INFOMCV Assignment 1

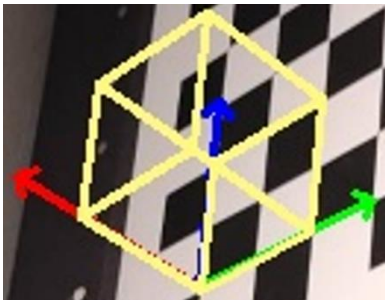
Marios Iacovou (1168533), Christos Papageorgiou (9114343) (Group 74)

Calibration

Run 1:

$$K = \begin{bmatrix} 606.81 & 0 & 302.45 \\ 0 & 609.8 & 416.63 \\ 0 & 0 & 1 \end{bmatrix}$$

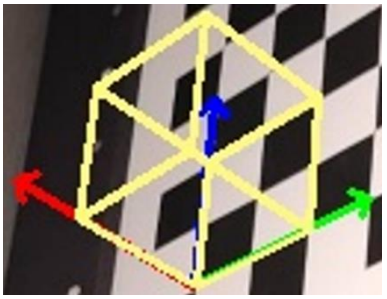
Image resolution:
- Initial image = **1197x1600** px.
- After resizing (final) = **598x800** px.



Run 2:

$$K = \begin{bmatrix} 607.44 & 0 & 353.68 \\ 0 & 607.4 & 424.8 \\ 0 & 0 & 1 \end{bmatrix}$$

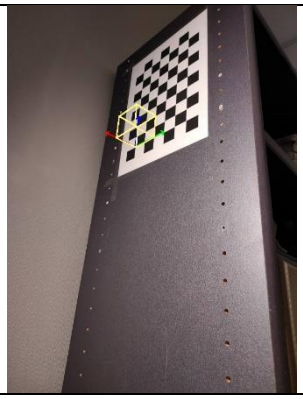
Image resolution:
- Initial image = **1197x1600** px.
- After resizing (final) = **598x800** px.

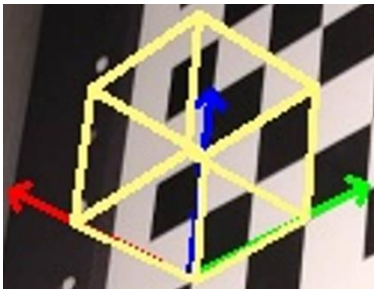


Run 3:

$$K = \begin{bmatrix} 659.47 & 0 & 439.45 \\ 0 & 661.22 & 532.76 \\ 0 & 0 & 1 \end{bmatrix}$$

Image resolution:
- Initial image = **1197x1600** px.
- After resizing (final) = **598x800** px.

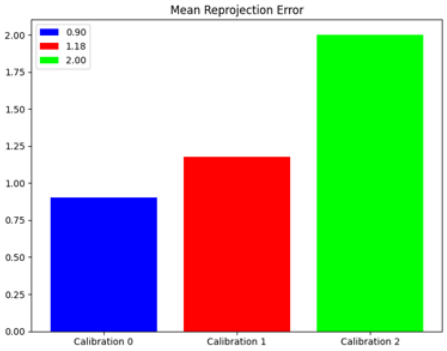


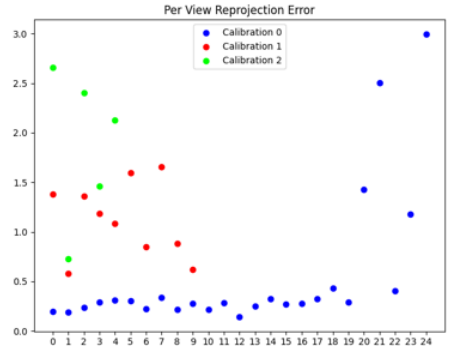


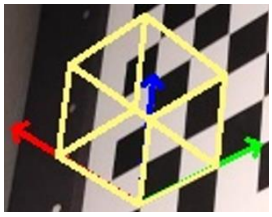
Discussion

Run 1 (25 images): The intrinsic matrix indicates moderate focal lengths (606.81 and 609.80 for x and y axes, respectively) and a fairly centered optical center (302.45, 416.63). This matrix likely provides a balanced representation of the camera's characteristics due to a larger dataset, which generally helps in averaging out the noise and inconsistencies present in individual images. The limitation in this run's calibration is the images where the corners had been manually selected, as the images themselves were difficult to process with precision in the first place. This can be seen in the chart (below) that shows the per view error for every image, where the 5 images with manual

selection were sorted to the end for presentation purposes and show higher per view error than the rest of the images. The Root Mean Square (RMS) Re-projection Error for run 1 is Re^1_{RMS}=0.90 (with 2-digit precision) which is the best out of the 3 runs. Tested with the image shown before we can see that this run has the best results overall, with the most aligned drawn cube on the titled image with the checkerboard near the edge and corner of the image. Tested with a video (choice option 4), this run continued to have the best overall results with the least flickering at various hard angles.
Run 2 (10 images): The focal lengths here are slightly increased compared to Set 1, but not significantly (607.44 and 607.40). The optical center has shifted more noticeably towards the right and slightly upwards (353.68, 424.80). This could be a result of less averaging of variations found in individual images due to the smaller dataset, leading to minor shifts in perceived camera characteristics. The RMS Re-projection Error is Re^2_{RMS}=1.18 > Re^1_{RMS}. Compared to run 1, the cube is slightly more misaligned on the test image, which applied to the test video as well.
Run 3 (5 images): There is a notable increase in focal lengths (659.47 and 661.22), and the optical center has shifted considerably to the right and upwards (439.45, 532.77). This significant shift could be attributed to the small sample size, which might not represent the camera's characteristics as accurately. The calibration is more susceptible to outliers or specific characteristics of the few images used, leading to a potentially less reliable estimation of the camera parameters. The RMS Re-projection Error is Re^3_{RMS}=2.00 > Re^2_{RMS} which was the worst out of all 3 runs. Compared to the other 2 runs, it is apparent that this run has the worst alignment, which also applied to the video.







Choice tasks

1) Improving the localization of the corner points in your manual interface: 10. Make the estimation of (a) the four corner points or **(b) the interpolated points more accurate**: We used the manual corners to compute a perspective transformation matrix using cv2.getPerspectiveTransform. We sampled an M x N grid of points, with vertical and horizontal steps calculated using the number of squares on each axis of the chessboard and max distances from corners. Each point is stored in a vector with Z=1 and multiplied by the inverse transformation matrix normalized by Z and stored as a 2D interpolated point. The results are significantly better than the results of a flat interpolation that we compared them against. We also use cv2.cornerSubPix to increase the precision of each corner.
2) Iterative **detection and rejection of low quality input images in offline phase**: We implemented a function that removes each image iteratively from the pool of images and checks if that improves the overall error given a threshold. Using this with the 25 images of run 1, we removed 1 outlier image that improved the error to 0.66 (see plots folder). This 24-image run is labeled as run 4 in our files and in tests it had the most accurate alignment for the test image (see cube above) and video.
3) Implement a function that can provide a **confidence for how well each variable has been estimated**: We used the extra parameters returned by cv2.calibrateCameraExtended to get access to standard deviations of each estimated variable (Fx, Fy, Cx, Cy) and plotted them. The resulting plots showcase the better precision with lower standard deviations of run 1 (and run 4) against the rest.
4) We additionally, as already mentioned, implemented a function to **test the estimated camera matrices on a video** to further test the precision. The video tests the calibrations at different angles while the light and position of the board changes. The results can be generated in the evaluation results folder by running online.py. Videos of the 4 runs were linked throughout this report.