Project 4: Calibration and Augmented Reality

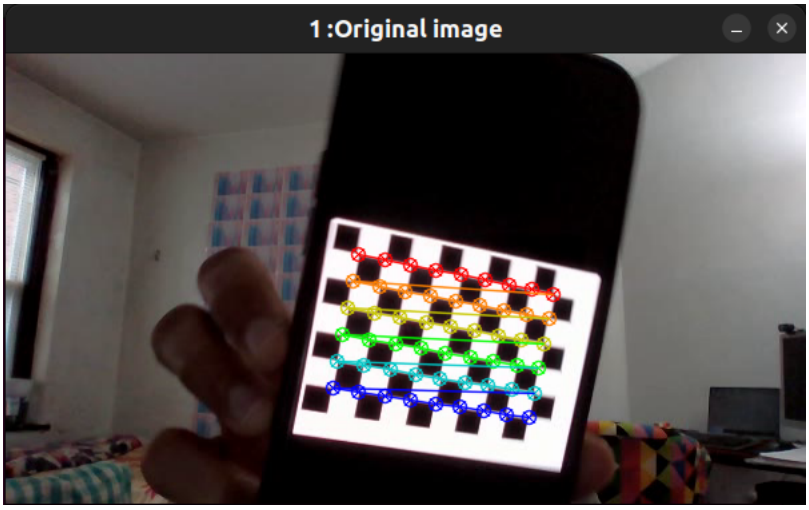
Name: Dev Vaibhav

Email: vaibhav.d@northeastern.edu

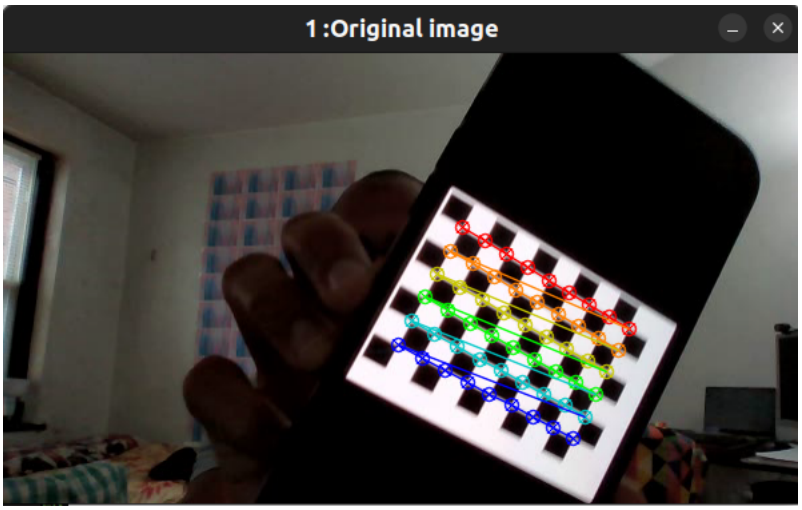
Class: CS 5330 Pattern Recognition and Computer Vision

Description: The goal of this project is to learn how to calibrate a camera and then use that calibration to create virtual objects in a scene. To begin, we must extract many frames from the video in order to obtain a set of picture coordinates and corresponding real-world coordinates with which to calibrate the camera matrix and distortion coefficients. Then, by converting the real-world coordinates to image coordinates and drawing lines between them, we might render a virtual object onto the target.

Task 1) Detect and Extract Chessboard Corners: This task was implemented using the findChessboardCorners(), cornerSubPix() functions to find the chessboard corners in live video. The following image shows the number of corners detected and the coordinates of the first corner i.e the top left corner.



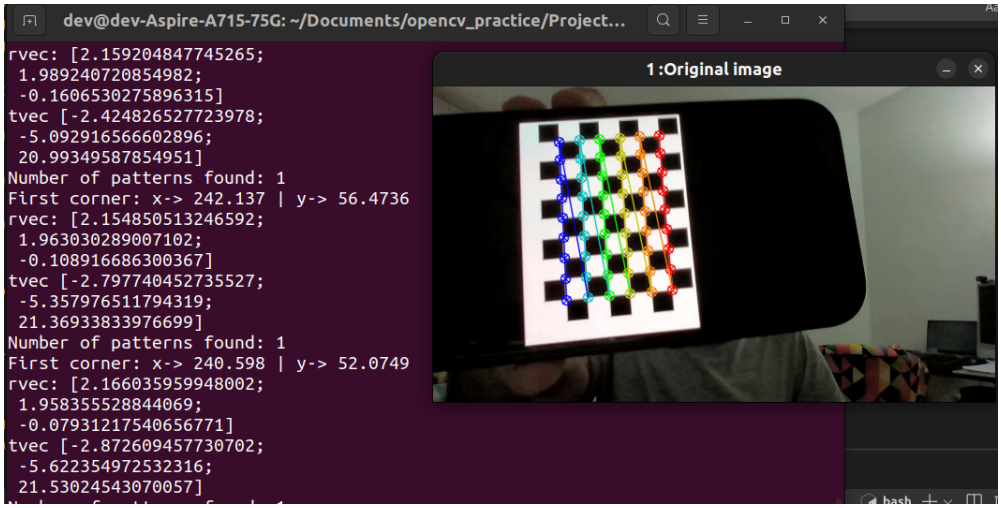
Task 2) Select Calibration Images: The code has the functionality to save live photos to disk when the user presses 's' into a directory ".. /calibration_images". 15 photos are captured in total which are taken by moving the checkerboard randomly in different pose with respect to the camera. Below is an example of a calibration image.



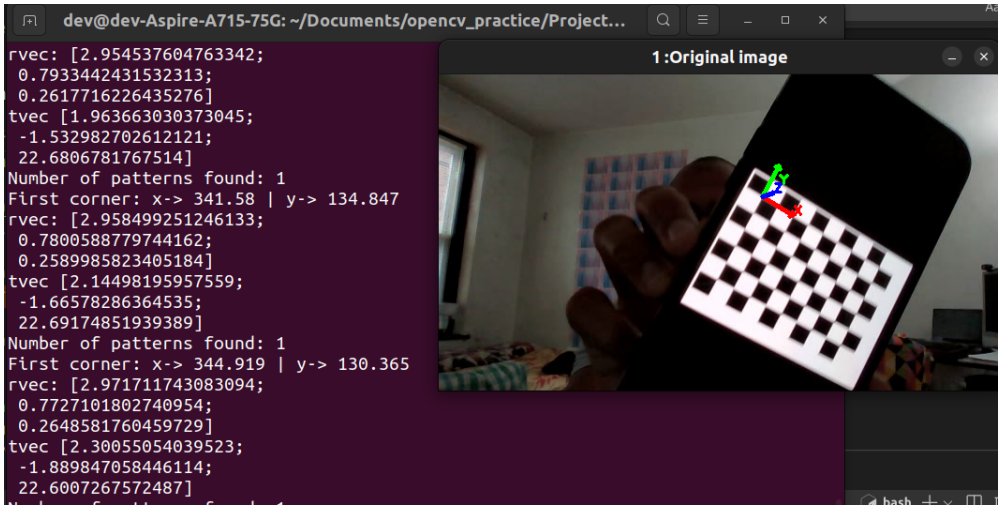
Task 3) Calibrate the Camera: This part of the code reads the images in the directory "../calibration_images". If the contents of this directory are empty, user can save the images for calibration by pressing the 's' button. Once images are saved and if the contents of the directory are non-empty, code reads images one by one, calculates the intrinsic and extrinsic camera calibration parameters, rvec, tvec, and Point_List, and writes them to a yaml file "calibparam.yaml". 15 images were used for camera calibration.

```
Calibrating...
Before calibration
cameraMatrix =
[1, 0, 300;
 0, 1, 168.5;
 0, 0, 1]
distCoeffs=
[0, 0, 0, 0, 0]
Reprojection error = 0.232171
cameraMatrix =
[448.2060566325312, 0, 299.5;
 0, 448.2060566325312, 168;
 0, 0, 1]
distCoeffs=
[-0.124084, 0.229533, 0, 0, 0]
Write Done.
```

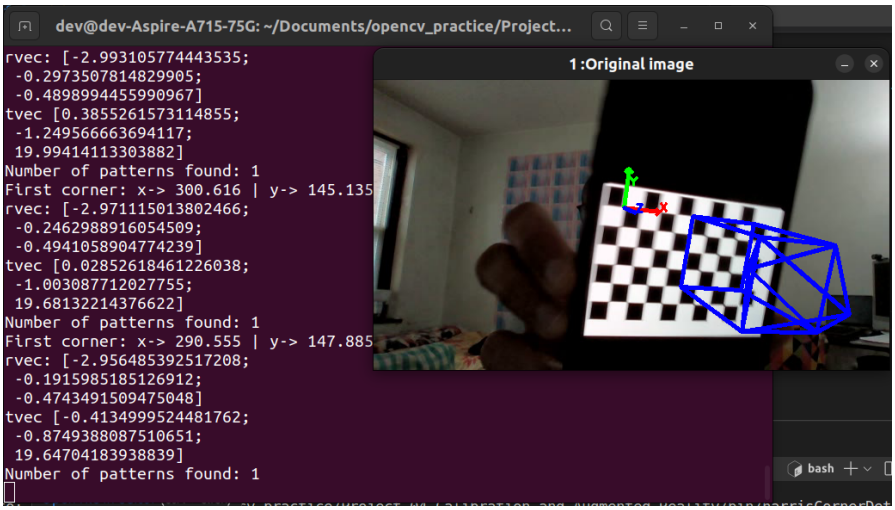
Task 4) Calculate Current Position of the Camera: This task involved calculating the pose of the camera in real-time based on the camera calibration parameters present in the yaml file. This is done using the cv::solvePNP() function. The following image shows the program printing out the rotation and translation vector associated with each frame in real time.



Task 5) Project Outside Corners or 3D Axes: This task draws the 3D axes on the board attached to the origin.

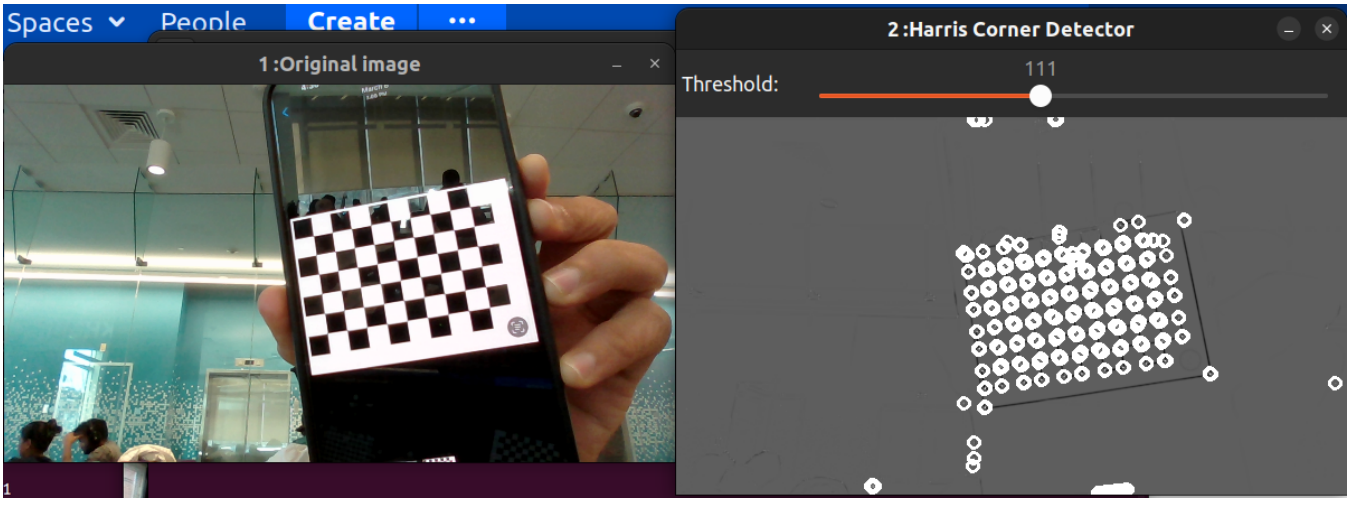


Task 6) Create a Virtual Object: I tried to project the Washington Monument on the checkerboard as shown below by defining the corners of the point and making edges between them.



task6-2023-03-20_18.13.53.mp4

Task 7) Detect Robust Features: Harris corner detector is implemented in which the threshold is variable (trackbar on GUI). The features detected are not stable (change a lot). It is not advisable to use it with AR.

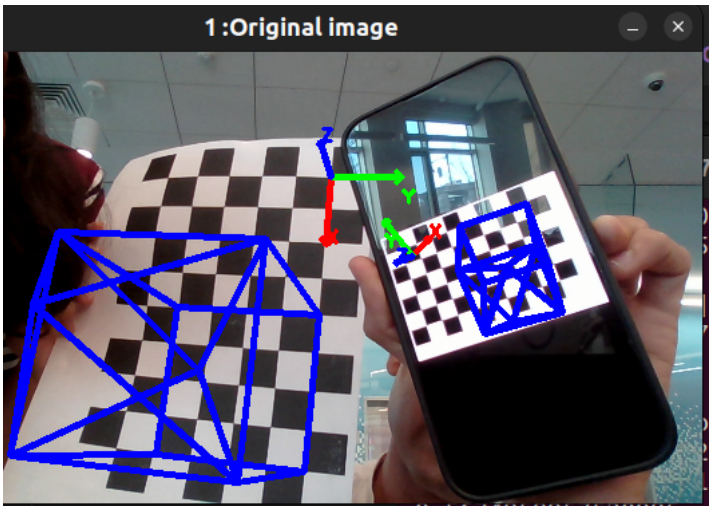


task7-2023-03-20_18.16.28.mp4

EXTENSIONS:

1) Get your system working with multiple targets in the scene

The findChessboardCorners() function can only detect one set of chessboard corners in a frame. After some searching on the internet, a different workaround was required to detect them in a frame. It is done by looking for chessboard corners in a loop, and every time the program finds one, the polygon consisting the detected corners is painted in white and this modified frame is searched for the corners again. The following screenshot shows the program detecting multiple targets in the frame in real-time while projecting the virtual object on all of them.

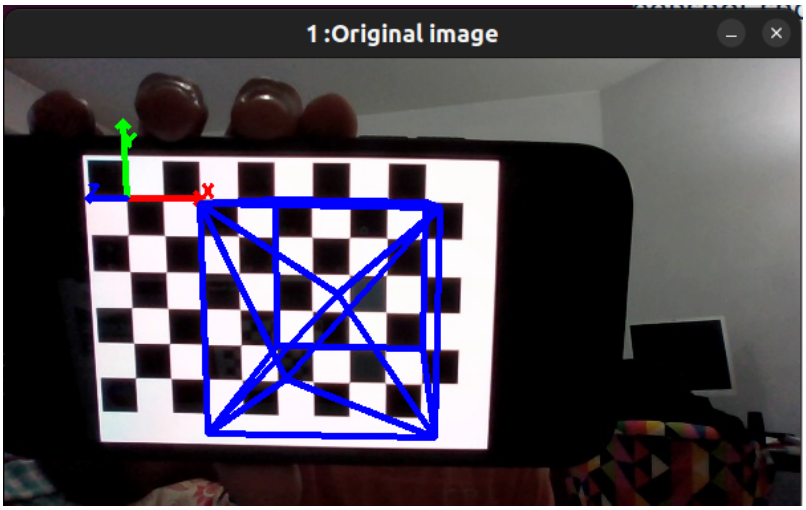


ext1_multiple_t...20_18.19.08.mp4

2) Enable your system to use static images or pre-captured video sequences with targets and demonstrate inserting virtual objects into the scenes

The program gives the user the option to insert virtual objects in either static images, pre-recorded videos, and live video. It automatically detects if the input is an image or video by counting the number of frames in the input.

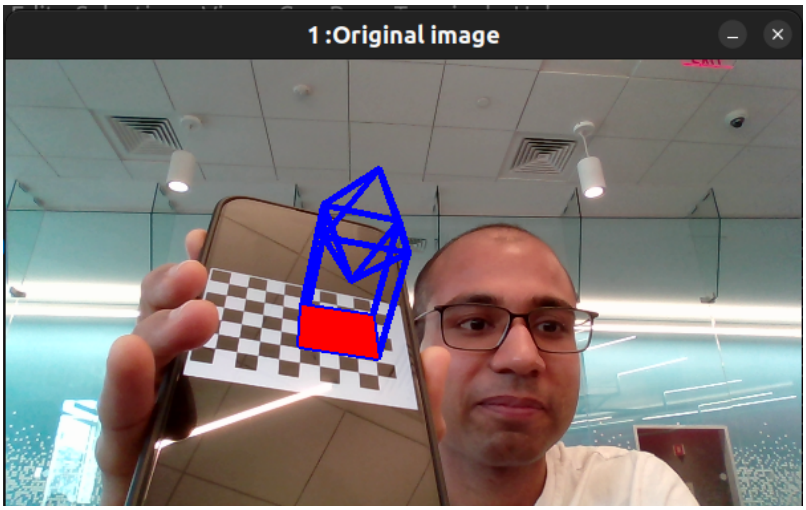
Projection on static image



Projection on recorded video:

ext2_recorded_...0_18.22.05.mp4

3) Alter target to make it not look like a target anymore: The program also has the capability to change the target (all targets in case of multiple) into other image which just paints the detected corners polygon in red. This altered target is also able to align correctly as orientation changes. The following images and videos show the system in action.



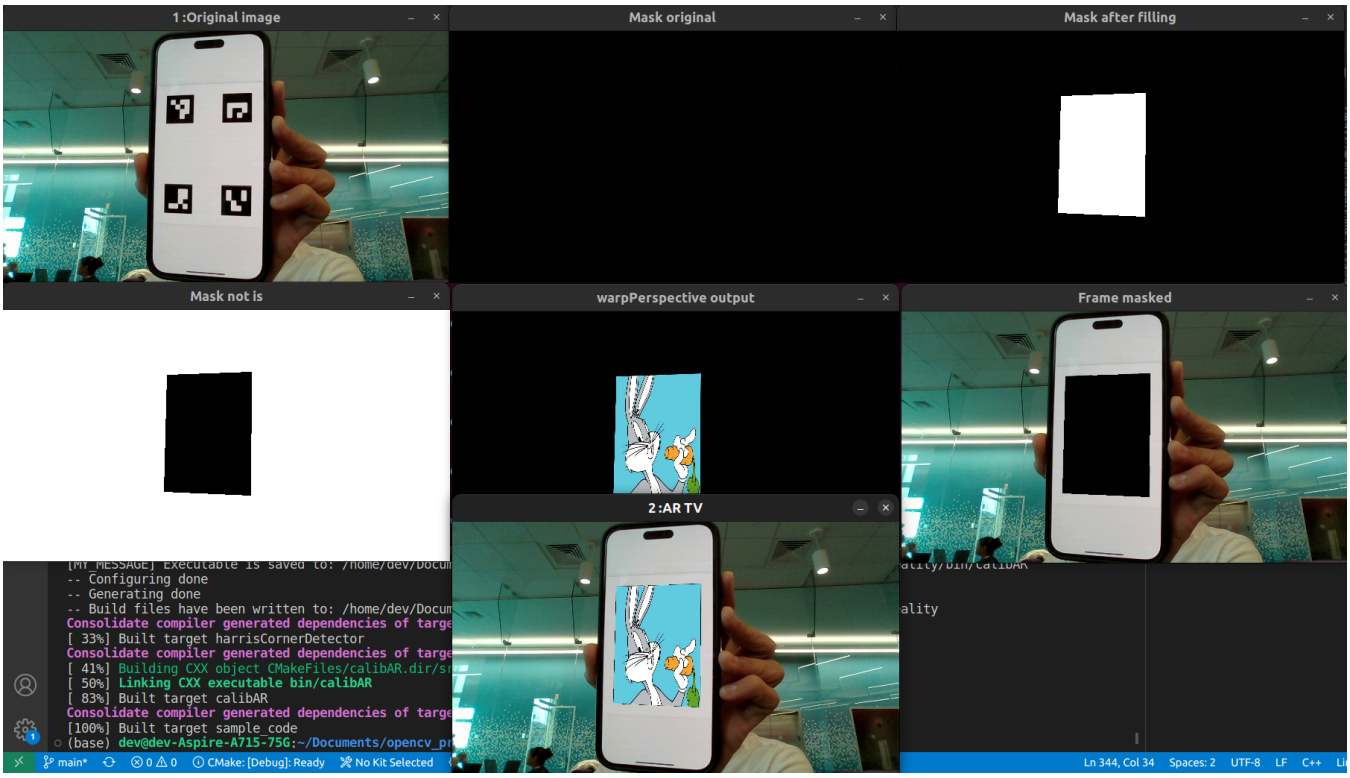
ext3_alter_targ...20_18.24.40.mp4

4) Test out several different cameras and compare the calibrations and quality of the results: The 2 cameras tested out were Acer A715-75G-50SA camera and XPCAM N5 USB webcam. The re-projection error for the external camera was 1.04350 as compared to 0.23217 for the laptop camera which is indicative of the better performance of the laptop camera. Both the devices had relatively similar performance in terms of the quality of the results.

laptop_cam_stab...calibparam.yaml

skanda_cam_mov...alibparam.yaml

5) Augmented Reality Television: I generated 4 markers with ID 0 to 3 and printed them on paper. The program looks for 4 Arucotag markers generated from dictionary using 4x4 dictionary. It then finds out the outermost 4 corners and projects an image/ video (specified by the user) on it. If the four markers are not detected, then projection won't happen. If they are detected, next step is to overlap the AR frame within the 4 corners. It can be achieved using two methods: By calculating a transformation matrix using either *cv::findHomography* or *cv::getPerspectiveTransform* which is then fed to *cv::warpPerspective* to get a warped image. A mask is generated using the 4 corners and bitwise and/ or/ not to get the final output. An interesting find, in this case, was that the Bugs bunny photo was too large compared to the resized live feed. Because of this, the warped image extended beyond the frame bounds and the AR frame was not showing within the 4 corners. To overcome this, bugs bunny photo/ any AR frame is first resized to live video frame size and then the warped output and AR TV worked nicely.



Reflections:

This project was a very fun introduction to Augmented Reality and all the various steps that go into it such as Camera calibration, calculating the pose etc. Augmented reality was one of the things that I always thought is complicated but this project helped me understand how to go about it. It was also a very fun creative outlet. I got to know converting 3D points to 2D coordinates, got to know a bit about arUco tags. In addition, I now grasp the concept of perspective transformation as well as the OpenCV methods that were used to convert a marker into video frames.

Acknowledgments: I'd like to thank the resources listed below for their assistance in understanding and referencing various concepts while working on this project. I'd also like to express my gratitude to Professor Bruce for his great video explanations of these ideas. I have mentioned many more references inside the code marked with "Src: "

OpenCV documentation

https://stackoverflow.com/questions/2802188/file-count-in-a-directory-using-c

https://docs.opencv.org/4.x/d4/d15/group__videoio__flags__base.html#ggaeb8dd9c89c10a5c63c139bf7c4f5704da6223452891755166a4fd5173ea257068

https://stackoverflow.com/questions/17158602/playback-loop-option-in-opencv-videos

https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a

https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/

https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/

https://kediarahul.medium.com/augmented-reality-television-with-aruco-markers-opencv-python-c-c81823fbff54,

https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a