Computer Vision, Assignment 2
Calibration and DLT

1 Instructions

In this assignment you will study camera calibration, the projective ambiguity, and the DLT method. You will solve the resection and triangulation problems using DLT and compute inner parameters using RQ factorization. In addition you will try out SIFT for feature detection/matching.

Please see Canvas for detailed instructions on what is expected for a passing/higher grade. All exercises not marked **OPTIONAL** are “mandatory” in the sense described on Canvas.

The maximum amount of points for the theoretical exercises in Assignment 2 is 24. 50% of 24 is 12.

2 Calibrated vs. Uncalibrated Reconstruction.

Theoretical Exercise 1 (4 points). Show that when estimating structure and motion (3D points and cameras) simultaneously, under the assumption of uncalibrated cameras (i.e. unknown calibration matrices), there is a projective ambiguity so that the solution can only be recovered up to an additonal unknown projective transformation of 3D space (which cannot be determined using only image projections). That is, if $\mathbf{X}$ are the estimated 3D-points, show that a new solution with identical image projections can always be obtained from $T\mathbf{X}$ for any projective transformation T of 3D space. (Hint: Look at the camera equations.)

For the report: Submit a complete (but short) solution.

Computer Exercise 1. Figure 1 shows an image of a scene and a reconstruction using uncalibrated cameras. The file `compEx1data.mat` contains the 3D points of the reconstruction `X`, the camera matrices `P`, the image points `x` and the filenames `imfiles` of the images. Here `X` is a 4×9471 matrix containing the homogeneous coordinates for all 3D points, `x{i}` is a 3×9471 matrix containing the homogeneous coordinates of the image points seen in image i (NaN means that the point has not been detected in this image). `P{i}` contains the camera matrix of image i and `imfiles{i}` contains the name of that image.

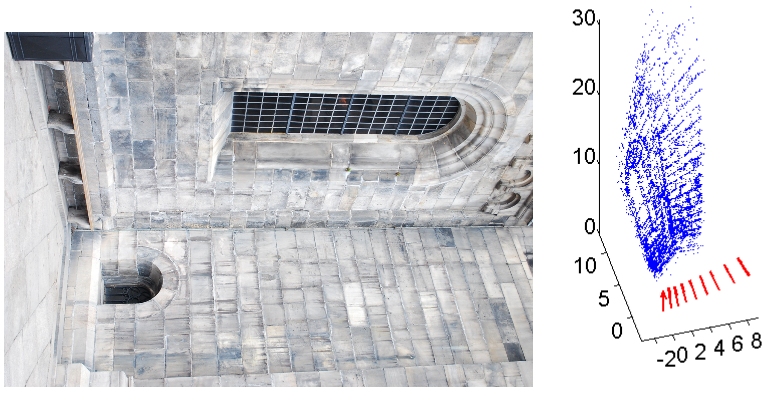

Figure 1: Left: An image of the scene. Right: A projective (uncalibrated) reconstruction.

Plot the 3D points of the reconstruction. Using the file `plotcams.m` to plot the cameras in the same figure. Do the physical properties (such as the relative heights of each of the two walls, the angle in the corner, and the relation between the width, height, and depth dimensions) look realistic in the reconstruction? Don't forget to use `axis equal` when plotting – otherwise you may get additional distortion.) The reconstruction should look like the one shown in Figure 1.

Project the 3D points into a camera of your choice (only points that have been detected in that camera). Plot the image, the projected points, and the image points in the same figure. Do the projections appear to be close to the corresponding image points? (If not: Did you forget to divide by the third coordinate?)

Using the two projective transformations

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/8 & 1/8 & 0 & 1 \end{pmatrix} \text{ and } T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/16 & 1/16 & 0 & 1 \end{pmatrix}, \tag{1}$$

transform the 3D points ($\mathbf{X} \mapsto T_i\mathbf{X}$), and accordingly the cameras (as in Exercise 1), so that two new projective solutions are obtained. For each of the solutions, plot the 3D points and cameras in the same figure. (Don't forget to divide the points by the fourth coordinate before plotting, and again don't forget to use `axis equal`. Feel free to use your `pflat.m` function.) What has happened to the 3D points? Does any of them appear reasonable?

For one of the 9 views, and for each of the new reconstructions (resulting from T_1 and T_2), project the new 3D points into its corresponding transformed camera. Plot the pair of images along with the corresponding projected points, and the image points in the same figure. (Write a function `project_and_plot(P, Xs, image)` that does this for you, it will be useful in the future.) How well do the projections and the image points align for the new reconstructions?

```
Useful matlab commands:

im = imread(imfiles{i}); %Reads the imagefile with name in imfiles{i}

visible = isfinite(x{i}(1,:));
% Determines which of the points are visible in image i

plot(x{i}(1,visible), x{i}(2,visible),'*');
%Plots a '*' at each point coordinate

plot(xproj(1,visible), xproj(2,visible),'ro');
%Plots a red 'o' at each visible point in xproj

plot3(X(1,:),X(2,:),X(3,:),'.','Markersize',2);
%Plots a small '.' at all the 3D points.
```

For the report: Submit the m-file, the 3D-plots, image plots with reprojected points, and answers to the questions.

Theoretical Exercise 2 (6 points). Explain why we can not get the same projective distortions as in Computer Exercise 1 when we use calibrated cameras (i.e. calibration matrices are known). Furthermore, how is the projective ambiguity (described in Exercise 1) affected when the cameras are calibrated – is there still some ambiguity to the reconstruction?

For the report: Submit a short explanation.

3 Camera Calibration

Theoretical Exercise 3 (8 points). Suppose that a camera has got the inner parameters

$$K = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

Verify that the inverse of K is

$$K^{-1} = \begin{pmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

and that this matrix can be factorized into

$$K^{-1} = \underbrace{\begin{pmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}}_{=B}. \tag{4}$$

What is the geometric interpretation of the the transformations A and B?

When normalizing the image points of a camera with known inner parameters we apply the transformation K^{-1}. What is the interpretation of this operation? Where does the principal point (x_0, y_0) end up? And where does a point with distance f to the principal point end up?

Suppose that for a camera with resolution 800×600 pixels we have the inner parameters

$$K = \begin{pmatrix} 400 & 0 & 400 \\ 0 & 400 & 300 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

Normalize the points $(0, 300)$, $(800, 300)$.

What is the angle between the viewing rays projecting to these points?

Show that the camera $K[R\ t]$ and the corresponding normalized version $[R\ t]$ have the same camera center and principal axis (viewing direction). HINT: For a camera matrix P, note that the null space of P defines the camera center, and let the last row P_{31}, P_{32}, P_{33} define the principal axis. NOTE: There is a special family of camera matrices known as affine cameras, for which the last row is 0, and for which the principal axis can not be determined in this way. However, you can disregard this, as the calibration matrix structure given above can not yield any affine camera.

For the report: Complete solution.

4 RQ Factorization and Computation of K

Theoretical Exercise 4 (4 points). Consider an upper triangular matrix

$$K = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}, \tag{6}$$

where the diagonal elements of K are positive. Verify by matrix multiplication that

$$KR = \begin{pmatrix} aR_1^T + bR_2^T + cR_3^T \\ dR_2^T + eR_3^T \\ fR_3^T \end{pmatrix}, \text{ where } R = \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix}. \tag{7}$$

If

$$P = \begin{pmatrix} 2400\sqrt{2} & 0 & 800\sqrt{2} & 4000 \\ 700\sqrt{2} & 2800 & -700\sqrt{2} & 4900 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 3 \end{pmatrix}, \tag{8}$$

what is R_3 (R is an orthogonal matrix) and f?

Using the second row of the camera matrix, and the fact that $R_3 \perp R_2$, $||R_3|| = ||R_2|| = 1$ compute R_2,d,e. (Hint: If $v = dR_2 + eR_3$ how do you compute the coefficient e?)

Similarly, using the first row compute, R_1,a,b and c. What is the focal length, skew, aspect ratio, and principal point of the camera?

For the report: Complete solution.

5 Direct Linear Transformation DLT

Theoretical Exercise 5 (**OPTIONAL,** 15 **optional points**). Show that the linear least squares system

$$\min_v ||Mv||^2 \tag{9}$$

always has the minimum value 0. In the DLT algorithm we use the least squares system

$$\min_{||v||^2=1} ||Mv||^2 \tag{10}$$

to remove the zero solution. Show that if M has the singular value decomposition $M = U\Sigma V^T$ then

$$||Mv||^2 = ||\Sigma V^T v||^2 \tag{11}$$

and

$$||V^T v|| = 1 \text{ if } ||v||^2 = 1. \tag{12}$$

Consider a new optimization problem

$$\min_{||\tilde{v}||^2=1} ||\Sigma \tilde{v}||^2. \tag{13}$$

Explain why (13) gives the same minimal value as (10). How can you obtain a solution to the first problem from the second (determine v^* given $\tilde{v}^*$)? Also, explain why there are always at least two solutions to these (equivalent) problems.

Finally, for the $m \times n$ matrix M, prove that the last column of V is an explicit solution to (10), in the case when $\text{rank}(M) < n$. Consider what happens when $m \geq n$, as well as when $m < n$. HINT: The singular values are in decreasing order on the diagonal of Σ. NOTE: The assumption $\text{rank}(M) < n$ is in fact not necessary but the general case is more difficult to prove, see lecture notes.

For the report: Complete solution.

Theoretical Exercise 6 (2 points). When using DLT it is often advisable to normalize the points before doing computations. Note that normalizing the points in this context should not be confused with the calibration of cameras and image points, which can also be referred to as normalization. Instead, this is just a technique which in practice yields higher quality solutions when using the DLT method.

Suppose the image points $\mathbf{x}$ are normalized by the mapping N by

$$\tilde{\mathbf{x}} \sim N\mathbf{x} \tag{14}$$

and that we compute a camera matrix in the new coordinate system, that is, we obtain a camera $\tilde{P}$ that solves

$$\tilde{\mathbf{x}} \sim \tilde{P}\mathbf{X}. \tag{15}$$

How do you compute the camera P that solves the original problem

$$\mathbf{x} \sim P\mathbf{X} \tag{16}$$

from $\tilde{P}$?

For the report: It is a very simple exercise, just give the formula.

Computer Exercise 2. Figure 2 shows two images `cube1.JPG` and `cube2.JPG` of a scene with a Rubik's cube. The file `compEx3data.mat` contains a point model `Xmodel` of the visible cube sides, the measured projections `x` of the model points in the two images and two variables `startind`, `endind` that can be used for plotting lines on the model surface.

The goal for this exercise is to determine the camera for each of the two views, i.e. resectioning. You should consider one view at a time, i.e. apply your code for each view separately.

Normalize the measured points by applying a transformation N that subtracts the mean of the points and then re-scales the coordinates by the standard deviation. Here, compute mean and standard deviation for x and y separately (separating the coordinates is not crucial, but it helps during grading if you do this in a coherent way). Plot the normalized points in a new figure. Verify that the points are centered around $(0, 0)$ with standard deviation 1 for each of the coordinates.

Implement a function `estimate_camera_DLT` that sets up the DLT equations for resectioning, and solves the resulting homogeneous least squares system (10) using SVD. Is the smallest singular value close to zero? How about $||Mv||$?

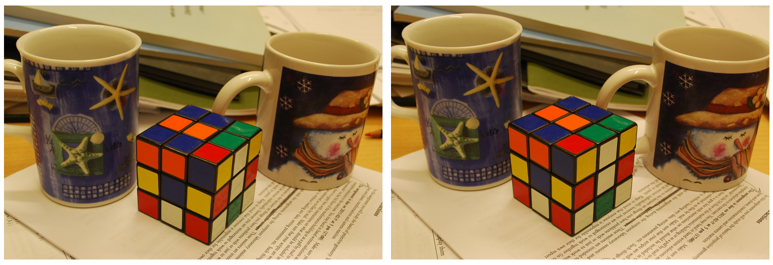
Figure 2: Two images of a scene containing a Rubik's cube.

Extract the entries of the camera from the solution and set up the camera matrix. Make sure that you select the solution where the points are in front of the camera. (If X has 4th coordinate 1 then the 3rd coordinate of PX should be positive for X to be in front of the camera.)

Project the model points into the images. (Don't forget to transform the camera matrix to the original (un-normalized) coordinate system, as in Exercise 6, before projecting.) Plot the measured image points in the same figure. Are they close to each other? Also make a 3D plot visualizing the camera centers, principal axes, as well as the 3D model points. Does the result look reasonable?

Compute the inner parameters of the first camera using the provided function `rq.m` that does RQ decomposition. How can we know that these are the "true" parameters? Why is there no ambiguity as in Exercise 1?

When you have achieved satisfactory results, save the camera matrices (for both views) to be used in further exercises.

The rest of this exercise: (**OPTIONAL,** 10 **optional points**)
You will now re-run your experiments in a few different settings to be compared, investigating the importance of normalization. You only need to do this part for the first view (corresponding to 2D points `x{1}`). For each experiment, measure the performance with the RMS error

$$e_{RMS} = \sqrt{\frac{1}{n}||x_{meas} - x_{proj}||_F^2}, \tag{17}$$

where x_{meas}, x_{proj} are the Cartesian coordinates ($2\times n$-matrices) for the measured points and projected model points respectively, and n is the number of points. Here $||\cdot||_F^2$ denotes the (squared) Frobenius norm, in MATLAB `norm(dx,'fro').^2`.

First compute RMS values when using normalization (as you have done so far). Next, repeat the whole resectioning experiment one more time, but this time don't normalize the points. (The easiest way of doing this is probably to use $N = I$ as normalization matrix and run the same code again.) Is the difference large? Now repeat the resectioning experiments (with and without normalization), but this time estimate the camera using only points number $1, 4, 13, 16, 25, 28, 31$. Still, measure the RMS error in a unified way, always using all points!. What can you conclude from the experiments that you have run?

```
Useful matlab commands:

mean(x{i}(1:2,:),2) %Computes the mean value of the 1st and 2nd rows ow x{i}

std(x{i}(1:2,:),0,2) %Standard deviation of the 1st and 2nd rows ow x{i}

[U,S,V] = svd(M); %Computes the singular value decomposition of M

P = reshape(sol(1:12),[4 3])';
%Takes the first 12 entries of sol and row-stacks them in a 3x4 matrix

plot3([Xmodel(1,startind); Xmodel(1,endind)],...
        [Xmodel(2,startind); Xmodel(2,endind)],...
        [Xmodel(3,startind); Xmodel(3,endind)],'b-');
%Plots the lines of the cube model (works only if all points are included)
```

For the report: Submit the m-file, and the figures for both cameras (normalized coordinates, 3D-plot, and reprojected points). Also specify the mean and standard deviation you used for normalizing x and y, for each camera, as well as the calibration matrix of the first camera, normalized s.t. $K_{33} = 1$. Explain why there is no ambiguity. For the OPTIONAL part, report RMS numbers and explain your findings.

6 Feature Extraction and Matching using SIFT

Computer Exercise 3. In this exercise you will get to try feature extraction using SIFT.

First you will have to download and start VLFeat. Go to `http://www.vlfeat.org/download.html` and extract the binary package to a directory of your choice, e.g. `H:\vlfeat`. Then start Matlab, go the `H:\vlfeat\toolbox` subdirectory and run `vl_setup`. Now you should see the following message:

```
** Welcome to the VLFeat Toolbox **
```

You will now be able to use VLFeat throughout this Matlab session.

First load the images `cube1.jpg` and `cube2.jpg` from Exercise 2.

Compute sift features using vlfeat.

```
[f1 d1] = vl_sift( single(rgb2gray(im1)), 'PeakThresh', 1);
```

The SIFT detector searches for peaks in scale space (similar to peaks in the autocorrelation function, see lecture notes). The second argument filters out peaks that are too small.

The vector `f1` contains 4 rows. The first two rows are the coordinates of the detected features. The second two contains an orientation and scale for which the the feature was detected. Plot the features together with the images using the command:

```
vl_plotframe(f1);
```

Compute the features for the second image and match the descriptors using the command:

```
[matches ,scores] = vl_ubcmatch(d1,d2);
```

Thus for each descriptor in image 1 the function finds the two best matches. If the quality of the best and the second best match is similar then the match is rejected, otherwise the best match is selected.

We can now extract matching points using:

```
x1 = [f1(1,matches(1,:));f1(2,matches(1,:))];
x2 = [f2(1,matches(2,:));f2(2,matches(2,:))];
```

The following code randomly selects 10 matches, plots the two images next to each other and plots lines between the matching points.

```
perm = randperm(size(matches ,2));
figure;
imagesc([im1 im2]);
hold on;
plot([x1(1,perm(1:10)); x2(1,perm(1:10))+size(im1,2)], ...
        [x1(2,perm(1:10)); x2(2,perm(1:10))],'-');
hold off;
```

How many of the matches appear to be correct?

For the report: Nothing, but you will need the point sets $x1$ and $x2$ for the next exercise.

7 Triangulation using DLT

Computer Exercise 4. Using the estimated cameras from Computer Exercise 2 you will now triangulate the points detected in Computer Exercise 3.

Implement a function `triangulate_3D_point_DLT` that set ups the DLT equations for triangulation, and solves the homogeneous least squares system. (You will have to do this in a loop, once for each point.)

Project the computed points into the two images and compare with the corresponding SIFT-points in a 2D plot.

Now, re-run the triangulation, but first normalize 2D points and cameras using the inverse of K. Again, compare with the SIFT points in a plot. Just as when normalizing with N (based on mean/std) for the resection problem in Computer Exercise 2, you should expect a better result from normalization. Note that using K or N for normalization doesn't matter much, but K was not available for the previous problem (resection).

From now on, consider only the normalized case.

As we saw in Computer Exercise 3, a portion of the SIFT matches will be incorrect. Most of the time (but not always) this will result in triangulations with large error. Compute the average pixel error (for each of the two images) between the projected 3D points and the corresponding SIFT points (remember to multiply with K to retrieve pixel coordinates rather than normalized coordinates). Remove those points for which the error in at least one of the images is larger than 3 pixels. Plot the remaining 3D points, the cameras and the cube model in one and the same 3D plot. Can you distinguish the dominant objects (the cups and the paper)?

```
Useful matlab commands:

xproj1 = pflat(Pa*X);
xproj2 = pflat(Pb*X);
%Computes the projections

good_points = (sqrt(sum((x1-xproj1(1:2,:)).^2)) < 3 &  ...
                sqrt(sum((x2-xproj2(1:2,:)).^2)) < 3);
%Finds the points with reprojection error less than 3 pixels in both images

X = X(:,good_points);
%Removes points that are not good enough.
```

For the report: Submit the m-file, and the plots.