1 Camera Calibration

1.1 Direct Linear Transform

For the data normalization part we first compute the centroid of the object coordinates as well as the camera coordinates, this is done by computing the mean as it yields the center of the data cluster. We then compute the distance between each coordinate and the centroid, the mean of this distance is then used as the scaling factor. Since the average distance for the camera coordinates should be the square root of two we calculate the scaling factor $w_{xy} = \sqrt{2}/meanDist_{xy}$, the scaling factor w_{XYZ} is computed equivalently.

Using the scale and centroid values we build the transformation matrices T and U.

$$T = \begin{bmatrix} w_{xy} & 0 & 0 \\ 0 & w_{xy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c^1_{xy} \\ 0 & 1 & -c^2_{xy} \\ 0 & 0 & 1 \end{bmatrix}$$

This will lead to the point p_{xy} being first translated and then scaled when transformed by T. The matrix U is built equivalently to T for three dimensions. Applying these transformation matrices to the coordinates will result in normalized coordinates $\hat{p}_{xy}$ and $\hat{p}_{XYZ}$.

We then run the direct linear transform algorithm which takes these normalized coordinates and computes a normalized projection matrix $\hat{P}$. This is done by assembling matrix A using equation (1) provided on the exercise sheet. The matrix is then decomposed using SVD where the eigenvector belonging to the smallest singular value is extracted. This vector $\tilde{p}$ will give the best approximation for a non-trivial solution to $A\tilde{p} = 0$.

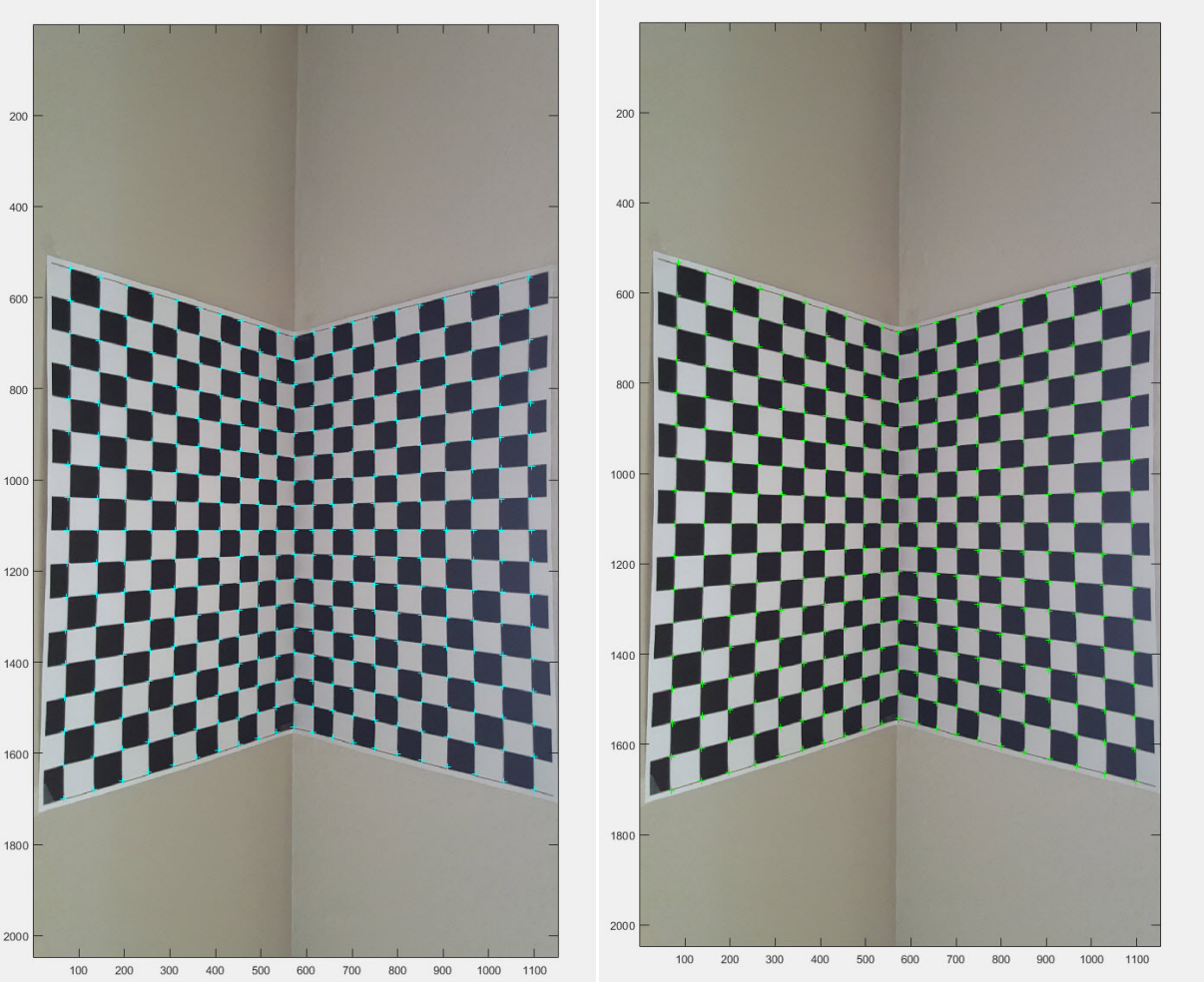
The vector $\tilde{p}$ is restructured into a matrix $\hat{P}$ representing the normalized projection matrix. The projection matrix P is then given by $P = T^{-1}\hat{P}U$ which corresponds to the denormalized projection matrix. This denormalized projection matrix can then be decomposed into the intrinsic camera matrix K, the rotation matrix R and the camera center vector C. This is achieved by extracting the 3×3 matrix M from P and applying the QR algorithm on its inverse, inverting the resulting matrices gives us R and K, C is the null vector of the projection matrix $PC = 0$ so we repeat the procedure for finding it equivalently to finding $\hat{P}$.

In the final part of the DLT algorithm we compute the error and reproject the manually selected points as well as newly generated ones into the camera space. The error is computed as the mean of the distances between the reprojected points and their original positions. If we were to use the unnormalized points the projection matrix P would have a higher condition number and thus be more unstable resulting in large differences in projection for small differences in data points.

1.2 Gold Standard Algorithm

For the Gold Standard Algorithm we also normalize the data and use it to compute the normalized projection matrix $\hat{P}$ equivalently to the description in the DLT section. The projection matrix is then passed to the fminGoldStandard function which is called by the Matlab built-in fminsearch method. The function fminGoldStandard is our minimization function, we compute the squared geometric error as part of the cost function which should be minimized. Using a weight w we can shift the cost function to more strongly focus on either the squared geometric error or the squared skew error and squared focal difference error. Since the camera matrix is modeled after the ideal pinhole camera in which no skew or focal difference error exists we may want to weigh the squared error of these terms differently depending on the sensor used.

After fminGoldStandard has run for twenty iterations the normalized projection matrix is returned and denormalized equivalently to the DLT section. The remaining part of reprojecting and calculating the points is also equivalent to section 1.1. The results can be viewed in fig. 1 as well as fig. 2, for the Gold Standard Algorithm we used the default weight $w = i/5$, the error being slightly higher at around 3 in contrast to DLT with an error of around 1.5, this makes sense as the Gold Standard Algorithm computes the projection matrix based on the minimization of the cost function which also considers the skew and focal difference errors and not only the geometric error.



(a) Direct Linear Transform

(b) Gold Standard Algorithm

Figure 1

2 Bouguet's Toolbox

For the calibration in Bouguet's Toolbox we used nine different images of the checkerboard (see fig. 3). The corner extraction worked well except for the second image in the third column. The angle at which this image was taken is too small and thus the corner detection is imprecise, this can then also be seen on the reprojection error (see fig. 5) where the blue error values correspond to this image. The focal length computed by Bouguet's Toolbox is very close to the value in the intrinsic camera matrix K from the DLT and Gold Standard algorithm, as was expected since all images were taken with the same camera settings. All focal length values are approximately 1500. The console output after calibration is:

```
Calibration results after optimization (with uncertainties):

Focal Length:    fc = [1503.19334 1500.53363] +/- [5.32234 4.95336]
Principal point: cc = [1051.49202 571.15782] +/- [4.87418 6.71913]
Skew:       alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:      kc = [0.19796 -0.43101 -0.00315 -0.00080  0.00000] +/- [0.01152 0.04048 0.00121 0.00142 0.00000]
Pixel error:    err = [0.51670 0.90197]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

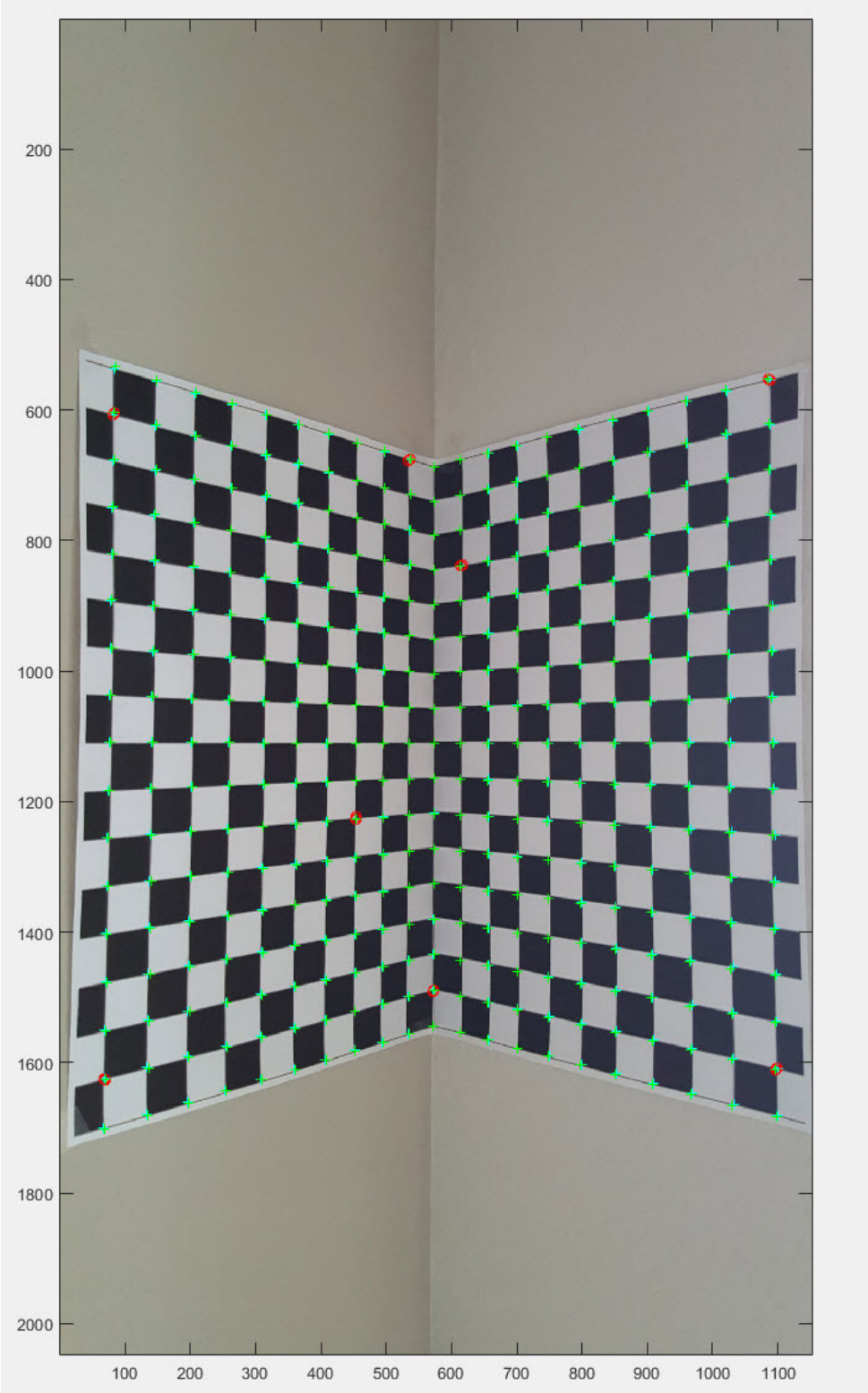


Figure 2: green points from Gold Standard, cyan points from Direct Linear Transform

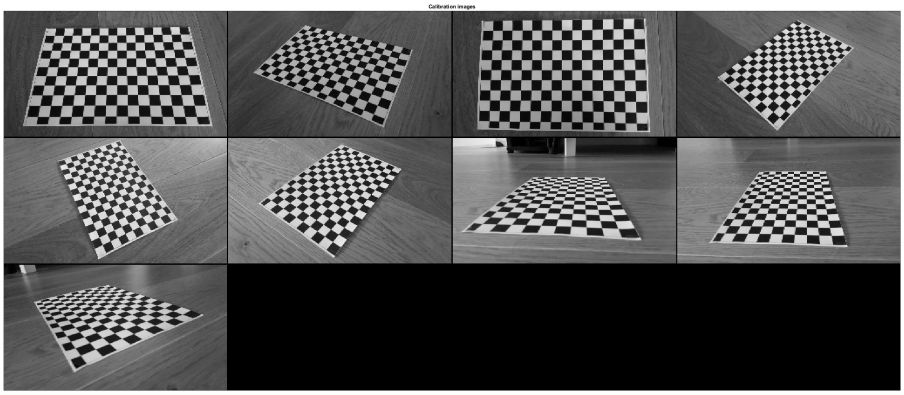
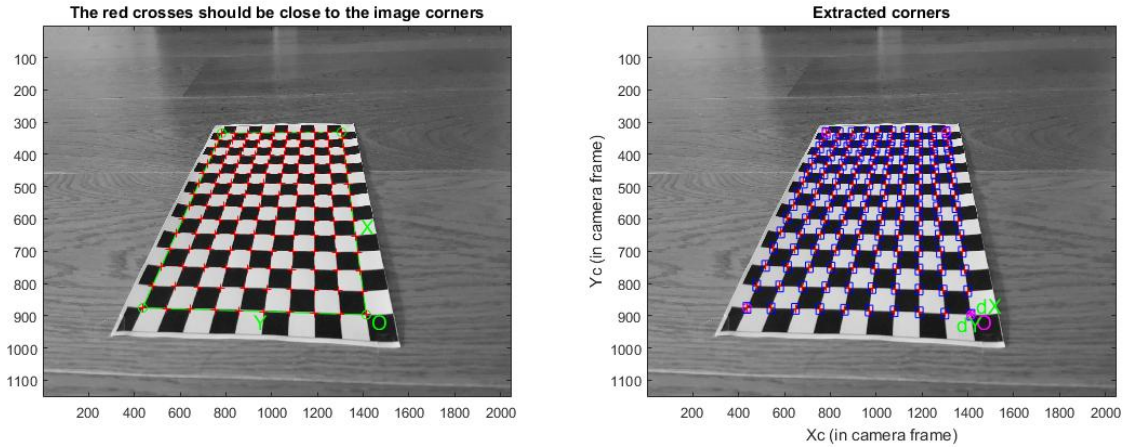
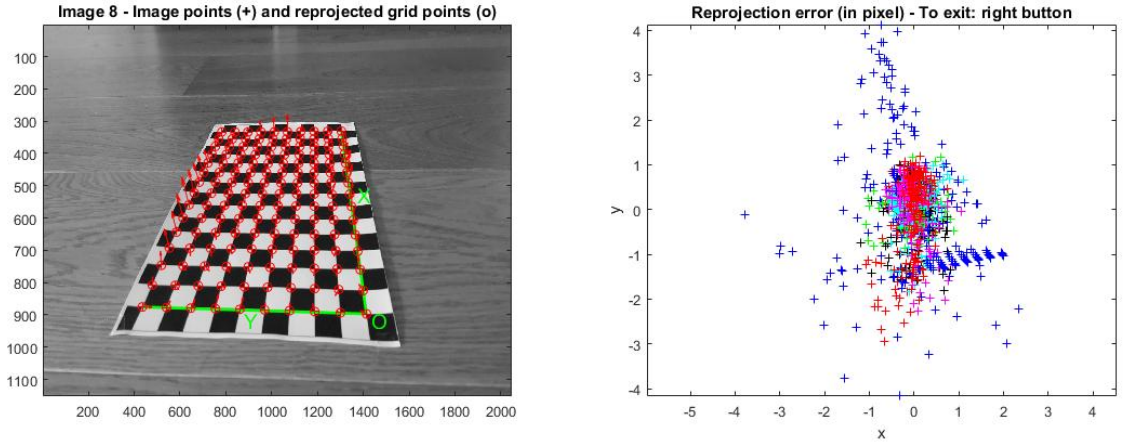


Figure 3: bouguet's toolbox calibration images



(a) (b)

Figure 4



(a) (b)

Figure 5