Zhimin Liang

Spring 2024 CS5330

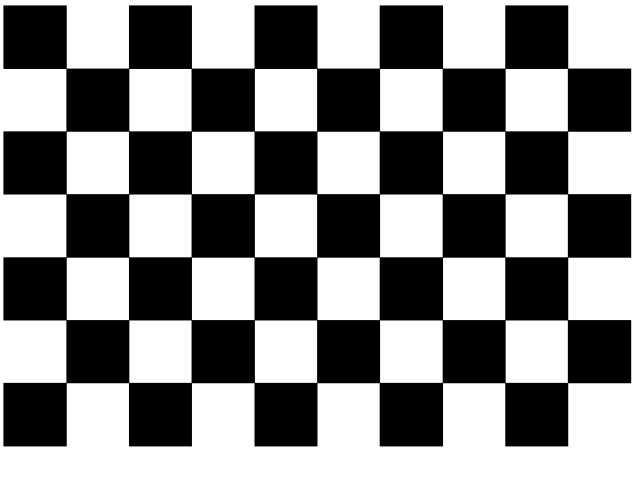
Project 4

1.A short description of the overall project in your own words. (200 words or less)

The project focuses on implementing augmented reality (AR) functionalities using computer vision techniques. It involves developing software that can analyze real-world images or video streams captured by a camera, detect distinctive features within these images, and overlay virtual content onto the scene in real-time. Key components of the project include camera calibration, feature detection and tracking, pose estimation, and integration of virtual objects into the real-world environment. By accurately estimating the camera's position and orientation relative to the scene and mapping virtual objects onto specific points or surfaces within the image, the project aims to create immersive and interactive AR experiences. These experiences could range from enhancing visualizations in educational settings to enabling interactive gaming or providing informative overlays in industrial applications. Overall, the project aims to showcase the potential of computer vision and AR technologies to create engaging and impactful user experiences in various domains.

2.Any required images along with a short description of the meaning of the image.

Task1:Indicate in your report which type of target you are using. Describe any limitations or challenges the system has in locating the target.



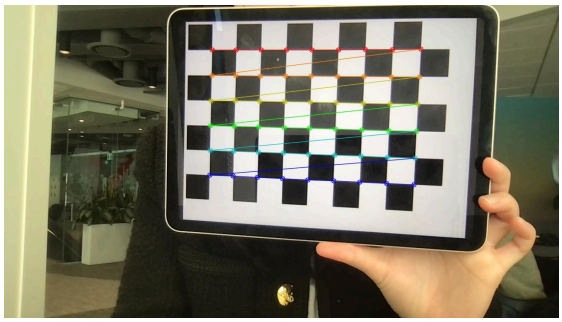
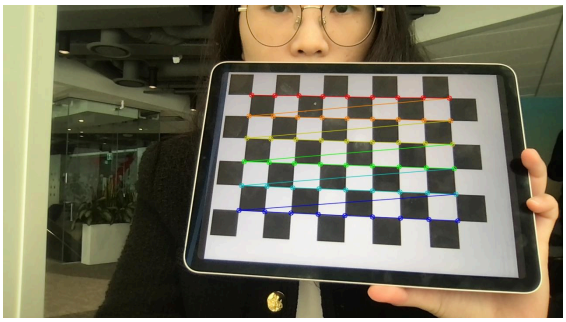
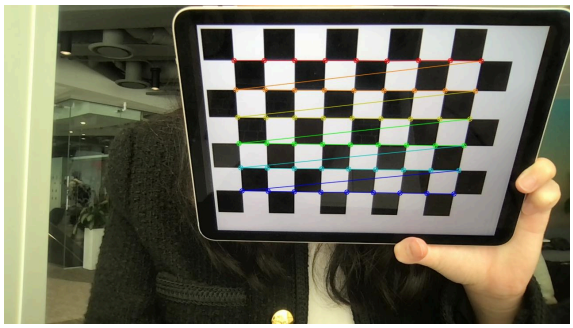
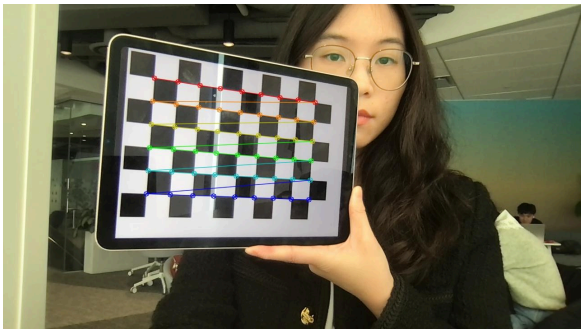
I use a checkerboard pattern as the target for corner detection and extraction. The checkerboard pattern comprises a grid of alternating black and white squares arranged in a specific configuration, typically with a known number of rows and columns.

Limitations/Challenges:

Symmetry: Checkerboard patterns are highly symmetric, which can sometimes lead to false positive corner detections, especially when the pattern is viewed from certain angles.

Lighting Conditions: Variations in lighting conditions, such as shadows or uneven illumination, may affect the contrast between squares in the checkerboard pattern, impacting corner detection accuracy.

Task 2: You may want to store the images themselves that are being used for a calibration. Include a calibration image with chessboard corners highlighted in your project report.

```
Corner list contents:
(479.506, 103.453) (535.495, 103.423) (592.686, 103.288) (651.086, 103.136) (710.469, 102.907) (770.897, 102.705) (833.053,
102.339) (896.378, 101.968) (961.227, 101.438) (478.517, 161.74) (534.196, 162.204) (590.911, 162.53) (649.24, 162.803) (708
.359, 163.269) (768.644, 163.795) (830.333, 164.266) (893.239, 164.531) (957.703, 164.761) (477.289, 219.542) (532.635, 220.
442) (589.256, 221.203) (647.304, 222.187) (706.354, 223.207) (766.349, 224.331) (827.736, 225.57) (890.199, 226.652) (954.2
39, 227.534) (476.023, 277.373) (531.114, 278.623) (587.453, 280.08) (645.521, 281.509) (704.372, 283.166) (764.112, 284.914
) (825.243, 286.826) (887.489, 288.608) (950.838, 290.384) (474.847, 334.371) (529.458, 336.389) (585.718, 338.407) (643.654
, 340.428) (702.387, 342.55) (761.879, 344.905) (822.721, 347.048) (884.467, 349.505) (947.757, 351.723) (473.709, 390.775)
(528.288, 393.457) (584.222, 396.123) (641.848, 398.901) (700.328, 401.673) (759.579, 404.297) (820.197, 407.142) (881.608,
409.903) (944.68, 412.931)
(567.758, 206.653) (621.449, 207.535) (675.997, 208.49) (731.475, 209.573) (787.341, 210.599) (843.727, 211.637) (901.536, 2
12.532) (959.933, 213.27) (1019.56, 214.159) (562.295, 255.418) (617.133, 256.444) (672.685, 257.794) (729.397, 259.222) (78
6.307, 260.618) (843.848, 262.19) (902.627, 263.401) (962.18, 264.663) (1022.98, 265.847) (556.577, 306.243) (612.492, 307.7
11) (669.374, 309.373) (727.279, 311.226) (785.306, 312.877) (843.863, 314.644) (903.777, 316.479) (964.649, 318.167) (1026.
66, 319.797) (550.682, 359.539) (607.663, 361.532) (665.781, 363.601) (724.782, 365.652) (784.072, 367.763) (843.875, 369.81
9) (905.265, 372.059) (967.297, 374.193) (1030.63, 376.4) (544.785, 414.732) (602.878, 417.311) (661.941, 419.712) (722.292,
 422.277) (782.771, 424.539) (843.915, 427.011) (906.504, 429.53) (969.914, 432.418) (1034.71, 435) (538.755, 471.901) (597.
863, 475.025) (658.248, 478.024) (719.535, 480.813) (781.41, 483.599) (843.824, 486.568) (907.98, 489.638) (972.967, 493.008
) (1039.22, 496.348)
(519.636, 124.82) (588.035, 125.009) (656.996, 125.192) (726.48, 125.385) (796.189, 125.487) (866.214, 125.596) (937.338, 12
5.461) (1008.51, 125.473) (1079.37, 126) (523.851, 191.082) (590.161, 191.34) (656.713, 191.685) (724.263, 192.139) (791.425
, 192.637) (859.041, 192.974) (927.65, 193.229) (996.562, 193.335) (1065.69, 193.665) (527.734, 253.197) (591.821, 253.54) (
656.618, 254.027) (722.035, 254.627) (787.226, 255.407) (852.454, 256.026) (918.617, 256.557) (985.29, 256.898) (1052.5, 257
.291) (531.427, 311.902) (593.559, 312.465) (656.491, 313.143) (719.945, 313.809) (783.164, 314.622) (846.224, 315.412) (910
.355, 316.219) (974.625, 316.673) (1039.75, 317.342) (534.816, 366.949) (595.312, 367.78) (656.414, 368.576) (717.973, 369.4
39) (779.312, 370.189) (840.426, 370.856) (902.56, 371.656) (964.879, 372.509) (1027.86, 373.319) (538.396, 418.698) (596.91
3, 419.779) (656.3, 420.748) (715.899, 421.519) (775.451, 422.349) (834.911, 423.141) (895.292, 423.947) (955.64, 425.011) (
1016.94, 425.943)
```

When users press “s”, the system allows users to save the corner locations and the corresponding 3D world points, which can be used for camera calibration. Users can capture images of the calibration pattern from different viewpoints and orientations to allow the calibration algorithm to accurately estimate the intrinsic and extrinsic camera parameters.

Task 3: Enable the user to write out the intrinsic parameters to a file: both the camera_matrix and the distortion_ceofficients. Include your calibration matrix in your report, along with the corresponding re-projection error.

build > calibration.csv

```
%YAML:1.0
---
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.5774674978639928e+02, 0., 6.4718730937839223e+02, 0.,
      9.5774674978639928e+02, 3.6748915096251750e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ -2.1607889588644216e-01, 2.1124482803771389e+00,
      -5.8462717313574740e-05, 3.7838859091253955e-03,
      -5.8342904641620397e+00 ]
```

Users can write out the intrinsic parameters to a yaml file with camera_matrix and the distortion_ceofficients.

```
lzm@xiaozhuizideMBP build % ./main
Expected size: 1280 720
Not enough calibration frames. Please capture 5 frames.
Not enough calibration frames. Please capture 5 frames.
Not enough calibration frames. Please capture 5 frames.
Not enough calibration frames. Please capture 5 frames.
Initial Camera Matrix:
[1, 0, 0;
 0, 1, 0;
 0, 0, 1]
Initial Distortion Coefficients:
[0;
 0;
 0;
 0;
 0]
Calibrated Camera Matrix:
[957.7467497863993, 0, 647.1873093783922;
 0, 957.7467497863993, 367.4891509625175;
 0, 0, 1]
Calibrated Distortion Coefficients:
[-0.2160788958864422;
 2.112448280377139;
 -5.846271731357474e-05;
 0.003783885909125395;
 -5.83429046416204]
Re-projection Error: 0.127358
Calibration parameters saved to calibration.csv
lzm@xiaozhuizideMBP build %
```

The Calibrated Camera Matrix:

[957.7467497863993, 0, 647.1873093783922;
0, 957.7467497863993, 367.4891509625175;
0, 0, 1]

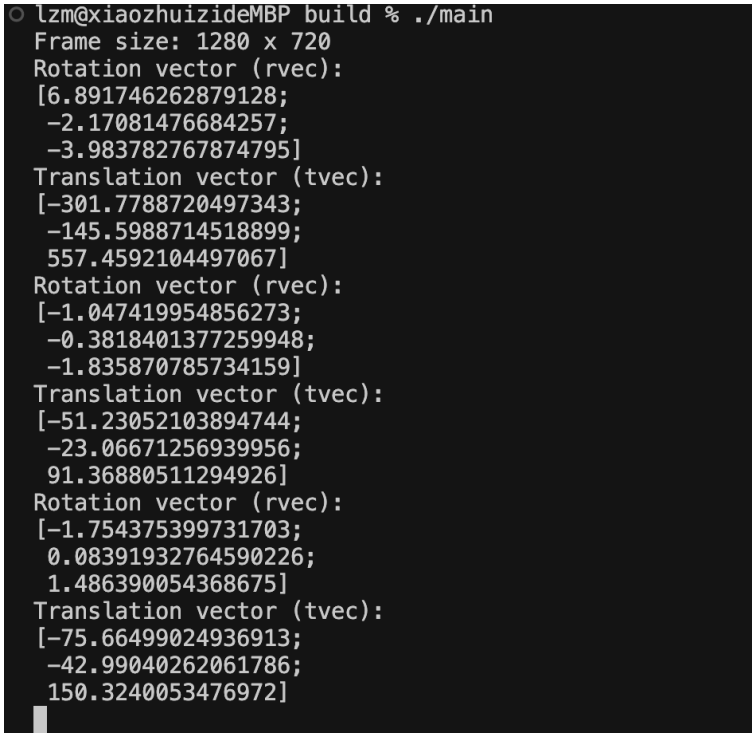
The distortion_ceofficients:

[-0.2160788958864422;
2.112448280377139;
-5.846271731357474e-05;
0.003783885909125395;
-5.83429046416204]

The Re-projection Error: 0.127358

This means the corners of the calibration pattern as projected using the estimated camera parameters are about 0.127358 pixels away from where they were actually detected in the images. This is a very low value, suggesting that the calibration process has been highly accurate and the camera model fits the observed data well.

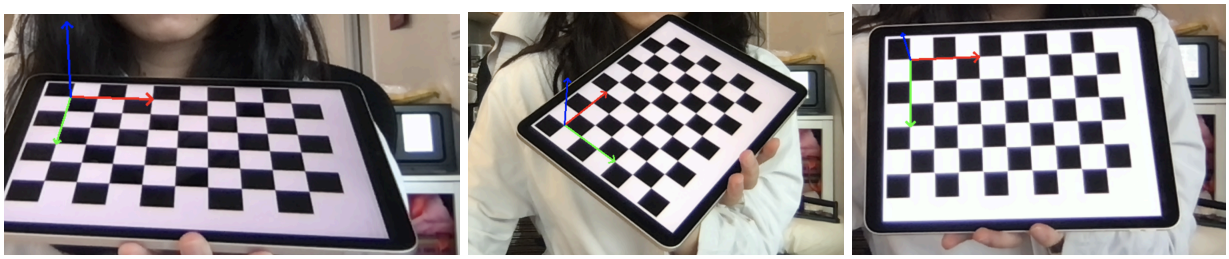
Task 4:Have your program print out the rotation and translation data in real time, as you are testing this task. Include in your report a description of how these values change as you move the camera side to side. Do they make sense?



Initially, significant rotations are observed, particularly around the x-axis, indicating substantial changes in pitch. Subsequent observations show smaller rotations, suggesting minor adjustments in the camera's orientation.These changes align with expectations, as the camera's pitch, yaw, and roll would naturally adjust as it moves relative to the target.

The translation vector reflects the camera's displacement relative to the target in its coordinate system. Initially, there is a significant displacement, with notable changes in all three components of the translation vector. As the camera moves side to side, subsequent observations show smaller translations, indicating minor movements.These changes are consistent with expectations, as the camera would experience larger displacements initially when transitioning from one side to another, followed by smaller adjustments to maintain alignment with the target.

***Task 5:** Do the reprojected points show up in the right places? Include at least one image from this step in your report.*



I choose to put 3D axes on the target attached to the origin, and we can see that it shows up on the right place

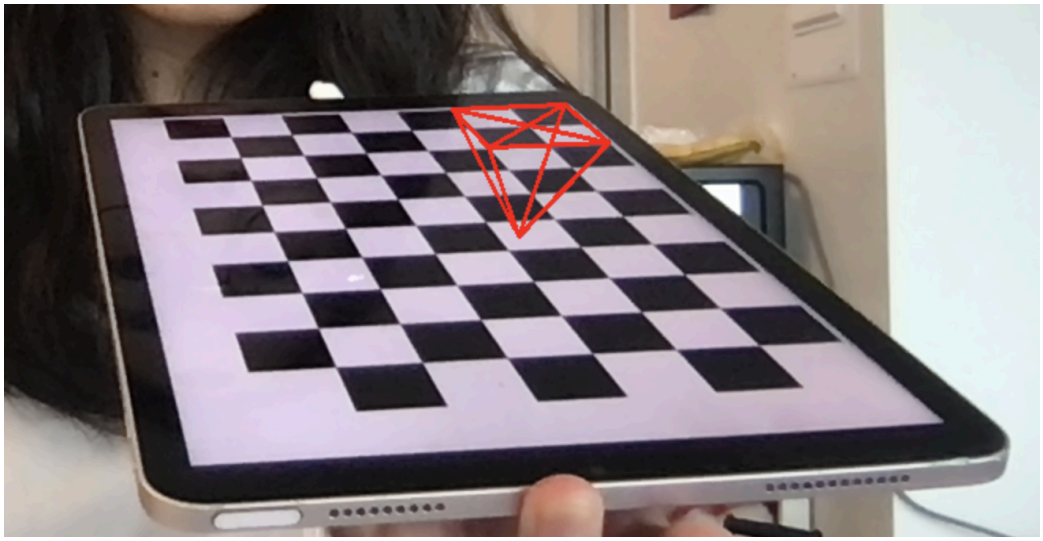
First, the code detects the checkerboard corners in the captured frame and refines their positions using cornerSubPix. Then, it estimates the pose (rotation and translation vectors) of the calibration target using solvePnP function.

After obtaining the rotation and translation vectors, the code defines 3D points representing the axes (X-axis, Y-axis, and Z-axis) attached to the origin of the calibration target's coordinate system. These points are stored in the axisPoints vector.

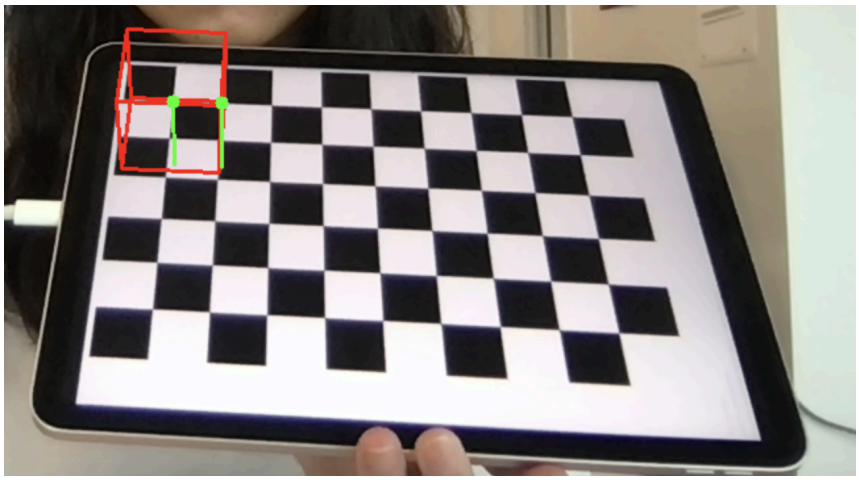
The projectPoints function is then used to project these 3D points onto the image plane using the estimated rotation and translation vectors, camera matrix, and distortion coefficients. This step computes the corresponding 2D image coordinates of the 3D points.

Finally, the code draws the projected axes onto the captured frame using arrowedLine. Each axis is drawn as a line connecting the origin (projected point of the origin) to the projected points of the X, Y, and Z axes. Each axis is assigned a color (red for X-axis, green for Y-axis, and blue for Z-axis) to distinguish them.

***Task 6:** Describe your virtual object and take some screen shots and/or videos of your system in action for your project report.*

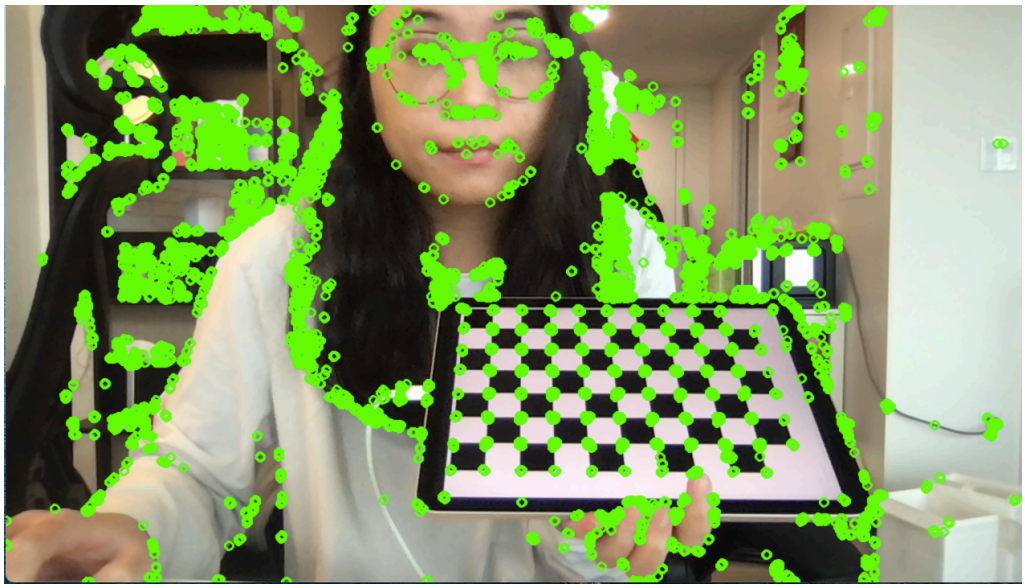


This is a reverse pyramid



This it cube with cylinder onto image plane

Task 7: In your report, explain how you might be able to use those feature points as the basis for putting augmented reality into the image.Take some screen shots and/or videos of the system working.



Feature points can be used to estimate the pose (position and orientation) of the camera relative to the scene. By identifying and tracking these distinctive points across frames, algorithms can compute the camera's movement in real-time. This information is crucial for overlaying virtual objects onto the image in a way that aligns with the camera's perspective, and it also provides unique landmarks on objects or scenes, enabling robust tracking of these objects as they move within the camera's field of view.

3. A description and example images of any extensions.

NO extensions

4.A short reflection of what you learned.

Throughout this project, I gained valuable insights into the practical implementation of augmented reality (AR) using computer vision techniques. I learned how to calibrate cameras, detect and track features in real-time, estimate camera pose, and integrate virtual objects into live video streams. This project provided hands-on experience with popular computer vision libraries like OpenCV and helped me understand their capabilities and limitations in the context of AR applications.One significant aspect I learned was the importance of robust feature detection and tracking for accurate pose estimation. Experimenting with different feature detection algorithms and tuning their parameters allowed me to understand how they perform under various conditions and scenes. I also realized the critical role of camera calibration in ensuring accurate mapping between the real-world and virtual coordinates, which is essential for realistic AR experiences.

5.Acknowledgement of any materials or people you consulted for the assignment.

- **OpenCV Tutorials:** https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html
- **YouTube:**

  Computer Vision Lab, “OPENCV & C++ TUTORIAL”,https://www.youtube.com/playlist?list=PLUTbi0GOQwghR9db9p6yHqwvzc989q_mu [access, Feb,29,2024]