# Assignment 3

## Puja Chaudhury

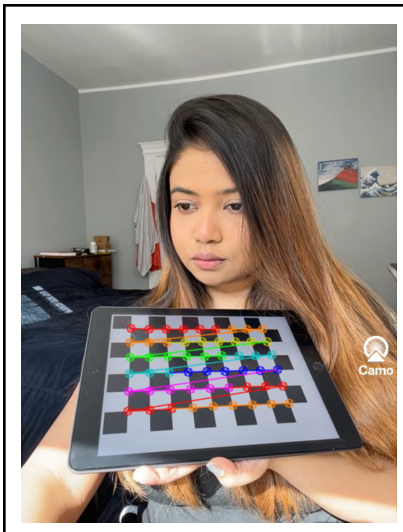
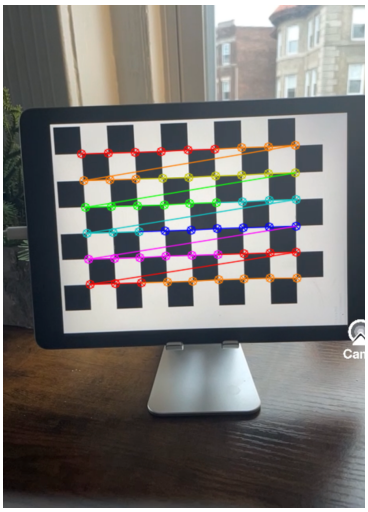
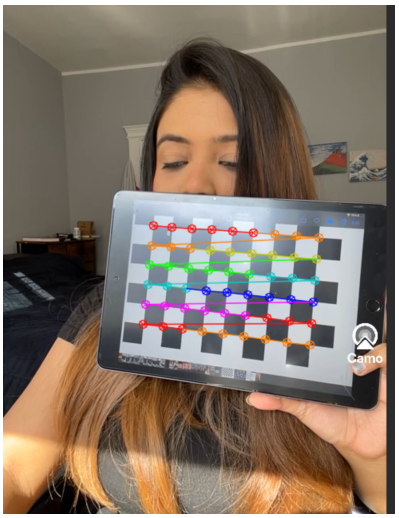
### *A short description of the overall project in your own words. (200 words or less)*

During the course of this project, I had the opportunity to acquire valuable skills in camera calibration by utilizing functions from the openCV library. I successfully applied the calibration parameters to identify target points within our scenes. Not only that, but I was also able to estimate our camera's position and orientation relative to the target scene.

This new understanding enabled me to place virtual objects within the scene in relation to the target. What made this particularly fascinating was that the virtual object could adapt its movements and orientation according to changes in the camera's position or the target's motion. Overall, this project has significantly expanded my knowledge and skills in camera calibration and openCV library functions.

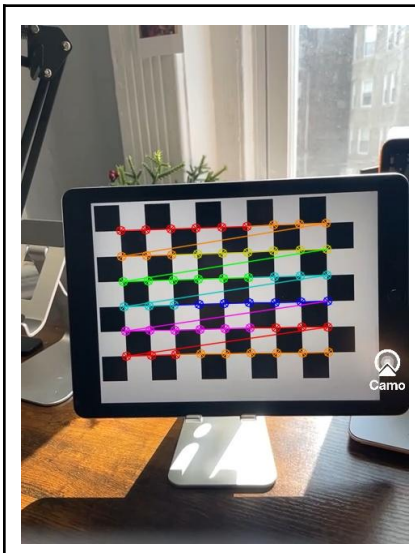
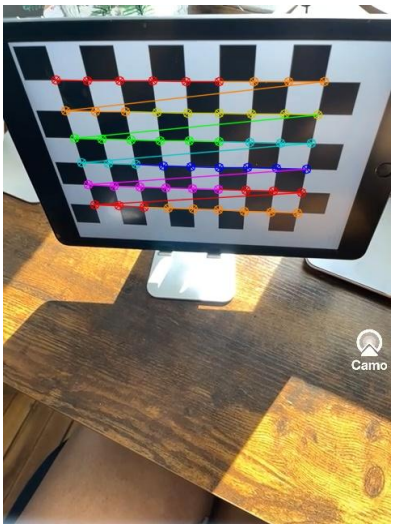
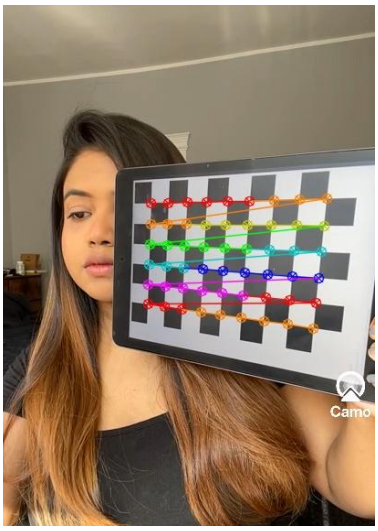
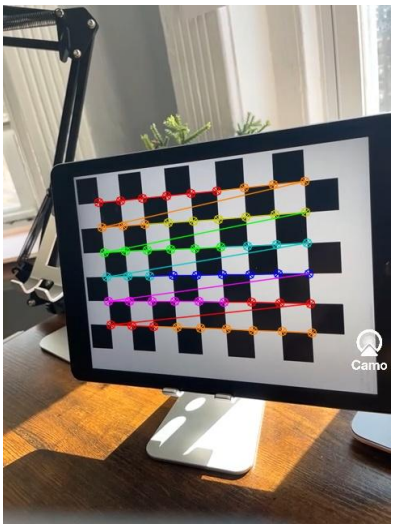
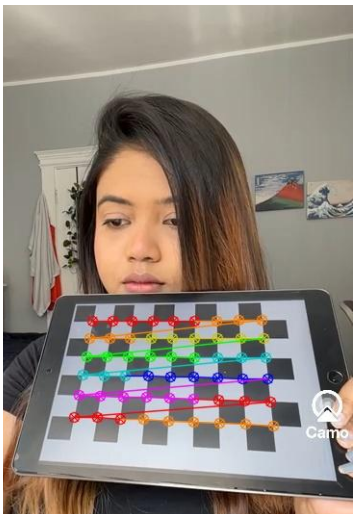
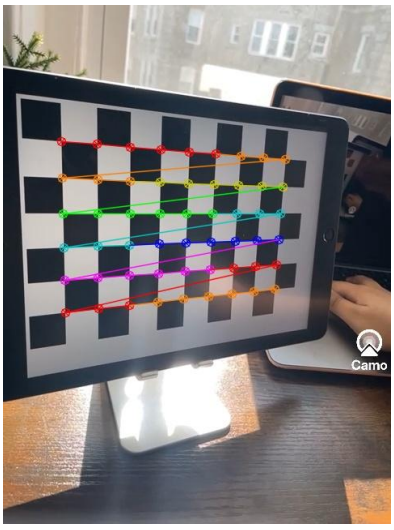
### *Any required images along with a short description of the meaning of the image.*

**Task 1: Detect and Extract Chessboard Corners**
The GetChessboardCorners function from opencv takes the input video frame and outputs the resulting frame with the corners drawn on it (outputFrame), as well as a vector of corner points (corners) and a boolean value indicating whether or not the corners were successfully detected (found). The cornersDrawn variable is a boolean that determines whether or not the output frame will have the corners drawn on it.

**Task 2:** **Select Calibration Images**

The current image frame can be saved for calibration purposes by pressing the 's' key. When the 's' key is pressed, a point set in 3D world coordinates and the corresponding pixel coordinates on the image frame are saved into vectors that will be used later for calibration. Each calibration is saved as a JPEG image, and the point set and corresponding corner set in the current frame are printed out. The point set consists of the coordinates of the corners in the 3D world, while the corner set represents the corresponding coordinates of the corners on the image frame.

### Task 3 - Calibrate the Camera

After collecting more than five calibration images, the calibration system updates the calibration each time the user adds a new calibration image. The system displays the current per pixel reprojection error and distortion coefficients after each calibration. The system will prompt the user to press the 'c' key if the two focal lengths have the same value, and the u0 and v0 values are close to the initial estimates of the center of the image. Pressing the 'c' key will save the current calibration statistics into a CSV file, which can be used later to calculate the position of the camera.

```
Calibrating with 6 frames...
Camera matrix before calibration:
[473.0986747129526, 0, 240.7717631549211;
 0, 473.0986747129526, 308.915390392403;
 0, 0, 1]
Calibrated camera matrix:
[475.4087966597929, 0, 240.6790926751339;
 0, 475.4087966597929, 308.8152106918061;
 0, 0, 1]
Re-projection error: 0.0898962
Distortion coefficients: [0.1668658274066525, -0.7411412427761697, 6.361252294863793e-05, 0.001114064954353976, 1.279900279966282]
```

### Task 4 - Calculate Current Position of the Camera

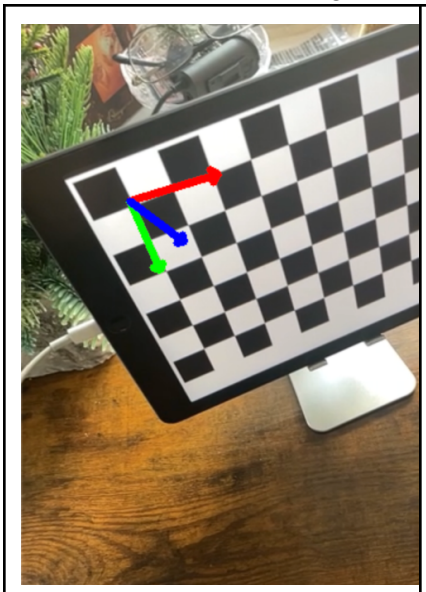
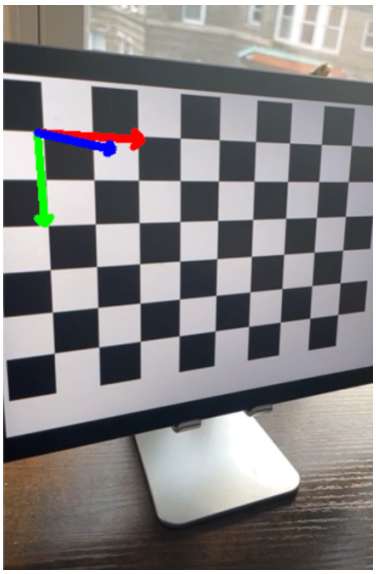
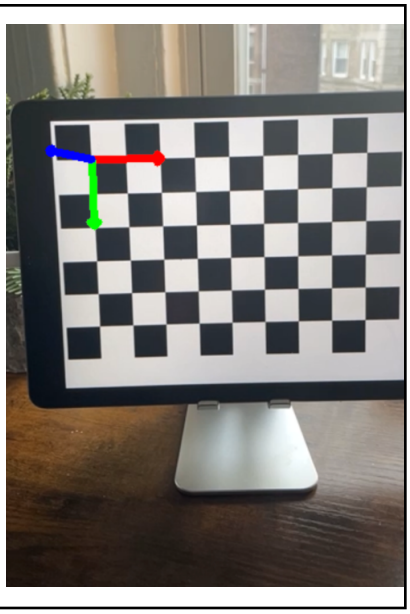
The code is used to retrieve calibration parameters from a CSV file and detect corners for each video frame. The detected corners are then used to calculate the target's pose (rotation and translation) relative to the camera using OpenCV's solvePnP function. Once the current camera position is successfully calculated, the system displays the rotation and translation data in real-time. An example of the output log containing the rotation and translation data is provided in the code.

```
pujachaudhury@Pujas-MacBook-Air augmented-reality-main 2 % ./main
Expected size: 1920 1080
Loading the saved calibration
Reading chessboard_intrinsics.csv...
...finished reading CSV file

retrieved calibrated camera matrix:
[982.350330078125, 0, 639.3306274414062;
 0, 982.350830078125, 360.5378112792969;
 0, 0, 1]
distortion coefficients: [0.1592, -1.1581, 0.0035000001, -0.00060000003, 2.9662001]

rotation matrix: [-3.130820479807535;
 0.1410169087193231;
 0.09986259200870334]

translation matrix: [-14.50253178802405;
 -2.772477945619816;
 25.64309759663749]

rotation matrix: [-3.131692865527312;
 0.1414243312111276;
 0.09998260840567]

translation matrix: [-14.55736859509761;
 -2.747801163446818;
 25.70250968490404]

rotation matrix: [-3.131581854665653;
 0.1420595467929274;
 0.1011201874115133]

translation matrix: [-14.58157957721772;
 -2.736715605872995;
 25.72219288232194]

rotation matrix: [-3.132351646817835;
 0.143997615408867;
 0.1020865811467397]

translation matrix: [-14.6208356594552;
 -2.723322013515466;
 25.75435043474477]

rotation matrix: [-3.132158293718152;
 0.1448350701726967;
 0.1025357518329009]

translation matrix: [-14.64067485997869;
 -2.708872036946939;
 25.77677404324281]

rotation matrix: [-3.132792579892109;
 0.1451892232339707;
 0.102608237433277]
```

### Task 5 - Project Outside Corners or 3D Axes

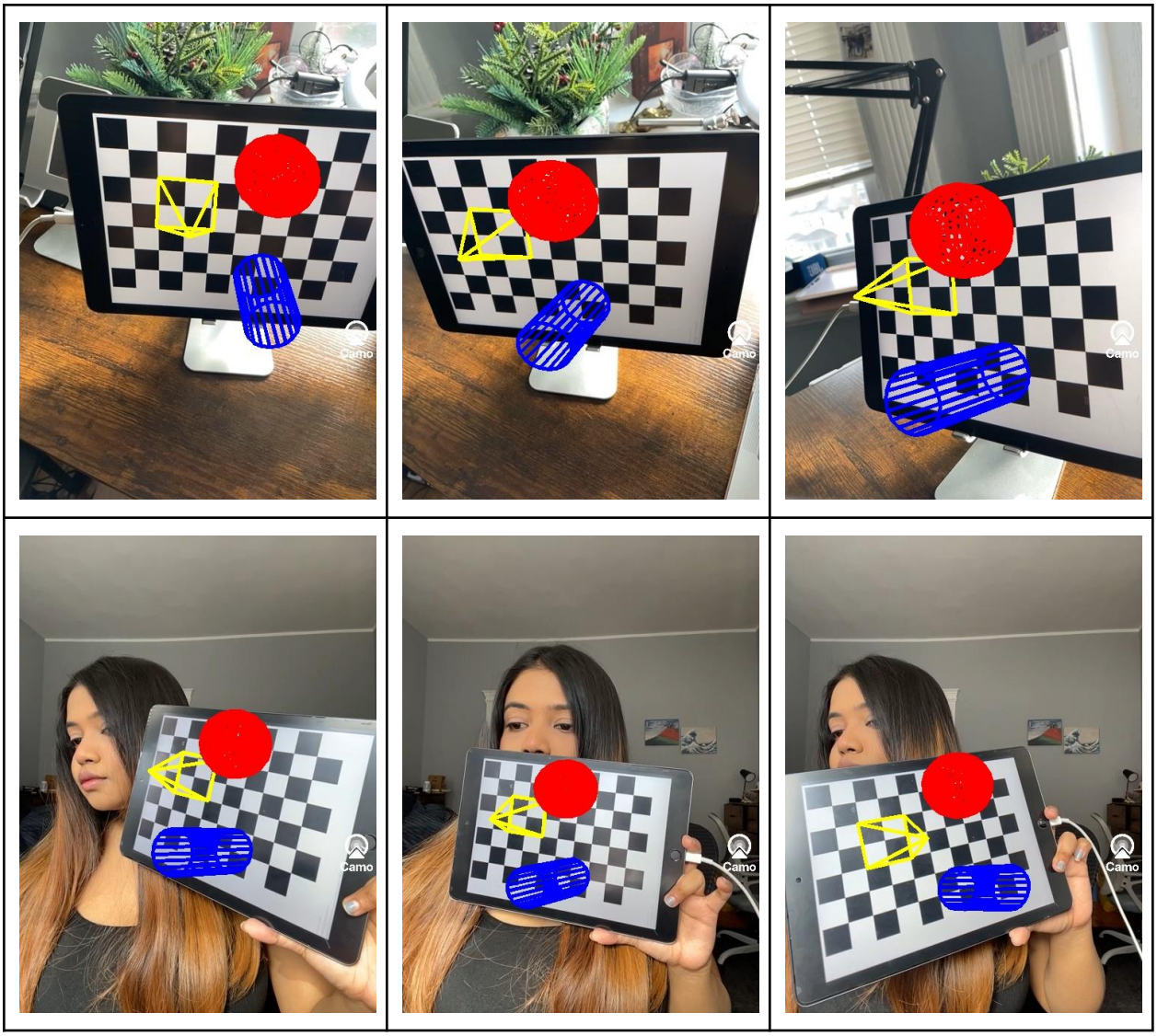
After determining the position and orientation (pose) of the camera in relation to the chessboard target, the next step is to map the 3D points in world coordinates onto the 2D image plane using OpenCV's projectPoints function. This allows us to display the 3D points as 2D points in the image frame, with the correct perspective and orientation.

When the user presses the 'x' key, the program projects a set of 3D axes (representing the x, y, and z axes) onto the origin point in world coordinates, which is located on the chessboard target. These axes are then displayed on the image frame, providing a visual reference for the orientation and positioning of the camera in relation to the target.

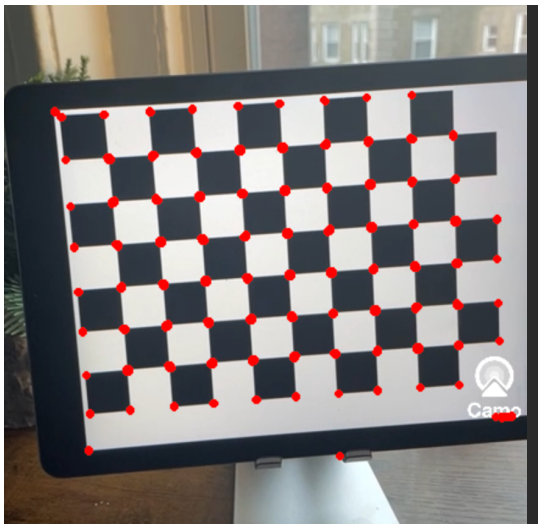
**Task 6 - Create a Virtual Object**

Next we draw a 3D triangle, cylinder, and sphere on the chessboard using the OpenCV projectPoints function to map 3D points to 2D points in image frame coordinates based on camera calibration parameters. The triangle is drawn with yellow lines, the cylinder with blue lines, and the sphere with red lines. The number of segments used to approximate the shapes can be adjusted with the "num_segments" variable.



**Task 7 - Detect Robust Feature**

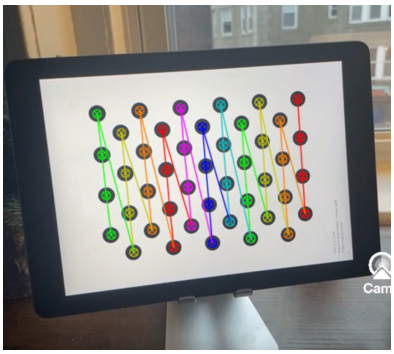
Next we use the Harris corner detection method to detect corners in an input image frame, and draw them on the output frame. It takes input image frame and output frame as its parameters. The corners are detected based on certain parameters such as threshold, block size, aperture size, and coefficient k.

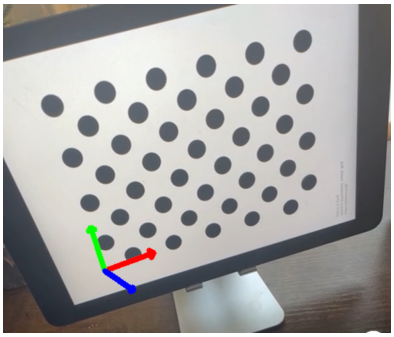


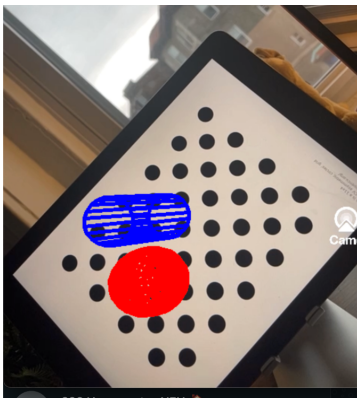
***Extensions:***

**1.Get your AR system working with a target other than the checkerboard, like a photo, painting, or object of your choice that is not a checkerboard. Make it something where it is easy to find 3-4 points on the object and determine which point is which. Place an AR object somewhere in the world relative to the object.**

For this extension, I used a different type of target called the asymmetric circle grid and calibrated my system again in a similar manner as done for the checkerboard target. The asymmetric circle grid consists of a 4x11 circle grid, detected using the OpenCV function findCirclesGrid, with adjacent sets of circles staggered. I constructed the 3D coordinates for each circle using the same rule as the chessboard.

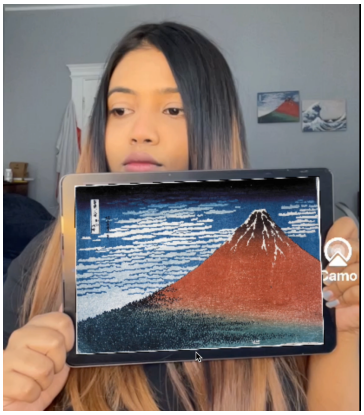


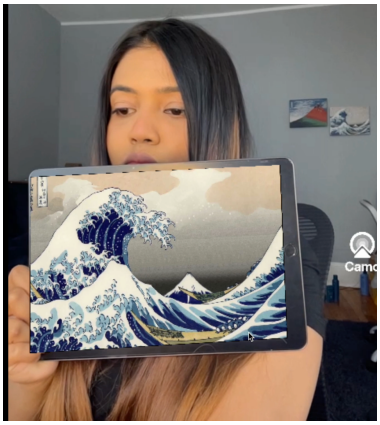




## 2.Not only add a virtual object, but also do something to the target to make it not look like a target any more.

In this extension, I used the perspective transform method to project two of my favorite artworks ‘The Great Wave off kanagawa’ and ‘Fine Wind, Clear Morning’ paintings onto the target, as can be seen in the background of my room. First, I detected the corners of the target image and formed a polygon in 3D world coordinates, which was then projected onto the video frame image coordinates. Then, I laid the artwork inside the polygon formed by the outer edges, giving the impression that it's replacing the canvas. Finally, I used a binary mask to get the details outside the target, blacking out the canvas polygon while keeping the rest of the video frame white, creating the illusion that everything around the target remains the same while the target is transformed into an image canvas.





***A short reflection of what you learned.***

I really enjoyed working on this project as it gave me a better understanding of the basics of Augmented Reality. During the project, I was introduced to many OpenCV functions that helped me in achieving my objectives. Overall, this was my favorite project so far!

I learned a lot of things during this project. For instance, I learned about camera calibration using detected feature points, and how intrinsic or extrinsic parameters affect the calibration. Additionally, I learned about mapping 3D points in the target world coordinate to image pixel coordinates, which helps in placing virtual 3D objects on the scene that align with camera movement.

One of the most important aspects of AR systems is figuring out how real-world 3D objects can be projected onto a 2D computer screen and which point in the object collapses to which pixel of the video frame. The extension of transforming a target into an image canvas taught me the concepts of perspective transformation.

Overall, this was a very fun project to work on, and it allowed me to be creative while learning new concepts.

***Acknowledgement of any materials or people you consulted for the assignment.***

- The official OpenCV documentation, which was the primary reference for implementing image processing and computer vision algorithms using OpenCV in C++. https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a
- https://longervision.github.io/2017/03/18/ComputerVision/OpenCV/opencv-internal-calibration-circle-grid/
- The C++ documentation, which was used as a reference for the C++ programming language.