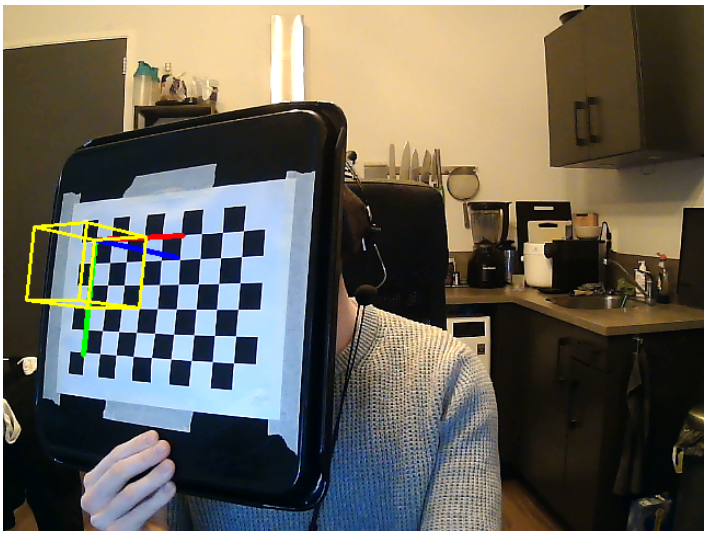
CV - Report Assignment 1

Computer Vision

GitHub repository: https://github.com/rickdott/mcv

Run 1 (all data)

Example

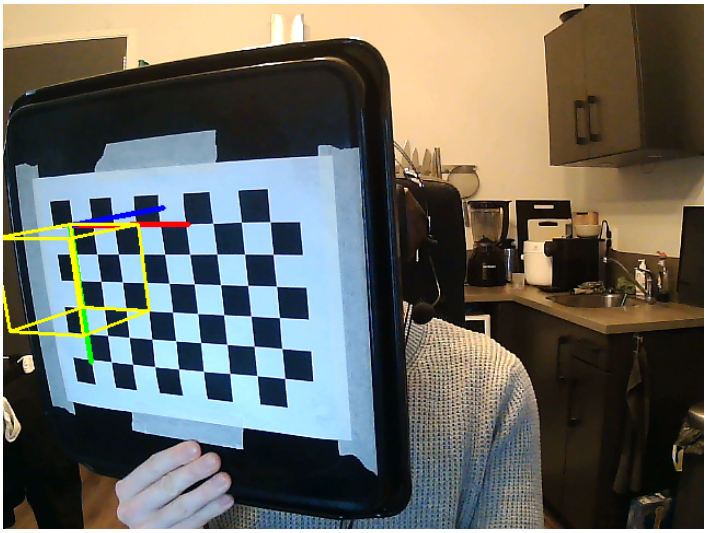


Intrinsic Matrix

```
[[674.91292803   0.         282.62116579]
 [  0.         667.99988379 247.93848211]
 [  0.           0.           1.        ]]
```

Run 2 (10 detected images)

Example

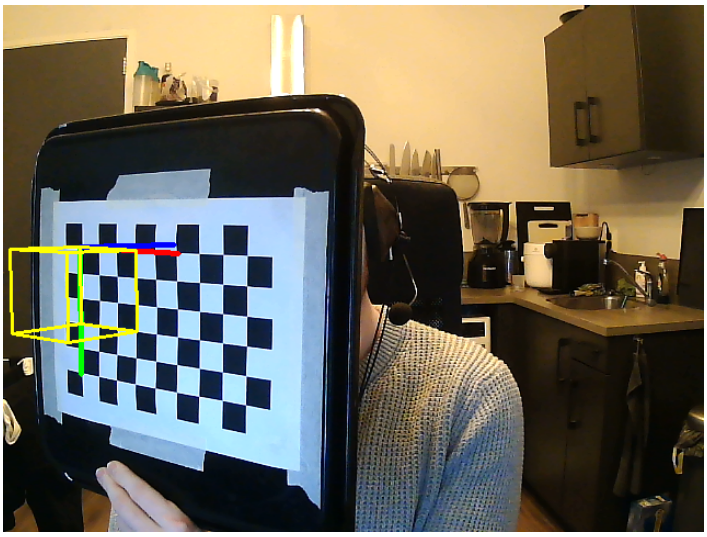


Intrinsic Matrix

```
[[651.50394369   0.         322.29588642]
 [  0.         649.06109259 245.37462324]
 [  0.           0.           1.        ]]
```

Run 3 (5 detected images)

Example



Intrinsic Matrix

```
[[647.51056127   0.         320.74925774]
 [  0.         644.15686895 252.02583847]
 [  0.           0.           1.        ]]
```

Intrinsic matrix change

Since I implemented the iterative detection and rejection of low quality input images, I noticed that most of the corners that I manually entered were deemed not good enough. This shows in the reprojection error for every run. The first run (with manual images) had a reprojection error of 1.402. The second run (with 10 automatically detected images) had a reprojection error of 0.1975. In the third run I did end up seeing the effect of not having enough data, since the reprojection error was 0.1636. I think the difference is due to my implementation and the order of my images. I can see the first 5 images are quite simple, causing a low reprojection error. I believe that the manually annotated images do not improve the reprojection error since their perspective is already skewed a lot.

Choice tasks

Real-time performance with webcam in online phase

I decided to implement this because I wanted to see the projection move in real time. An important thing to consider with real-time performance is that detecting and drawing information on the image frame does not take longer than it takes for a new frame to arrive. To combat this issue, I set the flag `cv.CALIB_CB_FAST_CHECK` in the `findChessboardCorners()` function so that it runs faster when no chess/checkersboard is present.

Iterative detection and rejection of low quality input images in offline phase

To implement this, I calibrated the camera after every new image. If the difference between the new reprojection error and the previous one is bigger than a predefined epsilon value ($\epsilon = 0.01$), I removed the object and image points from their array, removing them from consideration during the rest of the calibration.

Improving localization of corner points in manual interface

To improve the finding of corners I used a two-pronged approach. First, I use the four user-supplied points to create a polygon mask used in `cv.goodFeaturesToTrack()` to find a maximum of $\#cols \times \#rows$ corners within the user-defined polygon. If this function does not successfully find all corners, I use linear

interpolation. This does not always work perfectly due to skewed perspective. After finding the corners, I use `cv.cornerSubPix()` to refine the corner locations.