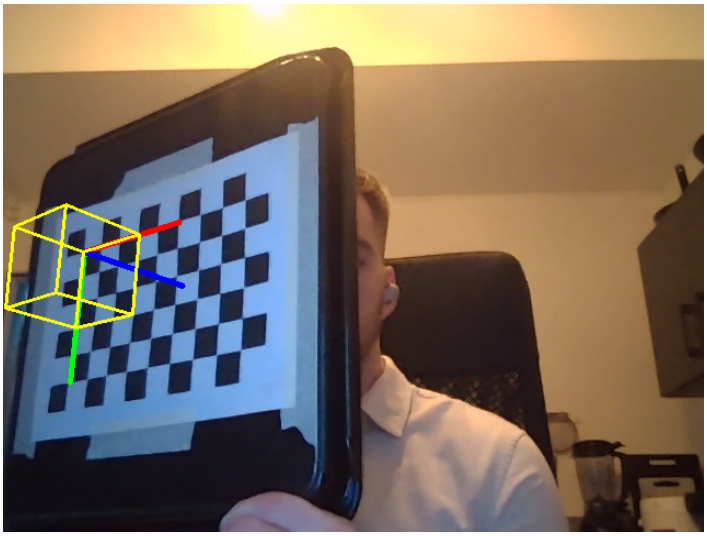
CV - Report Assignment 1

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

Run 1 (all data)

Example

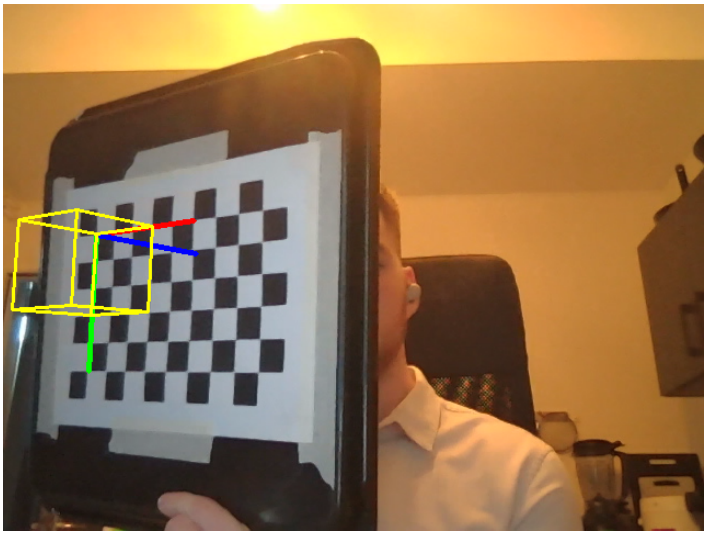


Intrinsic Matrix

```
[[678.26074176   0.         302.654443  ]
 [  0.         676.78509733 233.7038784 ]
 [  0.           0.           1.        ]]
```

Run 2 (10 detected images)

Example

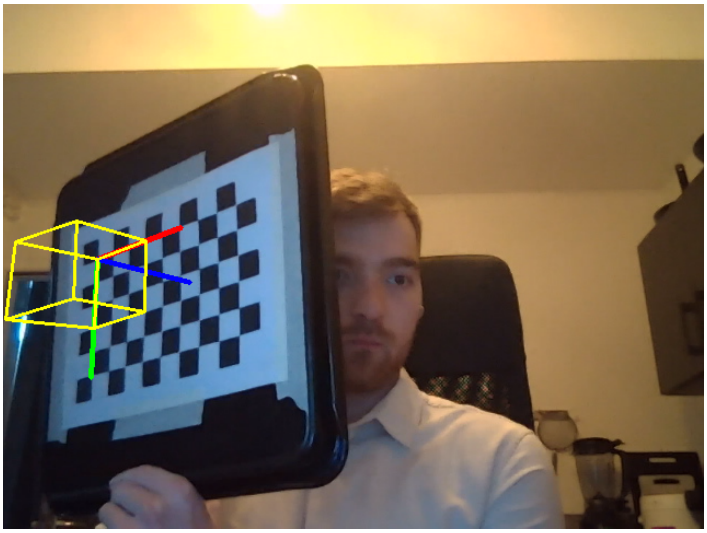


Intrinsic Matrix

```
[[651.65707916   0.         322.46415839]
 [  0.         649.20048196 245.30582641]
 [  0.           0.           1.        ]]
```

Run 3 (5 detected images)

Example



Intrinsic Matrix

```
[[687.44796818   0.         337.16182301]
 [  0.         681.52743619 235.48562585]
 [  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 0.739. The second run (with 10 automatically detected images) had a reprojection error of 0.198. In the third run I did end up seeing the effect of not having enough data, since the reprojection error was 0.8945. What these results show is that my manually interpolated points were not of the highest quality, but that the camera still calibrated just fine with only the automatically detected images.

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, I removed the object and image points from their array, removing them from testing.

Improving localization of corner points in manual interface

I used `cv.cornerSubPix()` for this. I am not very happy with the way this works, since manually annotated images are often rejected by the previous choice task. Visually, the inferred dots look very close to the corners. If I had more time I would want to look into this.