CS5330: Pattern Recognition and Computer Vision - Project 4 Report

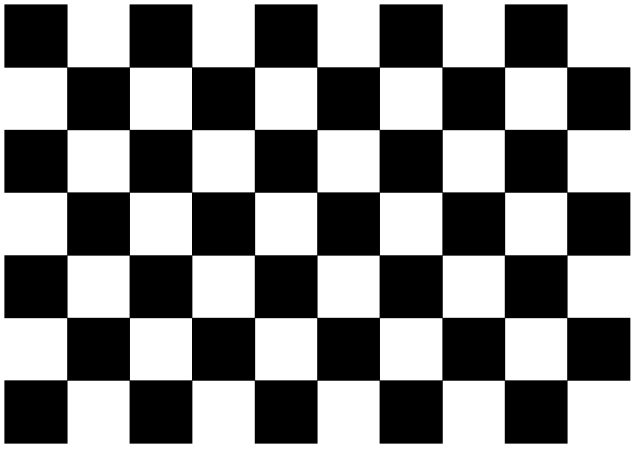
Project 4: Calibration and Augmented Reality

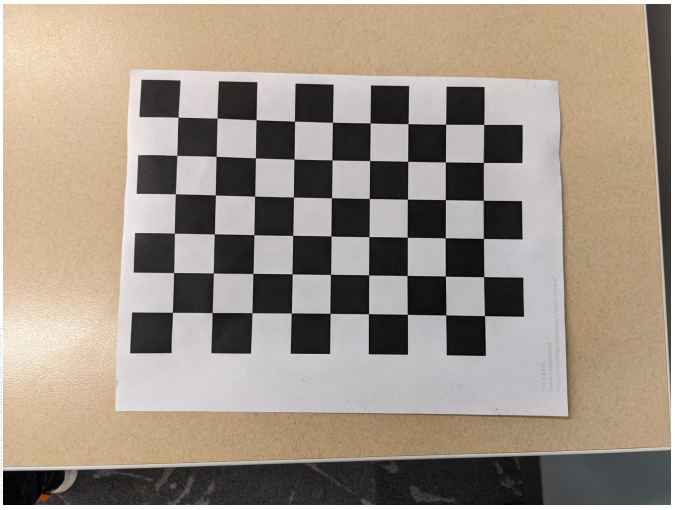
Brief Description of the Project:

This project intends to introduce us to the world of Augmented Reality. It briefly introduces us to the workings and processes behind calibrating a camera and its necessity before it can be utilised to project virtual objects into the real world. To project a virtual object, we need an anchor in the real world to project that object w.r.t. the anchor. It can be done by identifying certain robust features in an environment and by tracking the change in their position, we can change the position of the object. The deliverable of this project is to calibrate the camera, detect a target and then place a virtual object in the scene relative to the target that moves and orients itself correctly given the motion of the camera or target. Checkerboard is used as the intended target.

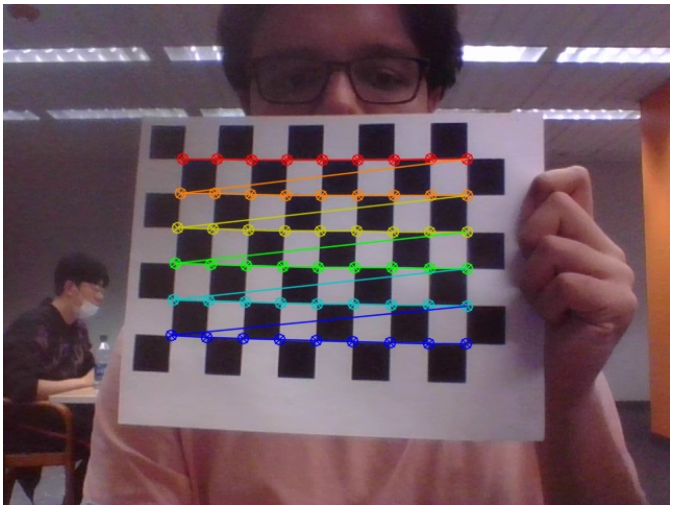
Detect and Extract Chessboard Corners:

This task accomplishes detecting a target (checkerboard) in our case and extracting the desired target corners. The following image is used as a target. It is printed on A4 paper.





Calibration Images:



The above image shows one of the images that was used as one of the calibration images for getting the calibration data. On the keypress 's', the program stores the vector of corners found by findChessboardCorners into a corner_list. It also saves the point_set that specifies the 3D position of the corners in world coordinates. It also saves the frame itself as an image for reference. This is all conditional on the fact that the target is actually detected in the frame. Otherwise, the user gets a prompt to clearly display the target in front of the camera.

Calibrate the Camera:

If the user has selected at least 5 frames which are the minimum requirement for calibrating the camera, on the keypress 'c', the program reads the data, and inputs it to the cv::calibrateCamera function to generate the calibration data. The following data is generated with the help of 5 images.

```
Expected size: 640 480
1 0 320
0 1 240
0 0 1
Frame 1 successfully saved for Calibration.
Frame 2 successfully saved for Calibration.
Frame 3 successfully saved for Calibration.
Frame 4 successfully saved for Calibration.
Frame 5 successfully saved for Calibration.
Calibrating the Camera...
Printing the Camera Matrix:
556.27 0 356.945
0 556.27 217.18
0 0 1
Printing the Distortion Matrix:
0.107693 -0.564851 -0.0272235 -0.00867842 0
Error: 0.302487
```

The initial matrix is the initialisation of the camera matrix. Five frames are then saved for calibration along with their corresponding vectors and then passed into the calibration function. The screenshot displays the resultant Camera Matrix, Distortion Matrix and the Reprojection Error.

The reprojection error comes out to be 0.3 approximately which is an acceptable value. It could have been finer but it's this high probably because the laptop is old and the webcam might have been compromised in quality because of scratches or age.

Once the data is generated, the calibration data can be saved into a file on the keypress 'w'.

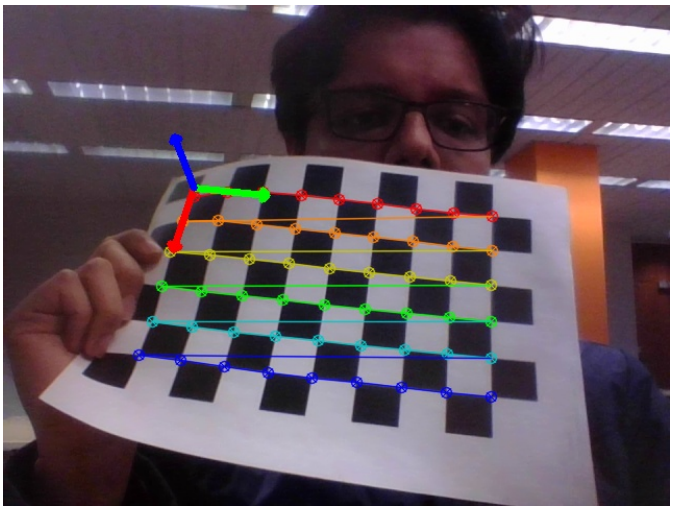
Calculate Current Position of the Camera:

This program reads the camera calibration parameters written in the previous file (camera matrix and distortion matrix) and inputs the data into solvePNP to get the board's pose.

I printed out the rotation and translation data in real-time as I was testing it.

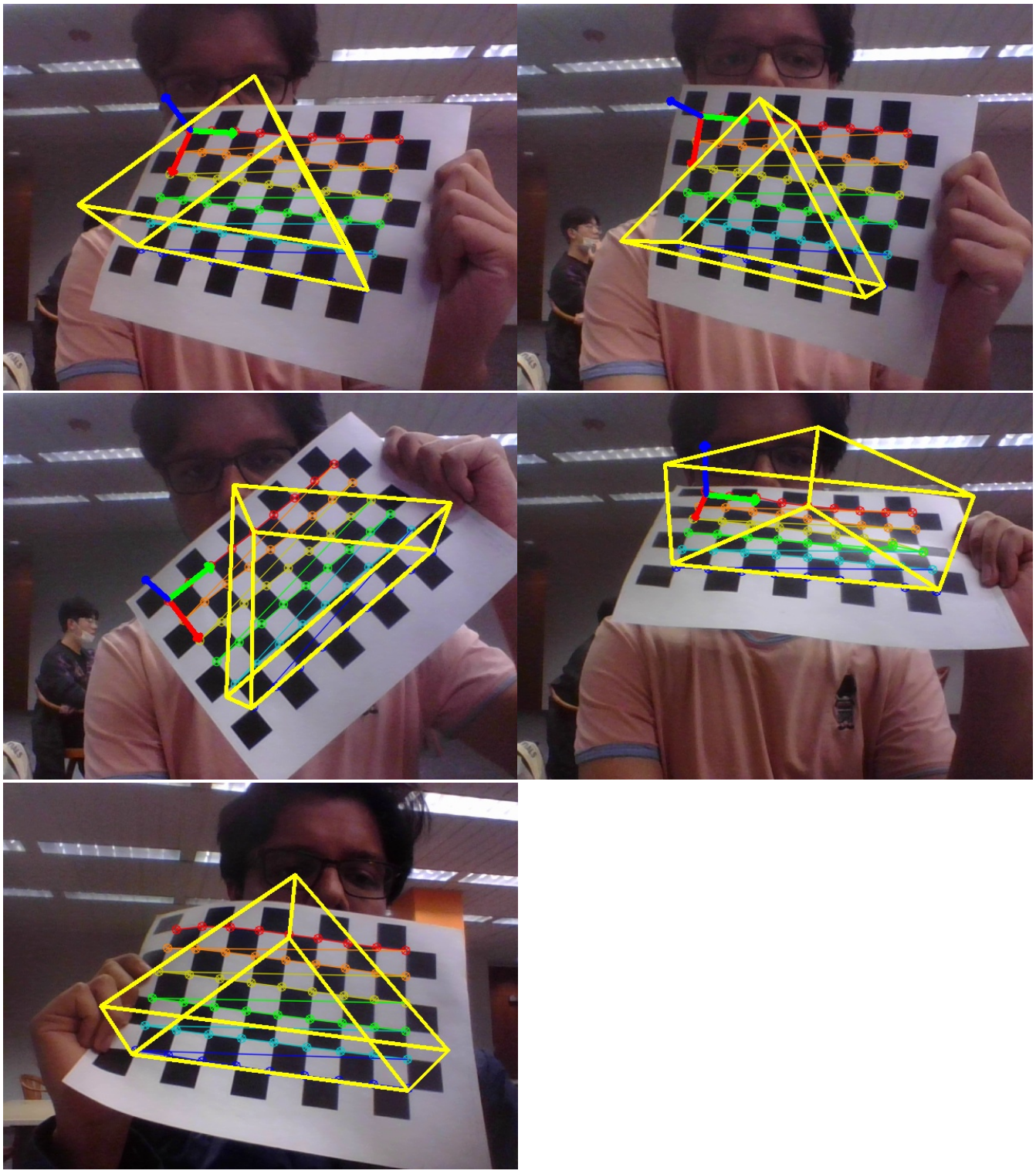
Project Outside Corners or 3D Axes:

This task is to project 3D axes on the board attached to the origin using the projectPoints function. It is anchored to the origin of the chessboard and moves with the target.



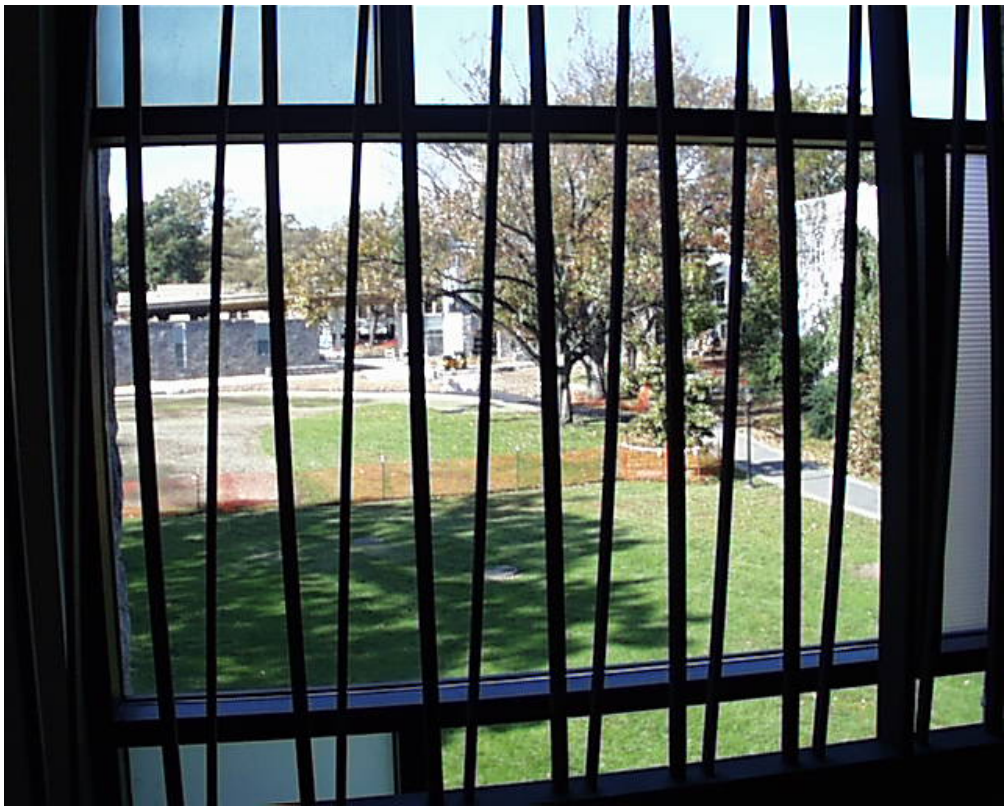
I have taken the X-axis to be the red line, Y-axis to be the green line and Z-axis to be the blue line.

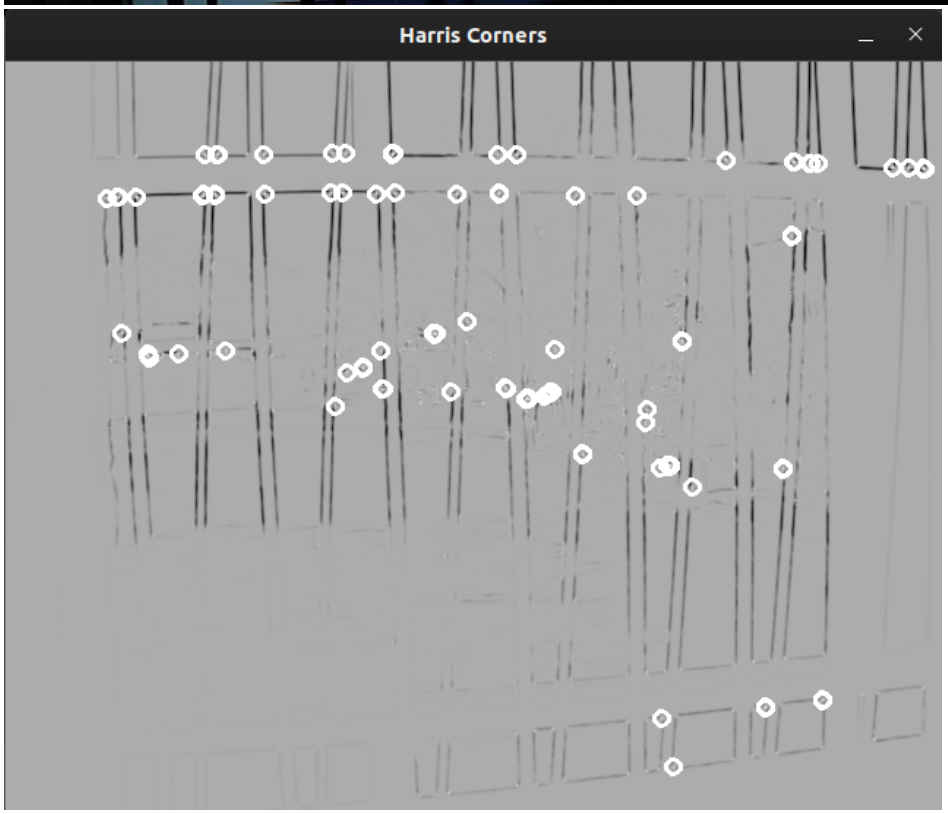
Create a Virtual Object:



I have constructed a triangular prism as a virtual object. These are the screenshots I took and you can also see the system in action from the link provided in the readme.txt.

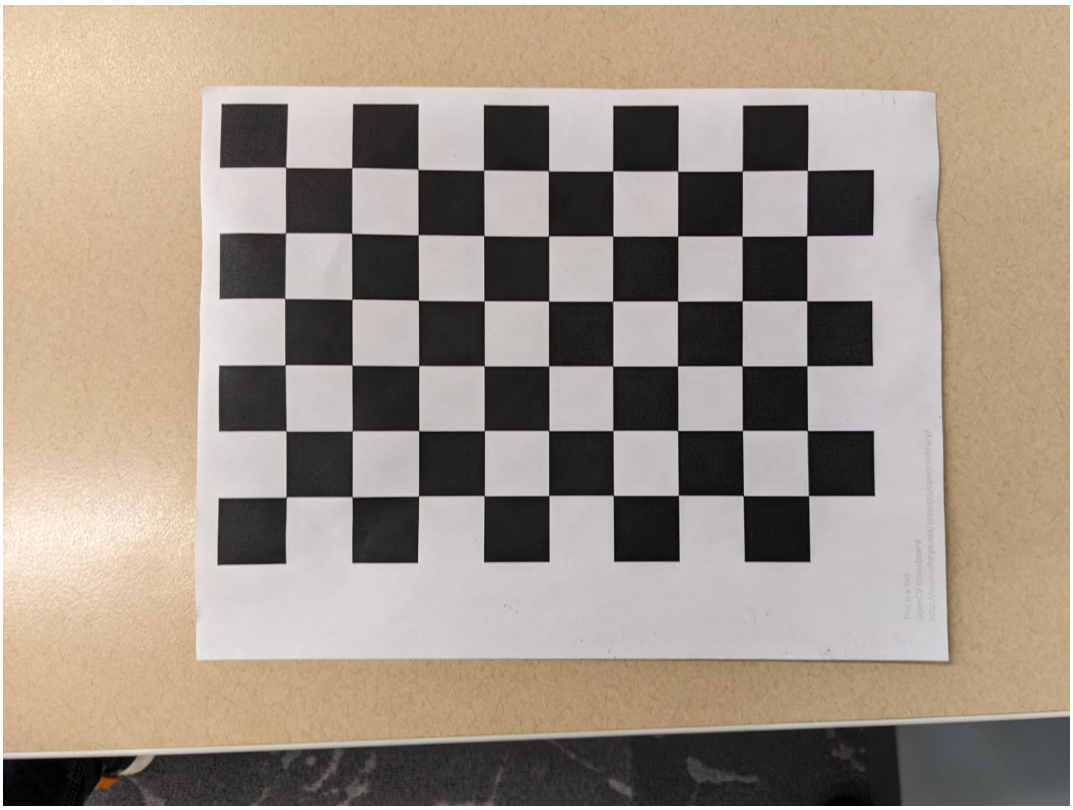
Detect Robust Features:

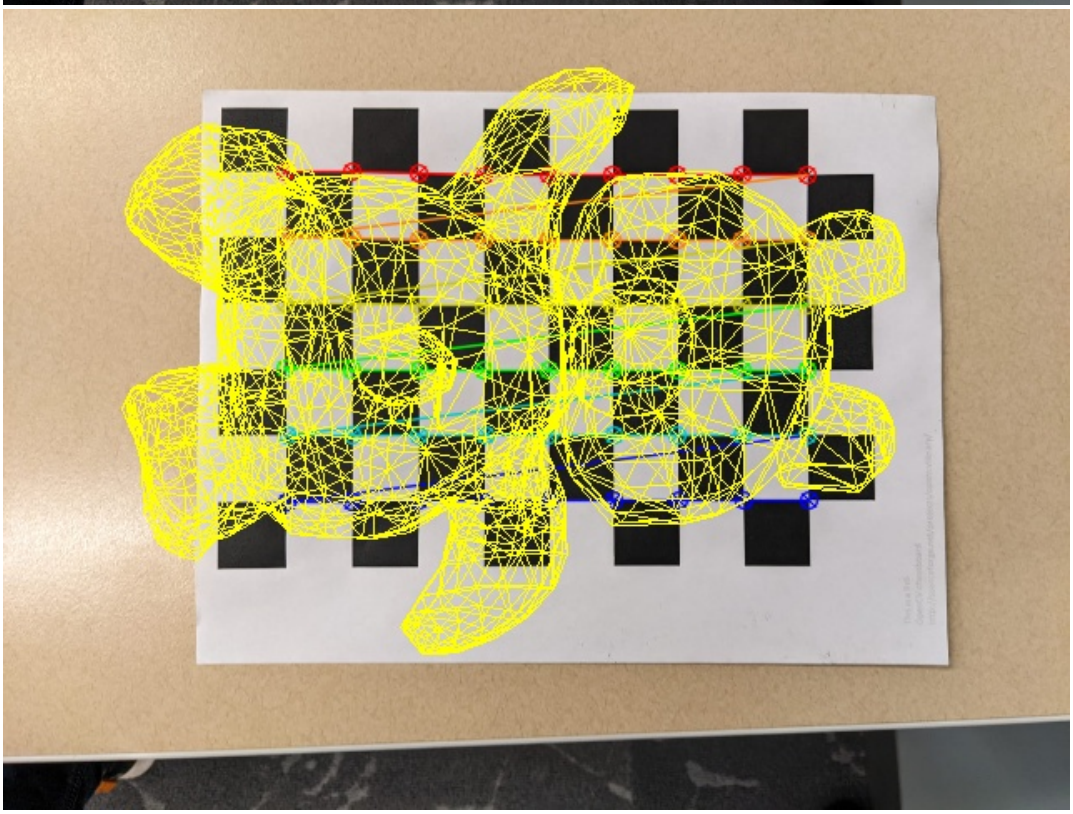




Picked Harris Corners as a feature and wrote the program that showed where the features were. These features are distinct to the target. If I were to select these feature points as the target, I could specify a point_set as per their distribution and project a virtual object into the world anchoring it to these specific set of feature points. Basically, the feature points detected from Harris Corners can be understood as analogous to the chessboard corners extracted from the target image.

Extension 1: Adding a virtual object in static images or pre-captured video sequences:





This program takes an input image and determines if it can locate a target is present in the frame. If not, it prompts the user that no pattern is found. If detected, the user is prompted to select what object would they like to insert the options being the 3D axes at the origin, the triangular prism, the diamond, the space shuttle, the skyscraper or the teddybear (inserted in this scenario). The frame is then saved to the directory of the program.

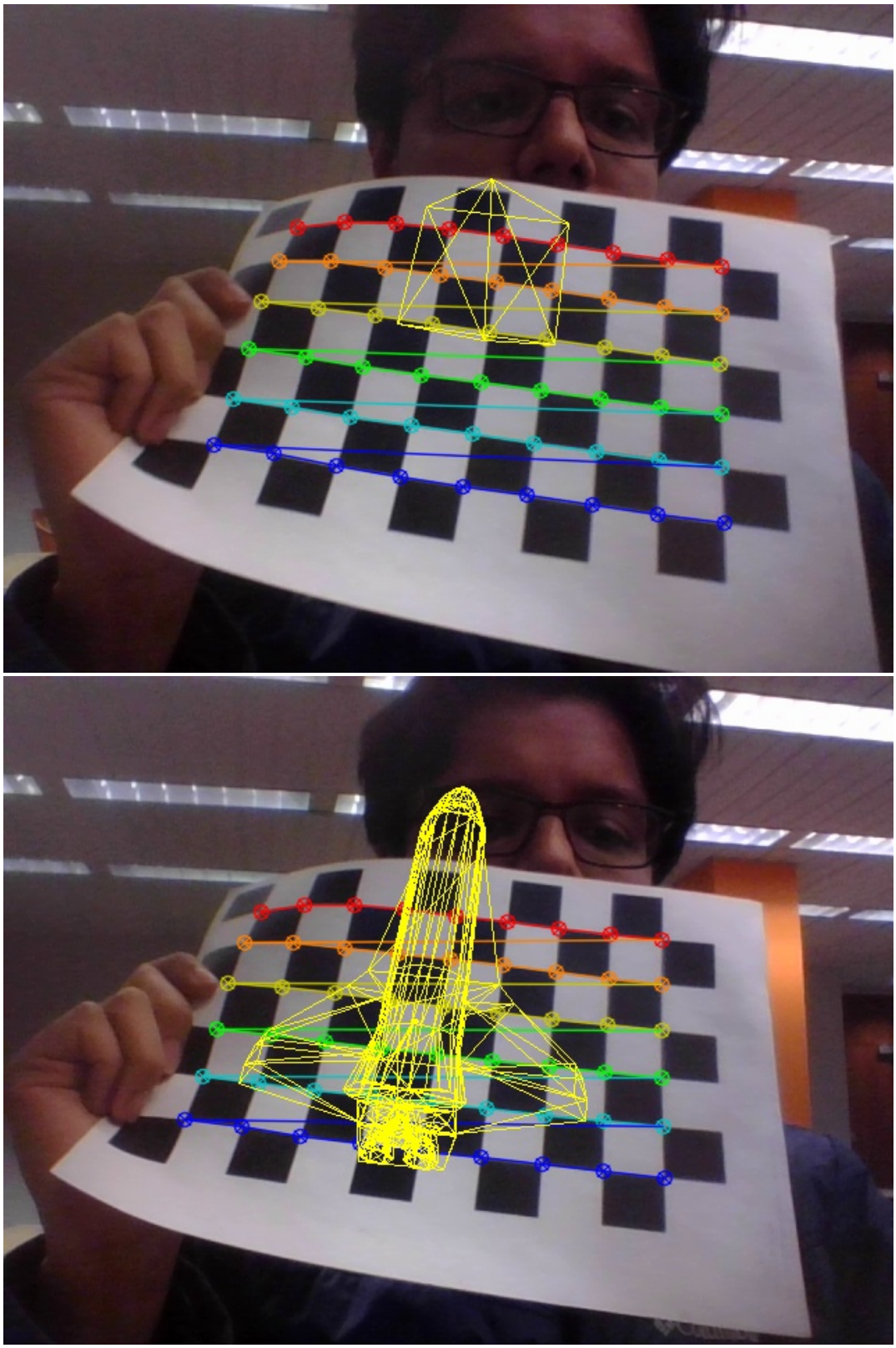
Extension 2: Testing out several different cameras and comparing the calibrations and the quality of results:

The camera of the mobile phone was tested apart from the webcam that is inbuilt into the laptop. The laptop is nearly 5 years old and so the quality of the webcam and the lens has been compromised and it gave an approximate error of 0.3. The approximation of the projection error from the mobile phone camera was found to be in the order of 0.1 along with significantly less distortion coefficients though the values weren't zero implying that some distortion is also found in the mobile phone camera which needs to be corrected.

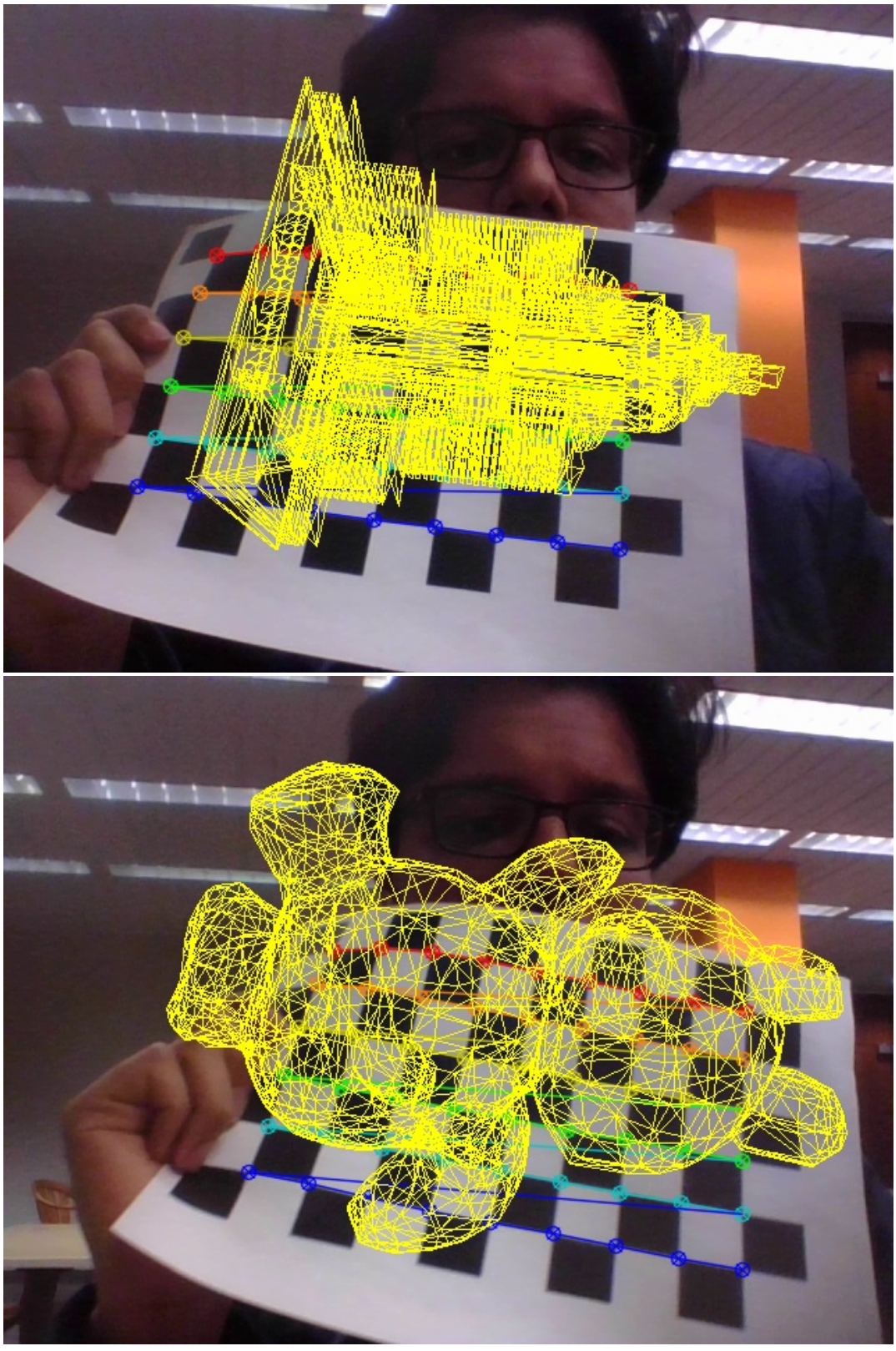
Extension 3: Creating a custom script to process the data in the object file and project them on the checkerboard after processing the read data:

I downloaded 4 open-source object files containing vertex and face information. The first task was to understand the data in the object files and gain an insight into how the data is stored and how can it be recreated. That is to say, the process with which the object can be recreated with the data from the object file. Also, a solid object isn't required and we need to form the object in a manner fit to project it onto the checkerboard. I have implemented my own C++ functions to parse the data from the object file and create the point_set which can then be projected onto the target.

The function reads the vertices argument and stores the coordinates. Then the function parses through the face attributes in order to determine what vertices need to be joined to form that specific face and then the object. Below images show the objects projected in a live video stream.



The first one is the diamond and the second one is the space shuttle.



The first one is the skyscraper and the second one is the teddy bear.

The objects needed to be scaled and translated in order for them to be projected such that their centre coincides with the centre of the checkerboard and they are of a size that is comparable to that of the checkerboard.

Extension 4: GUI to integrate everything and switch between different objects:

I have provided the functionality of giving the user, the functionality of what object they want to display on the checkerboard. The user gets a prompt with various keypress relating to different objects and the user can choose what object to display and toggle from one to another while the video is streaming.

For the program to not break on a keypress or in general when the pattern is not found, care is taken to provide appropriate prompts to the user as a guidance manual so that they can easily get an intuitive idea of how the program works.

I also wanted to provide the user with the functionality of moving the projected image with the help of arrow keys, scaling the images up and down using the '+' and '-' keys which I will definitely be doing after I submit this project.

Extension 5: Dynamic calibration data gathering:

The program has the functionality of capturing and storing calibration data and outputting the calibration data and writing the calibration data to a file bound to distinct key presses. This gives the user the flexibility to save as many images as required and take more images if the reprojection error is not to the satisfaction of the user and he can gather more of the calibration data in order to improve upon the error.

Reflection:

This project introduced me to the world of Augmented Reality which seemed far-fetched before and now, seems accessible. It is a subject that has always interested me but it was all about finding the right path into the world of Augmented Reality which this project served very well. It seemed like a daunting task before but now I am comfortable enough to experiment a lot on this and would love to generate something that is truly remarkable. To be honest, I would have loved to do a lot of amazing extensions for this project but I wasn't able to get started on it until very late and hence, I, myself am not satisfied with the quality of work I have produced. I still feel I could have done a lot better and applied this to some interesting applications which I will surely implement irrespective of my submitting this project.

Acknowledgement:

1) OpenCV Documentation

2) https://people.sc.fsu.edu/~jburkardt/data/obj/obj.html

3) http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/

I would like to thank Professor Bruce Maxwell for this project wouldn't be possible without the knowledge gained from attending his lectures and overall guidance. I would also like to thank my classmate and friend Sumegha Singhania for various brainstorming sessions and the exchange of ideas.