Computer Vision, Assignment 4
Model Fitting and Local Optimization

1 Instructions

In this assignment you will first study model fitting, in particular estimating essential matrices. You will use random sampling consensus RANSAC to robustly estimate essential matrices. Secondly, you will use study local optimization. You will use Levenberg-Marquardt to refine 3D points obtained from an initial solution to the SfM problem. After completing this assignment you will have a simple functional SfM pipeline.

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 4 is 16. 50% of 16 is 8.

2 Robust Epipolar Geometry and Two-View Reconstruction

In this first section of Assignment 4, you will work on robust estimation using RANSAC. Refer to Lecture Slides 7 for material covering RANSAC.

Theoretical Exercise 1 (**3 points**).

Suppose that two calibrated cameras are represented by the camera matrices $P_1 = [R_1\ t_1]$ and $P_2 = [R_2\ t_2]$. What is the essential matrix for this pair of cameras?

For the report: Complete solution.

Theoretical Exercise 2 (**5 points**).

Suppose that we want to find an essential matrix that relates two sets of 2D points. How many degrees of freedom does an essential matrix have?

What is the minimal number of point correspondences that you need to determine E? How many point correspondences are required by the eight point solver to determine E?

If the proportion of incorrect correspondences is 25%, how many iterations of RANSAC with an eight point solver do you need to find an outlier free sample set with 99% probability?

For the report: Answers are enough.

Computer Exercise 1.

Figure 1 shows two views of a round church and a set of 3D points that can be obtained with a

Figure 1: Image round_church1.png, round_church2.png and an example of the obtained 3D reconstruction.

3D reconstruction pipeline (just for visualization purposes). The goal of this exercise is to robustly estimate essential matrix E from point matches between the two images. For this, we will use RANSAC with an eight point solver.

The file `compEx1data.mat` contains calibration matrix K, which is the same for both images, and a cell $\mathbf{x}$ with matched image points for the two images. Note that $\mathbf{x}$ are noisy and contain some fraction of outliers.

First, solve for the essential matrix E with an eight point algorithm using all the point correspondences. Remember to normalize image points with K beforehand. Then convert the essential matrix E into the fundamental matrix F for the un-normalized coordinate system. Recall that in Assignment 3, you created functions `estimate_F_DLT(x1, x2)`, `enforce_essential(E_approx)`, and `convert_E_to_F(E,K1,K2)`. You are encouraged to re-use them in this computer exercise.

Compute the epipolar lines $\mathbf{l}_2 = F\mathbf{x}_1$ and $\mathbf{l}_1 = F^T\mathbf{x}_2$. Compute the RMS distance between the image

points ($\mathbf{x}_1$ and $\mathbf{x}_2$) and corresponding epipolar lines ($\mathbf{l}_1$ and $\mathbf{l}_2$, respectively)

$$e_{RMS} = \sqrt{\frac{1}{2m}\left(\sum_{i=1}^{m} d^2(\mathbf{x}_{1i}, \mathbf{l}_{1i}) + \sum_{i=1}^{m} d^2(\mathbf{x}_{2i}, \mathbf{l}_{2i})\right)},$$
$$\text{where} \quad d(\mathbf{x}, \mathbf{l}) = \frac{|l_1 x_1 + l_2 x_2 + l_3|}{\sqrt{l_1^2 + l_2^2}}. \tag{1}$$

In Assignment 3, you implemented the distances $d(\mathbf{x}, \mathbf{l})$ from image points to the corresponding epipolar lines for the second image (function `compute_epipolar_errors(F, x1s, x2s)`). You can re-use it to compute the errors in the first image by calling `compute_epipolar_errors(F', x2s, x1s)`. Plot two separate histograms with 100 bins for the epipolar errors $\{d(\mathbf{x}_{1i}, \mathbf{l}_{1i})\}$ in the first image and $\{d(\mathbf{x}_{2i}, \mathbf{l}_{2i})\}$ in the second image, respectively.

Pick 20 points in the first image at random and plot these in the same figure as the image. Also plot the corresponding epipolar lines in the same image using the function `rital.m`. Repeat the plot for the second image. Do the plots look reasonable, are points close to the epipolar lines?

Now, use RANSAC to robustly estimate E. Create a function `estimate_E_robust(K,x1,x2)` that does it. Note that for this problem setting, a point match $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$ represents a single measurement. Use the same eight point solver, however, note that now only a subset of matches will be used by the solver. As a squared error of the estimated essential matrix E with respect to the point match $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$, use $\frac{1}{2}\left(d^2(\mathbf{x}_{1i}, \mathbf{l}_{1i}) + d^2(\mathbf{x}_{2i}, \mathbf{l}_{2i})\right)$. A suggested inlier threshold is 2px. How many inliers did you get?

Again, compute the RMS distance between the image points and corresponding estimated epipolar lines in both images. Also plot a histogram of epipolar errors for both images as before. Which is the better estimate of the essential matrix, and why? Repeat plotting 20 random points with the corresponding epipolar lines, but this time pick random points from the subset of inliers.

```
Useful matlab commands:

E = enforce_essential(estimate_F_DLT(x_norm1(:,randind), x_norm2(:,randind)));
%Computes essential matrix from a sample set.

err_threshold = err_threshold_px / K(1,1);
inliers = (compute_epipolar_errors(E, x_norm1, x_norm2).^2 + ...
    compute_epipolar_errors(E', x_norm2, x_norm1).^2) / 2 < err_threshold^2;
%Finds the indices for which the symmetrized epipolar error is less than a
%pre-defined threshold re-scaled by the focal length.


RMS_error = rms(all_errors); %Computes the RMS error
```

For the report: Submit the m-file as well as RMS values, figures with epipolar lines and histogram plots (both when using all points and for RANSAC). Finally, provide answers to all questions.

Computer Exercise 2.

In this exercise you will build a 2-view reconstruction pipeline that connects feature extraction, matching, robust essential matrix estimation from the previous CEx, and triangulation.

You can use the two images `fountain1.png` and `fountain2.png` (see Figure 2), and the intrinsics calibration matrix K provided in `compEx2data`, but feel free to use other images if you want to (as long as you have access to the corresponding intrinsics calibration matrix K). You will need to use VLfeat as in Assignment 2 to generate potential matches, and then determine inliers using RANSAC.

Begin by loading the two images in Matlab and display them. Use VLFeat to compute SIFT features for both images and match them.

Figure 2: Image fountain1.png, fountain2.png and an example of the obtained 3D reconstruction.

```
Useful matlab commands:

[fA dA] = vl_sift( single(rgb2gray(A)) );
[fB dB] = vl_sift( single(rgb2gray(B)) );

matches = vl_ubcmatch(dA,dB);

xA = fA(1:2,matches(1,:));
xB = fB(1:2,matches(2,:));
```

How many SIFT features did you find for the two images, respectively? How many matches did you find?

Now you should find the essential matrix describing the transformation between the two images. Because not all matches are correct, you need to use RANSAC to find a set of good correspondences (inliers). To estimate the essential matrix use the function `estimate_E_robust(K,x1,x2)` that you created in the previous computer exercise.

How many inliers did you find?

After getting the robust essential matrix estimation, you should find the camera matrix of the second view. Remember that there are 4 possible solutions (see Lecture 6 or the Theoretical Exercise 7 of HA3)! You should pick the solution that has more points in front of the camera.

Which of the solutions seems correct to you?

In summary, the 2d-view reconstruction pipeline should consist of the following steps:

1. Load the two images, find SIFT features and match them;
2. Use the function `estimate_E_robust(K,x1,x2)` to robustly estimate the essential matrix between the two views;
3. Find the 4 possible camera matrices pairs for the essential matrix that you estimated, and for each them:
   (a) Triangulate the 3D points using the camera matrix pair and image points;
   (b) Get the camera centers and principal direction of both cameras;
   (c) Plot everything in 3D;
4. Pick the solution (out of the 4 possibilities) that has more 3D points (inliers) in front of the camera. The final results should look similar to the right-most image in Figure 2 after removing most of the outliers.

For the report: The m-file, a plot of the reconstructions for the four possible solutions, and answers to all the questions.

3 Levenberg-Marquardt for Structure from Motion Problems

In this second section of Assignment 4, you will study Levenberg-Marquardt for optimizing non-linear least squares objectives, in particular minimizing the reprojection error of a SfM solution. Refer to Lecture Slides 9 for related material.

Suppose that the 2D-point $x_{ij} = (x_{ij,1}, x_{ij,2})$ is an observation of the 3D-point $\mathbf{X}_j$ in camera P_i. The model configuration (points and cameras) that maximizes the likelihood of obtaining the observations $x_{ij} = (x_{ij,1}, x_{ij,2})$ under the assumption of i.i.d. Gaussian noise on the observations is then found by minimizing the reprojection error, i.e. solving

$$\arg\min_{P_i, \mathbf{X}_j} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \left(x_{ij,1} - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j},\ x_{ij,2} - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j} \right) \right\|^2 . \tag{2}$$

Here P_i^ℓ denotes the ℓ'th row of P_i, and we write $\mathbf{X}_j$ in homogeneous coordinates. The solution to (2) is called the Maximum Likelihood (ML) estimate. Unfortunately there is no closed-form solution for computing the ML estimate when we use general pinhole cameras. The only way to find the ML estimate is to try to improve a starting solution using local optimization.

Theoretical Exercise 3 (8 points). In this exercise you will derive the Jacobian and residual expressions needed for the refinement of the 3D points using the Levenberg-Marquardt method. You will then in the next exercise implement these formulas and refine the 3D points you obtained in Computer Exercise 2.

We write the reprojection error for a particular 3D point $\mathbf{X}_j$ as

$$\sum_{i=1}^{m} \|\mathbf{r}_i(\mathbf{X}_j)\|^2 \tag{3}$$

where

$$\mathbf{r}_i(\mathbf{X}_j) = (r_{i,1}(\mathbf{X}_j), r_{i,2}(\mathbf{X}_j)) = \left(x_{ij,1} - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j},\ x_{ij,2} - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j} \right) \tag{4}$$

is the residual vector of $\mathbf{X}_j$ in image i with individual components $r_{i,1}, r_{i,2}$. In total we get two residuals per image and 3D point, so $2 \cdot m$ residuals for every 3D point.

When using Levenberg-Marquardt (and related methods) for minimizing (3) we approximate the residual vector at an updated 3D point $\mathbf{X}_j + \delta\mathbf{X}_j$ using a first order Taylor approximation at $\mathbf{X}_j$, i.e.,

$$\mathbf{r}_i(\mathbf{X}_j + \delta\mathbf{X}_j) \approx \mathbf{r}_i(\mathbf{X}_j) + J_i(\mathbf{X}_j)\delta\mathbf{X}_j, \tag{5}$$

where $J_i(\mathbf{X}_j)$ is the Jacobian of $\mathbf{r}_i(\mathbf{X}_j)$.

a) Show that the Jacobian matrix of $\mathbf{r}_i(\mathbf{X}_j)$ relative to $\mathbf{X}_j$ for camera i is a 2×4 matrix given by

$$J_i(\mathbf{X}_j) = \begin{bmatrix} \frac{(P_i^1 \mathbf{X}_j)}{(P_i^3 \mathbf{X}_j)^2} P_i^3 - \frac{1}{P_i^3 \mathbf{X}_j} P_i^1 \\ \\ \frac{(P_i^2 \mathbf{X}_j)}{(P_i^3 \mathbf{X}_j)^2} P_i^3 - \frac{1}{P_i^3 \mathbf{X}_j} P_i^2 \end{bmatrix}. \tag{6}$$

HINT: To simplify the derivations you can use an intermediate representation $Z_{ij} = P_i \mathbf{X}_j$. This will allow you to decompose the calculation into two (simpler) parts by using the chain rule for derivatives. For instance, for the first row of J_i, i.e. the derivative of $r_{i,1}$ w.r.t. $\mathbf{X}_j$, using $Z_{ij} = P_i \mathbf{X}_j$ and the chain rule we get

$$\frac{\partial r_{i,1}}{\partial \mathbf{X}_j} = \frac{\partial r_{i,1}}{\partial Z_{ij}} \frac{\partial Z_{ij}}{\partial \mathbf{X}_j}, \tag{7}$$

where $\frac{\partial r_{i,1}}{\partial Z_{ij}}$ (1×3 vector) and $\frac{\partial Z_{ij}}{\partial \mathbf{X}_j}$ (3×4 matrix) are much easier to find than $\frac{\partial r_{i,1}}{\partial \mathbf{X}_j}$. A first step would be to write (4) replacing $Z_{ij} = P_i \mathbf{X}_j$ and find $\frac{\partial r_{i,1}}{\partial Z_{ij}}$ and $\frac{\partial r_{i,2}}{\partial Z_{ij}}$. For computing $\frac{\partial Z_{ij}}{\partial \mathbf{X}_j}$ you might find the

relation $\frac{\partial(Ax)}{\partial x} = A$ useful, for a matrix A and a vector x. Other (possibly) useful matrix relations can be found in the Matrix Cookbook.

b) Show how to build $\mathbf{r}(\mathbf{X}_j)$ and $J(\mathbf{X}_j)$ from $\mathbf{r}_i(\mathbf{X}_j)$ and $J_i(\mathbf{X}_j)$ such that

$$\sum_{i=1}^{m} \|\mathbf{r}_i(\mathbf{X}_j) + J_i(\mathbf{X}_j)\delta\mathbf{X}_j\|^2 = \|\mathbf{r}(\mathbf{X}_j) + J(\mathbf{X}_j)\delta\mathbf{X}_j\|^2 . \tag{8}$$

In particular, what are the dimensions of $\mathbf{r}(\mathbf{X}_j)$ and $J(\mathbf{X}_j)$?

For the report: Complete derivations.

Computer Exercise 3. In this exercise you will use the solution from Computer Exercise 2 (CE2) as a starting point and locally improve it using the Levenberg-Marquardt method. If you have doubts about the correctness of your solution from the earlier exercise, you can instead use the provided data in `CompEx3.mat` as starting solution.

The goal is to refine the solution **3D points** (not camera matrices, those you can assume fixed). You will loop over the 3D points and update them one by one using Levenberg-Marquardt (LM). LM is an iterative method for minimizing a non-linear least squares objective

$$F(v) = \|\mathbf{r}(v)\|^2, \tag{9}$$

with respect to v. In LM, the update is given by

$$\delta v = -(J(v)^T J(v) + \mu I)^{-1} J(v)^T \mathbf{r}(v), \tag{10}$$

where $J(v)$ is the Jacobian of $\mathbf{r}(v)$ at v. Here $\mu > 0$ is a damping factor that is adjusted adaptively (see Lecture Slides 9) and v is the previous solution. The new solution is $v + \delta v$ and we iterate until convergence (or for a fixed number of iterations). In our setting, we have $v = \mathbf{X}_j$ and $\mathbf{r}(v) = \mathbf{r}(\mathbf{X}_j)$ with the $\mathbf{r}$ from (8).

You will have to implement three functions in this exercise:

- `[err,res] = ComputeReprojectionError(P_1,P_2,X_j,x_1j,x_2j)`: a function that computes the reprojection error (3) given two cameras `P_i`, a 3D point `X_j` and 2D points `x_ij` in the images corresponding to the cameras `P_i` and the 3D point `X_j`. It also returns the values of all the individual residuals as a second output, i.e., the residual vector $\mathbf{r}(\mathbf{X}_j)$;
- `[r,J] = LinearizeReprojErr(P_1,P_2,X_j,x_1j,x_2j)`: a function that computes the linearization ($\mathbf{r}$ and J) as described in (8);
- `delta_X_j = ComputeUpdate(r, J, mu)`: a function that computes the LM-update given $\mathbf{r}$, J and μ as described in (10).

The idea is now to use "decoupled" LM, i.e. loop over the 3D points and refine them one by one.

Compare the total reprojection error (sum of (3) over all 3D points) before and after running LM. Also compare the median reprojection error (median of (3) over all 3D points) before and after running LM. Finally plot the refined 3D points in the same plot as the originals. What do you observe?

```
Useful matlab commands:

% Computes the LM update.
C = J.'*J + mu*eye(size(J,2));
c = J.'*r;
delta = -C\c;
```

For the report: 3D plot, error comparison, m-files.

Computer Exercise 4. (**OPTIONAL, 10 optional points**.) Perform an empirical noise sensitivity analysis of your LM-solver from the previous exercise. Add i.i.d. mean-zero Gaussian noise with standard deviation $\sigma_X \in \{0, 0.1\text{m}\}$ to the 3D points and $\sigma_x \in \{0, 3\text{px}\}$ to the 2D points (from SIFT), yielding (at least) four noise combinations (σ_X, σ_x) to try. See how the total reprojection error and median reprojection error as computed in the previous exercise varies before and after LM with the added noise. If you want to, you can test other noise types as well.

For the report: Report your findings with plots and numbers in some reasonable manner.

Theoretical Exercise 4. (**OPTIONAL, 15 optional points**.) A direction d is called a descent direction of a function $F : \mathbb{R}^N \to \mathbb{R}$ at the point v if

$$\nabla F(v)^T d < 0, \tag{11}$$

since this means that the directional derivative in the direction d is negative (see your multidimensional calculus book).

A matrix M is called positive definite if $w^T M w > 0$ for any w such that $\|w\| \neq 0$. Show that if M is positive definite, the direction

$$d = -M \nabla F(v) \tag{12}$$

is a descent direction for F at v (provided v is not a stationary point).

Is the update chosen in Levenberg-Marquardt (10) in a descent direction for the $F(v)$ in (9)?

For the report: Complete solution.