hiding target and making a video of the movements was fun. Overall it was fun to work on the project and made me intrigued by AR in computer vision. Would work on AR in my final project.