CS5330: Project 4: Calibration and Augmented Reality

Created by Sida Zhang, last modified on Mar 27, 2022

@Sida Zhang and @Hongyu Wan

March 20, 2022

Introduction

This project allow users to calibrate a camera and use the calibration to general virtual objects on checkerboard and ArUco board. The program can detect a target, or a checkerboard and then place several different virtual objects in the scene relative to the checkerboard that moves and orients itself.

> Our operating system is Windows11 and the IDE is Visual Studio Code.
>
> Please feel free to post any comment or suggestion on this page, and thank you for viewing our Wiki Page 👍.

Over all thinking

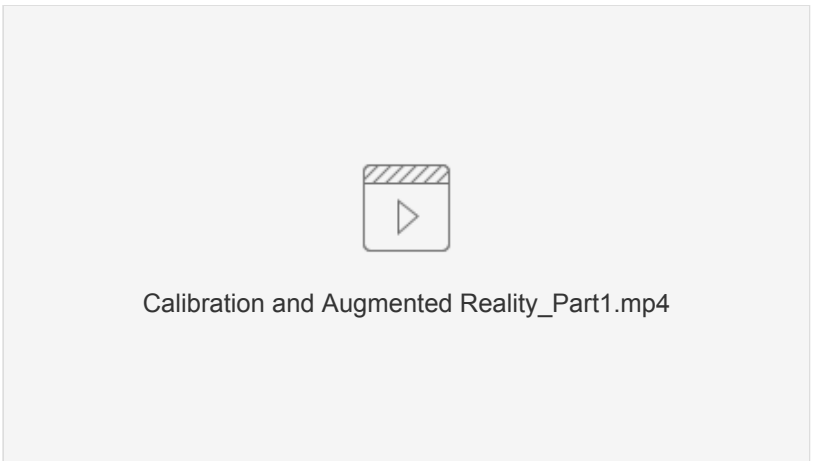
I think the biggest challenge we had for this project is that we couldn't find a way to draw complexed objects at first because we couldn't parse data from obj files. We also tried to use OpenGL for the first extension but we got stuck on OpenGL configuration for many days. We will continue work on OpenGL for this project in the future and hopefully we will have more cool stuff coming up.

> Overall, the tasks are pretty straightforward and I will now demonstrate each of our tasks in below.
>
> Please feel free to watch the video demonstrations (Two Parts and Two Videos) on all the tasks.
>
> Enjoy our project and please feel free to leave a comment or a like 😃.

Tasks and functionalities:

livevid_I.cpp:

Video Demonstration Part I

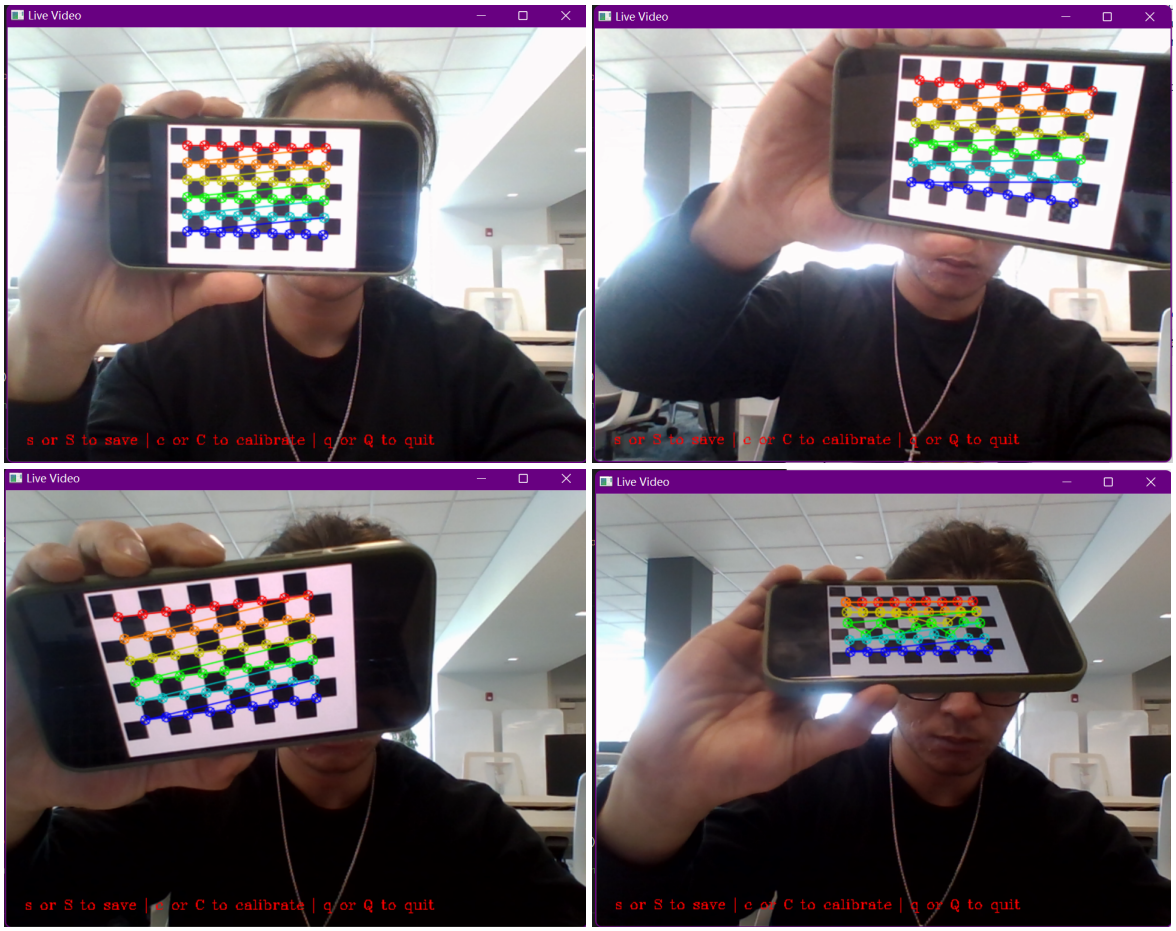


Task 1: Detect and Extract Chessboard Corners

For the first task, we spent a lot of time on reading the calibration documentation and the github code. I thought the documentations are really helpful for understanding the martials and OpenCV built-in functions.

We have also encountered an issue with the responding speed. I figured it is because of the quality of the camera, nicer camera tends to have slower results. We solved this issue by adding another parameter when using cv::findChessboardCorners. By adding CALIB_CB flags provide much faster speed when searching for chessboard. (**Very helpful**)

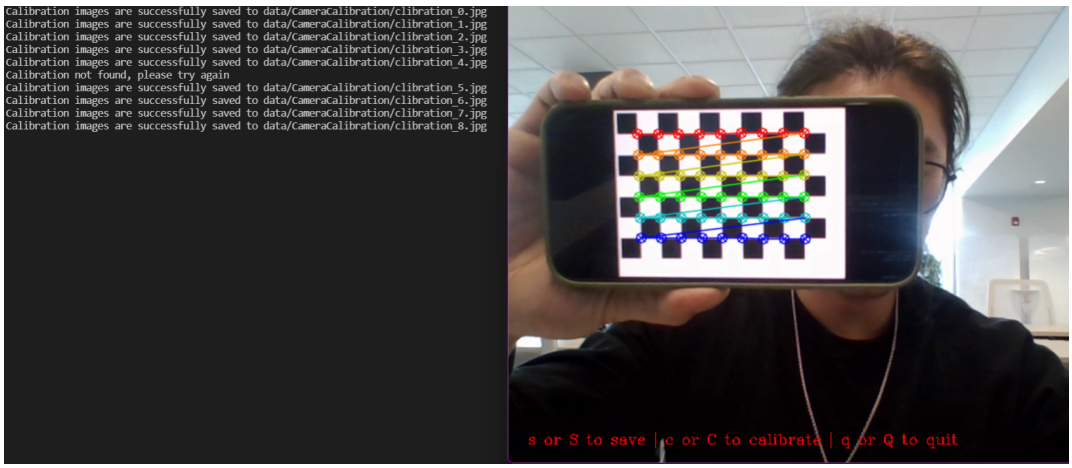
Here are some results for this task from different angles:

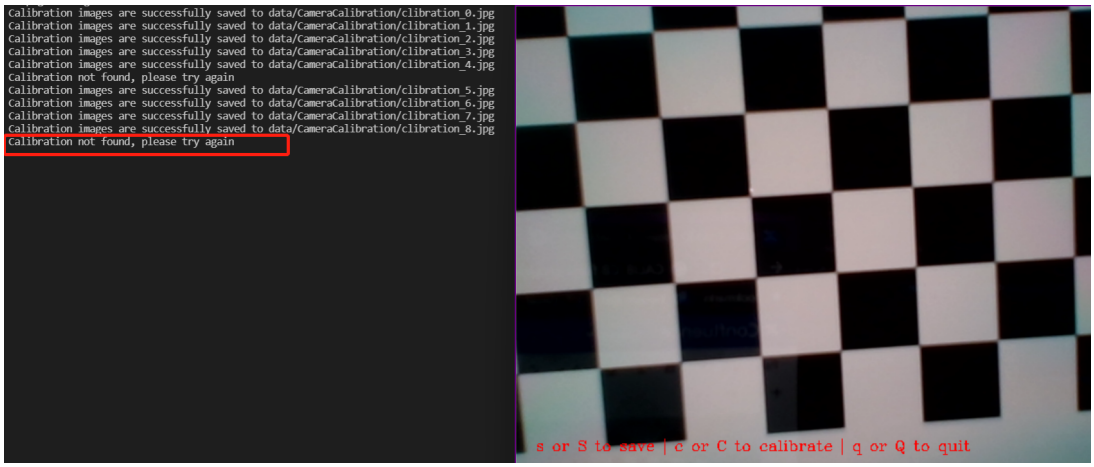


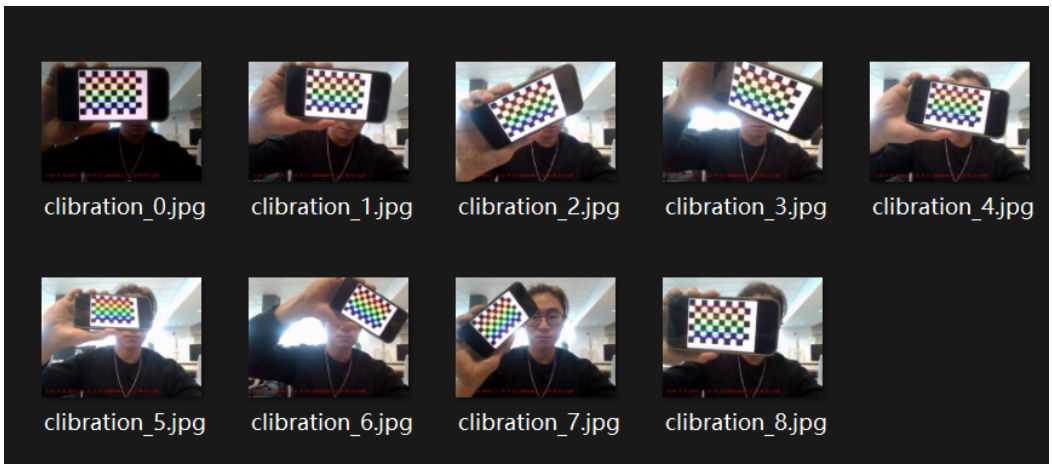
Task 2: Select Calibration Images

For this task, this program allow user to press "s" or "S" to save and store the current calibration and the vector of corners by findChessbordCorners into the corner_list and specifies 3D position of corners in world coordinates and pushed the data into the point_list.

Here are some results showing calibration saved successfully and when chessboard is not found:







Task 3: Calibrate the Camera

For the last task in the first program, we implemented a function to run calibration and prints the pixel errors after each calibration. The function also returns the camera matrix and the distortion coefficients and save them to data/camera_matrix.txt and data/distoration_coefficients.txt. We have used the flag CV_CALIB_FIX_ASPECT_RATIO for this task.

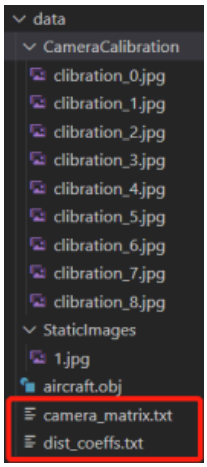
In order to run Calibration, at least five frames are required.

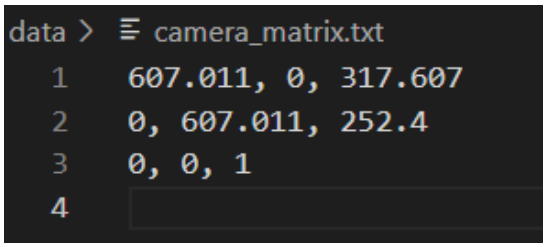
```
std::cout << "At least 5 calibration frames is required for this task." << std::endl;
std::cout << "You currently only have " + std::to_string(num) + " images saved." << std::endl;
```

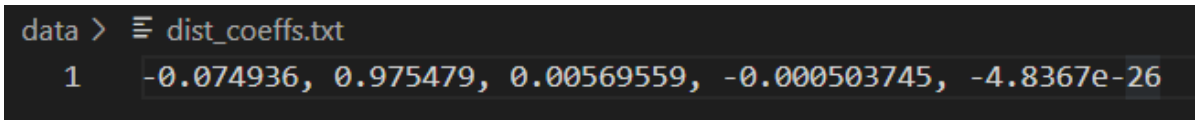
```
Calibration images are successfully saved to data/CameraCalibration/clibration_0.jpg
Calibration not found, please try again
Calibration images are successfully saved to data/CameraCalibration/clibration_1.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_2.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_3.jpg
At least 5 calibration frames is required for this task.
You currently only have 4 images saved.
```

Here are some results from calibration.

```
Writting data to files...
Finished writing data to data/camera_matrix.txt and data/dist_coeffs.txt
Camera Matrix:
607.011 0       317.607
0       607.011 252.4
0       0       1
Distortion Coefficients:
-0.074936       0.975479        0.00569559      -0.000503745    -4.8367e-26
0.0970347 Re-Projection Error found
Writting data to files...
Finished writing data to data/camera_matrix.txt and data/dist_coeffs.txt
```



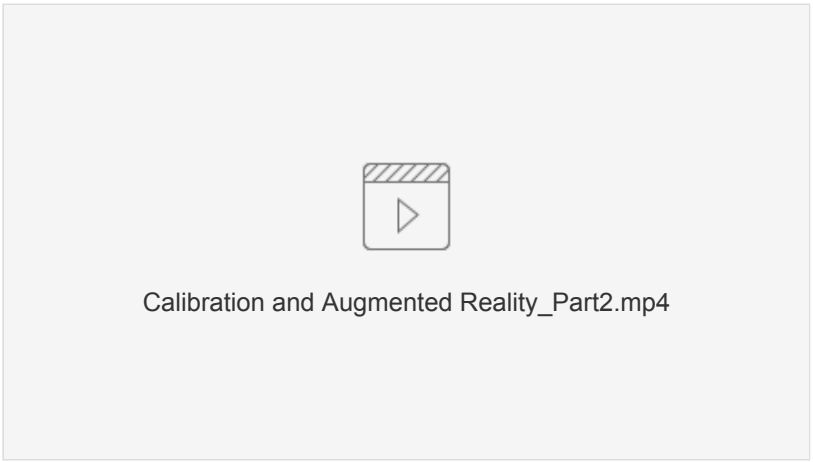




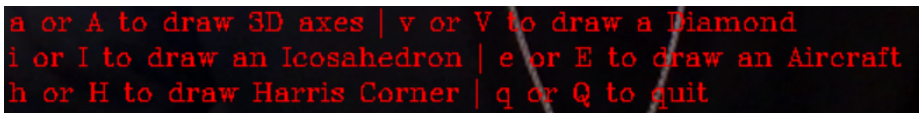
Now we have enough information to run AR system. Please check out our video demonstration for more information!

livevid_II.cpp:

Video Demonstration Part II



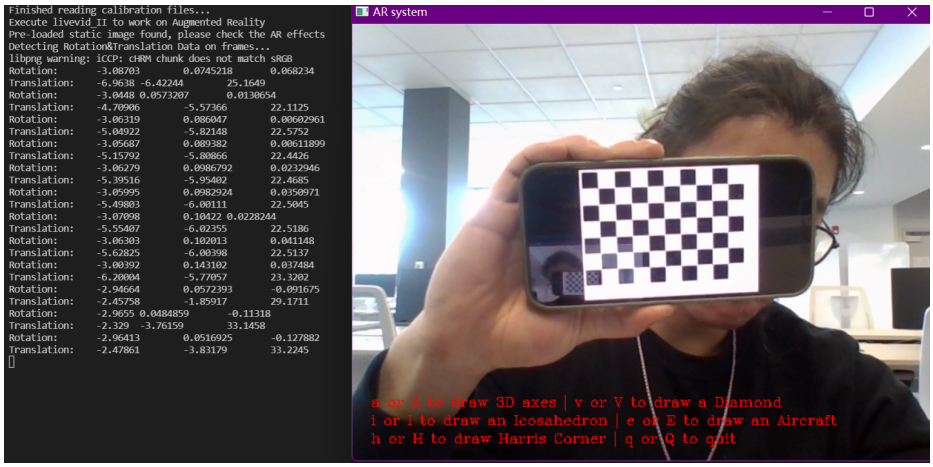
The second program has a little nice caption for different tasks.



Task 4: Calculate Current Position of the Camera

Now we will run our second program, this task would read the camera calibration parameters then detect the chessboard. If chessboard is found, it will print out the checkerboard's poses, rotation matrix and translation matrix in real-time.

Because it finds chessboard easily, we have decided to print the poses in every 25 frames



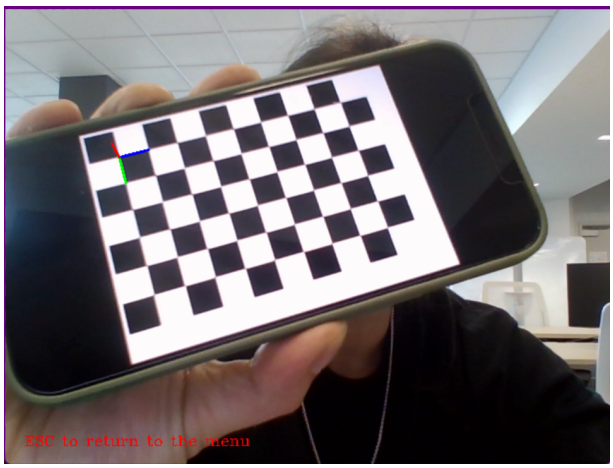
Task 5: Project Outside Corners or 3D Axes

For this task, we used cv::projectPoints to project the 3D axes corresponding to the outside comers of the chessboard onto the image plane as the chessboard moves around.

Press "a" or "A" to display 3D axes.

Press "ESC" to return to the menu.

See result below in top left of the checkerboard.

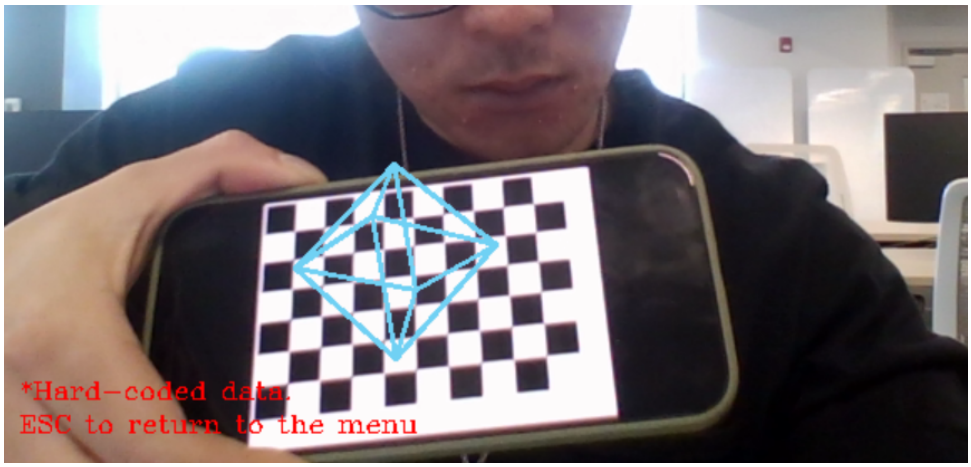


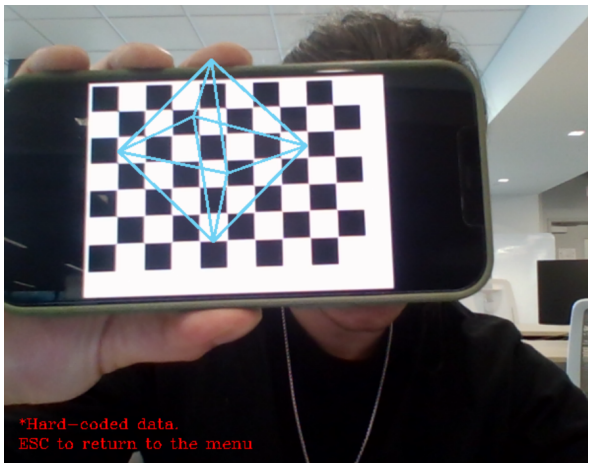
Task 6: Create a Virtual Object

For this task, we implemented two hard coded objects and one .obj file rendering.

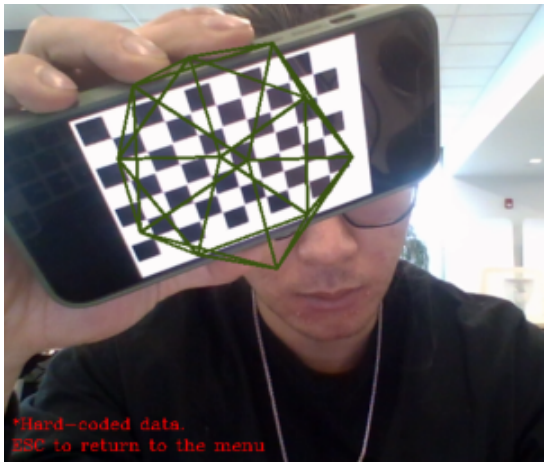
All of the obj files are found from here. We have spent lot of time on trying to read obj files and we ended up using this method from OpenGL tutorial.

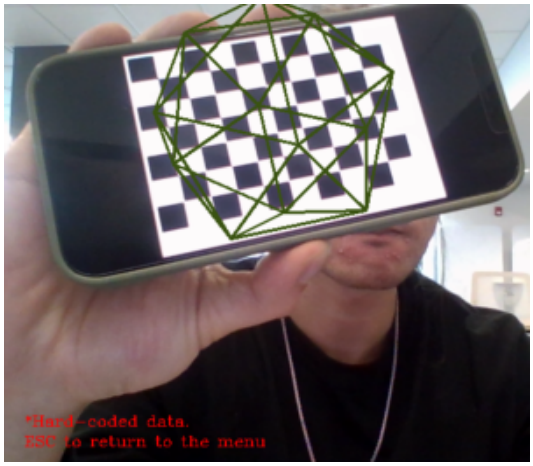
Press "v" or "V" to draw a tiffany blue diamond.





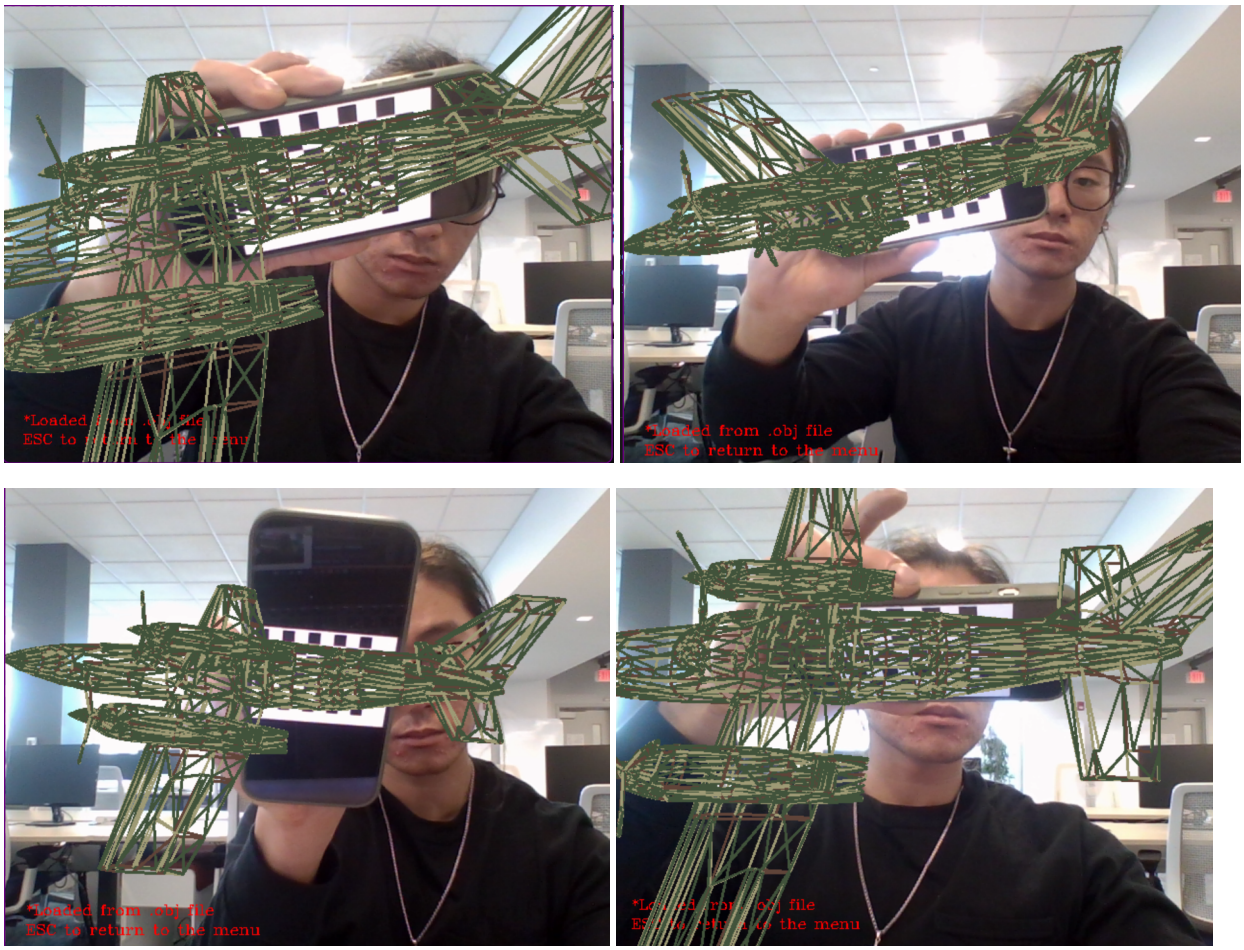
Press "i" or "I" to draw an 20 faces Icosahedron.





Press "e" or "E" to draw a camouflage aircraft! Very cool one!

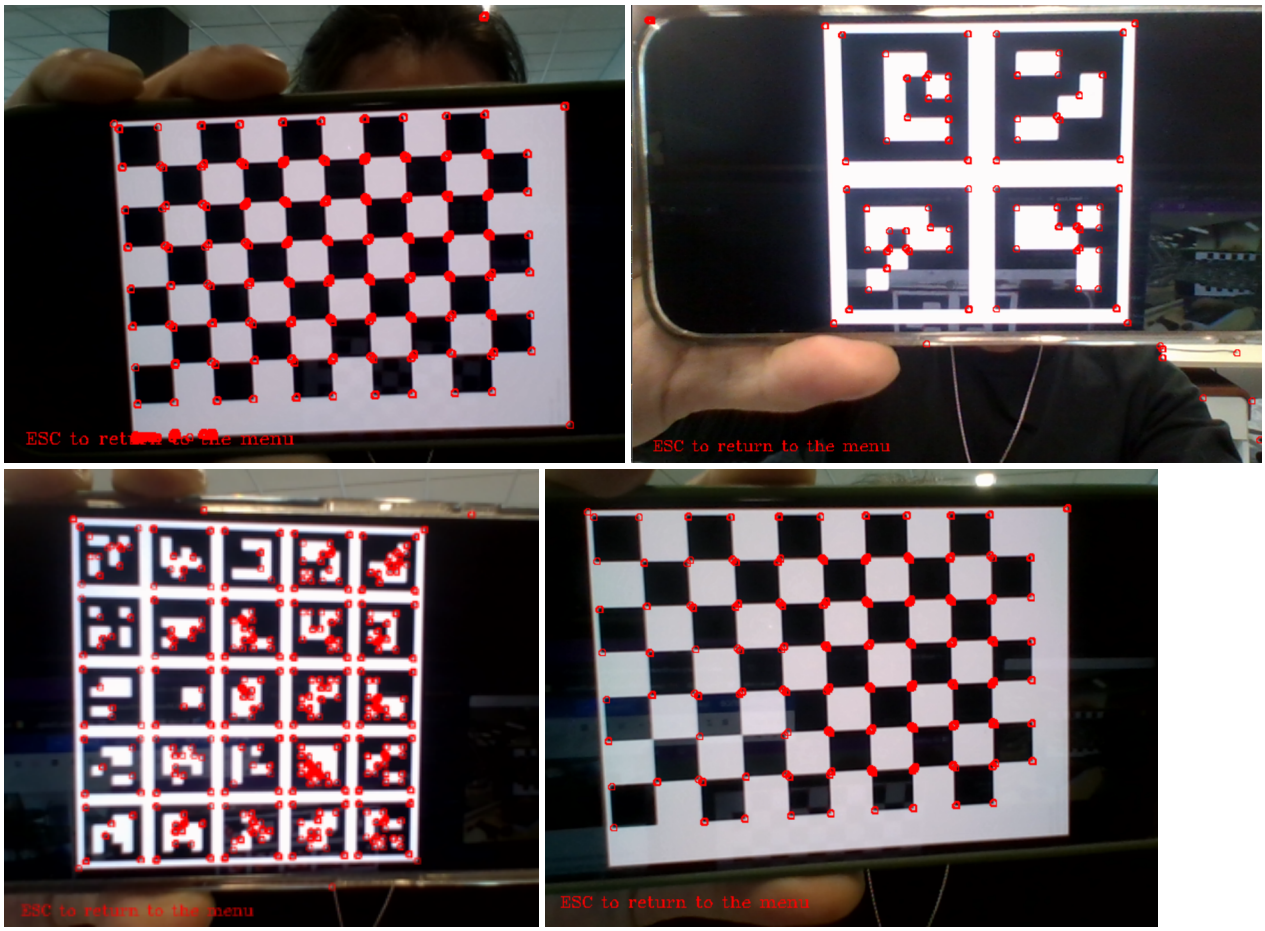
The color I used for this one is: cv::Scalar(57, 68, 96)/ cv::Scalar(117, 154, 158)/ and cv::Scalar(59, 83, 65)



Task 7: Detect Robust Features

For the very last task, we used Harris Corners feature that shows where the features are in a video stream. Seems like Harris Corner allows us to do more corner detection and use the point locations to associate with the world coordinates.

Overall, I think it's a pretty classic modeling process. It finds some special points to match the edge segments then characterizes the structure of the image, and represent the local features of the image.



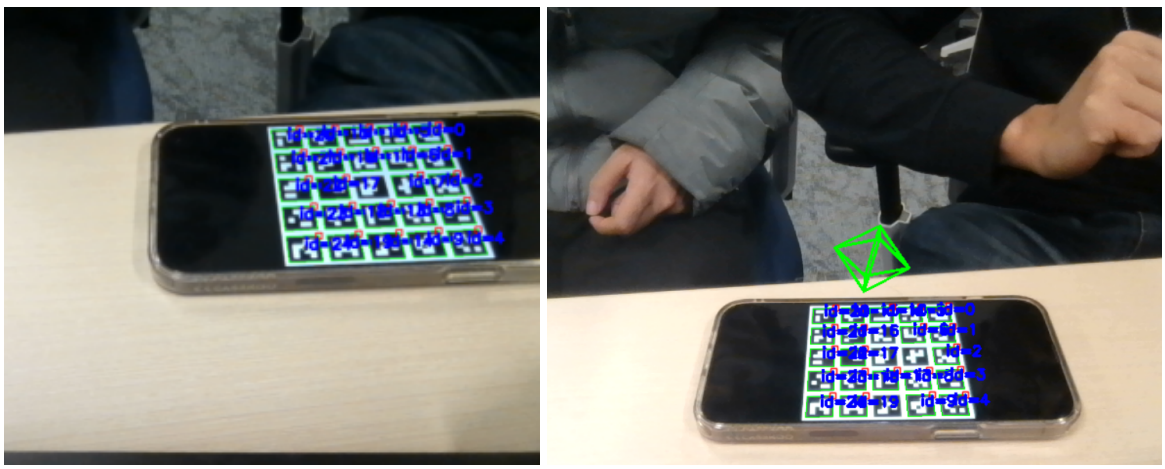
Extentions

Extension 1: ArUco functionatlities:

We have implemented the ArUco into the project in livevid_III.cpp.

This extension does different functionalities that you can do in OpenCV including detecting chessboard corner and calibration and draw a Diamond on the board.

As you can observe from below image, no mater how yo rotate the target, the ArUco maker alwyas have 25 ids.



Extension 2: Different cameras and comparsions:

For this extension, we have used different cameras to do calibrations on images and looks like they have different performance.

Without the CALIB_CB_FAST_CHECK flag, both of the performance was really bad because the quality of our cameras are kind high.

Results shown in below:

Setups: 5 required frames for calibrations on two different cameras

Camera I (720p HD) :

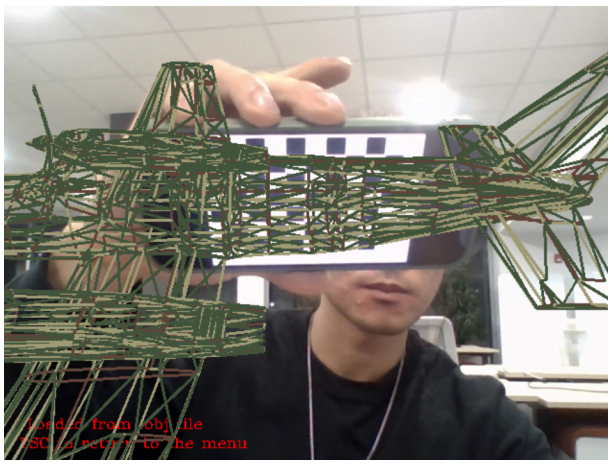
```
Calibration images are successfully saved to data/CameraCalibration/clibration_1.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_2.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_3.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_4.jpg
Camera Matrix:
604.176 0       318.633
0       604.176 247.015
0       0       1
Distortion Coefficients:
-0.0264416      0.510405        0.00200513      -0.000133545    -4.8367e-26
0.103262 Re-Projection Error found
Writting data to files...
Finished writing data to data/camera_matrix.txt and data/dist_coeffs.txt
```

Camera II (1080P HD):

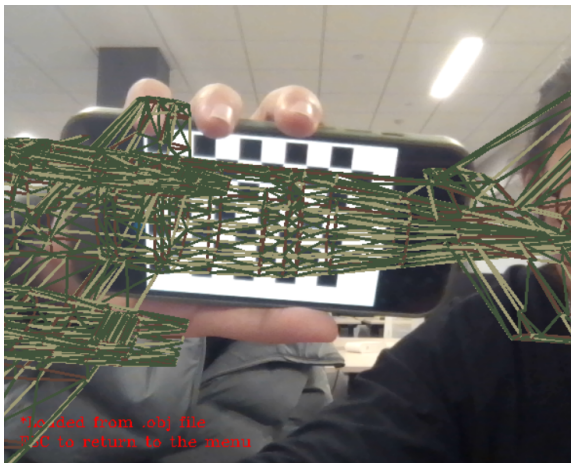
```
Calibration images are successfully saved to data/CameraCalibration/clibration_0.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_1.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_2.jpg
Calibration images are successfully saved to data/CameraCalibration/clibration_3.jpg
Calibration not found, please try again
Calibration images are successfully saved to data/CameraCalibration/clibration_4.jpg
Camera Matrix:
645.738 0       311.946
0       645.738 248.372
0       0       1
Distortion Coefficients:
-0.110172       0.509505        -0.0028508      -0.000872318    -4.8367e-26
0.128479 Re-Projection Error found
Writting data to files...
Finished writing data to data/camera_matrix.txt and data/dist_coeffs.txt
```

Result difference: Both of the camera prints about ~0.11 re projection error. I think they worked pretty the same because the quality of our cameras are similar.

Camera I:



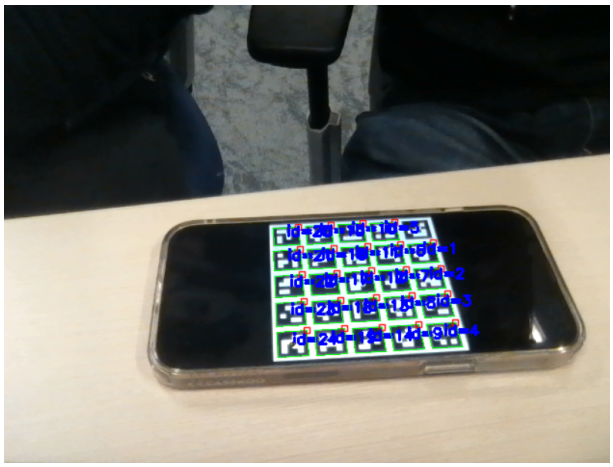
Camera II:



Camera I shows complexed virtual object pretty smooth because camera I has lower resolution. However, camera II runs complex object much slower I think because it tries too hard to find checkerboard.

Extension 3: AR system for other target, or board:

We used ArUco to detect different board. We used ArUco board and it efficently printed out the world coordinate (CHArUco Markers).

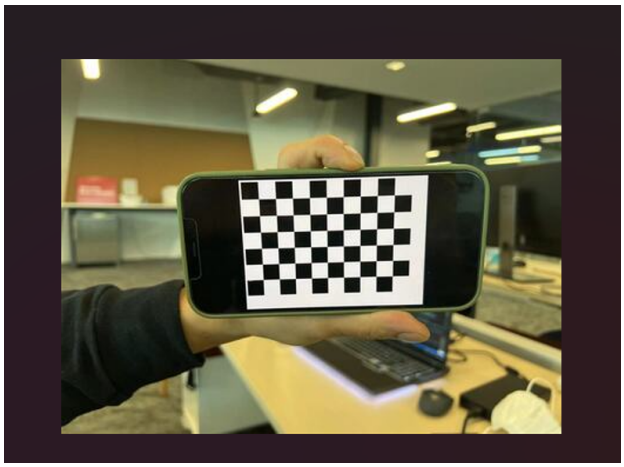


Extension 4: Allow Static Image to apply AR effects:

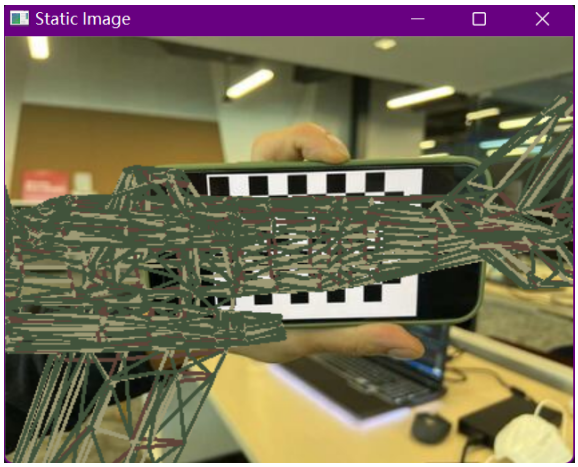
For this extension, if user runs the second program, livevid_II.cpp, it will automatically check if there is any picture in data/StaticImages folder. If yes, it will run cv::findChessboardCorners to find corners; if a chessboard is found, it will show the after effect in a new window.

```
if( (dp = readdir(dir_path)) == NULL ){
  std::cout << "No pre-loaded static image found in data/StaticImages" <<  std::endl;
} else {
  std::cout << "Pre-loaded static image found, please check the AR effects" << std::endl;
}
```

Original:

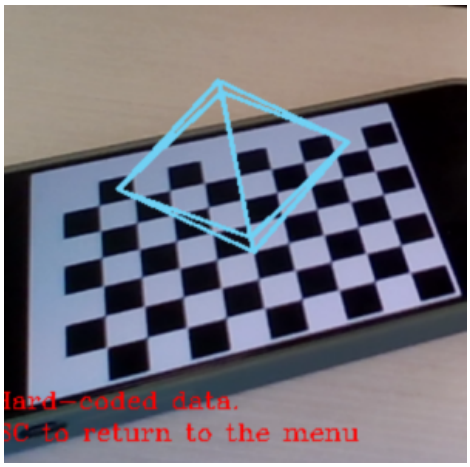


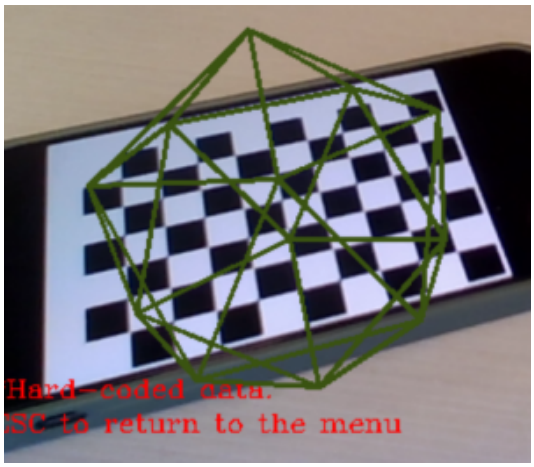
After Effect:

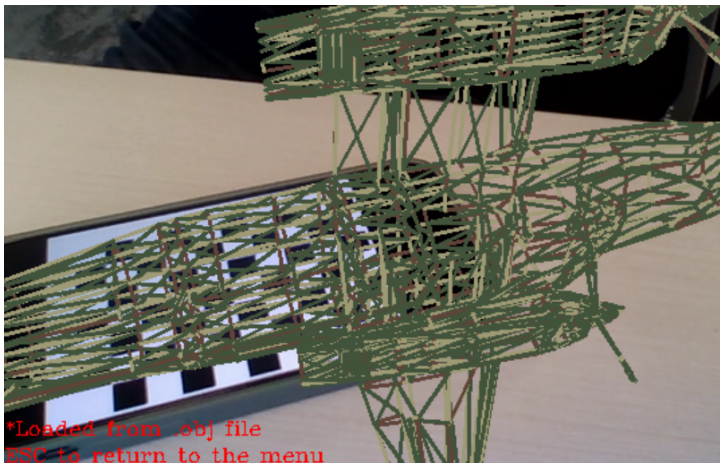


Extension 5: Multiple Objects:

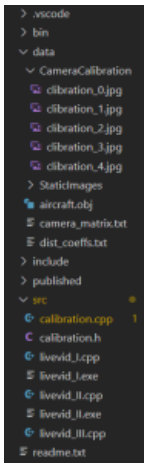
We have down many different objects, obj files, for this project in the /data folder. The experience was quite interesting to see all kinds of different objects to orient and move around.







Project Structure:



Project Reflection:

I think most of class martials were very complicated but this assignment definitely helps on understanding the concepts. I believe this project is my favorite project so far because I really enjoyed playing around with the virtual objects. The tasks are not too hard to understand but OpenGL was definitely something I need to spend more time on. I believe with OpenGL we could have done many more cooler effects to Augmented Reality objects. I also think it is really cool to use the idea of world coordinates and apply them on a single surface.

Overall, I think this project is a great learning process in how AR works because before this lecture I had no clue how to create virtual objects from scratch.

Acknowledgement:

Thanks to Dr. Maxwell and all the posts and discussions on Piazza.

https://github.com/opencv/opencv/blob/4.x/samples/cpp/tutorial_code/calib3d/camera_calibration/camera_calibration.cpp

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

https://people.sc.fsu.edu/~jburkardt/data/obj/cessna.obj

https://people.sc.fsu.edu/~jburkardt/data/obj/minicooper.obj

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

https://github.com/el1ven/NTU-OpenGL1/tree/master/obj

https://github.com/maijiaquan/blog-code/blob/master/opengl-loadObj/main.cpp

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

https://github.com/g-truc/glm

No labels