Samuel Lee

CS5330

Project 4

3/18/2024

**Project Overview**

The primary purpose of this project is to understand how to calibrate a camera and use the calibration data to project objects from 3D to the 2D image plane. This project first explores recognizing corners of a feature or target (for instance by using a chessboard). Once the target is found, we can save and use these images with the detected target to calibrate the camera. During calibration, it is important to have multiple different views of the target.

With enough images and corresponding corner sets, my program takes all the data to calibrate the camera. Calibration returns a refined camera matrix, distortion coefficients, and a reprojection error. Using this data, my program can recognize the same target in different orientations and poses in terms of rotation and translation. Now, with the camera matrix, distortion coefficients, rotation matrix, and translation matrix, the program is now able to project points of the 3D world onto the 2D image plane.

Overall, this project calibrates the camera with example images of the target and uses this calibration data to generate 3D objects in the 2D image plane.

**Required Images**

*Question 1*

I am using the chessboard (9x6) target.

One limitation I am noticing is with extreme rotations to the chessboard. For instance, when I rotate the chessboard at a certain angle, the program is not recognizing the chessboard. (SHOW EXAMPLE IMAGE).

Also, I can see how extreme lighting can prevent the chessboard from being recognized as a target. (SHOW EXAMPLE IMAGE).

TEST FOR SMALL CHESSBOARD

*Question 2*

My program stores the images used for calibration in the directory “calibration\_directory” within the project folder.

SHOW IMAGE OF CALIBRATION CORNERS

*Question 3*

After calibration, my program immediately creates a .yml file and stores the camera matrix and distortion coefficients found in that run. For simplicity, the program overwrites any previous values in the .yml file.

IS CAMERA MATRIX CALIBRATION MATRIX?

*Question 4*

SCREENSHOT OF TERMINAL CHANGING VALUES

Move camera side to side and see what happens

*Question 5*

Yes, the reprojected points show up in the expected place. I projected 3D axes on the target attached to the origin. The origin actually starts 1 square away from the top side and the left side (due to the nature of how chessboard recognition works in OpenCV).

Show image of XYZ axes

*Question 6*

I created a slanted 3D parallelogram. The top and bottom faces are both squares, but their X-coordinates and Z-coordinates differ. From the sides, the shape you can see is not a square but a parallelogram.

Show this image

*Question 7*

**Extensions**

*Multiple Chessboard Target Recognition*

I made a separate cpp file called “augmentedRealityMultiple.cpp” that projects 3D object points to multiple detected targets in the video feed. To incorporate this, in “detectCorners.cpp,” I added a function called “getMultipleCorners,” which stores all found set of corners in a vector<vector<Point2f>>. Once a corner set is found, the program masks out the corner set and reiterates this until no additional corner sets are found. Using this vector of corner sets, we can project multiple 3D objects if multiple targets are found.

ADD EXAMPLE

*Image and Video Processing*

Instead of just using a video feed to project 3D objects onto the current frame, I created a file called “generateAR.cpp” that allows a user to input a file path to a video or image to create an edited version of the video or image that shows the projected 3D object. Using the already existing calibration data in the .yml file, I was able to detect corners and the chessboard target as expected. This extension allowed me to explore VideoWriter in-depth to edit a video frame by frame. I also implemented a file and extension checker to see if the file is a valid video or image file and whether it was a valid file to open in the first place.

ADD EXAMPLE

*Advanced Virtual Objects*

Write my name ?

**Reflection**

This project helped me understand projective geometry and how I can use OpenCV to test this. In class, the concepts I learned were not intuitive and hard to visualize. I understood the theory and studied it repeatedly, but I did not know the practical applications of the material I learned. This project really benefited me in my learning because it bridged the gap between theory and application. The step-by-step instructions of the project were especially helpful for me to understand the concepts by cross-referencing each section to the lecture.

Now, I know that we need a target to anchor from such as a chessboard or anything unique that doesn’t vary much from one image to another. Using different views of the target, we can calibrate our camera to get our calibration matrix, reprojection error, and distortion coefficients, which we can use to get our rotation and translation matrices. Using these data points, we can then project 3D objects onto the 2D image plane.

Overall, I think this project was perhaps the most enjoyable and enlightening because I was able to concretely see the results of my program. Looking forward to the next projects, and I hope learning computer vision continues to be fun, exciting, and educational!

**Acknowledgement**

* Professor Maxwell’s lectures – for filling knowledge gaps
* OpenCV documentation – to learn more about calibration and projection
* TAs – to ask questions to whenever stuck
* Stack Overflow – some posts on this website aligned with what I struggled with; used this resource to understand OpenCV functions better
* Past project code – re-used some code to save time