

CS 5330 - Pattern Recognition and Computer Vision

Project 4 : Calibration and Augmented Reality

Group Members : Shirish Kishore Kumar (002980081), Soorya Vijayaragavan (002921909)

Introduction:

This project is mainly focussed on developing a program that can identify a target and create virtual objects in relation to the identified target. Camera calibration techniques are used to accurately determine the position and motion of the target and camera, which are then used to place the virtual objects in the scene. The methodology used for camera calibration, target detection, and virtual object placement, along with its implementation and testing are analyzed and understood.

Task 1: Detect and Extract Chessboard corners

In this part the system was able to detect the chess board and the corners in it were detected. For this process to be done, the opencv functions, `findChessboardCorners`, `cornerSubPix`, `drawChessboardCorners` were used. Finally the number of corners detected and the first corner values were printed.

Task 2: Select Calibration Images

In this task, when 's' is pressed the program saves the most recent calibration image and it also saves chessboard corners in a corner set. To build a point set the size of a checkerboard square in mm is measured and used.

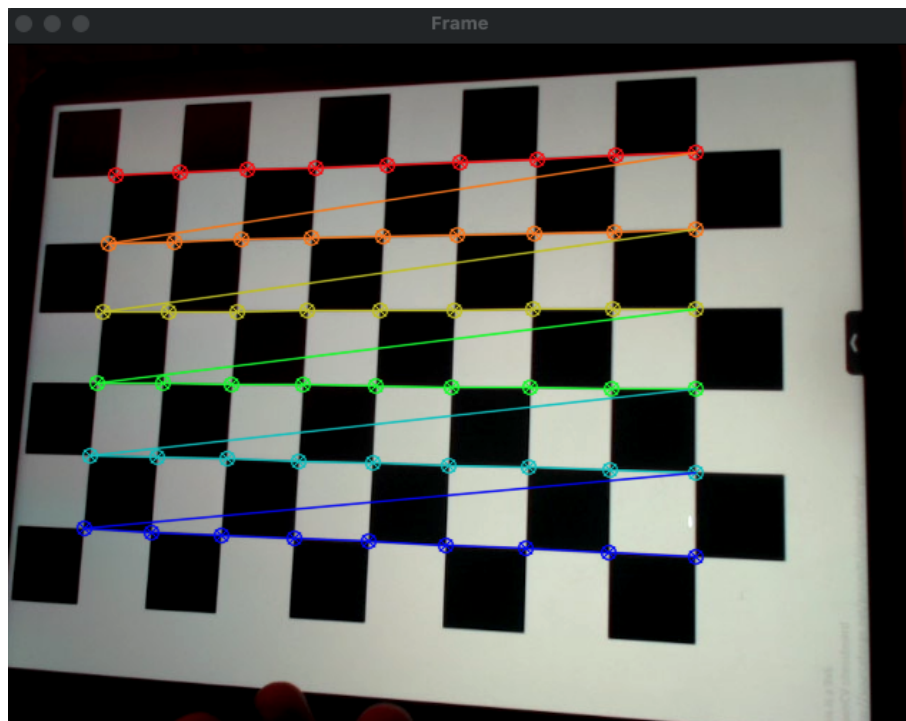


Fig.1 Calibration Image

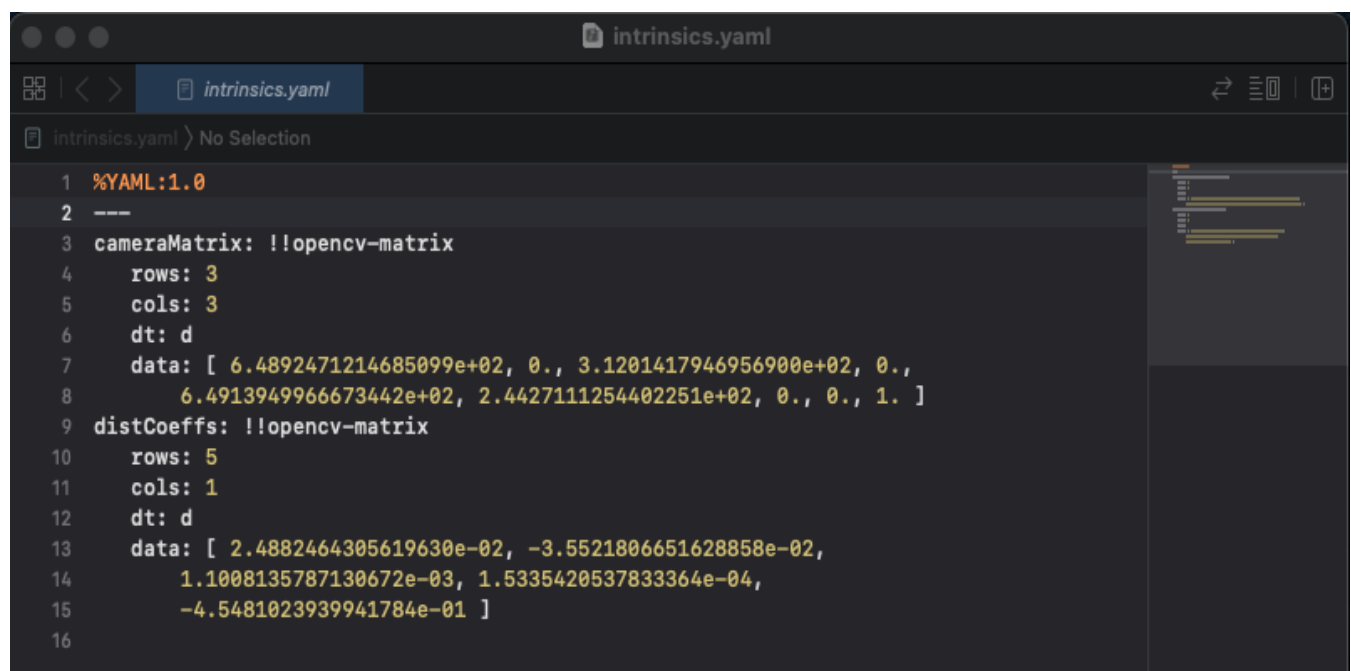
Task 3: Calibrate the Camera

In this task, "c" is pressed to start the camera calibration process but for that to be done it needs at least five corner data, so the user must have selected enough calibration frames i.e, atleast 5 or more frames. If the frames are sufficient, the calibration process is done, and the results i.e, camera matrix, distortion coefficient and the reprojection error values are printed. And the intrinsic parameters such as camera matrix and distortion coefficient values are stored in an YAML. Which can be used for future tasks.

```
Camera matrix:
[641.1899267476385, 0, 331.1990830925557;
 0, 635.4763473746539, 233.1224958676501;
 0, 0, 1]
RMS Reprojection error: 0.166537
```

Fig.2 Camera matrix and reprojection error being printed

Reprojection error calculated = **0.166537**

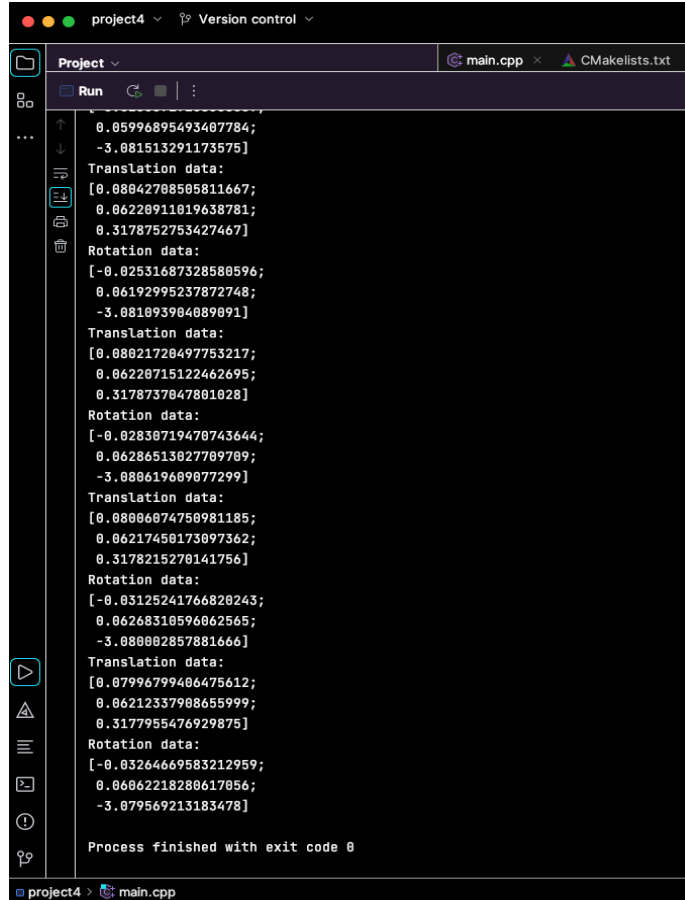


```
1 %YAML:1.0
2 ---
3 cameraMatrix: !!opencv-matrix
4   rows: 3
5   cols: 3
6   dt: d
7   data: [ 6.4892471214685099e+02, 0., 3.1201417946956900e+02, 0.,
8           6.4913949966673442e+02, 2.4427111254402251e+02, 0., 0., 1. ]
9 distCoeffs: !!opencv-matrix
10  rows: 5
11  cols: 1
12  dt: d
13  data: [ 2.4882464305619630e-02, -3.5521806651628858e-02,
14          1.1008135787130672e-03, 1.5335420537833364e-04,
15          -4.5481023939941784e-01 ]
16
```

Fig.3 Camera matrix and Distortion coefficient values saved in a XML file

Task 4: Calculate Current Position of the Camera

In this task, when "d" is pressed the intrinsic parameters stored in the YAML file in the previous task is read. And then when the chessboard is detected it stores the corner points and the solvePNP function is used to calculate the pose of the chessboard, i.e, it calculates the rotation and translation data by which the pose of the board is determined. Below is the example image of the rotation and translation values being calculated and printed for a live video stream.



```
project4 Version control
Project main.cpp CMakeLists.txt
Run
0.05996895493407784;
-3.081513291173575]
Translation data:
[0.08042708505811667;
0.06220911019638781;
0.3178752753427467]
Rotation data:
[-0.02531687328500596;
0.06192995237872748;
-3.081093904089091]
Translation data:
[0.08021720497753217;
0.06220715122462695;
0.3178737047801028]
Rotation data:
[-0.02830719470743644;
0.06286513027709709;
-3.080619609077299]
Translation data:
[0.08006074750981185;
0.06217450173097362;
0.3178215270141756]
Rotation data:
[-0.03125241766820243;
0.06268310596062565;
-3.08002857881666]
Translation data:
[0.07996799406475612;
0.06212337908655999;
0.3177955476929875]
Rotation data:
[-0.03264669583212959;
0.06062218280617056;
-3.079569213183478]
Process finished with exit code 0
project4 main.cpp
```

Fig. 4 Rotation and Translation data being printed for a live video stream

Task 5: Project Outside Corners or 3D axes

In this task, "e" is pressed to project the outside 4 corner points on the checkerboard, this done by using the projectpoint function. In the below image the four corners of the chess board are projected in light green color.

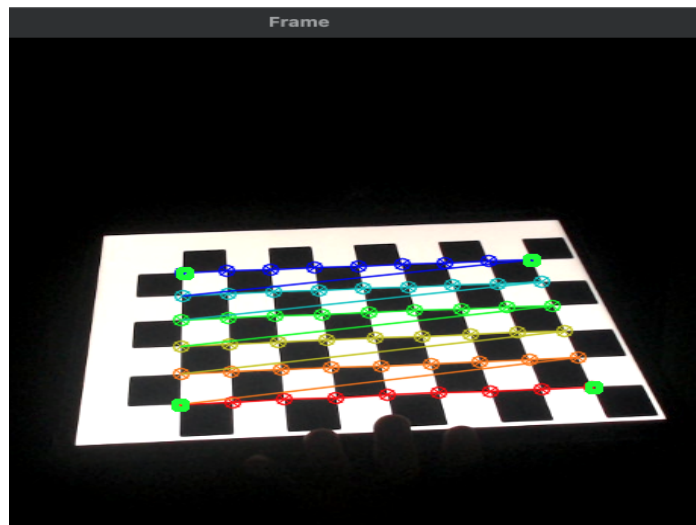


Fig. 5 Four corners projected in green color

Task 6: Create a Virtual Object

In this task, when “v” is pressed a virtual object is being displayed over the chessboard. Orientation of the virtual object remains the same, even when the chess boards orientation keeps changing in all directions.

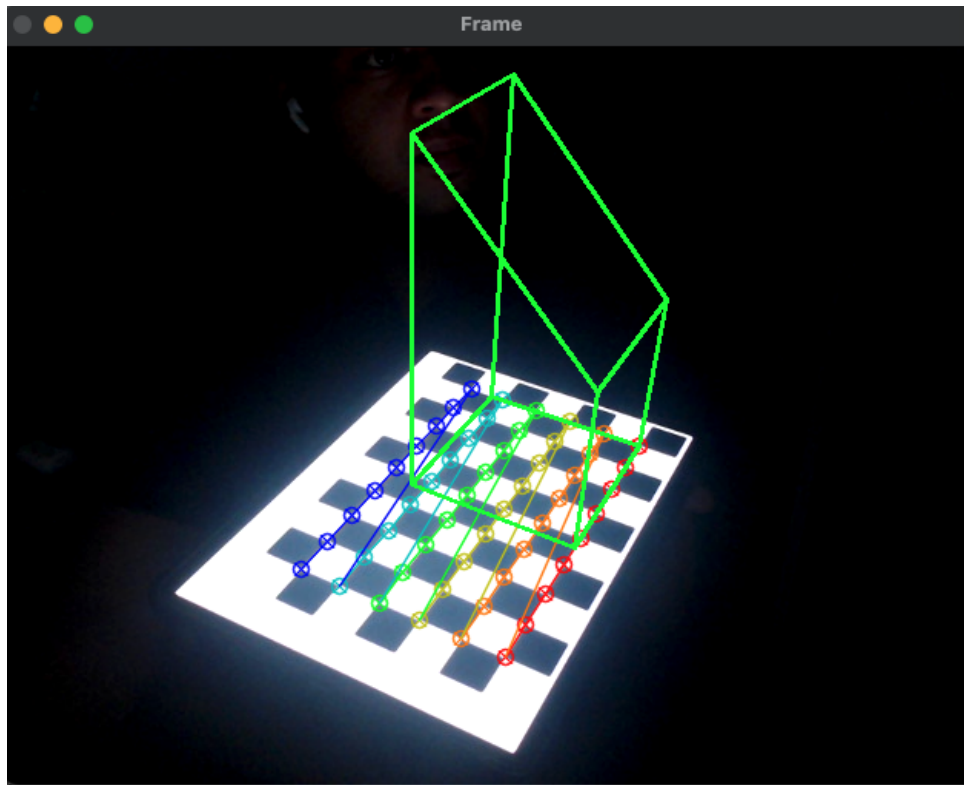


Fig.6 An asymmetrical object is displayed over the chessboard

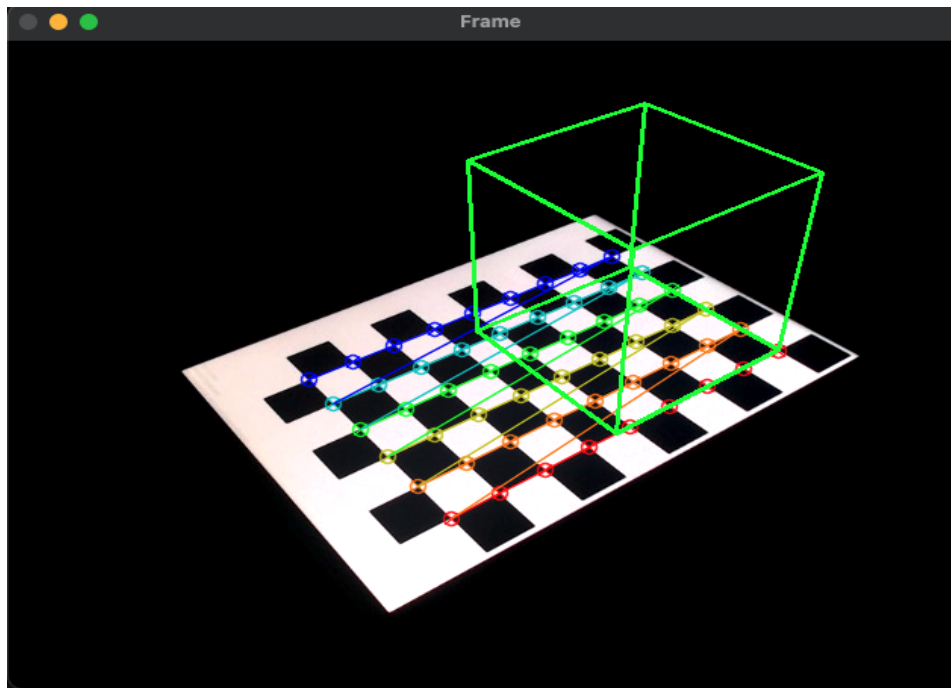


Fig. 7 A symmetrical cube is displayed over the chessboard

Task 7: Detect Robust Model

In this task, a new program is created to detect the features of the target in a live video stream. We have implemented the Harris corner detection and we have displayed the corner features in the video stream. Explanation regarding using feature points as basis for AR is given below.

Example 1:

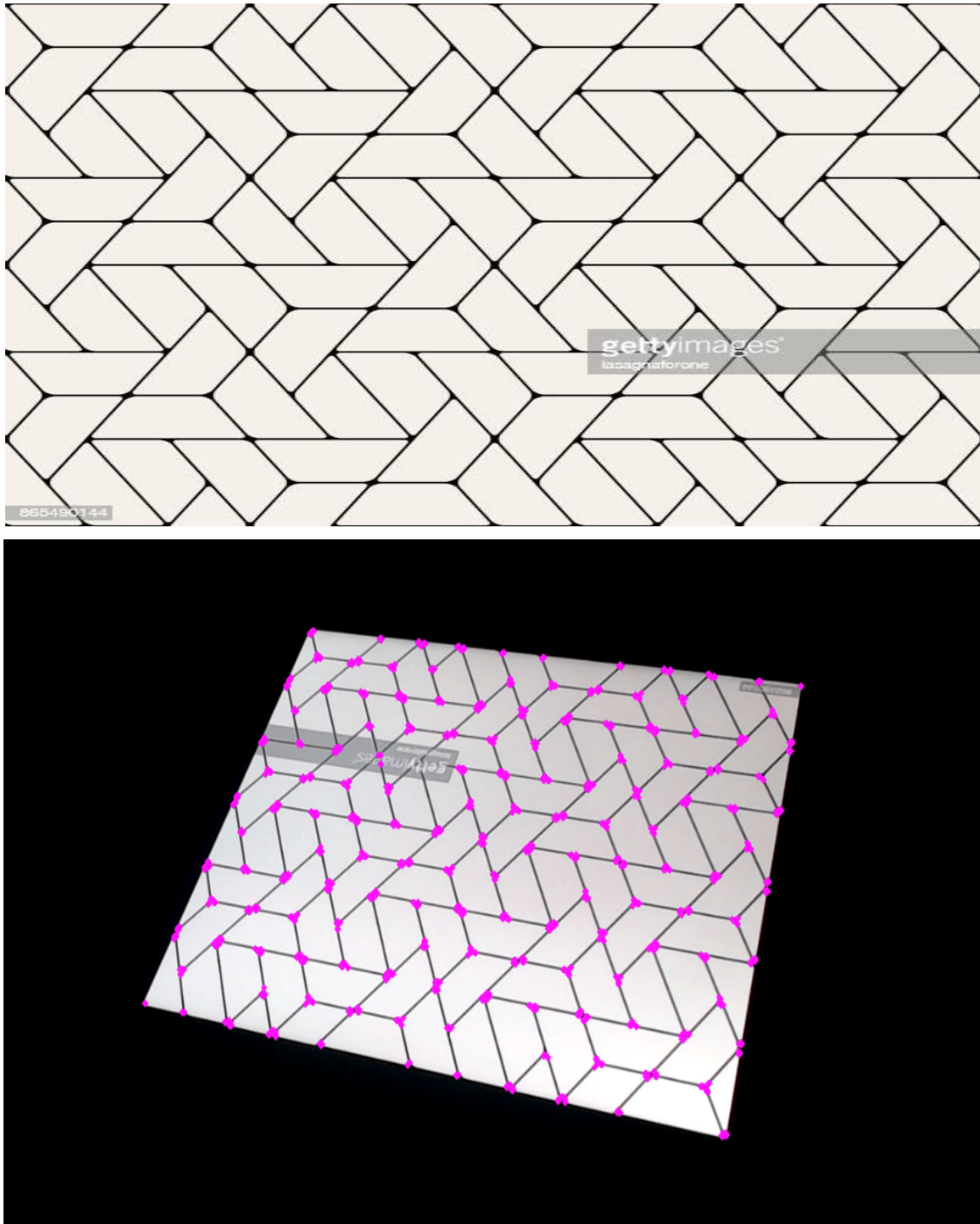


Fig. 8 Original image and the Harris corner feature displayed image

Example 2:

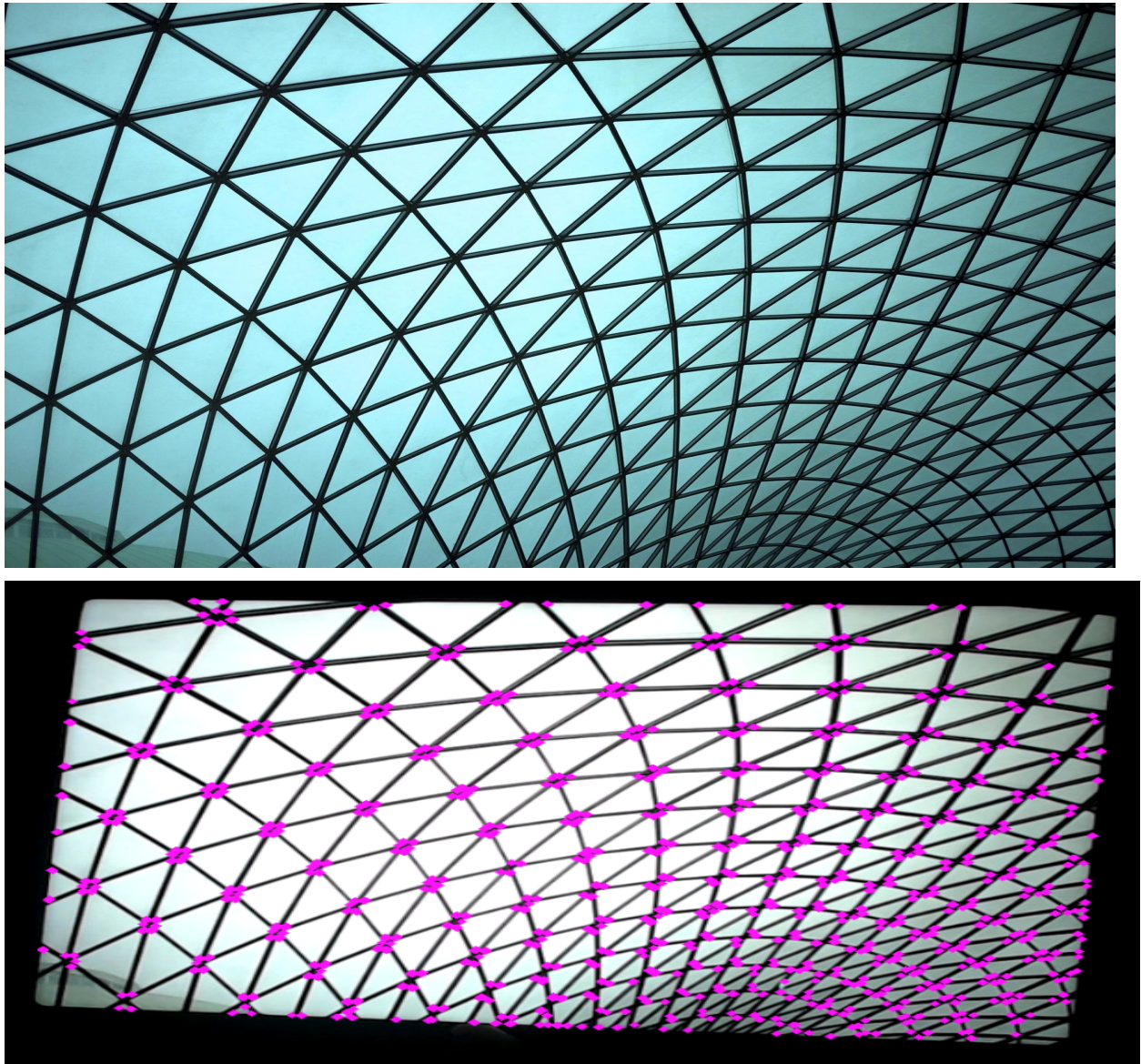


Fig. 9 Original image and the Harris corner feature displayed image

Question: How you might be able to use those feature points as the basis for putting augmented reality into the image?

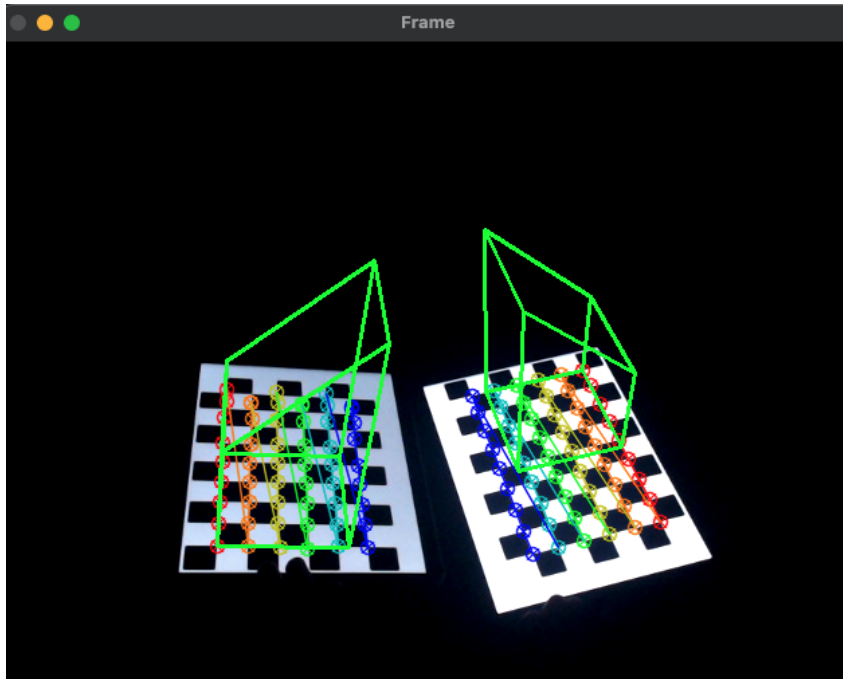
In order place the object on the world what we need to do is first generate a 3d map, in order to do that first we have to move the camera around the world and get multiple images for example $c_1, c_2, c_3, \dots, c_n$ then for each image we compute the harris features and we find the matching points in the image pairs from example c_1 and c_2 .

Then we estimate the transformation, fundamental matrix and the essential matrix from the Harris features and find the relative camera location and estimate the 3d geometry/map. Now using these 3d points we can place an object in the world.

Extension:

1. System Working with Multiple Targets in the Scene

In order to perform this we split the image into two halves and performed the same process for each halves of the image and placed the object if a checkerboard is found in the region.



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

Below are the calibration parameters of 2 different cameras and it's error estimates,

CAMERA 1: Logitech C925-e Webcam



```
Camera matrix:
[641.1899267476385, 0, 331.1990830925557;
 0, 635.4763473746539, 233.1224958676501;
 0, 0, 1]
RMS Reprojection error: 0.166537
```

Reprojection Error for Camera 1 = **0.166537**

```
1 %YAML:1.0
2 ---
3 cameraMatrix: !!opencv-matrix
4   rows: 3
5   cols: 3
6   dt: d
7   data: [ 6.4892471214685099e+02, 0., 3.1201417946956900e+02, 0.,
8         6.4913949966673442e+02, 2.4427111254402251e+02, 0., 0., 1. ]
9 distCoeffs: !!opencv-matrix
10  rows: 5
11  cols: 1
12  dt: d
13  data: [ 2.4882464305619630e-02, -3.5521806651628858e-02,
14        1.1008135787130672e-03, 1.5335420537833364e-04,
15        -4.5481023939941784e-01 ]
16
```

CAMERA 2: Logitech C922x Pro Stream Webcam



```
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Corners detected & saved
Camera Calibration Results
Camera matrix:
[833.8214443178936, 0, 248.9370337078972;
 0, 2281.682422009005, 341.9387439160873;
 0, 0, 1]
RMS Reprojection error: 0.184359
```

Reprojection Error in Camera 2 = **0.184359**

```
1 %YAML:1.0
2 ---
3 cameraMatrix: !!opencv-matrix
4   rows: 3
5   cols: 3
6   dt: d
7   data: [ 8.3382144431789357e+02, 0., 2.4893703370789723e+02, 0.,
8         2.2816824220090048e+03, 3.4193874391608733e+02, 0., 0., 1. ]
9 distortion_coefficients: !!opencv-matrix
10  rows: 5
11  cols: 1
12  dt: d
13  data: [ -1.7156181029487966e-01, 1.2891792329814697e+00,
14        -1.806348646442052e-03, -1.7061341001861432e-02,
15        -1.9732895189778862e-01 ]
16
```


3. Integrate OpenCV with OpenGL to render shaded objects

We tried to implement this but we kept facing issues and could not find the bug, as of now the video freezes if it detects the checkerboard.

We have uploaded the code as well, even though it is not working as expected. Due to time constraints we are leaving this as such and in future we will fix the bug.

Conclusion:

The project helped in deeply understanding the camera calibration techniques, intrinsic parameters of the camera and how it determines the error estimate. Working with augmented reality by learning it by practically implementing things, like displaying a virtual object was interesting. Trying out to integrate openCV with openGL was great learning.

References:

- [1]https://docs.opencv.org/4.x/d2/de8/group_core_array.html#ga0a1acc042551eb596589d38f7bcac738
- [2]https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html
- [3]<https://stackoverflow.com/questions/16641209/reading-and-storing-values-from-obj-files-using-c>
- [4]<https://www.geeksforgeeks.org/split-string-by-space-into-vector-in-cpp-stl/>
- [5]<https://stackoverflow.com/questions/61890766/how-can-i-split-a-vector-string-and-access-what-i-need-in-it>
- [6]<https://stackoverflow.com/questions/10962232/wavefront-obj-files-what-does-the-f-command-do>
- [7]https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector.html
- [8]<https://www.tutorialspoint.com/detecting-corners-using-harris-corner-detector-in-python-opencv>
- [9]https://docs.opencv.org/3.4/d2/de8/group_core_array.html#gab473bf2eb6d14ff97e89b355dac20707
- [10]https://docs.opencv.org/3.4/d9/d0c/group_calib3d.html#ga1019495a2c8d1743ed5cc23fa0daff8c
- [11]https://docs.opencv.org/3.4/d9/d0c/group_calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d
- [12]https://docs.opencv.org/3.4/d9/d0c/group_calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a
- [13]https://docs.opencv.org/4.x/dd/d1a/group_imgproc_feature.html#ga354e0d7c86d0d9da75de9b9701a9a87e
- [14]https://docs.opencv.org/3.4/d9/d0c/group_calib3d.html#ga6a10b0bb120c4907e5eabbcd22319022