Project 4 Report: Calibration and Augmented Reality

Name: Yao Zhong, zhong.yao@northeastern.edu

This project is about camera calibration and virtual object generation. By using a chessboard pattern, I extracted features (the corners) from calibration images and then used those images for the camera calibration. With the information about the intrinsic parameters of the camera, I was able to estimate the camera position when given a chessboard pattern. Using the camera position(rvec and tvec), I was able to further reproject virtual objects to the pattern. As an extension, I also tried to use robust features (SURF) as the feature and place virtual objects on the target pattern.

Task1: Detect and Extract Chessboard Corners

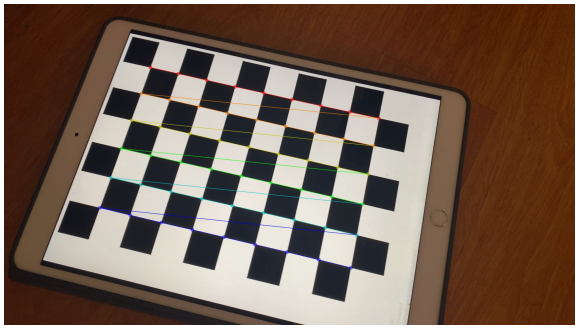
In this task, the program was able to use findChessboardCorners to find the corners of the chessboard pattern. When the corners are found, the cornerSubPix() function was used to refine the locations of the corners. Finally, through drawChessboardCorners(), the program is able to mark out the corners they found.

Task2: Select Calibration Images

In this task, the program enabled the users to specify a particular image that should be used for the calibration by pressing the ‘s’. The program is able to save both the calibration image and the cornerSet.

For the 3-D object points, the program adopted the units of chessboard squares. The upper left corner is (0,0,0), the first point on the next row is (0,-1,0), so on.

Here are two examples of chessboard corners found:



Task3: Calibrate the Camera

In this task, when the user have selected enough calibration frames(5 or more), a calibration will run automatically, and the result (cameraMatrix and distCoeffs) will be saved in an XML file as the intrinsic parameter of the camera.

Here is the result of this step:

Initial Value			After Calibration		
1	0	960	1379.53	0	968.401
0	1	540	0	1379.53	529.272
0	0	1	0	0	1

Camera Matrix

Initial Value	After Calibration
0	0.255919
0	-1.27151
0	0.000568941
0	0.000574337
0	1.88869

distortion coeffecients

Error estimate: 0.231089

Task4: Calculate Current Position of the Camera

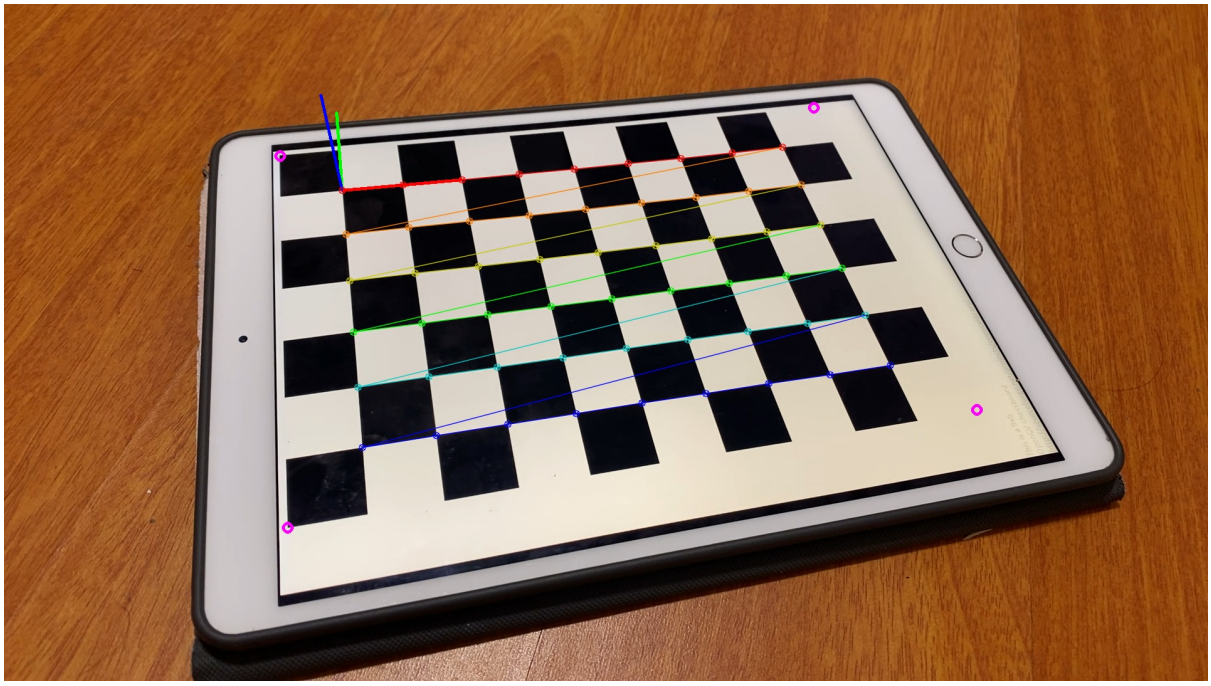
In this task, another program called “objProjection” is developed. The program will read the intrinsic parameters of the camera from the XML file saved in previous step. And then for each frame, it will detect the chessboard, if found any, it will grab the locations of the corners and then use solvePNP to get the board’s pose(rvec and tvec)

Task5: Project Outside Corners or 3D Axes

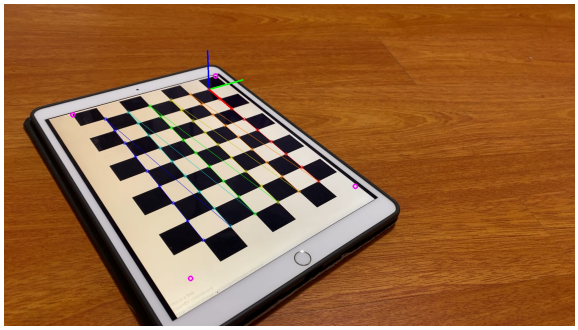
Given the pose estimation from task4, the program was able to use the projectPoints() function and the drawing function of openCV to project the outside corners of the chessboard onto the image plane.

In addition, the program also puts 3D axes on the board attached to the origin at the same time.

Here is an example of this step.



An example of this step, the four outside corner are in pink, and you can also see the axes



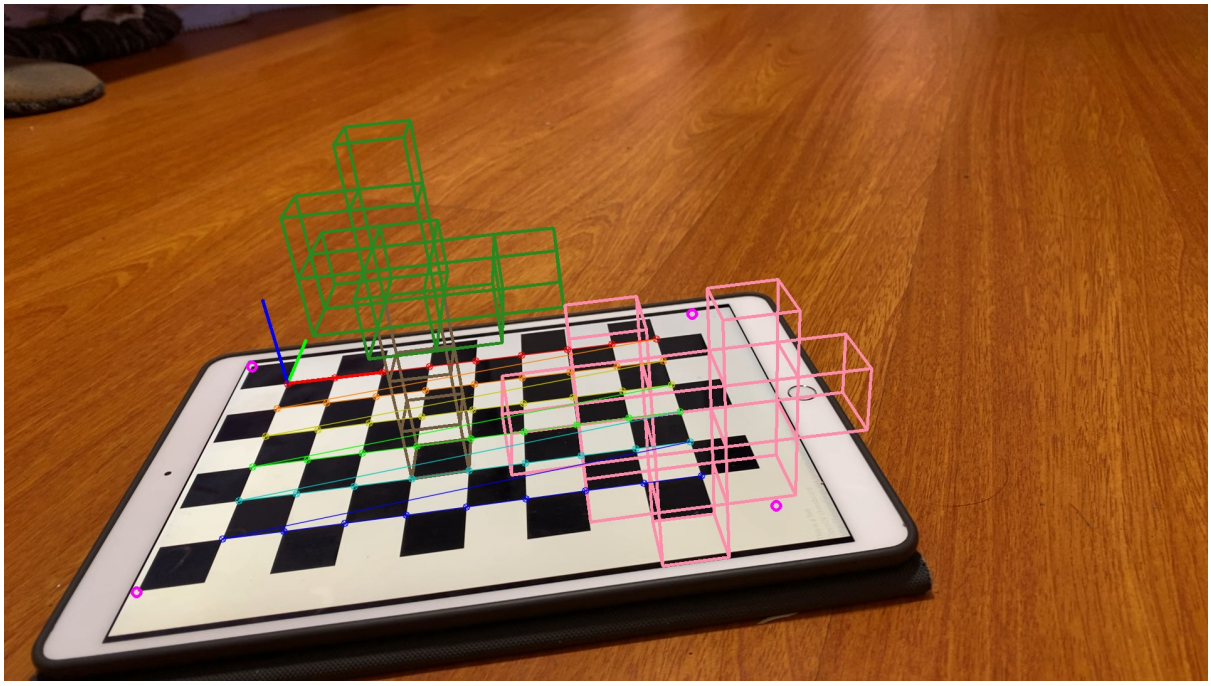
A view from right

A view from more right

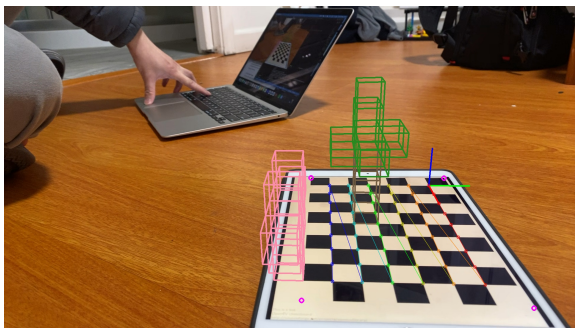
Task6:Create a Virtual Object

In this task, I created a virtual object in 3D world space made out of lines. It is a composition of cubes so that the whole object is asymmetrical. From the result, we can tell that when the camera moves around, the object is always in the right orientation.

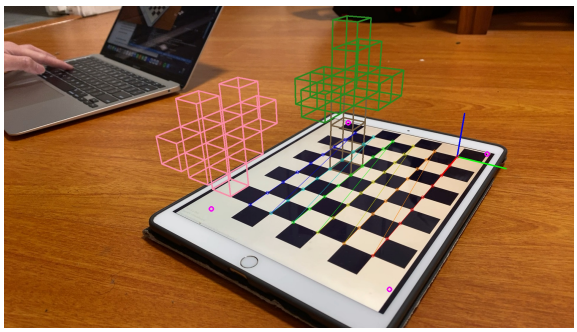
Here are some examples:



Here, we projected a tree and a heart shape on the chessboard(front view)



A view from the right



A view from the right-behind

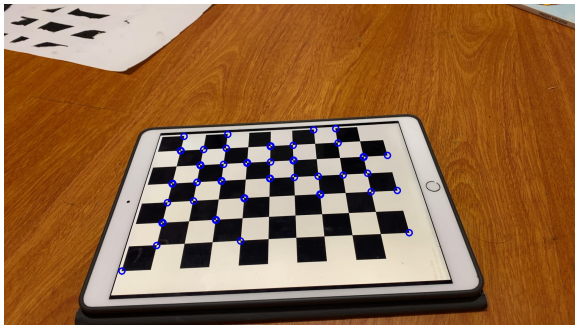
Task7:Detect Robust Features

In this task, another program named “patternDetection” was developed.

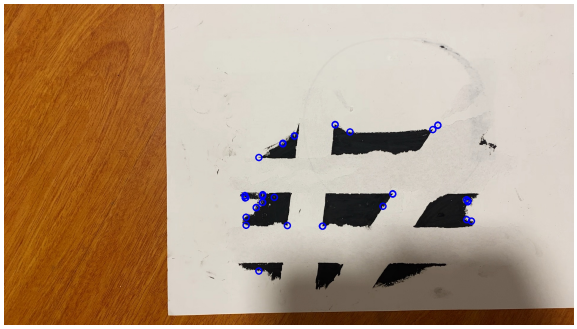
This program implemented both Harris corners detection and SURF features, in which SURF is further used in the next steps.

Here are examples of the features found

Harris Corners:



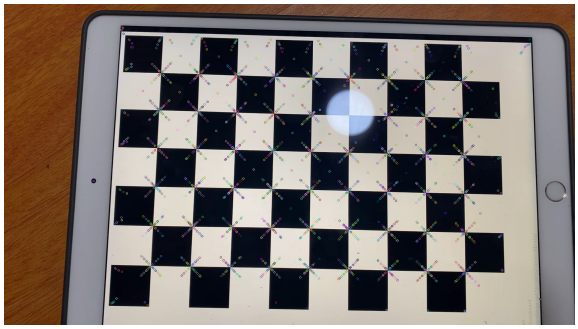
Harris Corners on a chessboard



Harris Corners on a irrigular patter

SURF Features:

SURF feature on a painting (need to look more carefully)



SURF feature on a chessboard

SURF feature on an irregular pattern

Using these features. We are able to put augmented reality into the images too. To do this, we first need to get pre-saved image of the pattern and the robust feature. Then, in the video feed, we can try to extract the features of each frame. Then we need to do a match between the pre-saved features and frame features. After the matching, we can use a solvePNP function to get the rotation and translation vectors. And the following steps of creating a virtual object will be the same with task 6.

Extension 1: Placing a virtual object on a picture using SURF features.

For extension, the program utilized the SURF features from the task7 and tried to get the AR system work with a target other than just the checkboard.

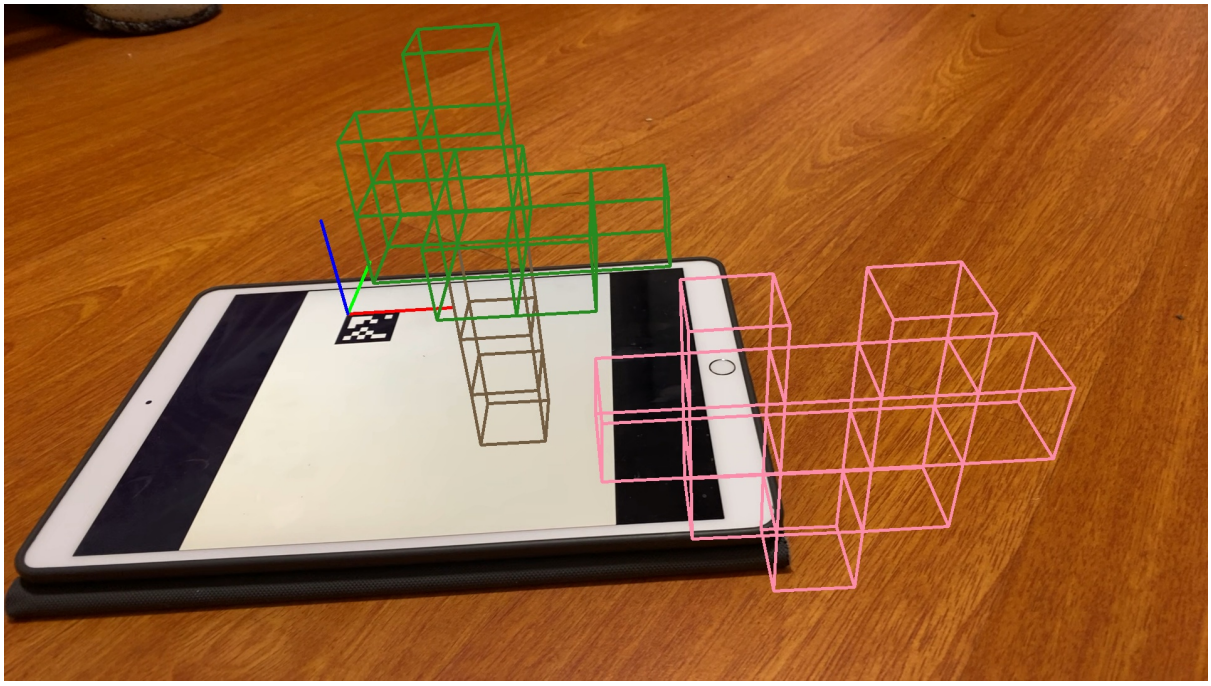
Firstly, the program needs a pre-saved image of the pattern and then extracts the SURF feature from it. Then, in the video feed, the program extracts the SURF features of each frame. Then did a FLANN-based match between the pre-saved features and frame features. After the matching, the program used a solvePNPRansac() function to get the rotation and translation vectors. With the rvec and tvec from, the program is able to call pointsProject to project 3D points on the image plane, and then draw out the lines.

Here are examples of the system working(this extension was not very successful, but I did two extensions)

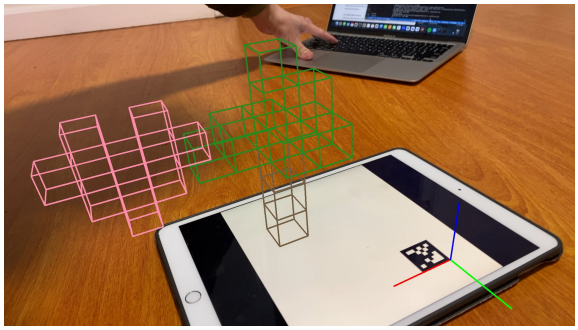
The three lines of the axes is not differenciable. But the orientation is always right when changing positions.

Extension 2: Integrate ArUco into the system

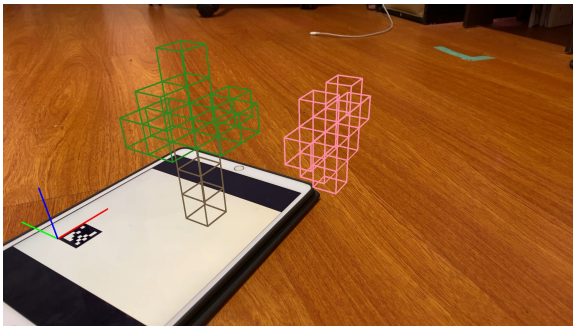
For this extension, the program enabled to use the ArUco as the feature to detect rather than a chessboard. From the program, we are able to generate a ArUco label. When there is one label can be detected, the program will project the virtual objects on the image plane.



A view from the front



A view from the behind



A view from the left-front

Reflection:

From this project, I have a deeper understanding of the augmented reality. I have read several tutorials to understand each step of this project. I now know why the corners are important. Also I learned how to use other features of the object such as the Harrris corner and the SURF features. I also learned how to do the feature matching using functions like FLANN matching. More importantly, I now know the whole process of how to calibrate a camera, and how to generate the rotation and translation vectors. And then use those information to project points form the 3D world to the image plane.

AR is a very interesting area, I will keep digging in this area and try to build more interesting things with it.

Acknowlegements:

online tutorials I have used:

https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html

https://docs.opencv.org/4.x/d4/d7d/tutorial_harris_detector.html

https://docs.opencv.org/4.x/d7/d66/tutorial_feature_detection.html

https://docs.opencv.org/4.x/d5/d6f/tutorial_feature_flann_matcher.html

https://docs.opencv.org/4.x/dc/d2c/tutorial_real_time_pose.html