COMPUTER VISION - ASSIGNMENT 3 - REPORT

CALIBRATION

General approach

The first part of the task consisted in the implementation of the Gold-Standard algorithm for the estimation of the projection matrix P. Given n ≥ 6 world to image point correspondences $Xi \leftrightarrow xi$, our goal is to determine the Maximum Likelihood estimate of the camera projection P.
First of all, we compute an initial estimate of P using a linear method such as the DLT algorithm (computationally efficient) .To avoid numerical instability, due to noise in the image, it is necessary to normalize the data.
Then, we refine our estimate of P, by minimizing the geometric error:
$\Sigma d(xi, PXi)2$
In order to solve this non-linear minimization problem an iterative algorithm is used. The computed linear estimate is the starting point of this algorithm.
After having denormalized P, we can decompose it (using QR decomposition) to compute an accurate estimate of the calibration matrix K, of the rotation matrix R and of the vector t. The goal of the task is to estimate the intrinsic (K) and extrinsic (R,t) camera parameters.
The approach proposed in the assignment does not involve any method to estimate the **distortion effects**, such as radial distortion. We can basically assume that between the extrinsic matrix and the intrinsic matrix, there is a [Id I 0] 3 x 4 matrix.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \simeq \begin{bmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\intercal & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

NORMALIZATION and DLT

Let's suppose we have a set of world to image point correspondences $Xi \leftrightarrow xi$. Then it holds that:
$x \propto PX \rightarrow x \times PX = [x]_{\times} PX = 0$

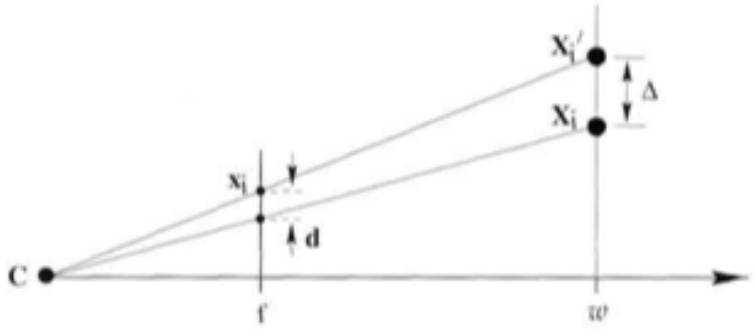
As suggested in the assignment, we can formalize the constraints by defining a vectorized version of the projection matrix P: $vec(P) = [P11,P12,...,P34]T$ and a constraint matrix A obtained by stacking up the **two independent constraint vectors** we have for **each correspondence**. In other words, the matrix A is composed by stacking up **n** matrices "**a**", where a is the constraint matrix obtained for a single correspondence and n is the number of available correspondences.
In particular for any given correspondence $Xi \leftrightarrow xi$, 3D point X and its correspondent image point x = [x1 x2 1]T, we compute the matrix a as follows:

a = $\begin{bmatrix} \mathbf{0}^T & -\boldsymbol{X}^T & x_2\boldsymbol{X}^T \\ \boldsymbol{X}^T & \mathbf{0}^T & -x_1\boldsymbol{X}^T \end{bmatrix}$

In presence of noise in the data, **normalization** is an essential step before implementing the DLT algorithm. Without normalization, typical image points in homogeneous coordinates xi are of the order $(x, y, w) = (100,100,1)$, where the x and y components are usually much larger than the w component. Therefore, without normalization, the order of magnitude of the second order entries of the matrix A may end up being too big. If the observations are noisy, this may lead to **numeric instability** and to the **divergence of the solution** from the correct result. However, it is worth noticing that for exact data and infinite precision arithmetic the results will be independent of the normalizing transformation. Furthermore, since the normalization nullifies the effect of coordinate changes, the algorithm will be invariant with respect to scale and coordinate origin. In the code, a similarity transformation T2D is used to normalize the image points, while a second similarity transformation T3D is used to normalize space points.

ALGEBRAIC ERROR AND GEOMETRIC ERROR

The DLT algorithm minimizes the so called algebraic error, meaning that the vector p, containing the components of the projection matrix is obtained by minimizing ||Ap|| under the constraint ||p|| = 1. If we calculate the 2-norm of the matrix P_hat estimated by the EstimateProjectionMatrix function, we can observe that ||P_hat|| = 1, in accordance to the constraint. The solution is obtained from the unit singular vector of A corresponding to the smallest singular value. Therefore given a set of correspondences, the quantity $\varepsilon = Ap$ is the algebraic error vector, whose norm is the algebraic distance. With a proper choice for data normalization, minimization of the algebraic error can yield excellent results. The advantages are a linear (and thus unique solution) and computational cheapness. As already mentioned, the solution can be easily computed via SVD decomposition. The main drawback of using the algebraic error is that it is neither geometrically not statistically meaningful.That is why, we have used P_hat only as a starting point for non-linear minimization. The geometric solution is generally considered the most plausible solution, since the assumptions about the noise are the most sensible in the majority of the case. Penalizing the squared distance between the 2D observation and the projection of the 3D point amounts to assuming that the most relevant contribution to the noise arises from the imaging process (camera / lens / sensor imperfections). We furthermore assume that this noise is i.i.d Gaussian distributed in the image plane.



Let Xi be a 3D point and xi be its observation. The plane w contains Xi and it is parallel to the image plane. The algebraic error is Δ, the distance between Xi and the back projection ray in the plane w . The geometric error d is the distance between xi and projection of Xi onto the image plane, f. Note that as the 3D point moves farther from the camera, the algebraic error increases, while the geometric error remains constant.

In contrast, roughly speaking, the algebraic error measures the distance between the known 3D point and the observation's backprojection ray. This implies errors arise from noise in 3D points as opposed to the camera itself, and tends to overemphasize distant points when finding a solution.
Generally speaking, the choice of error function can be seen as a trade-off between quality and speed. The minimization of the geometric cost function is a non-linear optimization problem, where the cost function grows linearly with the number of parameters.

```
Reprojection error before optimization: 0.0006316426059796243

Reprojection error after optimization: 0.0006253538899291337
```

By printing the reproduction error calculated before and after optimization, respectively on P_hat and P_hat_opt matrices, we can see that, after optimization, the reproduction error has been reduced. However, there is not a significant difference before and after optimization.

DENORMALIZING THE PROJECTION MATRIX

To normalize the point coordinates, two similarities transformation T2D, T3D were defined. In particular:

$$xnorm = T2D * x$$
$$Xnorm = T3D * X$$

If P norm is the projection matrix such that:

$$xnorm = Pnorm * Xnorm,$$

Then, we can substitute xnorm and Xnorm from the previous equations:

$$T2D * x = Pnorm * T3D * X$$

To conclude:

$$x = (T2D)^{-}1 * Pnorm * T3D * X.$$

That means:

$$P = (T2D)^{-}1 * Pnorm * T3D$$

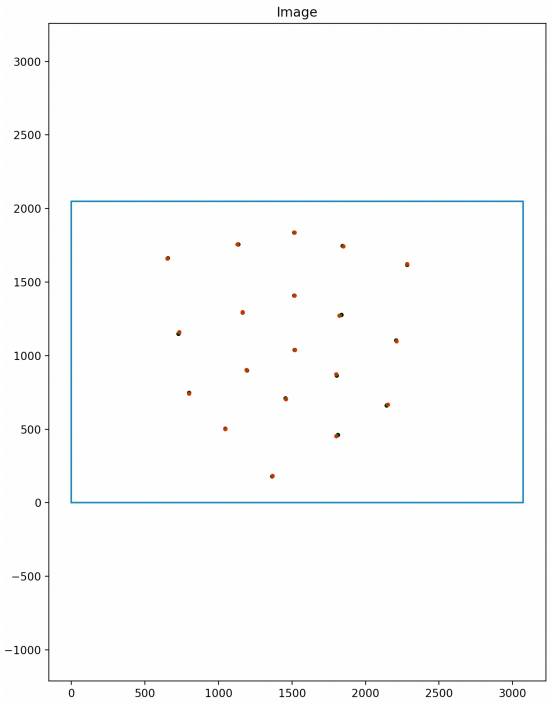
DECOMPOSING THE PROJECTION MATRIX

After the decomposition of the estimated P matrix, via the QR decomposition, the following values for K,R,t were obtained:

```
K=[[2.713e+03 3.313e+00 1.481e+03]
 [0.000e+00 2.710e+03 9.654e+02]
 [0.000e+00 0.000e+00 1.000e+00]]

R =[[-0.774  0.633 -0.007]
 [ 0.309  0.369 -0.877]
 [-0.552 -0.681 -0.481]]

t = [[0.047 0.054 3.441]]
```



At first sight the results seem indeed **reasonable**.

The K matrix is of course **upper-triangular** the element(3,3) equals 1 as expected. The elements (1,1) and (2,2)) are equal, which is a correct result assuming the **density of pixels** in image coordinates is the same both in the x and y directions. Finally, the coordinates of the **principal point** in the last column are located almost in the middle of the image, which is once again a reasonable result. The principal point is, in fact, the point where the principal axis meets the image plane. Finally, the **skew factor** in position (1,2) is almost negligible if compared to the elements on the diagonal.

We have calculated the t vector by setting:

$$t = -RC$$

where C represents the coordinates of the camera centre in the world coordinate frame and R is a 3 × 3 rotation matrix representing the orientation of the camera coordinate frame. That means that **t is the vector between the origin and camera centre expressed in the camera frame**. Since the z-axis of the camera points towards the image plane, it is reasonable that the we almost have a pure translation on the camera z-axis. The z- coordinate of the t vector should be way larger than the x and y coordinates. We notice that in our t-vector the z-coordinate is two order of magnitude bigger than the x and y coordinates.
In addition, we can also compute:
C = [1.918 2.294 1.703],
which is consistent with the plotted 3D scene.

SFM

As suggested in the assignment, DLT has to be used to reformulate the constraint:

$$x'_1{}^T E \; x'_2 = 0$$

Here $x1 \leftrightarrow x2$ are corresponding points in the two images. Moreover,

$x' = K^{-1}x = [R \mid t] X$

Thus, x' is the image point expressed in **normalized coordinates.** It may be thought as the image of the point X with respect to a camera [R | t] having the identity matrix as calibration matrix. Such a camera is called a **normalized camera matrix**. This procedure aims at removing the effects of the known calibration.

Let us the define two corresponding image points represented in normalized image coordinates x'1 = [x'1 y'1 1], x'2 = [x'2 y'2 1]. The constraint can be rewritten as:

$$x'1x'2e11 + x'1y'2e12 + x'1e13 + y'1x'2e21 + y'1y'2e22 + y'1e23 + x'2e31 + y'2e32 + e33 = 0$$

or **e * $\bar{X}$ = 0**

where: **$\bar{X}$ = [x'1x'2 x'1y'2 x'1 y'1x'2 y'1y'2 y'1 x'2 y'2 1]**

e = [e11 e12 e13 e21 e22 e23 e31 e32 e33]

After having obtained an estimate from DLT it is necessary to fulfill the essential matrix constraints. For a proper essential matrix, the first singular values need to be equal (and > 0), and the third one is 0. Since *E* is up to scale, we can set the first singular values to 1. After the estimation of the E matrix, it is necessary to pick the correct relative pose between the two cameras. At first we can set one camera's pose to [Id I 0]. By doing that, we define the new coordinate system to be aligned with this camera. Then, to pick the correct pose of the other camera, we triangulate points with each pose and pick the one that yields the most point in front of both the cameras. Then the final step is to triangulate new points with all pairs of registered images. Following a suggestion on Moodle, I have preferred to change the initial pair of images to 5 and 6, since they yield better results for the reconstruction. This behavior is probably due to the lack of the non-linear optimization for the pose estimation.