Computer Vision: Project Description

Fredrik Kahl

2025-12-11

Project

- Note: Projects should be done individually! No groups!
- 50 optional points will ensure you grade 4. (Oral for grade 5).

Project choices

Two options:

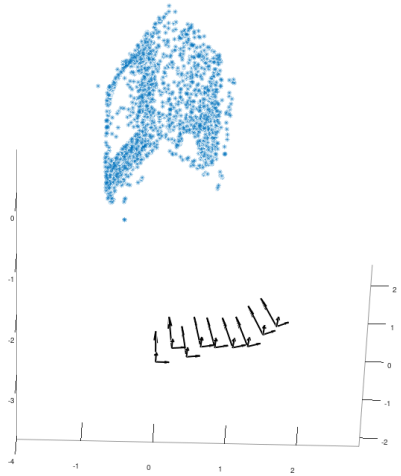
1. Default project, described in the following slides
2. Your own project
   - Suggest a project to Fredrik and get it approved
   - For example, "I want to explore COLMAP/GLOMAP/Gaussian Splattings/NeRFs/VGGT/RoMa ..."
   - Strongly recommended: Create a video of your visualizations that can be shown on a project webpage for the course

Overview of the default project

Implement a 3D reconstruction system!

In short: From

to



Algorithm: Structure from Motion

1. Calculate relative orientations $(R_{i,i+1} \mid T_{i,i+1})$ between images i and $i+1$
2. Upgrade to absolute rotations R_i
3. Reconstruct initial 3D points from an initial image pair i_1 & i_2
4. For each image i *robustly* calculate the camera center C_i / translation T_i
   Reduced camera resectioning problem (since R_i is known at this stage)
5. Refine camera centers (or translation vectors) using Levenberg-Marquardt method (optional)
6. Triangulate points for all pairs $(i, i+1)$ and visualize 3D points + cameras

Algorithm: Structure from Motion

1. Calculate relative orientations $(R_{i,i+1} \mid T_{i,i+1})$ between images i and $i+1$
   - $P_{i,1} = (I \mid 0)$ & $P_{i,2} = (R_{i,i+1} \mid T_{i,i+1})$
   - Some datasets have a dominant plane and then the 8-point algorithm may fail! It is a degenerate case!
   - (Optional): For dominant planar structures:
     Implement robust estimation of $(R \mid T)$ that works for (nearly) planar and general non-planar scenes (see following slides) and keep the inliers!
2. Upgrade to absolute rotations R_i
3. Reconstruct initial 3D points from an initial image pair i_1 & i_2
4. For each image i *robustly* calculate the camera center C_i / translation T_i
   Reduced camera resectioning problem (since R_i is known at this stage)
5. Refine camera centers (or translation vectors) using Levenberg-Marquardt method (optional)
6. Triangulate points for all pairs $(i, i+1)$ and visualize 3D points + cameras

Algorithm: Structure from Motion

1. Calculate relative orientations $(R_{i,i+1} \mid T_{i,i+1})$ between images i and $i+1$
2. Upgrade to absolute rotations R_i
   - Set $R_0 = I$ and chain rotation matrices $R_i = R_{i-1,i}R_{i-1}$
   - You now know the rotation matrices for all cameras!
3. Reconstruct initial 3D points from an initial image pair i_1 & i_2
4. For each image i *robustly* calculate the camera center C_i / translation T_i
   Reduced camera resectioning problem (since R_i is known at this stage)
5. Refine camera centers (or translation vectors) using Levenberg-Marquardt method (optional)
6. Triangulate points for all pairs $(i, i+1)$ and visualize 3D points + cameras

Algorithm: Structure from Motion

1. Calculate relative orientations $(R_{i,i+1} \mid T_{i,i+1})$ between images i and $i+1$
2. Upgrade to absolute rotations R_i
3. Reconstruct initial 3D points from an initial image pair i_1 & i_2
   - For better accuracy: i_1 and i_2 should not be adjacent (have enough baseline) Suggested values provided by us (given in data file)
   - Calculate $(R_{i_1,i_2} \mid T_{i_1,i_2})$ and triangulate inliers: 3D points $\mathcal{X}_0$
   - Rotate $\mathcal{X}_0$ to world coordinates (via $R_{i_1}^T$). Note that the translation to the world coordinate system is not known at this stage!
   - Save the SIFT descriptors (coming from i_1 or i_2) for reconstructed 3D points $\mathcal{X}_0$
4. For each image i *robustly* calculate the camera center C_i / translation T_i Reduced camera resectioning problem (since R_i is known at this stage)
5. Refine camera centers (or translation vectors) using Levenberg-Marquardt method (optional)
6. Triangulate points for all pairs $(i, i+1)$ and visualize 3D points + cameras

Algorithm: Structure from Motion

1. Calculate relative orientations $(R_{i,i+1} \mid T_{i,i+1})$ between images i and $i+1$
2. Upgrade to absolute rotations R_i
3. Reconstruct initial 3D points from an initial image pair i_1 & i_2
4. For each image i *robustly* calculate the camera center C_i / translation T_i
   Reduced camera resectioning problem (since R_i is known at this stage)
   1. Establish correspondences between 2D points in image i and the 3D points $\mathcal{X}_0$ (using the saved SIFT descriptors)
   2. Estimate C_i/T_i robustly from these correspondences using RANSAC with a DLT based on 2 point correspondences.
5. Refine camera centers (or translation vectors) using Levenberg-Marquardt method (optional)
   - Modify CE3 in assignment 4
6. Triangulate points for all pairs $(i, i+1)$ and visualize 3D points + cameras

Things you need to implement

In order to complete the mandatory parts

1. A RANSAC algorithm robustly estimates R and T from correspondences
   - For scenes without and (optionally) with a dominant plane
   - Suggested approach: use a "parallel" version of RANSAC (estimate both essential matrix E and plane homography H)
   - Alternative: Implement a 5-point solver (optional)
     - In this case a standard RANSAC loop is sufficient, no dominant plane solution needed.
2. Extract R and T from either E or (optionally) H
   - $(R \mid T)$ from E: CE3 in assignment 3
   - $(R \mid T)$ from H: provided (`homography_to_RT`)
   - Use cheirality to determine the correct configuration
3. A method to estimate the translation T robustly from 2D-3D correspondences $x_i \leftrightarrow X_i$ and known absolute rotation R_i
   - E.g. `P = estimate_T_robust(xs, Xs, R, inlier_threshold)`
   - Modify `estimate_camera_DLT` from CE2 in assignment 2 to estimate T for a minimal sample
4. High-level wrapper code ("glue code")

RANSAC: Searching for E and H in parallel

Modify the vanilla RANSAC loop

- Maintain (and return) $(R^* \mid T^*)$
- Return best $(R^* \mid T^*)$ found after sufficient iterations for a good E or H

In each iteration

- Estimate E using the 8-point method from a minimal sample set of size 8
  - If E is the new best essential matrix:
    Extract new $(R^* \mid T^*)$ from E
- Estimate 2D-homography H using the DLT from a minimal sample set of size 4
  - If H is new best homography:
    Extract two solutions $(R_a \mid T_a)$ and $(R_b \mid T_b)$ from H (`homography_to_RT`)
    Test both essential matrices $E \in \{[T_a]_\times R_a, [T_b]_\times R_b\}$ for validity
    - If E is the new best essential matrix *and* is "acceptable"
      - Assign correct $(R \mid T)$ (extracted from E) to $(R^* \mid T^*)$

An essential matrix E is "acceptable" if exactly one of the 4 cheirality configurations has all inliers (w.r.t. the epipolar constraint) in front of both cameras

RANSAC: Searching for E and H in parallel

Modify the vanilla RANSAC loop

- Maintain (and return) $(R^* \mid T^*)$
- Return best $(R^* \mid T^*)$ found after sufficient iterations for a good E or H

In each iteration

- Estimate E using the 8-point method from a minimal sample set of size 8
  - If E is the new best essential matrix:
    Extract new $(R^* \mid T^*)$ from E
- Estimate 2D-homography H using the DLT from a minimal sample set of size 4
  - If H is new best homography:
    Extract two solutions $(R_a \mid T_a)$ and $(R_b \mid T_b)$ from H (`homography_to_RT`)
    Test both essential matrices $E \in \{[T_a]_\times R_a, [T_b]_\times R_b\}$ for validity
    If E is the new best essential matrix *and* is "acceptable"
    Assign correct $(R \mid T)$ (extracted from E) to $(R^* \mid T^*)$

An essential matrix E is "acceptable" if exactly one of the 4 cheirality configurations has all inliers (w.r.t. the epipolar constraint) in front of both cameras

How to run your software

- Once your implementation is complete, you should run it via

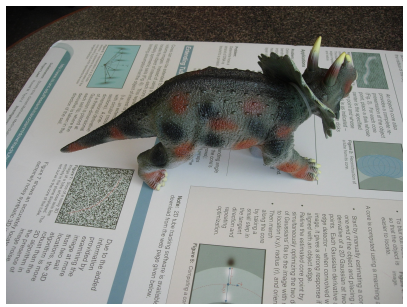
  `run_sfm(<dataset number>)`

- Datasets 1–9 are test datasets provided by us
  - Datasets 1 & 2: for debugging, do not include in the report
  - Datasets 3–7: increasing difficulty, you should include those in the report
  - Datasets 8 & 9: optional
- You may include your own datasets (optional, see later)







For the report...

- Provide a high-level explanation of your algorithmic design choices
  - Especially what you implemented beyond the routines from the computer exercises
  - Also if you modified/improved your implementation for the computer exercises
- Describe unexpected problems you encountered (if any)
- Document changes to the provided code
  - E.g. modifying inlier threshold or selecting a different initial pair
- Briefly describe extra datasets (if you have any)
  - If you picked something from the internet: link to the dataset(s)
  - Describe unexpected issues with the additional datasets
- For optional points: describe what you implemented

Deadline: Sunday, January 11, 2026, 23:59

Extracting R and T from H

- Bill Triggs, "Autocalibration from Planar Scenes", ECCV 1998 (Appendix 1)
  - `http://perception.inrialpes.fr/Publications/1998/Tri98/Triggs-eccv98-long.pdf`
  - Use the provided Python function
    `P = homography_to_RT(H)`

Importing dataset information

- There is a provided function `get_dataset_info(<dataset number>)`.
- This is probably one of the first routines to call in your `run_sfm` script
- *Provided together with the project datasets in the* `data` *folder*

Practical Suggestions

- Work with normalized image points $\tilde{x}_i = K^{-1}x_i$
  - Camera matrices are $P_i = (R_i \mid T_i)$
  - Do not forget to adjust inlier thresholds from pixels to normalized units (`pixel_threshold` is a suggested value in `get_dataset_info`)

```
epipolar_threshold    =     pixel_threshold / focal_length
homography_threshold  = 3 * pixel_threshold / focal_length
translation_threshold = 3 * pixel_threshold / focal_length
```

  - Use maybe `xn` for normalized and `xu` for unnormalized image points
- Filter 3D points excessively far away from the center of gravity, e.g.

$$\|X_j - \bar{X}\| \leq 5 \times 90\% \text{ quantile of } \{\|X_j - \bar{X}\|\}_j$$

- Check (visualize) triangulated points and cameras for each relative pose
  - $P_{i,1} = (I \mid 0)$ & $P_{i,2} = (R_{i,i+1} \mid T_{i,i+1})$
  - Verify correctness of relative pose and cheirality
- While in the development/debugging phase
  - Precompute and store keypoints and matches
  - Load those for faster time to reach your own code

Practical Suggestions

- Work first with the non-planar datasets 1-5 and start with the RANSAC implementation from CE2 in assignment 4
- RANSAC is a stochastic method, so it may *occasionally* fail
  - But there is likely a bug if your software frequently produces bad 3D models
- Reproducability while debugging: early in your code set the random seed via
  - Python: `np.random.seed(1)`

Optional Points for Additional Functionality

1. Run your software on datasets 8 & 9 (5 points each, max. 10 points)
2. Run your software on more datasets (10 points/dataset, max. 20 points)
   - Describe dataset (and its origin) and your experiences briefly
   - Check the next slide how to extract K
   - Images should have $\approx$ HD (1920×1080) resolution
   - Use dataset ids ≥ 10
3. Estimate homography in parallel RANSAC to handle scenes with a dominant plane (max. 30 points)
   - See description on earlier slides
4. Nonlinear refinement of camera pose (max. 20 points)
   - Optimize over camera translation (10 points) and rotation matrix (10 points)
5. Use 5-point method in OpenCV (max. 10 points)
6. Use RoMa or the recent RoMav2 (max. 50 points) to obtain dense matches and triangulate them (in a robust way) in the last step of your pipeline.
   - `https://github.com/Parskatt/RoMa`
7. Implement the 5-point method (max. 100 points)
   - Cf. lecture 10
   - No need for the parallel search for E and H
   - Note that the action matrix should be 10×10 so that you need to check for up to 10 solutions

How to obtain K?

- Approximate K from EXIF meta-data
- E.g. in a Linux terminal

```
$ exiftran -d kronan1.JPG | grep Focal
   0x920a  Focal Length                    30.0 mm
   0xa405  Focal Length in 35mm Film       45
$ exiftran -d kronan1.JPG | grep Dim
   0xa002  Pixel X Dimension               1936
   0xa003  Pixel Y Dimension               1296
```

- 30mm is the sensor-dependent focal length
- 45mm is the 35mm equivalent
- Focal length in pixels: $f = \text{X-Dimension} \times \frac{f_{35mm}}{35} \approx 2489$
- Calibration matrix

$$K = \begin{pmatrix} f & 0 & \text{X-Dimension}/2 \\ 0 & f & \text{Y-Dimension}/2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2489 & 0 & 968 \\ 0 & 2489 & 648 \\ 0 & 0 & 1 \end{pmatrix}$$

- Attention: image viewing software may display auto-rotated picture