# CS5330 – PROJECT 4

# Project 4: Calibration and Augmented Reality

## Authors:

**Name**: Ravi Shankar Sankara Narayanan

**NUID**: 001568628

## Short Description:

This project focuses on two key areas: calibration and Augmented Reality. The main project is divided into 3 programs:

1. Calibration and obtaining intrinsic calibration parameters.
2. Using the camera calibration parameters to calibrate the camera and project 3-D points onto the target frame (Chessboard).
3. Detect Robust features.

The first program calibrates a camera using a chessboard pattern. The program captures frames, finds, and refines the corners of the chessboard, and stores them if 's' is pressed. Then it uses these corners to estimate the camera's intrinsic parameters and distortion coefficients when enough frames are collected. The estimated parameters are saved to an xml file.

The second program reads camera parameters provided by the first program. It then detects a chessboard pattern in a video frame, estimates the chessboard's pose, and overlays virtual 3D objects (a pyramid and a cube) onto the video frame based on the estimated pose. The result is a video stream where virtual objects appear to exist in the real world.

And finally, the third program is a real-time feature detection program. It captures video from the default camera, converts each frame to grayscale, and then uses a variation of Harris corners method to detect corners in the grayscale image. The detected corners are then drawn on the original frame as circles. The frame with the detected corners is displayed in a window.

## File name and purpose:

- main_task1.cpp : Function implementation for task 1,2,3
- operations_task1.cpp: Function definition for task 1,2,3
- operations_task1.h : Function headers for task 1,2,3
- main_task4.cpp : Function implementation for task 4,5,6 and extensions
- operations_task4.cpp: Function definition for task 4,5,6 and extensions
- operations_task4.h : Function headers for task 4,5,6 and extensions
- main_task7.cpp : Function implementation for task

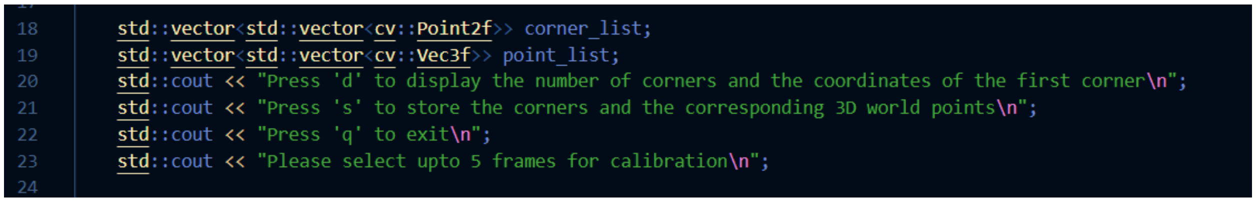
## Task outputs:

### Task 1: Detect and Extract Target Corners

The target I will be using for this project is the chessboard target. The functions provided on the canvas were very helpful in detecting the corners of the chessboard. The only challenging part was to reduce light reflections(illuminations) from the chessboard so that the function can detect the corners correctly.

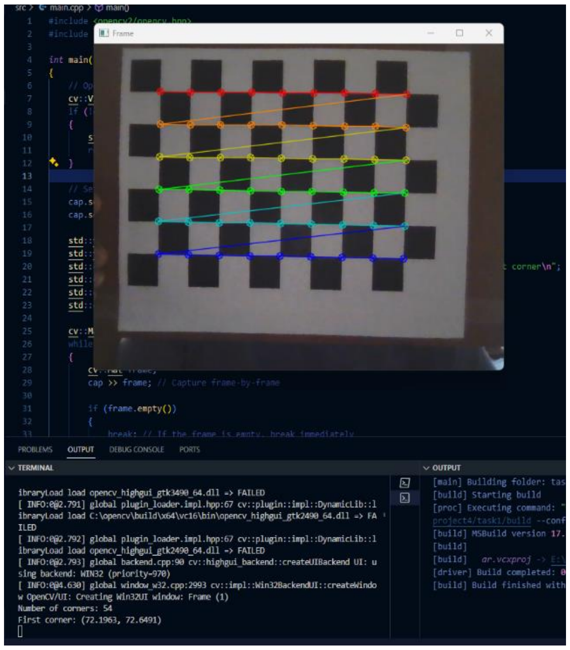
The program has a function 'findChessboardCorners' that first converts the frame to grayscale and then attempts to find the corners of a chessboard pattern in the frame using the cv::findChessboardCorners function. If the pattern is found, the corners are refined using the cv::cornerSubPix function and drawn on the frame with cv::drawChessboardCorners.

If the 'd' key is pressed, the function prints the number of corners found and the coordinates of the first corner in the terminal.

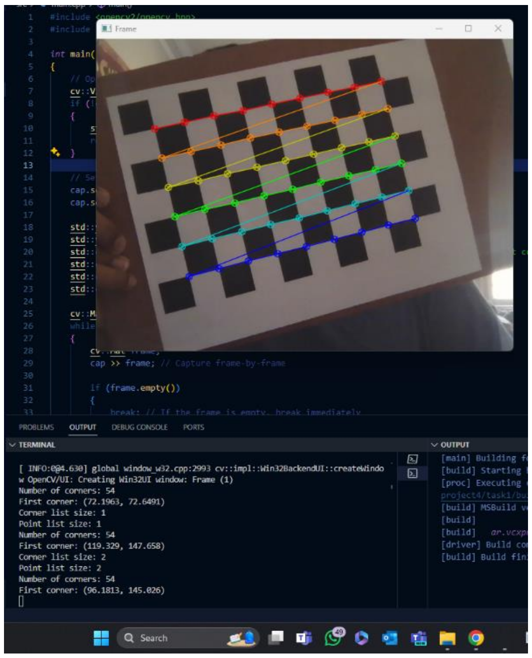


```cpp
    std::vector<std::vector<cv::Point2f>> corner_list;
    std::vector<std::vector<cv::Vec3f>> point_list;
    std::cout << "Press 'd' to display the number of corners and the coordinates of the first corner\n";
    std::cout << "Press 's' to store the corners and the corresponding 3D world points\n";
    std::cout << "Press 'q' to exit\n";
    std::cout << "Please select upto 5 frames for calibration\n";
```

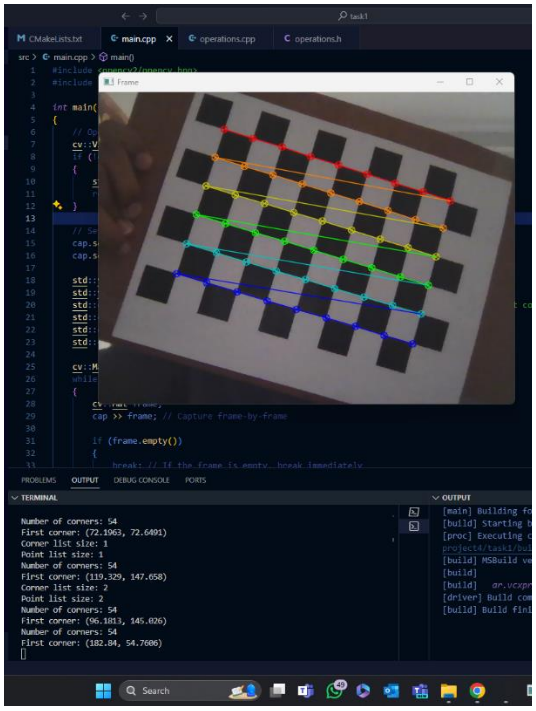
The image results are attached below:



```
ibraryLoad load opencv_highgui_gtk3490_64.dll => FAILED
[ INFO:0@2.791] global plugin_loader.impl.hpp:67 cv::plugin::impl::DynamicLib::l
ibraryLoad load C:\opencv\build\x64\vc16\bin\opencv_highgui_gtk2490_64.dll => FA
ILED
[ INFO:0@2.792] global plugin_loader.impl.hpp:67 cv::plugin::impl::DynamicLib::l
ibraryLoad load opencv_highgui_gtk2490_64.dll => FAILED
[ INFO:0@2.793] global backend.cpp:90 cv::highgui_backend::createUIBackend UI: u
sing backend: WIN32 (priority=970)
[ INFO:0@4.630] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindo
w OpenCV/UI: Creating Win32UI window: Frame (1)
Number of corners: 54
First corner: (72.1963, 72.6491)
```



```
[ INFO:0@4.630] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindo
w OpenCV/UI: Creating Win32UI window: Frame (1)
Number of corners: 54
First corner: (72.1963, 72.6491)
Corner list size: 1
Point list size: 1
Number of corners: 54
First corner: (119.329, 147.658)
Corner list size: 2
Point list size: 2
Number of corners: 54
First corner: (96.1813, 145.026)
```



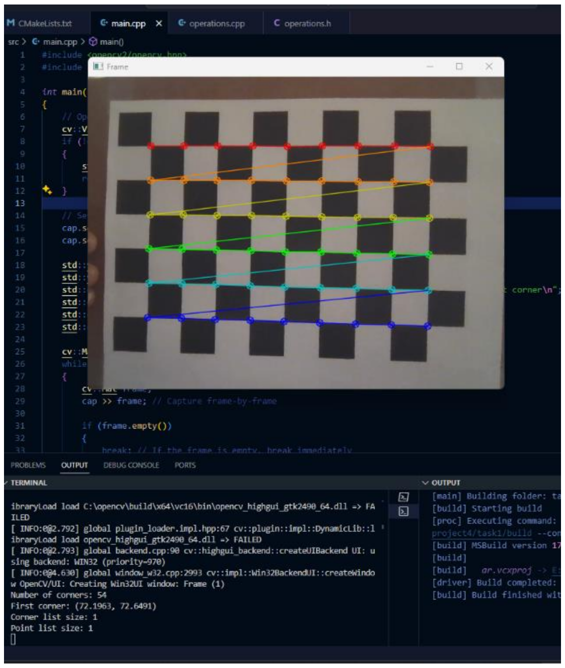
```
Number of corners: 54
First corner: (72.1963, 72.6491)
Corner list size: 1
Point list size: 1
Number of corners: 54
First corner: (119.329, 147.658)
Corner list size: 2
Point list size: 2
Number of corners: 54
First corner: (96.1813, 145.026)
Number of corners: 54
First corner: (182.84, 54.7606)
```

From the above three images we can observe that when the chessboard pose changes the co-ordinates of the first corner changes accordingly.

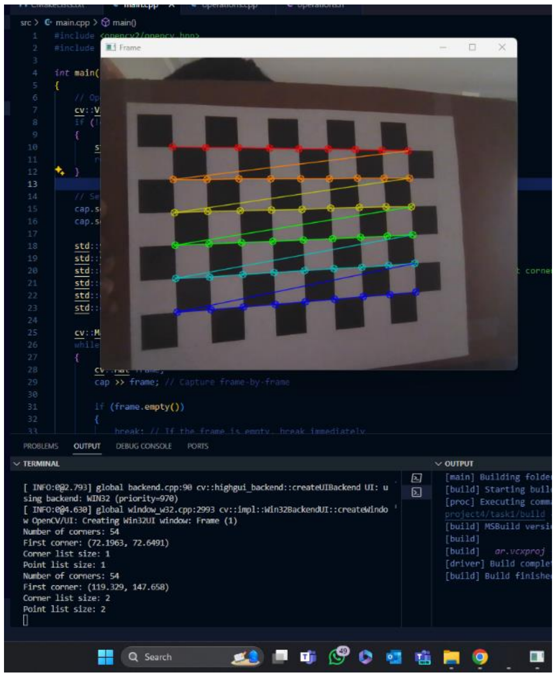
## Task 2: Select Calibration Images

As a part of the same function ‘findChessboardCorners’, If the 's' key is pressed, the function stores the corners in corner_list and creates a corresponding set of 3D points with size of the chessboard squares being 1 unit. These 3D points are stored in point_list. The function then also saves the current frame as an image file, incrementing a counter to ensure each saved image has a unique filename.

After the ‘s’ key is pressed, the size of corner_list and point_list is printed in the terminal as an acknowledgement to confirm that the data has been added to the list. The results are attached below:

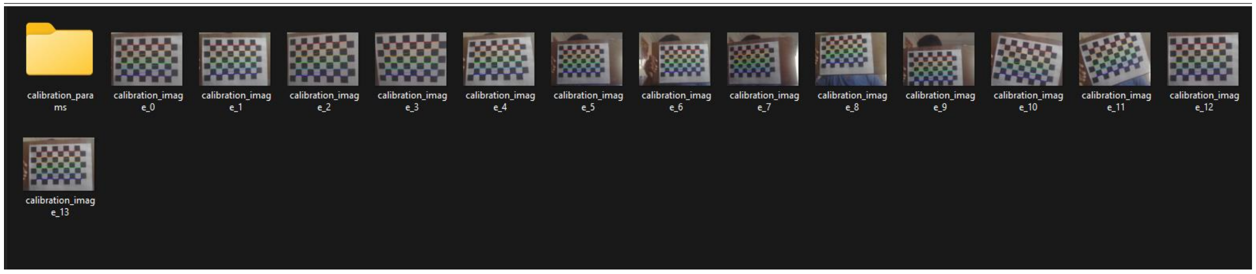


```
ibraryLoad load C:\opencv\build\x64\vc16\bin\opencv_highgui_gtk2490_64.dll => FA
ILED
[ INFO:0@2.792] global plugin_loader.impl.hpp:67 cv::plugin::impl::DynamicLib::l
ibraryLoad load opencv_highgui_gtk2490_64.dll => FAILED
[ INFO:0@2.793] global backend.cpp:90 cv::highgui_backend::createUIBackend UI: u
sing backend: WIN32 (priority=970)
[ INFO:0@4.630] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindo
w OpenCV/UI: Creating Win32UI window: Frame (1)
Number of corners: 54
First corner: (72.1963, 72.6491)
Corner list size: 1
Point list size: 1
```



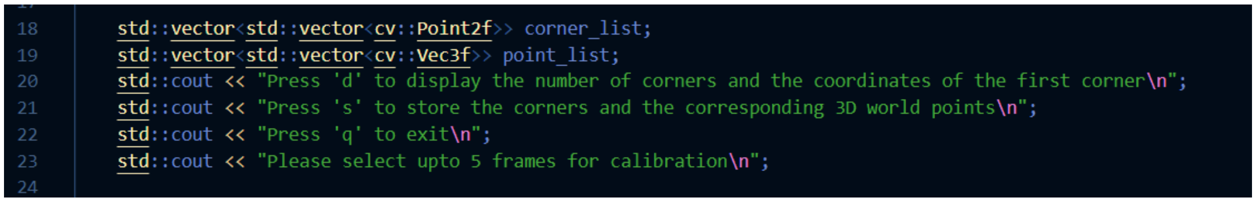
```
[ INFO:0@2.793] global backend.cpp:90 cv::highgui_backend::createUIBackend UI: u
sing backend: WIN32 (priority=970)
[ INFO:0@4.630] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindo
w OpenCV/UI: Creating Win32UI window: Frame (1)
Number of corners: 54
First corner: (72.1963, 72.6491)
Corner list size: 1
Point list size: 1
Number of corners: 54
First corner: (119.329, 147.658)
Corner list size: 2
Point list size: 2
```

From the above two images, we can observe that the corner and point list increases as ‘s’ key pressed for different poses of chessboard.



And the images are also stored each time the corner and point list are updated.

These are the instructions that the program will print at the beginning, so that it’s easy for the user:



```cpp
    std::vector<std::vector<cv::Point2f>> corner_list;
    std::vector<std::vector<cv::Vec3f>> point_list;
    std::cout << "Press 'd' to display the number of corners and the coordinates of the first corner\n";
    std::cout << "Press 's' to store the corners and the corresponding 3D world points\n";
    std::cout << "Press 'q' to exit\n";
    std::cout << "Please select upto 5 frames for calibration\n";
```

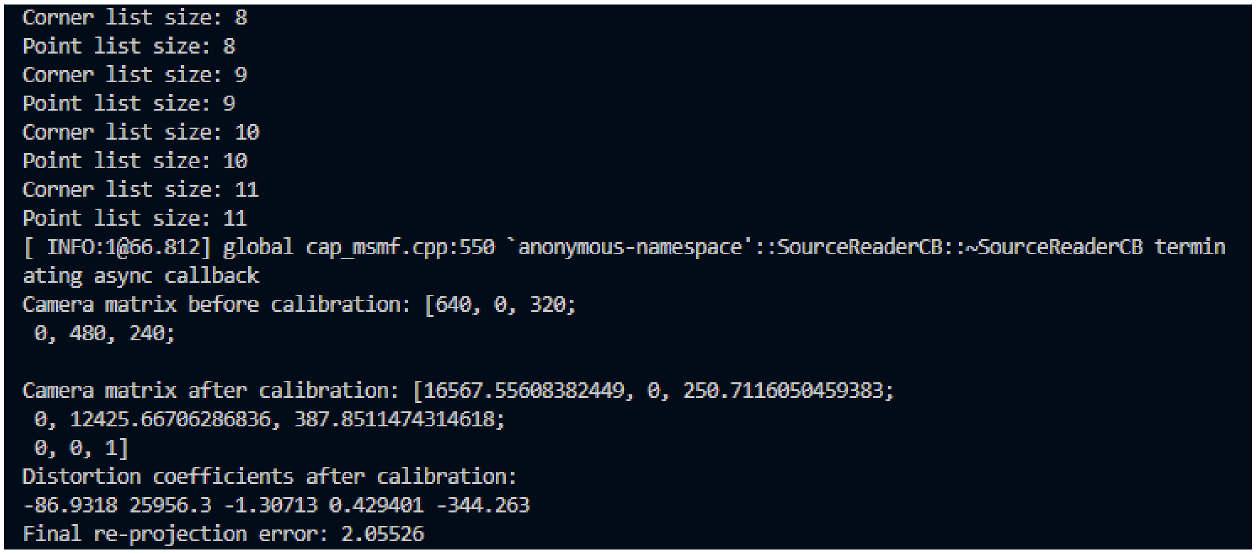
## Task 3: Calibrate the Camera

When key ‘q’ is pressed, the camera frame closes and then the program runs the runCalibration function. It takes as input a list of 2D image points (corner_list), a list of corresponding 3D world points (point_list), and a frame for size reference.

If at least 5 sets of points are provided, the function initializes a camera matrix and an empty list for distortion coefficients. It then calls cv::calibrateCamera to estimate the camera's intrinsic parameters and distortion coefficients, as well as the rotation and translation vectors for each image.

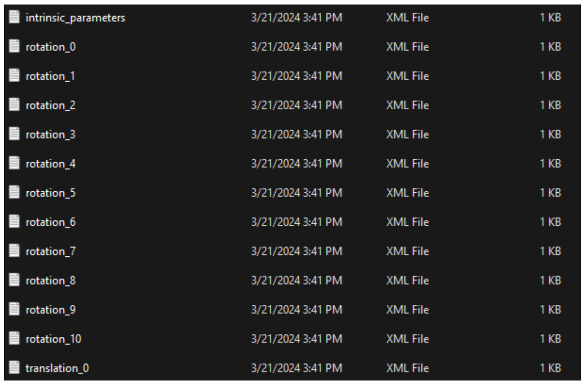
The estimated parameters are printed to the console and saved in an xml file. The function also saves each rotation and translation vector to a separate XML file. If fewer than 5 sets of points are provided, the function prints a message asking for more frames.

The results are attached below:



```
Corner list size: 8
Point list size: 8
Corner list size: 9
Point list size: 9
Corner list size: 10
Point list size: 10
Corner list size: 11
Point list size: 11
[ INFO:1@66.812] global cap_msmf.cpp:550 `anonymous-namespace'::SourceReaderCB::~SourceReaderCB termin
ating async callback
Camera matrix before calibration: [640, 0, 320;
 0, 480, 240;

Camera matrix after calibration: [16567.55608382449, 0, 250.7116050459383;
 0, 12425.66706286836, 387.8511474314618;
 0, 0, 1]
Distortion coefficients after calibration:
-86.9318 25956.3 -1.30713 0.429401 -344.263
Final re-projection error: 2.05526
```

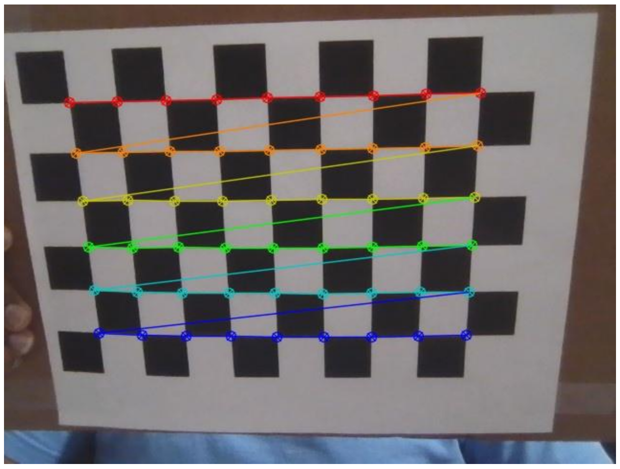
The output after the calibration function runs is provided in the above image. As we can observe, I got a re-projection error of 2.05 using the integrated webcam in my laptop. This reprojection error is decent enough for projecting 3-D points but the error can still be reduced.

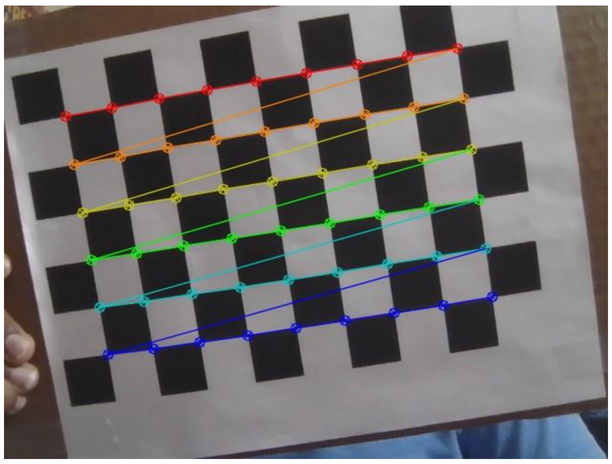


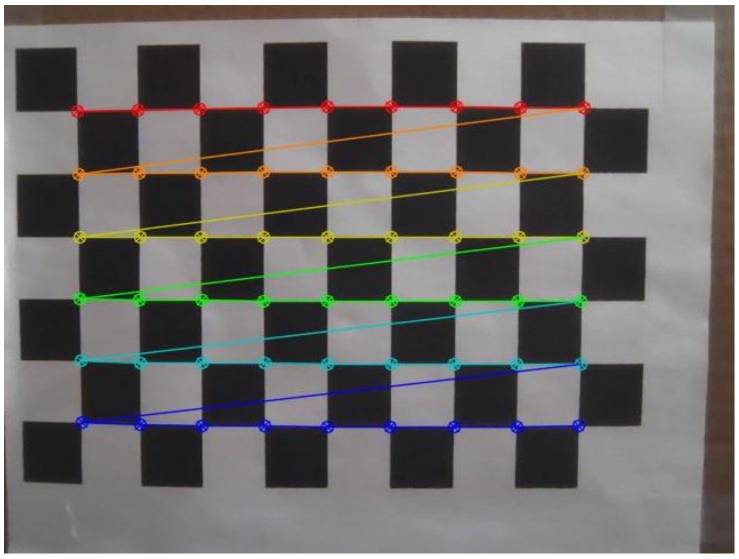
| | | | |
|---|---|---|---|
| intrinsic_parameters | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_0 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_1 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_2 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_3 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_4 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_5 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_6 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_7 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_8 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_9 | 3/21/2024 3:41 PM | XML File | 1 KB |
| rotation_10 | 3/21/2024 3:41 PM | XML File | 1 KB |
| translation_0 | 3/21/2024 3:41 PM | XML File | 1 KB |

The output files generated after the calibration is attached below. I used 11 images for calibration and 11 rotations and translations associated with each of the calibration image are generated.

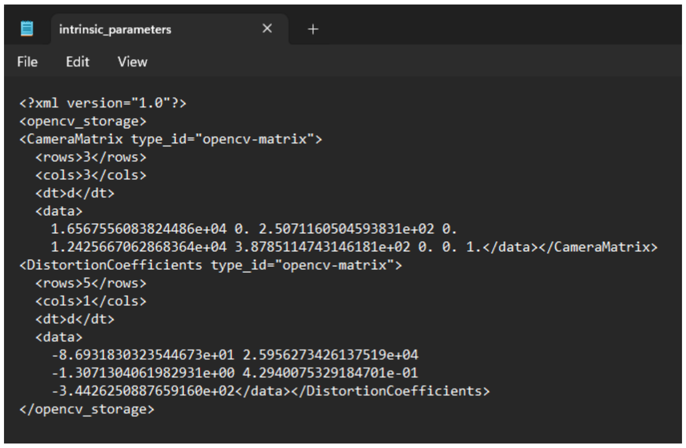
Some of the calibration images are attached below:





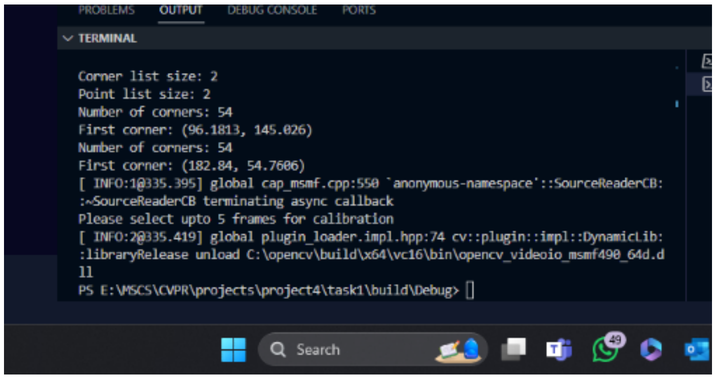


The intrinsic parameters xml file is attached below:



```
<?xml version="1.0"?>
<opencv_storage>
<CameraMatrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    1.6567556083824486e+04 0. 2.5071160504593831e+02 0.
    1.2425667062868364e+04 3.8785114743146181e+02 0. 0. 1.</data></CameraMatrix>
<DistortionCoefficients type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    -8.6931830323544673e+01 2.5956273426137519e+04
    -1.3071304061982931e+00 4.2940075329184701e-01
    -3.4426250887659160e+02</data></DistortionCoefficients>
</opencv_storage>
```

If the user has selected less than 5 images for calibration, then a message is displayed in terminal and the program ends.

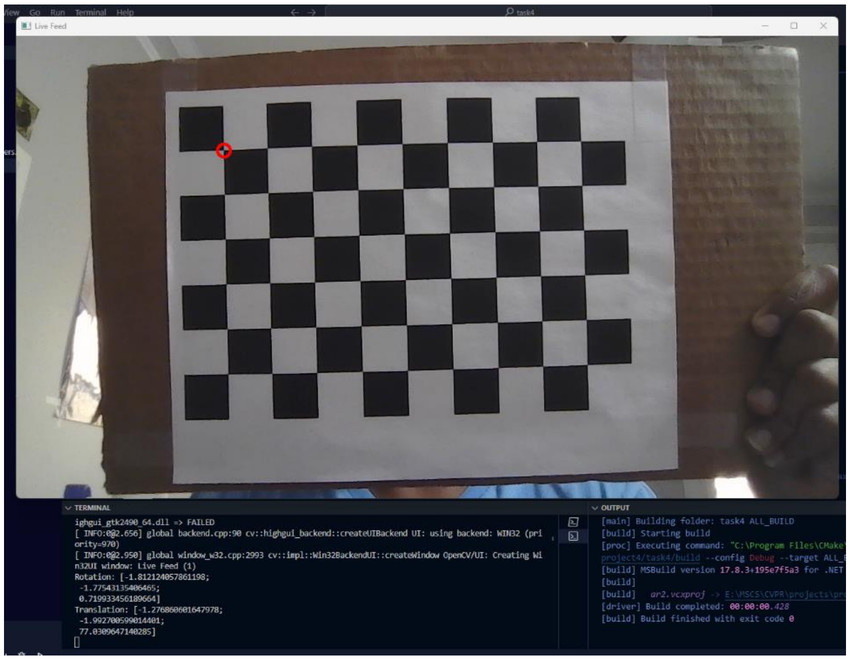


```
Corner list size: 2
Point list size: 2
Number of corners: 54
First corner: (96.1813, 145.026)
Number of corners: 54
First corner: (182.84, 54.7606)
[ INFO:1@335.395] global cap_msmf.cpp:550 `anonymous-namespace'::SourceReaderCB:
:~SourceReaderCB terminating async callback
Please select upto 5 frames for calibration
[ INFO:2@335.419] global plugin_loader.impl.hpp:74 cv::plugin::impl::DynamicLib:
:libraryRelease unload C:\opencv\build\x64\vc16\bin\opencv_videoio_msmf490_64d.d
ll
PS E:\MSCS\CVPR\projects\project4\task1\build\Debug> []
```

## <u>Task 4: Calculate Current Position of the Camera</u>

For task 4 I have written a separate program. This program has three functions. One reads the camera parameters from the xml file created in the previous task. The parameters include the camera matrix and distortion coefficients, which are essential for image undistortion and 3D projection. The second function attempts to find a chessboard pattern in each image. If the pattern is found, it returns true, and the corners of the chessboard are stored in the corners vector. The third function estimates the 3D pose (rotation and translation) of the chessboard relative to the camera. It uses the Perspective-n-Point (PnP) problem to solve for the pose that minimizes the re-projection error between the observed 2D points (chessboard corners) and the projected 3D points. I have also marked the origin (the first corner) with a red circle for reference.

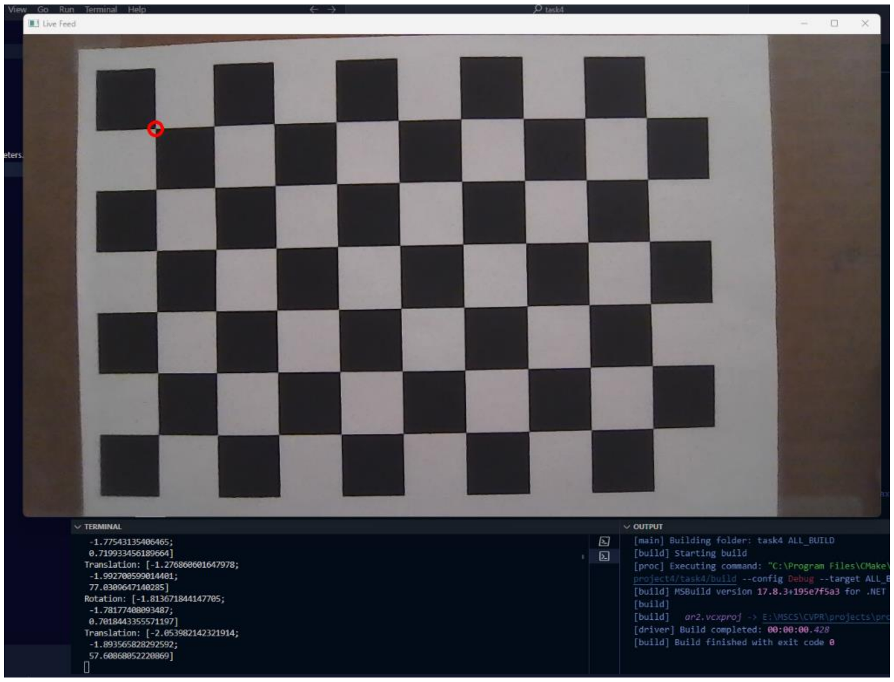
As I press the ‘p’ key the pose of the chessboard is printed in the terminal. The output is attached below:



```
ighgui_gtk2490_64.dll => FAILED
[ INFO:0@2.656] global backend.cpp:90 cv::highgui_backend::createUIBackend UI: using backend: WIN32 (pri
ority=970)
[ INFO:0@2.950] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Wi
n32UI window: Live Feed (1)
Rotation: [-1.812124057861198;
 -1.77543135406465;
 0.719933456189664]
Translation: [-1.276860601647978;
 -1.992700599014401;
 77.0309647140285]
```

```
[main] Building folder: task4 ALL_BUILD
[build] Starting build
[proc] Executing command: "C:\Program Files\CMake\
project4/task4/build --config Debug --target ALL_B
[build] MSBuild version 17.8.3+195e7f5a3 for .NET
[build]
[build]   ar2.vcxproj -> E:\MSCS\CVPR\projects\pro
[driver] Build completed: 00:00:00.428
[build] Build finished with exit code 0
```

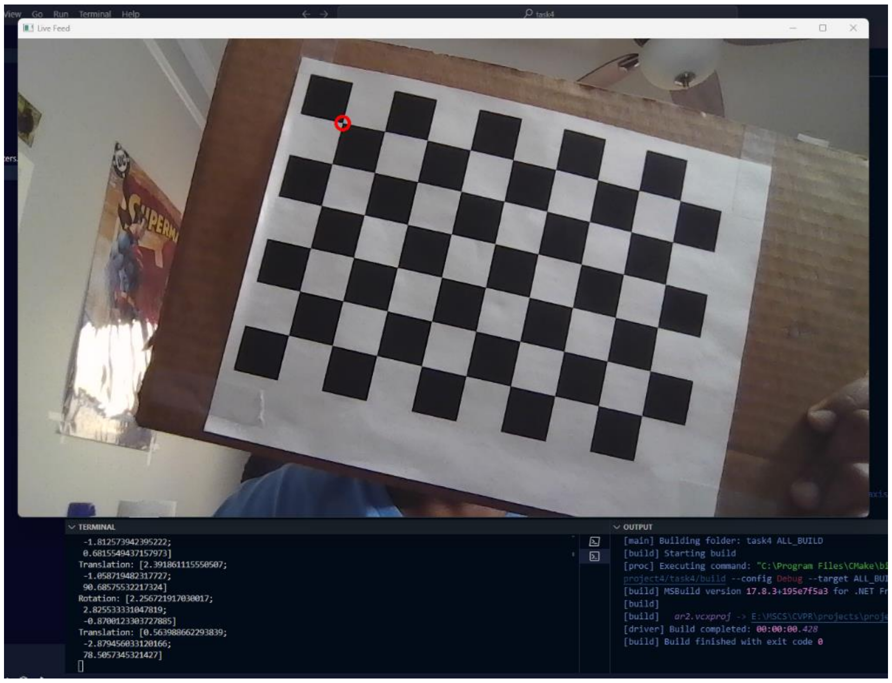
The pose is printed in the terminal below when I press key ‘p’.



```
 -1.77543135406465;
 0.719933456189664]
Translation: [-1.276860601647978;
 -1.992700599014401;
 77.0309647140285]
Rotation: [-1.813671844147705;
 -1.78177408093487;
 0.7018443355571197]
Translation: [-2.053982142321914;
 -1.893565828292592;
 57.60868052220869]
```

```
[main] Building folder: task4 ALL_BUILD
[build] Starting build
[proc] Executing command: "C:\Program Files\CMake\
project4/task4/build --config Debug --target ALL_B
[build] MSBuild version 17.8.3+195e7f5a3 for .NET
[build]
[build]   ar2.vcxproj -> E:\MSCS\CVPR\projects\pro
[driver] Build completed: 00:00:00.428
[build] Build finished with exit code 0
```

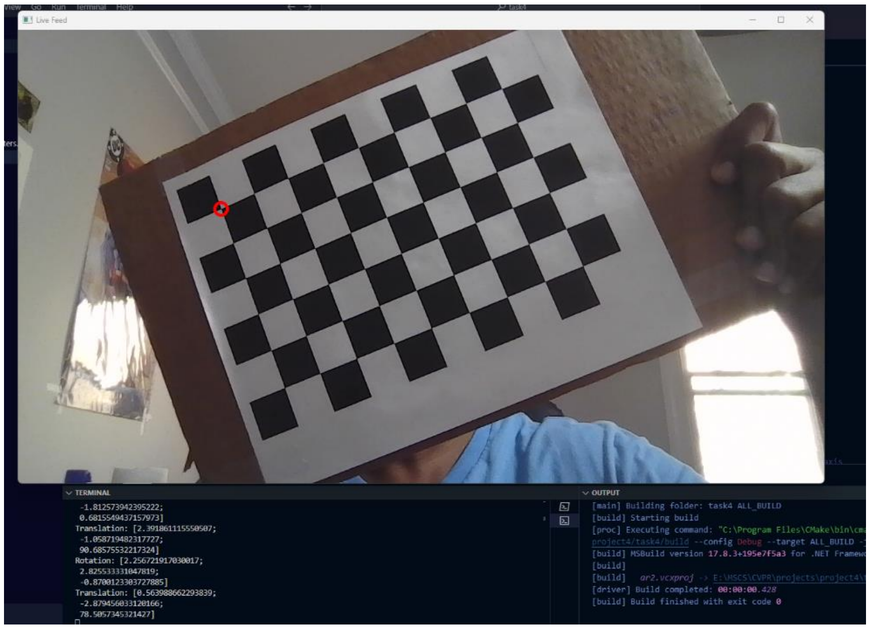
As I move the chessboard closer to the camera there is a decrease in Translation Z value.



```
 -1.812573942395222;
 0.6815549437157973]
Translation: [2.391861115550507;
 -1.058719482317727;
 90.68575532217324]
Rotation: [2.256721917030017;
 2.825533331047819;
 -0.8700123303727885]
Translation: [0.563988662293839;
 -2.879456033120166;
 78.5057345321427]
```

```
[main] Building folder: task4 ALL_BUILD
[build] Starting build
[proc] Executing command: "C:\Program Files\CMake\bi
project4/task4/build --config Debug --target ALL_BUI
[build] MSBuild version 17.8.3+195e7f5a3 for .NET Fr
[build]
[build]   ar2.vcxproj -> E:\MSCS\CVPR\projects\proje
[driver] Build completed: 00:00:00.428
[build] Build finished with exit code 0
```

Now I moved the chessboard farther from the camera and rotated it which is also displayed correctly in the terminal as there is a change in Rotation and Translation vector.



```
 -1.812573942395222;
 0.6815549437157973]
Translation: [2.391861115550507;
 -1.058719482317727;
 90.68575532217324]
Rotation: [2.256721917030017;
 2.825533331047819;
 -0.8700123303727885]
Translation: [0.563988662293839;
 -2.879456033120166;
 78.5057345321427]
```

```
[main] Building folder: task4 ALL_BUILD
[build] Starting build
[proc] Executing command: "C:\Program Files\CMake\bin\cma
project4/task4/build --config Debug --target ALL_BUILD -
[build] MSBuild version 17.8.3+195e7f5a3 for .NET Framewo
[build]
[build]   ar2.vcxproj -> E:\MSCS\CVPR\projects\project4\
[driver] Build completed: 00:00:00.428
[build] Build finished with exit code 0
```

Now I rotated the chessboard in the opposite direction and the rotation vector changes as needed.

From the above three images we can observe that as I change the pose of the chessboard in world frame, their respective changes in the pose are printed in the terminal when I press ‘p’ key.

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

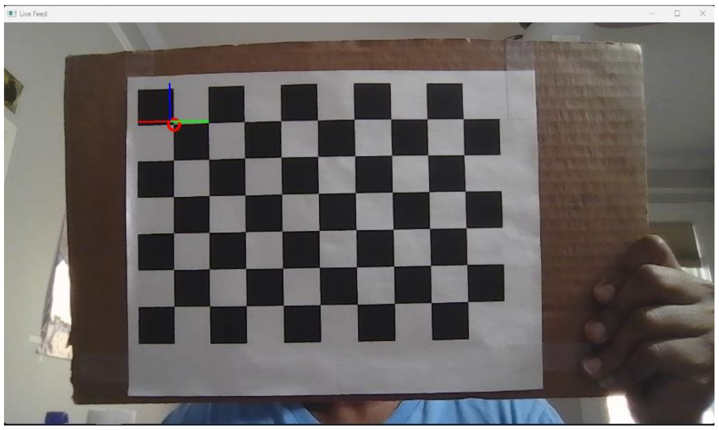
The program then defines a 3D coordinate axes, projects it onto the 2D image plane, and then draws the projected axes on the frame.

The axes are defined in 3D space with the origin at (0,0,0), the end of the X-axis at (3,0,0), the end of the Y-axis at (0,3,0), and the end of the Z-axis at (0,0,-3).

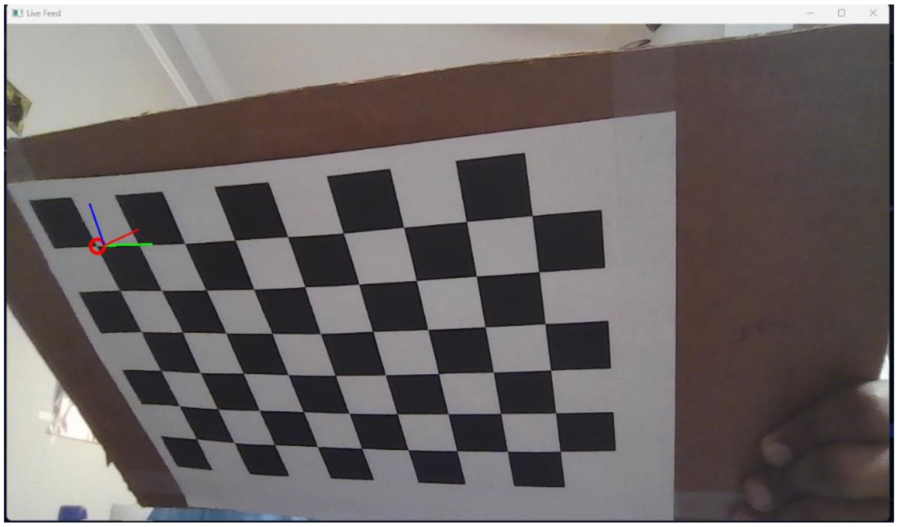
These points are then projected onto the 2D image plane using the camera parameters and the estimated pose of the chessboard.

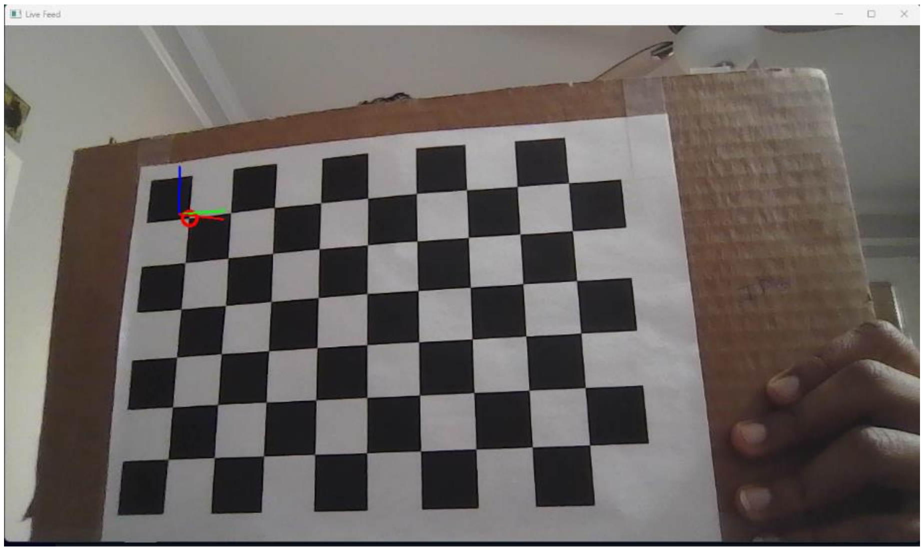
Finally, the projected axes are drawn on the frame: the X-axis in red, the Y-axis in green, and the Z-axis in blue.

The result is attached below:



The resulting 3-D axis is projected on the target. The z-axis might look like it's in x-axis but as I tilt the chessboard the result gets more obvious:



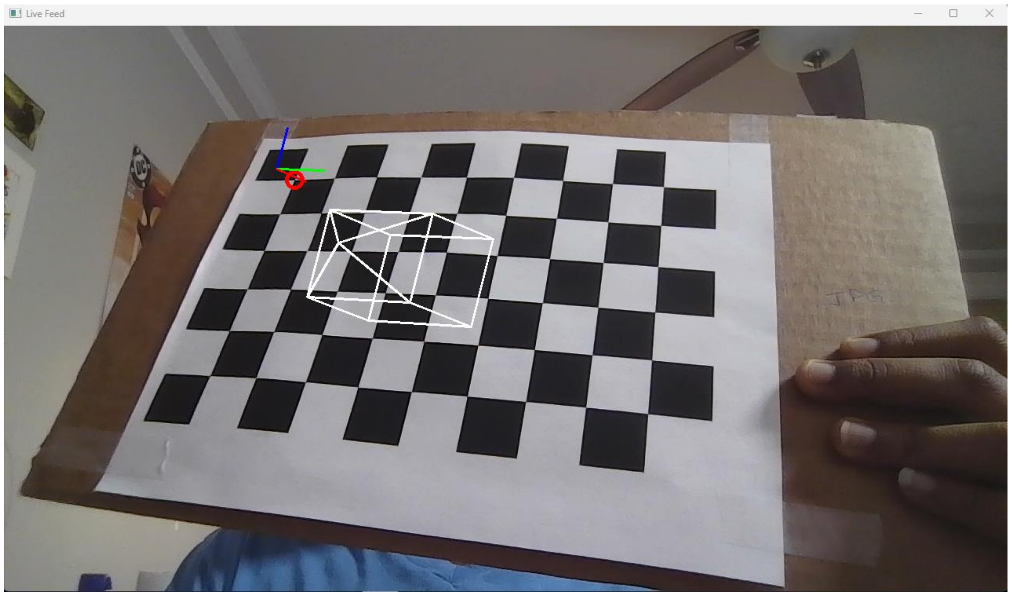


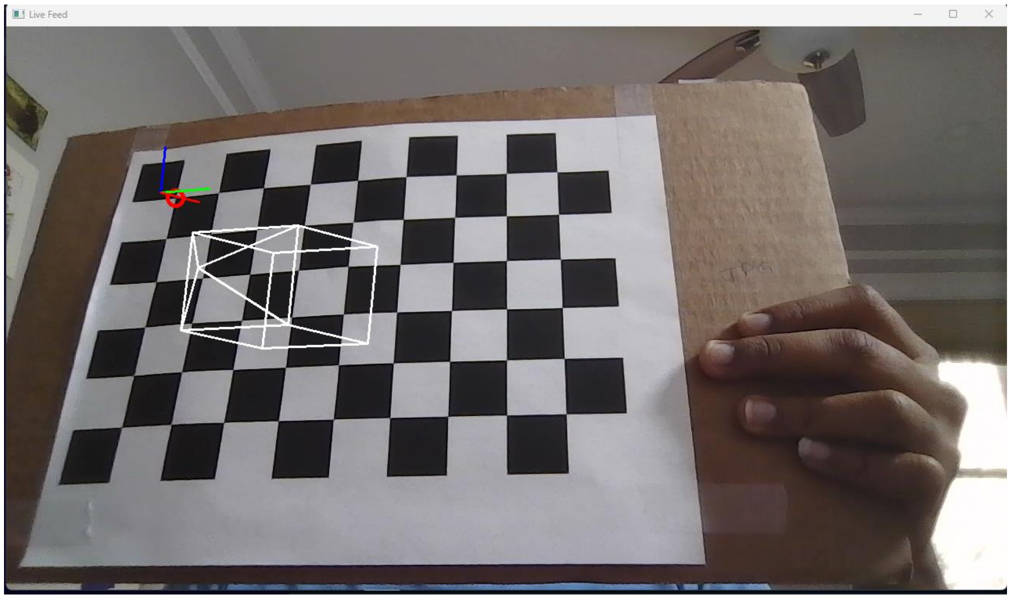
As I tilt the chessboard the z-axis (red color) points in the direction of the camera (away from the camera).

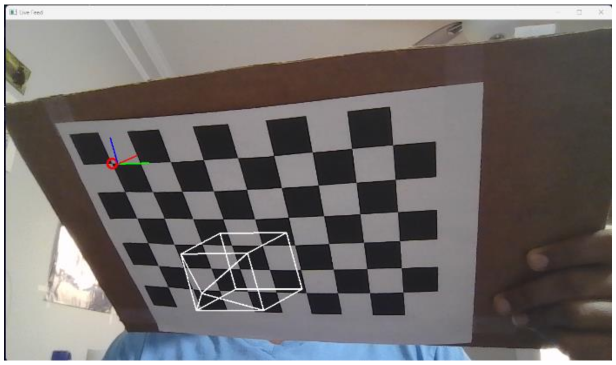
## **Task 6: Create a Virtual Object**

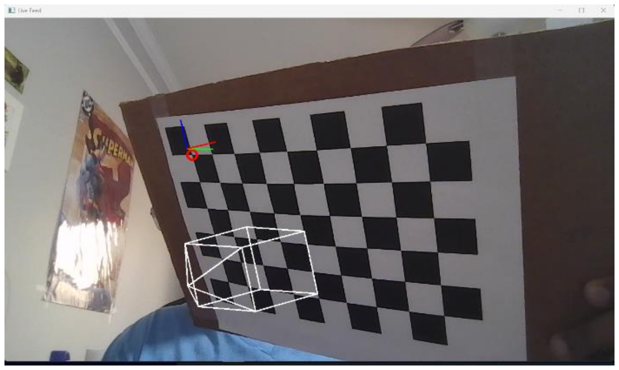
The program has a  drawHouse function that overlays a virtual 3D structure similar to Washington Monument onto a 2D image frame. It first defines the 3D points of the structure, then projects these points onto the 2D image plane using the camera parameters and the estimated pose. Finally, it draws lines between the projected points on the frame to form the figure.

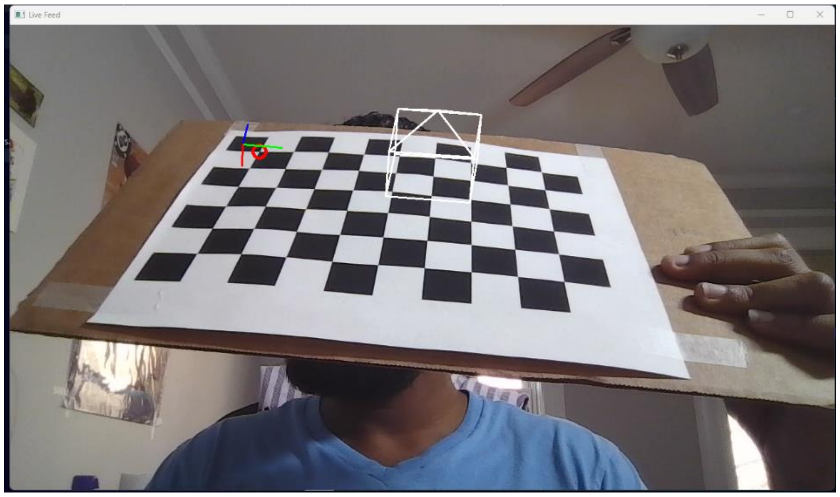
The results are attached below:











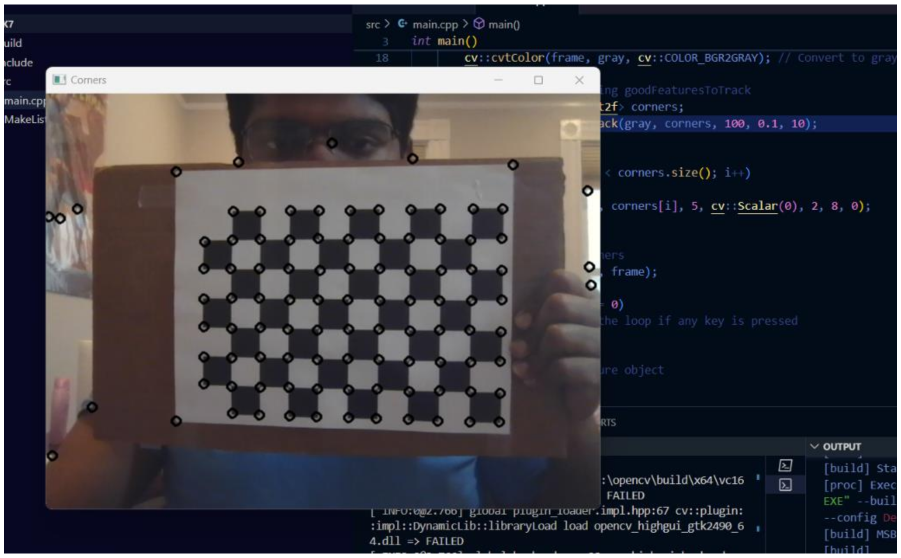
From the various images attached below, we can observe that the 3-D object changes in orientation as I move the pose of the chessboard(target)

## Task 7: Detect Robust Features

I have written a separate program for this task. I initially tried to use the Harris Corner detector but that proved to be more computationally expensive, so I used another OpenCV function that is based on Harris Corner but is more efficient and results in better output when compared to Harris Corner detector.

The program uses the goodFeaturesToTrack function to detect corners in the grayscale image. The function takes the grayscale image, an output vector to store the detected corners, the maximum number of corners to detect, the quality level (which is a parameter characterizing the minimal accepted quality of image corners), and the minimum possible Euclidean distance between the returned corners. The program then draws a circle around each detected corner on the original frame.

The results are attached below:

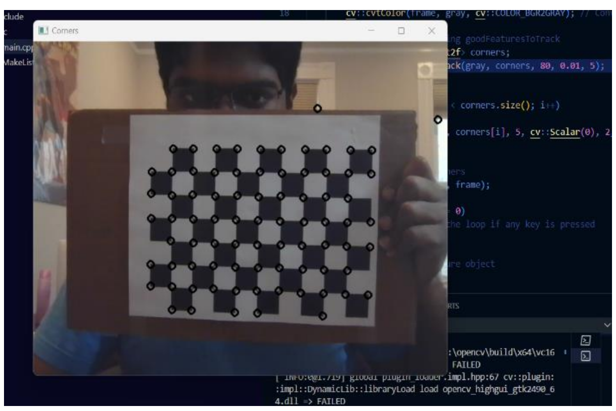


```
int main()
    cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY); // Convert to gray
ing goodFeaturesToTrack
t2f> corners;
ack(gray, corners, 100, 0.1, 10);

< corners.size(); i++)

, corners[i], 5, cv::Scalar(0), 2, 8, 0);

ners
, frame);

 0)
the loop if any key is pressed

ure object
```

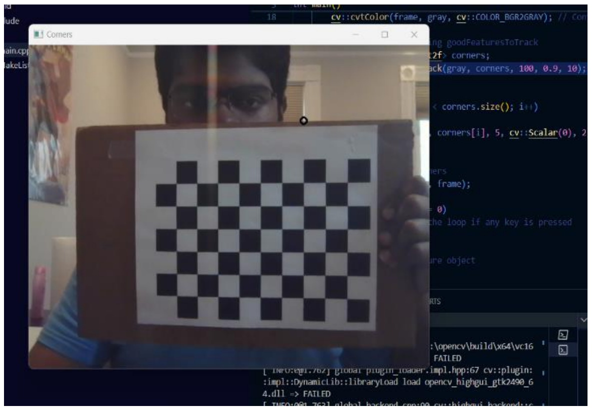
```
:\opencv\build\x64\vc16
 FAILED
[ INFO:0@2.766] global plugin_loader.impl.hpp:67 cv::plugin:
:impl::DynamicLib::libraryLoad load opencv_highgui_gtk2490_6
4.dll => FAILED
```

```
[build] Sta
[proc] Exec
EXE" --buil
--config De
[build] MSB
[build]
```

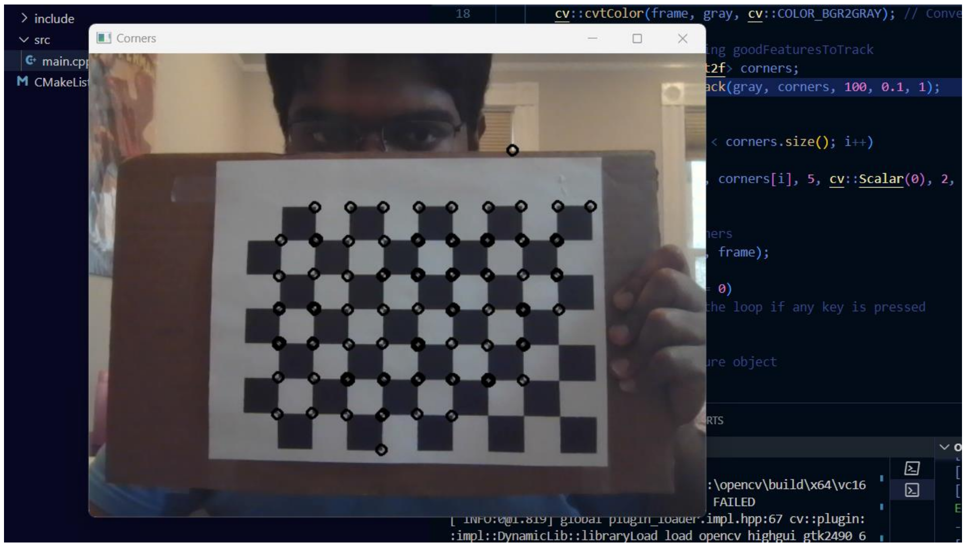
From the above image we can observe that all the corners of the chessboard are perfectly detected and marked with circles.
I changed the values of the function parameters like the number of circles , the distance between the circles and the quality of corners(threshold for sharp change in pixel values) and the results are attached below:



```
cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY); // Con
ng goodFeaturesToTrack
t2f> corners;
ack(gray, corners, 80, 0.01, 5);
< corners.size(); i++)
corners[i], 5, cv::Scalar(0), 2
ers
frame);
0)
the loop if any key is pressed
re object
:\opencv\build\x64\vc16
FAILED
[ INFO:0@1.719] global plugin_loader.impl.hpp:67 cv::plugin:
:impl::DynamicLib::libraryLoad load opencv_highgui_gtk2490_6
4.dll => FAILED
```



```
cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY); // Con
ng goodFeaturesToTrack
t2f> corners;
ack(gray, corners, 100, 0.9, 10);
< corners.size(); i++)
corners[i], 5, cv::Scalar(0), 2
ers
frame);
0)
the loop if any key is pressed
re object
:\opencv\build\x64\vc16
FAILED
[ INFO:0@1.762] global plugin_loader.impl.hpp:67 cv::plugin:
:impl::DynamicLib::libraryLoad load opencv_highgui_gtk2490_6
4.dll => FAILED
```



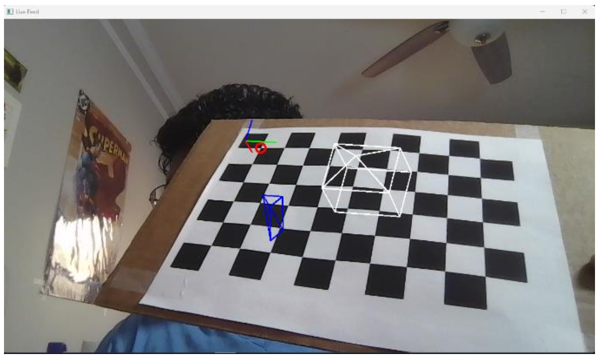
```
cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY); // Conve
ng goodFeaturesToTrack
t2f> corners;
ack(gray, corners, 100, 0.1, 1);
< corners.size(); i++)
corners[i], 5, cv::Scalar(0), 2,
ers
frame);
0)
the loop if any key is pressed
re object
:\opencv\build\x64\vc16
FAILED
[ INFO:0@1.819] global plugin_loader.impl.hpp:67 cv::plugin:
:impl::DynamicLib::libraryLoad load opencv_highgui_gtk2490_6
```

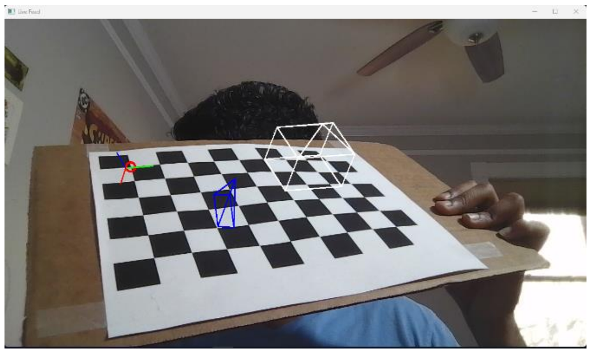
These features are invariant to rotation, scale, and illumination changes, and they provide a rich description of the local image structure. In Augmented Reality, these feature points will be helpful in perfecting the position and orientation of the projected points in the world frame. These feature points can be used to align a virtual object with a specific location in the real world. And these features can be used to track the position and orientation of the camera relative to a known object or environment. By tracking how the feature points move across frames, the system can estimate the camera's movement and adjust the position and orientation of the virtual objects accordingly.

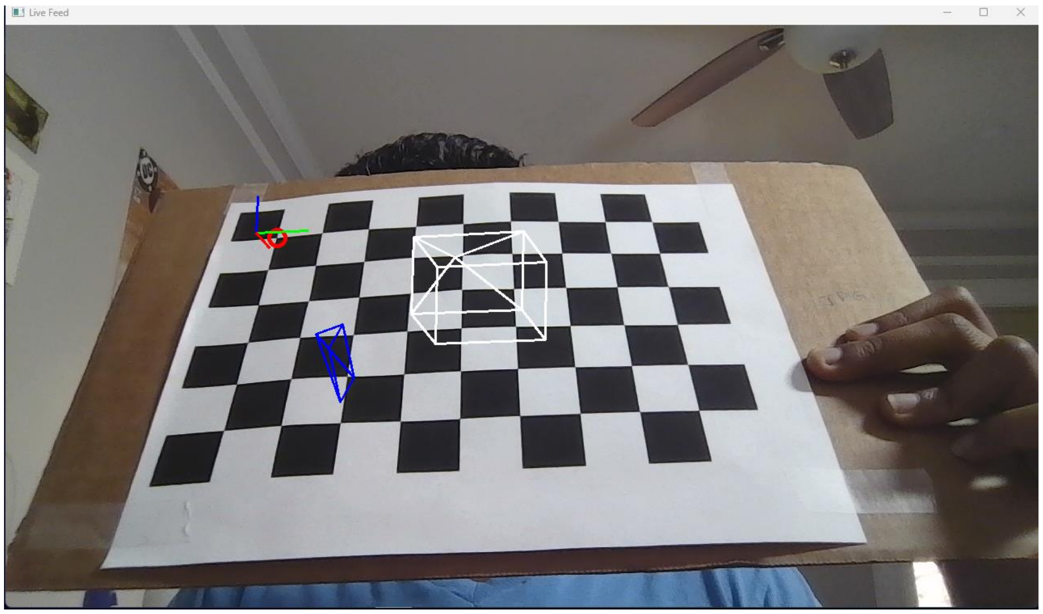
## Extensions:

### Extension 1: Project multiple 3-D objects.

I extended the project by adding another 3-D object to be projected onto the target plane. The results are attached below:



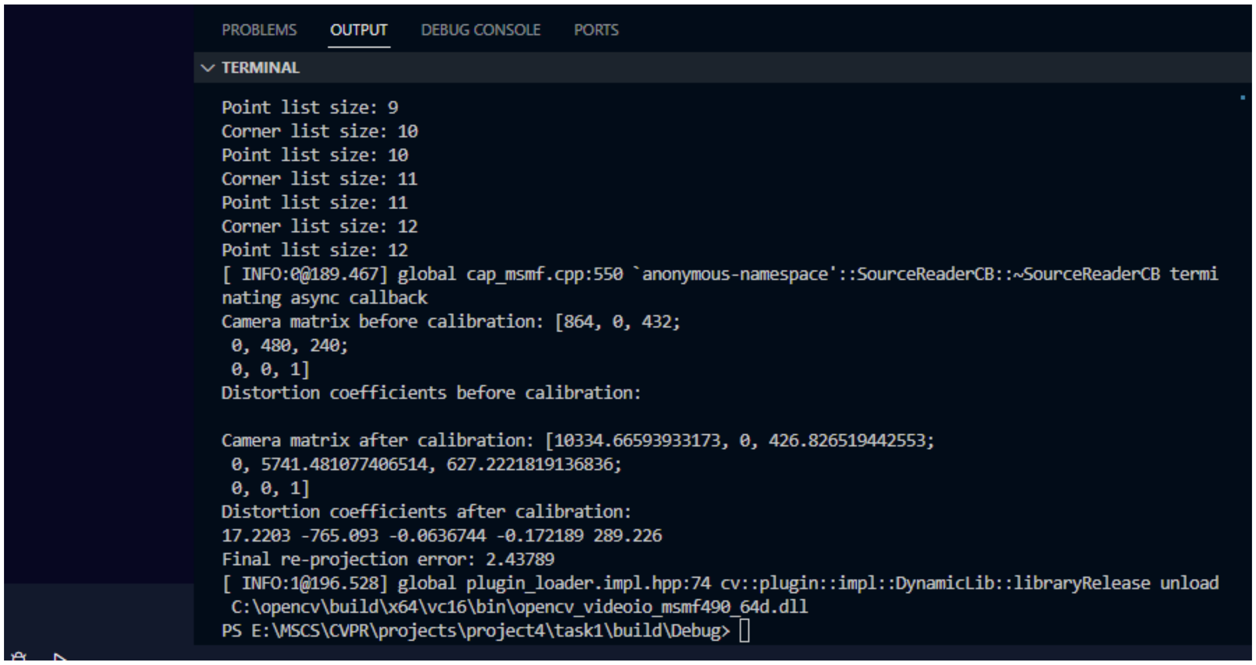




Both the objects change their pose as I change the target’s orientation and position.

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

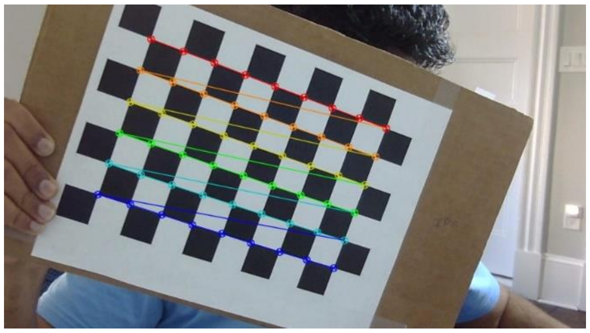
Apart from the integrated webcam, I also performed the entire project with an external webcam that had a better resolution. The calibration results are attached below:

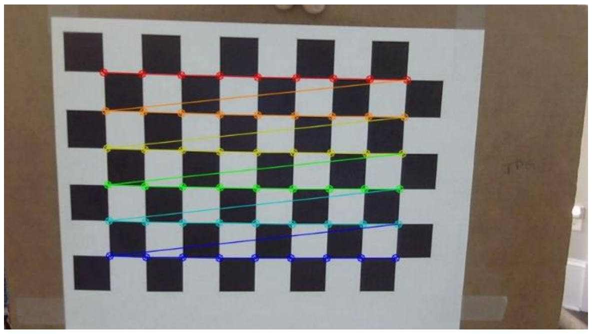


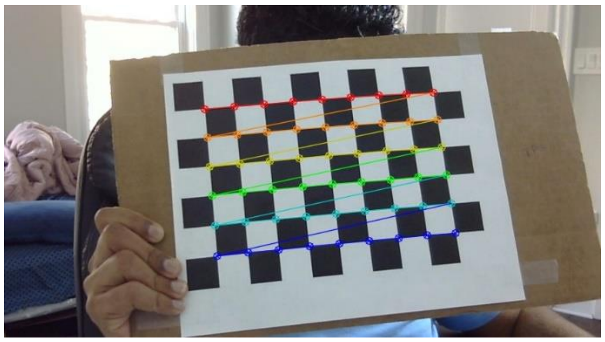
```
Point list size: 9
Corner list size: 10
Point list size: 10
Corner list size: 11
Point list size: 11
Corner list size: 12
Point list size: 12
[ INFO:0@189.467] global cap_msmf.cpp:550 `anonymous-namespace'::SourceReaderCB::~SourceReaderCB termi
nating async callback
Camera matrix before calibration: [864, 0, 432;
 0, 480, 240;
 0, 0, 1]
Distortion coefficients before calibration:

Camera matrix after calibration: [10334.66593933173, 0, 426.826519442553;
 0, 5741.481077406514, 627.2221819136836;
 0, 0, 1]
Distortion coefficients after calibration:
17.2203 -765.093 -0.0636744 -0.172189 289.226
Final re-projection error: 2.43789
[ INFO:1@196.528] global plugin_loader.impl.hpp:74 cv::plugin::impl::DynamicLib::libraryRelease unload
 C:\opencv\build\x64\vc16\bin\opencv_videoio_msmf490_64d.dll
PS E:\MSCS\CVPR\projects\project4\task1\build\Debug>
```

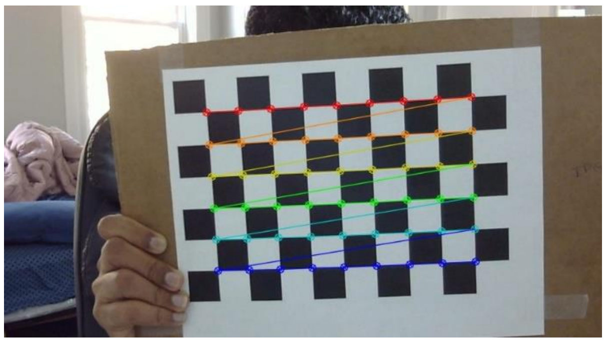
A total of 13 images were used for calibration and the re-projection error is 2.4 which is only 0.4 higher than the integrated webcam.

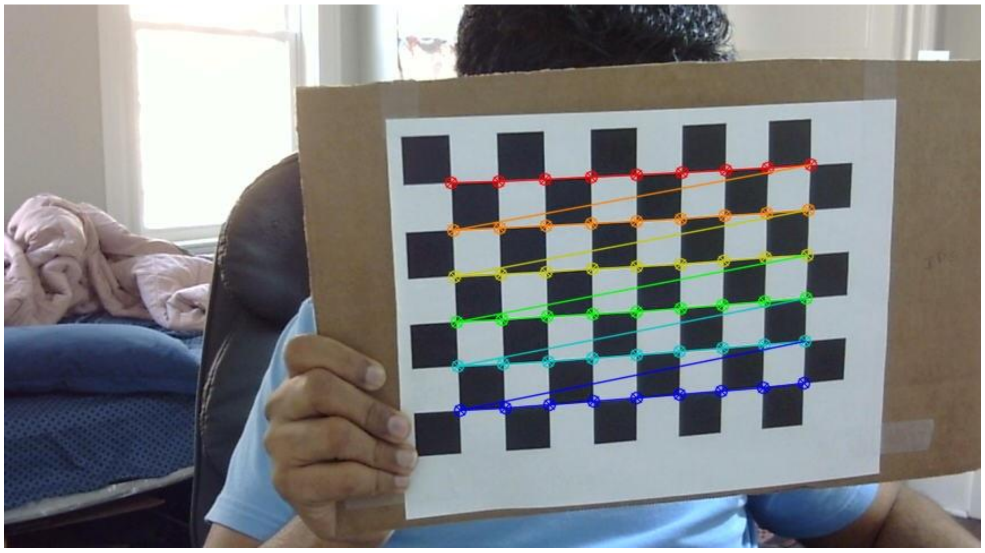
Some of the calibration images are attached below:



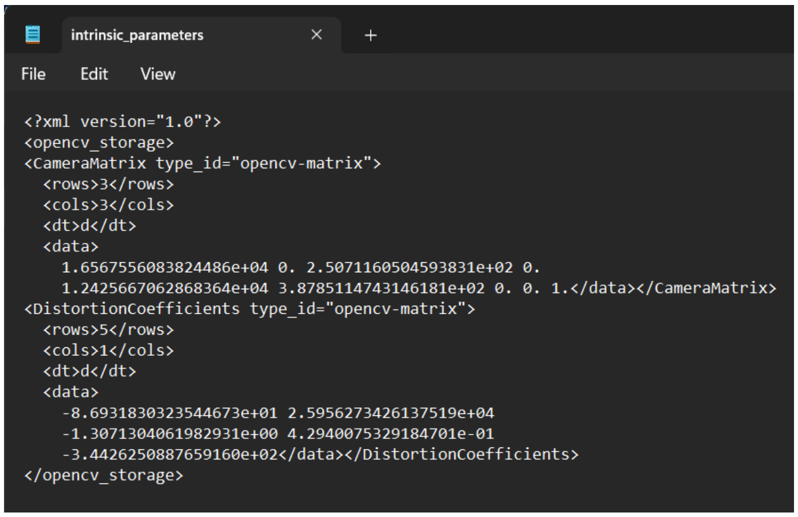






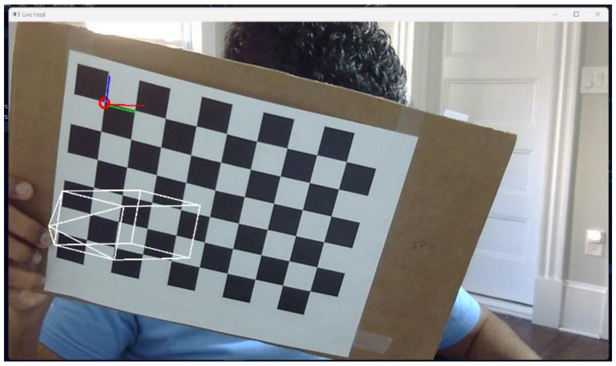
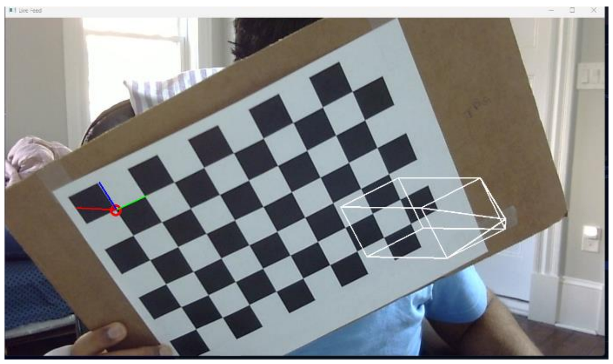
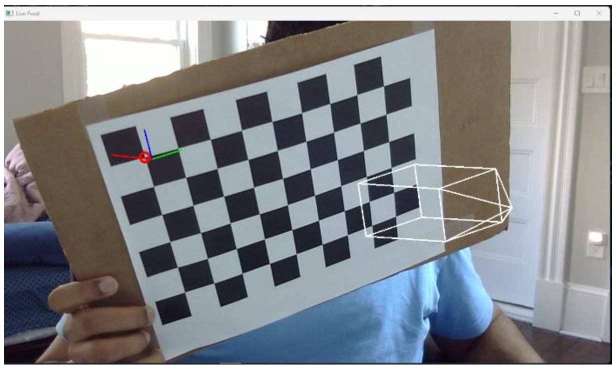
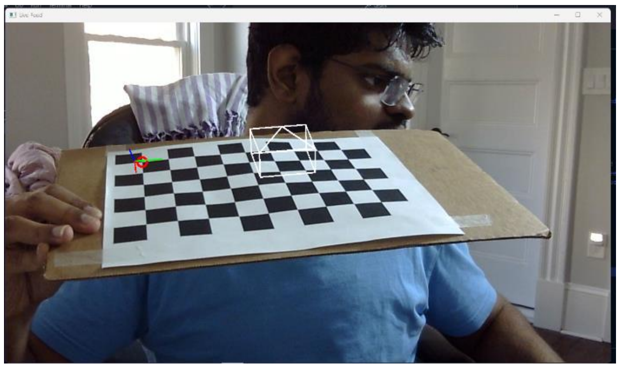
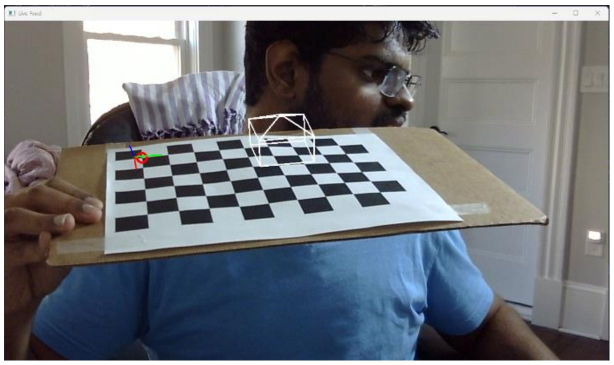
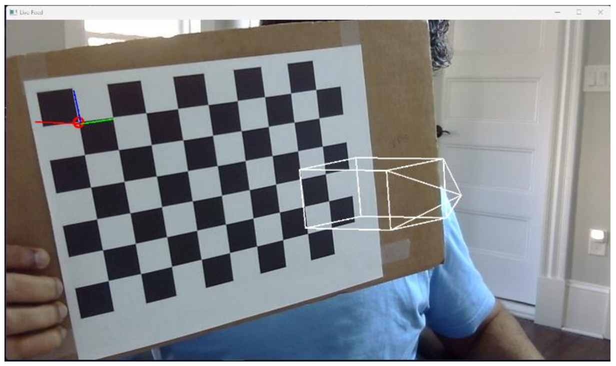


The intrinsic parameter is attached below:



```
<?xml version="1.0"?>
<opencv_storage>
<CameraMatrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    1.6567556083824486e+04 0. 2.5071160504593831e+02 0.
    1.2425667062868364e+04 3.8785114743146181e+02 0. 0. 1.</data></CameraMatrix>
<DistortionCoefficients type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    -8.6931830323544673e+01 2.5956273426137519e+04
    -1.3071304061982931e+00 4.2940075329184701e-01
    -3.4426250887659160e+02</data></DistortionCoefficients>
</opencv_storage>
```

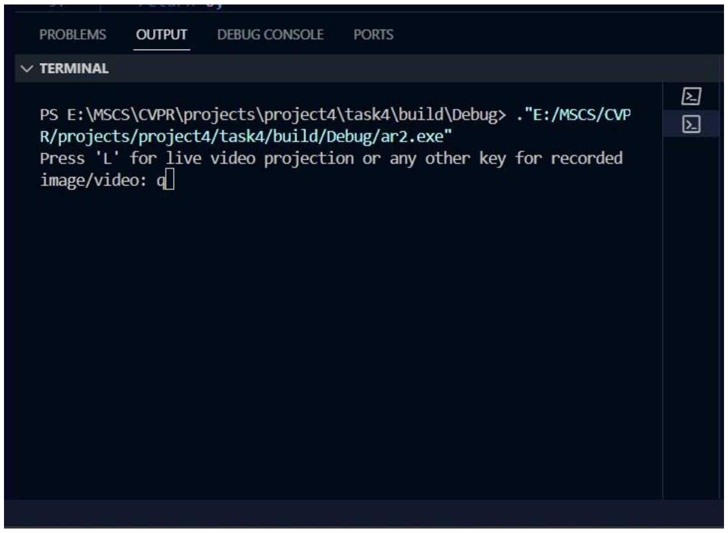
The projection of 3-D points using the external webcam is attached below:

In terms of quality both the integrated and external webcam performed in similar ways.

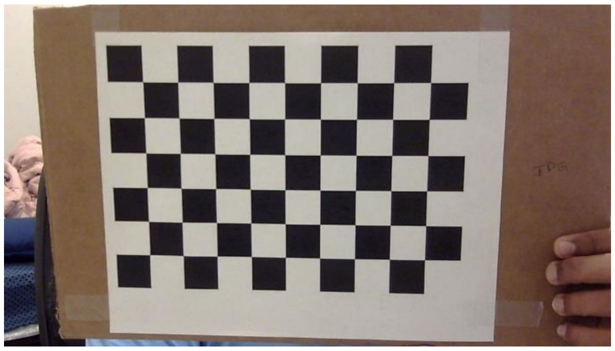
## <u>Extension 3: Enable your system to use static images or pre-captured video sequences with targets and demonstrate inserting virtual objects into the scenes.</u>

For this task I have reconfigured the AR system to take input from the user when the program begins. If the user presses 'L' or 'l' key the program goes into live video mode where the webcam will provide the input and the objects will be projected to the target frame in real time. If the user presses any other key, the program will fetch the image/video file mentioned in the program and detects for target and projects points onto it:

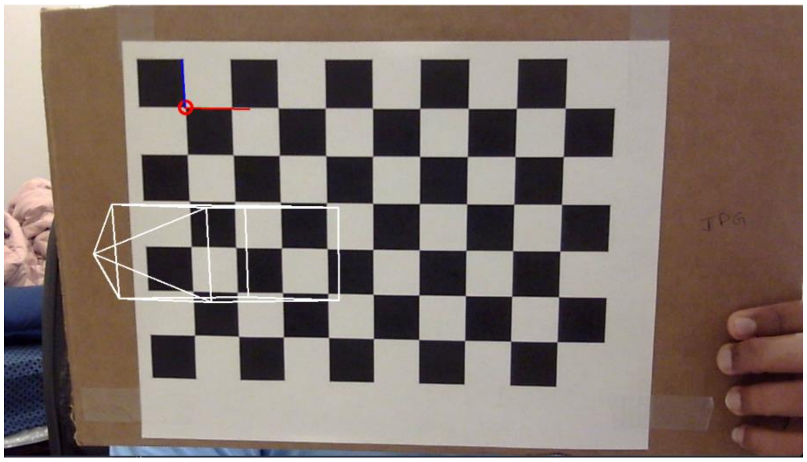


```
PS E:\MSCS\CVPR\projects\project4\task4\build\Debug> ."E:/MSCS/CVP
R/projects/project4/task4/build/Debug/ar2.exe"
Press 'L' for live video projection or any other key for recorded
image/video: q
```

As I press the q key, first an image saved in my laptop will be read and 3-d points will be projected. The input image:

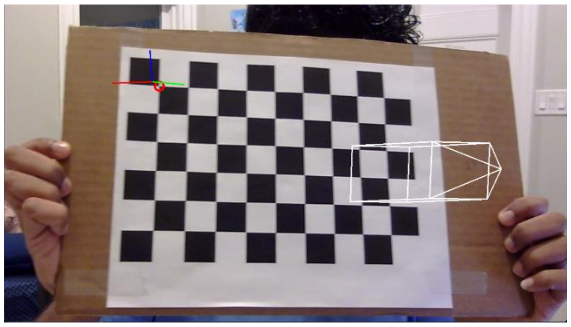


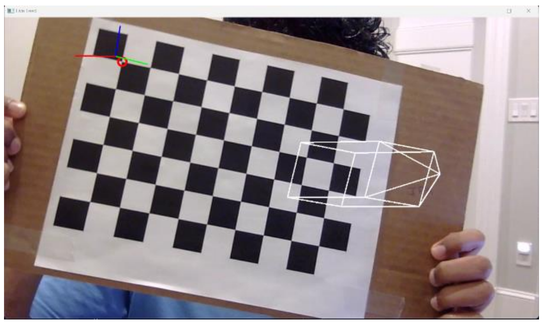
The result is attached below:



Secondly, I provided a pre-recorded video file as input. The link to input video is given below.
Input video link :
https://drive.google.com/file/d/152pqkfbP15JCdzcryF2z5wLwa0HN2klN/view?usp=sharing

The program was successfully able to detect target and project points in video as well. Screenshot of the results are attached below:





The link to output video recording is:
https://drive.google.com/file/d/14zSZohNGG9eO_9lT9KnLxvUXdFAtyNtu/view?usp=sharing

## Reflection:

This project was interesting to work with. Though I thought camera calibration would be easy, it did not prove to be easy. Irrespective of how many different tries and different set of images, I can never bring the re-projection error less than 0.15. And similarly in projecting points, I couldn't fix the tilted 3-D axis but still was able to project the 3-D figures. From this project I have learnt how to obtain calibration metrics using chessboard target and project 3-D points to a 2-D plane. I further plan to learn how to use the features obtained using Harris corners and project 3-D points using those features as reference.

## Acknowledgements:

I used the following websites and the official OpenCV documentation website as references for completing this project:

- https://theailearner.com/tag/cv2-goodfeaturestotrack/
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga1019495a2c8d1743ed5cc23fa0daff8c
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d
- https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a