Computer Vision 2 – Assignment 2
Structure from Motion

Yahui Zhang, Ozzy Ülger

Wednesday 20th April, 2022

Deadline: 06-05-2022, 23:59:59 (Amsterdam time)

General guidelines

Work on the assignments in **groups of 3**. Minor additions and/or changes might occur between the assignment release and deadline. Students will be informed for these changes on Canvas. For any questions or discussions, please also refer to Canvas.

For this assignment, please provide a requirements.txt with all external dependencies listed and a README.txt file with descriptions on how to reproduce results. Make sure to double check your code on bugs. Without running code, we cannot assess your submission, resulting in zero points.

Source code and report must be handed in together as a zip file (ID1_lastname1_ID2_lastname2_ID3_lastname3.zip) before the deadline on Canvas. For full credit, make sure your report follows these guidelines:

- Include an introduction and a conclusion to your report.
- The maximum number of pages is 10 (single-column, including tables and figures). Please express your thoughts concisely. The number of words does not necessarily correlate with how well you understand the concepts. Additional visualisations can be included in an appendix which is not counted in the page count, **but only if they are required to prove/augment/clarify your argument**.
- Answer all given questions. Briefly describe what you implemented. Reports comprised of sole results will reduce your grade.
- Try to understand the problem as much as you can. When answering a question, give evidences (qualitative and/or quantitative results, references to papers, etc.) to support your arguments.
- Analyze your results and discuss them, e.g. why does algorithm A work better than algorithm B in a certain problem?
- Tables and figures must be accompanied by a brief description. Do not forget to add a number, a title, and if applicable name and unit of variables in a table, name and unit of axes and legends in a figure.
- Overall, the purpose for the report here is to learn how to write a proper scientific report/paper in addition to solving the assignment and learning about the problem. So aim for such a format.

Late submissions are not allowed. Assignments that are submitted after the strict deadline will not be graded. In case of submission conflicts, TA's system clock is taken as reference. We strongly recommend submitting well in advance, to avoid last minute system failure issues.

Plagiarism note: Keep in mind that plagiarism (submitted materials which are not your work) is a serious crime and any misconduct shall be punished with the university regulations.

Points: Maximum points are mentioned in the section heading. Total possible points in this assignment is 30.

1 Introduction

In this assignment you will implement the Structure-from-Motion algorithm to recover three-dimensional structures from 2D images. All the information needed to complete the assignment is available in the lecture or tutorial slides on Canvas. Some parts of your code from the image alignment and stitching assignment in the Computer Vision 1 course may be reused.

2 Data

House dataset is provided in the same archive.

3 Fundamental Matrix (9pt)

Without any extra assumption on the world scene geometry, you cannot affirm that there is a projective transformation (homography) between two views. In this assignment, first, you will write a function that takes two images as input and computes the fundamental matrix by the Normalized Eight-point Algorithm with RANSAC. You will work with the supplied house images. The overall scheme of the homography estimation can be summarized as follows:

1. Detect interest points in each image.
2. Characterize the local appearance of the regions around interest points.
3. Get a set of supposed matches between region descriptors in each image.
4. Perform RANSAC to estimate the homography between images.
5. Estimate the fundamental matrix for the given two images.

The first three steps can be performed using the VLFeat functions (You can use the PyVLFeat library, which is a wrapper around the VLFeat library). We recommend to use SIFT features (`https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html` for interest points detection, however others are not prohibited. Make sure to use a recent version of OpenCV.

Hint: Eliminating detected interest points on background would help.

In the next stage, we will introduce in detail a method for estimating fundamental matrix [1]. For $n \geq 8$ known corresponding points' pairs in two stereo images, we can formulate a homogenous linear equation as follows:

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \underbrace{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}}_{F} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \ , \tag{1}$$

where x_i and y_i denote x and y coordinates of the i^{th} point p_i, respectively. Equation 1 can also be written as

$$\underbrace{\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_n' & x_ny_n' & x_n & y_nx_n' & y_ny_n' & y_n & x_n' & y_n' & 1 \end{bmatrix}}_{A} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0 \ , \tag{2}$$

where F denotes the fundamental matrix.

3.1 Eight-point Algorithm (3pt)

- Construct the $n \times 9$ matrix A
- Find the SVD of A: $A = UDV^T$
- The entries of F are the components of the column of V corresponding to the smallest singular value.

An important property of fundamental matrix is that it is singular, in fact of rank two. The estimated fundamental matrix F will not in general have rank two. The singularity of fundamental matrix can be enforced by correcting the entries of estimated F:

- Find the SVD of F: $F = U_f D_f {V_f}^T$
- Set the smallest singular value in the diagonal matrix D_f to zero in order to obtain the corrected matrix ${D'}_f$
- Recompute F: $F = U_f {D'}_f {V_f}^T$

3.2 Normalized Eight-point Algorithm (2pt)

It turns out that a careful normalization of the input data (the point correspondences) leads to an enormous improvement in the conditioning of the problem, and hence in the stability of the result [1]. The added complexity necessary for this transformation is insignificant.

3.2.1 Normalization:

We want to apply a similarity transformation to the set of points $\{p_i\}$ so that their mean is 0 and the average distance to the mean is $\sqrt{2}$.

Let
$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i$,
$m_y = \frac{1}{n}\sum_{i=1}^{n} y_i$,
$d = \frac{1}{n}\sum_{i=1}^{n} \sqrt{(x_i - m_x)^2 + (y_i - m_y)^2}$, and

$$T = \begin{bmatrix} \sqrt{2}/d & 0 & -m_x\sqrt{2}/d \\ 0 & \sqrt{2}/d & -m_y\sqrt{2}/d \\ 0 & 0 & 1 \end{bmatrix},$$

where x_i and y_i denote x and y coordinates of a point p_i, respectively.

Then $\widehat{p_i} = Tp_i$. Check and show that the set of points $\{\widehat{p_i}\}$ with homogeneous coordinates satisfies our criteria. Similarly, define a transformation T' using the set $\{\widehat{p_i}'\}$, and let $\widehat{p_i}' = T'p_i'$.

3.2.2 Find a fundamental matrix:

- Construct a matrix A from the matches $\widehat{p_i} \leftrightarrow \widehat{p_i}'$ and get $\widehat{F}$ from the last column of V in the SVD of A.
- Find the SVD of $\widehat{F}$: $\widehat{F} = U_{\widehat{f}} D_{\widehat{f}} {V_{\widehat{f}}}^T$
- Set the smallest singular value in the diagonal matrix $D_{\widehat{f}}$ to zero in order to obtain the corrected matrix ${D'}_{\widehat{f}}$
- Recompute $\widehat{F}'$: $\widehat{F}' = U_{\widehat{f}} D'_{\widehat{f}} {V_{\widehat{f}}}^T$

3.2.3 Denormalization:

Finally, let $F = {T'}^T \widehat{F}' T$.

3.3 Normalized Eight-point Algorithm with RANSAC (2pt)

Fundamental matrix estimation step given in Section 3.2.2 can also be performed via a RANSAC-based approach. First pick 8 point correspondences randomly from the set $\{\widehat{p_i} \leftrightarrow \widehat{p_i}'\}$, then, calculate a fundamental matrix F, and count the number of inliers (the other correspondences that agree with this fundamental matrix). Repeat this process many times, pick the largest set of inliers obtained, and apply fundamental matrix estimation step given in Section 3.2.2 to the set of all inliers.

In order to determine whether a match $p_i \leftrightarrow p_i{}'$ agrees with a fundamental matrix F, we typically use the Sampson distance as follows:

$$d_i = \frac{(p_i'^T F p_i)^2}{(Fp_i)_1^2 + (Fp_i)_2^2 + (F^T p_i')_1^2 + (F^T p_i')_2^2}, \tag{3}$$

where $(Fp)_j^2$ is the square of the j^{th} entry of the vector Fp. If d_i is smaller than some threshold, the match is said to be an inlier.

To check the fundamental matrix estimation we can also plot their corresponding epipolar lines. The epipolar line can be thought of as the projection of the line on which the point in the other image could have originated from.

3.4 Analysis (2pt)

Draw the epipolar lines based on your estimated fundamental matrix obtained using (1) eight-point algorithm, (2) normalized eight-point algorithm and (3) normalized eight-point algorithm with RANSAC. Do they look plausible? Is Epipolar constraint satisfied? Do you see any improvements? Analyse your findings.

4 Chaining (5pt)

The matching process described in Section 3 is performed across pairs of views. These matches can be represented in a single match graph structure. Intuitively, the set of views of the same surface point forms a connected component of the match graph, which can in turn be used to form a sparse point-view matrix whose columns represent surface points, and rows represent the images they appear in.

4.1 Point-view matrix (3pt)

Construct a point-view matrix for chaining multiple views with the matches found in last step using all consecutive house images (1-2, 2-3, 3-4, ..., 48-49, 49-1). Rows of the point-view matrix will be representing your images while columns will be points. For more details, you can refer to [2].

1. Start from any two consecutive image matches. Add a new column to point-view matrix for each newly introduced point.
2. If a point which is already introduced in the point-view matrix and another image contains that point, mark this matching on your point-view matrix using the previously defined point column. Do not introduce a new column.

Together with the assignment a sample point view matrix (PointViewMatrix.txt) is provided to test and finish your pipeline (for the ones who have difficult time to build a point view matrix). This data has more images than provided for the assignment, however you can still use PVM to check correctness of the last part of your algorithm.

4.2 Analysis (2pt)

Visualize obtained point-view matrix from the house dataset. Analyse the results. Compare it with PointView-Matrix.txt. Does it look plausible? How dense is your point-view matrix? How do you think you can improve results?

5 Structure from Motion (8pt)

In Section 4, you have created the point-view matrix to represent point correspondences for different camera views. You will use this matrix for the affine structure from motion in this part of the assignment.

5.1 Factorize and Stitch (5pt)

The point-view matrix is comparable to the measurement matrix used in factorization procedure of the affine structure from motion. If all the points appeared in all views, we could indeed factorize the matrix directly to recover the points' 3D configurations as well as the camera positions. However, in general, the point-view matrix is sparse, and we must find dense blocks (submatrices) to factorize and stitch. Remember to enable a sufficient number of points that persist throughout the sequence to perform factorization on a dense block. There is no need to fill in missing data for this problem now. Follow the general scheme described below:

1. Select a dense block from the point-view matrix and construct the $2M \times N$ measurement matrix D. Each column contains the projection of a point in all views, while each row contains one coordinate of the projections of all the points in a view.
2. Normalize the point coordinates by translating them to the mean of the points in each view in the dense block (see lecture slides for details).
3. Apply SVD to the $2M \times N$ measurement matrix to express it as $D = U * W * V'$ where U is a $2M \times 3$ matrix, W is a 3×3 matrix of the top three singular values, and V is a $N \times 3$ matrix. Derive the structure and motion matrices from the SVD as explained in the lecture.
4. Repeat initial steps for each image set which are composed of (1) three and (2) four consecutive images. Then you will have different 3D point sets. By an iterative manner, stitch each 3D point set to the main view using the point correspondence. Transformation between different sets can be found using Procrustes analysis. Scipy's procrustes (`https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.procrustes.html`) function can be used in the project.

5.2 Analysis (3pt)

Plot the 3D structure. In the report, please include snapshots from several viewpoints to show the structure clearly for:

- 3D point cloud obtained from 1 single dense block.
- 3D point clouds obtained from the step (4).
- The same but using provided PointViewMatrix.txt.

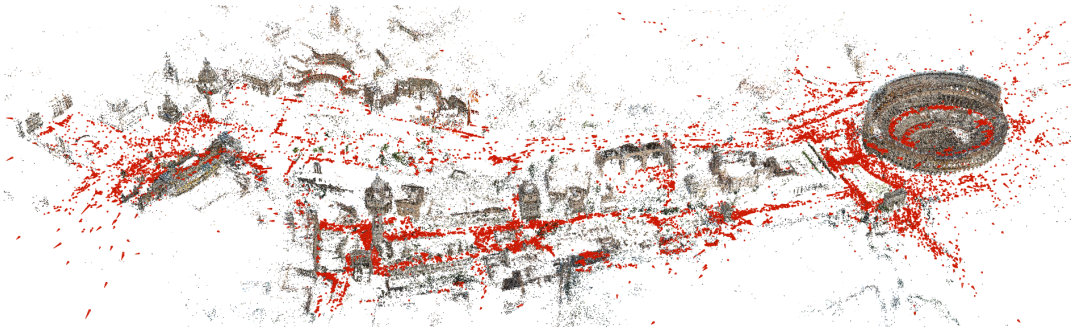
Try with different structure-motion decompositions. Analyze your findings.

6 Additional improvements (4pt)

Improve the results. You are free to explore other references outside provided resources. Example of directions for improvements:

- Affine ambiguity removal (check the lecture)
- Point-View matrix density improvement

Show qualitatively or/and quantitatively that result improves in comparison to the baseline. If it doesn't improve, please elaborate why it may be the case.



7 Real-world Data and Industry Tool (4pt)

Now that you have become familiar with the Structure-from-Motion (SfM) algorithm using your own implementation, we explore industry tools that achieve the same using real-world data. One such (free) tool is COLMAP `https://colmap.github.io`, a general-purpose Structure-from-Motion and Multi-View Stereo (MvS) pipeline with a graphical and command-line interface. You can use a image dataset of choice, or use images of the Sarphati monument, to be downloaded here.

1. Follow the installation steps in `https://colmap.github.io/install.html`. We recommend building from source on a Unix-based system.
2. Follow the tutorial to obtain a sparse model of the dataset with the SfM and MvS pipeline. Visualize and analyze your results at each step.
3. What could be done to further improve the results? Elaborate the problems you see in the reconstructions and how your suggestion could help.
4. What happens when the number of (adjacent) frames is reduced?
5. How can you reduce the number of background pixels included in the reconstruction? Try it out and compare your results.

8 Self-Evaluation

Please describe briefly in an appendix to the report about contribution of each team member to this assignment. Self-evaluation will be taken into consideration in the case if contributions are significantly unequal.

References

[1] RI. Hartley. In defense of the eight-point algorithm. *TPAMI*, 1997.

[2] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66:2006, 2006.