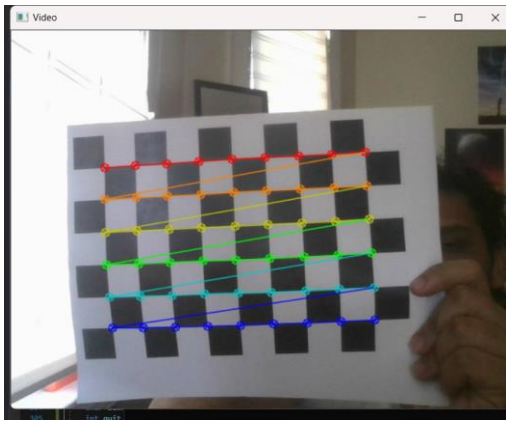
Project 4: Calibration and Augmented Reality

Brief Description:

The goal of this project was to enhance comprehension of the implementation of augmented reality. The concept involves using a camera to recognize a familiar object (in this case, a chessboard) and utilize it to create a mapping of image coordinates to real-world coordinates. Several calibration images are taken from various angles of the chessboard, with a minimum threshold of five images, and then these images are used to calibrate the camera. After calibration, it becomes feasible to insert virtual objects into the scene, as long as the chessboard is visible, resulting in an augmented reality experience.

Task1:Detect and Extract Chessboard Corners

The code has implemented the findChessboardCorners function to detect 54 corners for a 9 x 6 checkerboard. The precise coordinates of these corners are then obtained using cornersubpix. The program attempts to identify a chessboard from a live stream and marks the detected chessboard corners once it is found.

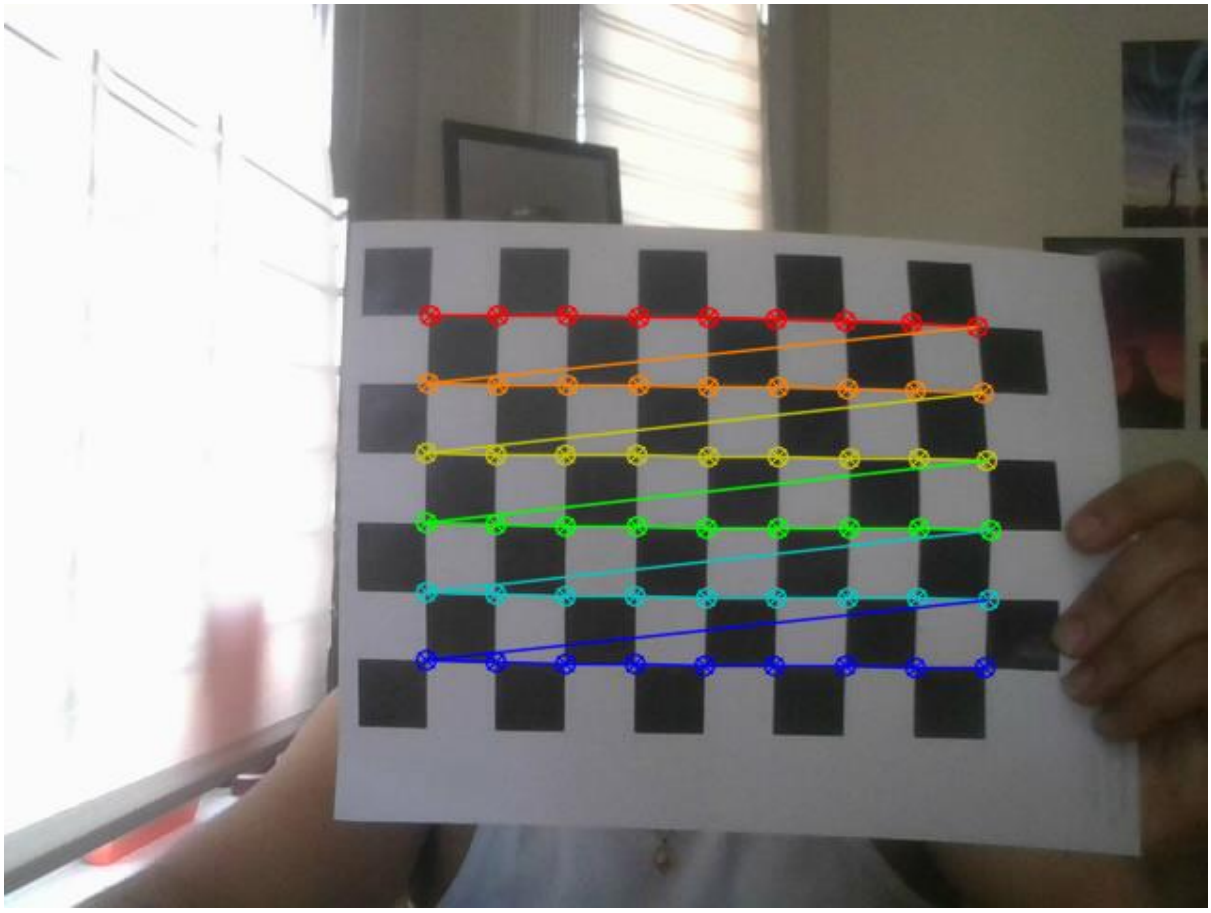


1 Chessboard detected and displayed on screen

Task2: Select Calibration Images

Upon clicking the '2' button, the program checks for the presence of a chessboard in the live stream. If detected, the corner points are gathered and saved for calibration, and the corresponding image is saved to the local drive. The drawChessboardCorners() function is utilized to highlight the chessboard corners on the saved calibration image. Here is an example of one such saved calibration image with

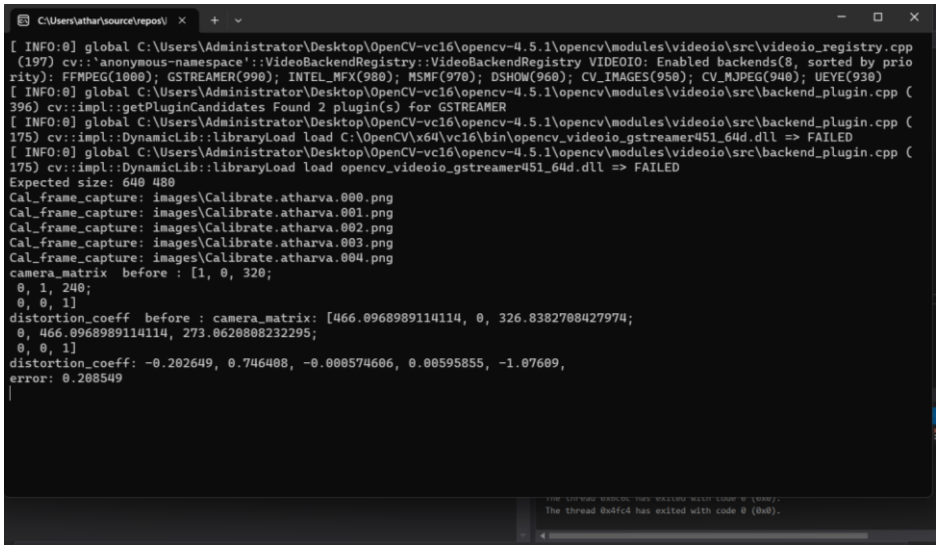
the chessboard corners marked.



2One of the calibration images used

Task3: Calibrate the Camera

Upon the user clicking '2' five times, the program collects five image frames along with their corresponding image and real-world coordinates. It then uses the calibrateCamera() function to calculate the camera matrix, distortion coefficients, and re-projection error. The resulting camera matrix, distortion coefficients, and error are displayed as output.

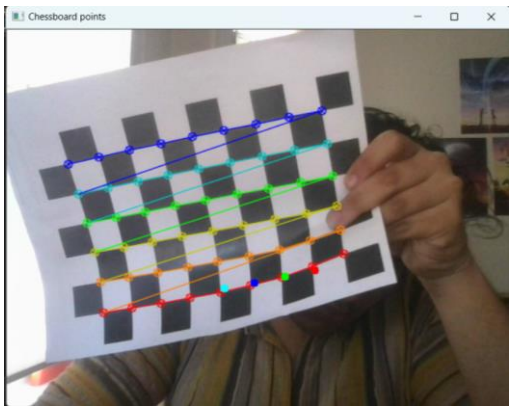


Task 4 Calculate Current Position of the Camera

The objective of this task was to compute the camera's pose in real-time, utilizing the camera calibration parameters obtained earlier. To achieve this, the program reads the camera calibration data stored in the intrensics.xml file and utilizes it as input for the solvePNP function to determine the pose of the chessboard. The program then updates the rotation and translation of the chessboard in real-time, based on its position in the live stream.

Task5: Project Outside Corners or 3D Axes

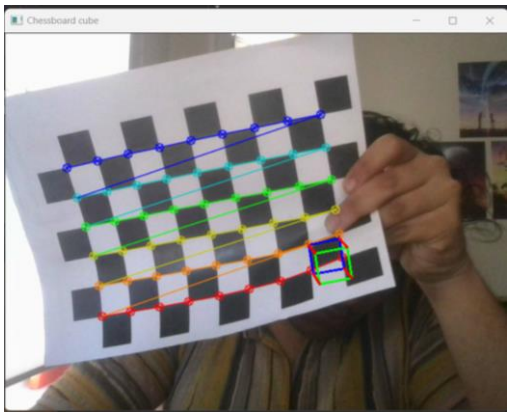
The objective was to display the 3D axes projected from the corner of a chessboard. To achieve this, the projectPoints() function was utilized to identify the 3D world points that corresponded to the specified image coordinates.



3 Four dots are drawn on the 3D Plane

Task6: Create a Virtual Object

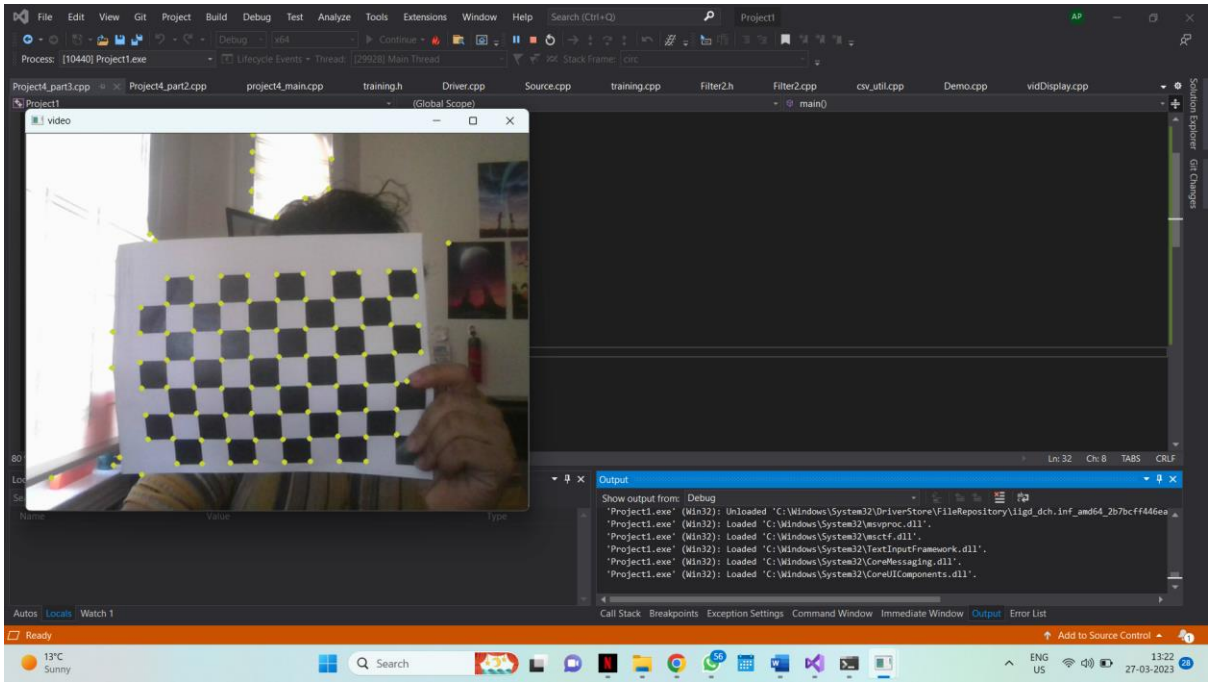
The task involved generating a virtual object comprising of lines and projecting it onto a chessboard while ensuring that its orientation remained constant even as the object moved around. The object created included a cube, and projectPoints() function was used to achieve this. The function was supplied with the image points corresponding to the shape. A screenshot of the resulting virtual object is displayed below.



4Cube drawn on the 3D Plane

Task7:Detect Robust Features

To detect features, I employed Harris corners detection using the goodFeaturesToTrack function on the chessboard. This involved selecting a random pattern of Aruco markers and applying Harris corner detection to identify the corners. The resulting image displays the detected Harris corners.



Reflection:

This project was an enjoyable experience for me. Initially, the project involved comprehending the concepts and implementing the appropriate functions with accurate parameters, which was relatively straightforward. However, as the project progressed and it came time to draw the 3D object and axes, it became more engaging. It required me to utilize my creativity to generate the cube shape. I unfortunately did not do any extensions, I plan on working on the next one better such that I can provide excellent extensions.

Acknowledgement:

I would like to acknowledge Professor Maxwell's conducting coding session. I referred to the

- OpenCV documentation
- https://datahacker.rs/opencv-harris-corner-detector-part2/
- https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d