TTK4255: Robotic Vision

Final project: Visual localization

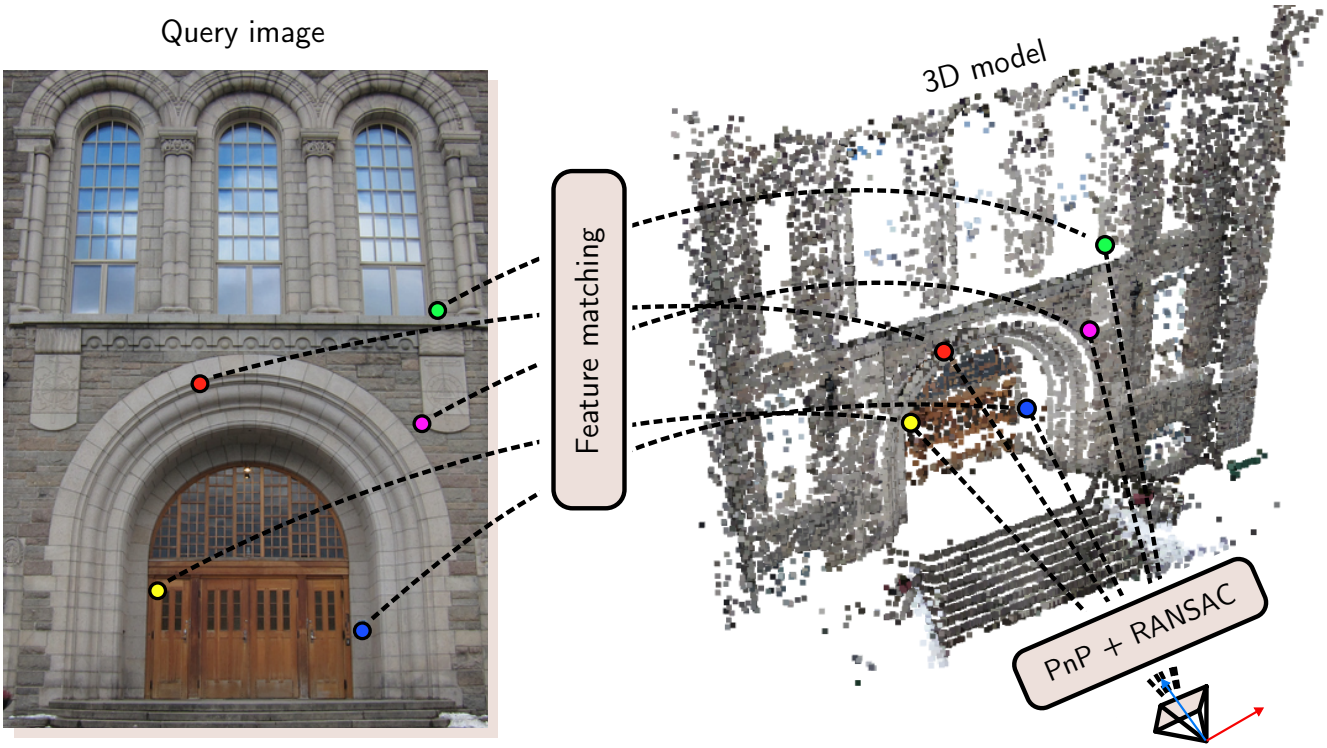

Figure 1: Operating principle behind the visual localization algorithm you will implement, here shown for a model of Hovedbygget, Gløshaugen.

Instructions

The final project will be conducted in the same way as the midterm project. As a reminder:

- The project can be done alone or in groups of two (no more).
- Your score counts toward 35% of your final grade. In groups each member gets the same score.
- To receive a score, you must upload a written report containing answers to the questions and requested program outputs within the deadline. The report must be a single PDF. You must also upload a zip containing your code. Only one group member needs to upload everything.
- You are not permitted to collaborate with other groups or copy solutions from previous years. Unlike the homework, we will look for plagiarism and non-permitted collaboration.
- During the project period and up to the deadline, the Piazza forum will only allow private posts. We may choose to not answer your question, but if we do we will make the answer public.

About the project

Visual localization is the problem of determining the 3D position and orientation from which a photo was taken. In this project you will implement a visual localization system for a small environment of your choice. Your system should be based on the following strategy:

1. *(Done once up front):* Create a 3D model of the environment, where each 3D point is associated with an image feature descriptor (e.g. SIFT).
2. Given a new image, establish 2D-3D correspondences between features detected in the new image and points in the model.
3. Estimate the camera pose using a PnP solver inside a RANSAC loop, followed by a non-linear refinement that minimizes the reprojection error.

This strategy is called structure-based localization. It is similar to SLAM, except that the mapping is done entirely up front, thereby eliminating the risks of live mapping. This is useful if the environment has a decent number of features that are persistent in their appearance and 3D position during the operational period.

Some of the tasks in Part 3 involve uncertainty analysis, which has been only briefly touched upon in the lectures. Some more background is provided in the relevant tasks, but if you find this insufficient then you are expected to consult the references (in the task text) to improve your understanding.

An extended version of the dataset from Homework 5 is provided on Blackboard. Although you must collect your own data to receive full score in Part 1 and Part 2, you may want to use this dataset initially and collect your own data later, if time allows. If you are working as a pair, then this also lets you do Part 1 concurrently with Part 2 and Part 3, which can be a natural division of work to begin with.

Part 1 Camera calibration (8 points)

To create your 3D model, you need to know the intrinsics for your camera and correct your images for lens distortion. Both can be done using either the Single Camera Calibrator App (link) from Matlab's Computer Vision Toolbox or the example script from OpenCV's Camera Calibration Module (link). Both of these are based on taking images of a planar pattern to estimate the intrinsics.

An extended version of the dataset from Homework 5 is provided on Blackboard. If you don't wish to calibrate your camera, then you can use this dataset for the rest of the project instead. You will not receive points for this part, but you can still receive full score for the remaining parts (except Task 2.1).

Task 1.1 (6 points)

Follow the Matlab or OpenCV tutorial linked above and calibrate the camera that you intend to use. A smartphone camera should be fine. Below are some additional tips for success.

- Display the calibration pattern on your TV or computer display. Turn off the lights and shut the curtains to avoid glare. To generate a calibration pattern you can use the Calib.io website (link).
- Keep the calibration pattern fully visible in each image, including the white outer border. If the software fails to find the pattern, it is usually because not enough of the outer border is visible.
- Include both frontoparallel images and images at 45 degrees tilt against the pattern.
- Lens distortion is most severe at the edges of the image. Therefore, including images with the pattern appearing near the edges is important to determine the distortion parameters.
- Use the same settings (focus, zoom and aperture) that you will use when collecting your data.
- The size of the pattern (e.g. the checkerboard squares) only affects the calibration extrinsics. When you only care about the intrinsics you can just specify an arbitrary size.

Include the following in your report:

(a) A bar chart of the mean reprojection error per image and a scatter plot of the 2D error vectors collected from all the images. See `showReprojectionErrors` in Matlab (link) for an example.

(b) The standard deviation of the intrinsics. In Matlab, the function `displayErrors` (link) prints these. In OpenCV, `calibrateCameraExtended` (link) returns the standard deviations.

Task 1.2 (2 points)

Write a script that undistorts a given image, using e.g. `undistortImage` in Matlab or `undistort` in OpenCV. Use your script to analyze the effect of uncertainty in the distortion parameters: generate e.g. ten hypothetical distortion parameter vectors, by sampling from a normal distribution with the above standard deviations, and undistort an image of your choice. Comment on the differences and whether undistortion seems necessary for your camera.

Part 2 Model creation (8 points)

Here you will create the model containing the 3D points and their associated feature descriptors. A two-view reconstruction is sufficient for solving the tasks in the next part. You can therefore apply the strategy from HW5, given that you first extract the 2D-2D correspondences and the feature descriptors.

Task 2.1 (2 points)

You can collect your own dataset if you calibrated your camera. Pick a location and capture images from different viewpoints. You want two images for creating the model. Keep in mind that you need sufficient overlap to find correspondences, but also a sufficient baseline for an accurate triangulation. You also want some additional test images for the next part. It will be useful to have images taken at different distances from the modeled scene.

Make sure to undistort the images that you intend to use. Note that the undistorted image may "modify" the intrinsic matrix obtained from calibration (due to offsetting or scaling). Read the documentation for your calibration package to see if this is the case. If so, then you will want to save the modified intrinsic matrix for use in the following tasks. You don't need to report anything for this task.

Task 2.2 (4 points)

Pick a pair of images and create a two-view reconstruction. Save the 3D point locations and a feature descriptor associated with each 3D point. If you use the provided dataset, then you still need to do this task to obtain the descriptors. The images in the provided dataset are undistorted.

Python users can follow OpenCV's tutorial for feature extraction and matching (link). SIFT is still a popular descriptor and detector, so you may want to use that. Matlab users can follow the examples in the documentation for `matchFeatures` (link). SURF is similar to SIFT and is probably a good choice. You are free to use HW5 or any external code to estimate the relative pose and the 3D point locations. You will receive a higher score if you refine the model by bundle adjustment (external code is OK).

Include the following in your report:

(a) The pair of images you used for your reconstruction.

(b) A description of how you extracted and matched features. Specify if you did anything beyond the above tutorials, such as cross-checking, bidirectional matching or using a different metric.

(c) A description of how you estimated the relative pose and the 3D point locations, including how you eliminated outlier correspondences and how you performed bundle adjustment (if you did).

Task 2.3 (2 points)

The reconstruction is only unique up to a rigid body transformation and a scaling factor. You can resolve the former by fixing the model coordinate frame, e.g. to one of the two cameras, but the scale is still meaningless. Impose a metric scale by measuring (or guessing) the real distance in meters between two points, and multiplying the 3D coordinates by an appropriate scaling factor. Describe how you determined this scaling factor in your report.

Part 3 Localization and uncertainty analysis (8 points)

Here you will use the model to estimate the camera pose of new images (*queries*). Your strategy should be to establish 2D-3D point correspondences between the query image and the model via descriptor matching, then estimate the camera pose by minimizing the reprojection error. Due to the possibility of outlier correspondences, you should first estimate a rough pose and the inlier set using RANSAC.

Task 3.1 (2 points)

Write a script named `localize` that estimates the camera pose of a given query image. The script `visualize_query_results` is provided for visualizing your results. Estimate the camera pose of three query images of your choice. Include the generated figure for each and describe your approach (e.g. how you selected the inlier threshold and any other constants).

You probably have to modify the visualization script to work with your results, particularly the data format and the viewpoints for the pseudo-3D figures. The script will use some sample data if you run it as-is, so that you can see what the figure should look like.

You are free to use external code to solve this task, for example: `solvePnPRansac` (OpenCV) and `estimateWorldCameraPose` (Matlab). To minimize the reprojection error and further refine the pose, you can use external code or your LM implementation from the midterm project.

Task 3.2 (2 points)

It's good practice to analyze the uncertainty of your estimates. For example, you may want to know how the pose estimate is affected by noise in the 2D feature locations, perhaps caused by imprecision in the keypoint detector. This requires you to estimate not only the "best" pose, but also its distribution. Due to the non-linearities involved, this distribution does not have a nice analytical form. However, if the measurements follow a Gaussian distribution, then the distribution of the estimated pose can be approximated by a Gaussian with the covariance matrix

$$\boldsymbol{\Sigma}_{\mathbf{p}} = (\mathbf{J}^T \boldsymbol{\Sigma}_{\mathbf{r}}^{-1} \mathbf{J})^{-1}, \tag{1}$$

where $\mathbf{p}$ is a vector parameterizing the pose, $\mathbf{J}$ is the Jacobian of the residuals, evaluated at the least squares minimum, and $\boldsymbol{\Sigma}_{\mathbf{r}}$ is the covariance matrix of all the measurements. This is an application of propagation of covariance, which is explained in Hartley and Zisserman 5.2 (available on BB). If you use a local parameterization, then $\mathbf{p}$ has three translational and three rotational components, and $\boldsymbol{\Sigma}_{\mathbf{p}}$ is a 6×6 matrix. If you have n correspondences, then $\boldsymbol{\Sigma}_{\mathbf{r}}$ is a $2n \times 2n$ matrix.

Compute the covariance $\boldsymbol{\Sigma}_{\mathbf{p}}$ for the three queries you used above. Assume that the measurement covariance is $\boldsymbol{\Sigma}_{\mathbf{r}} = \sigma_r^2 \mathbf{I}$ with $\sigma_r = 1$ pixel. Report the standard deviations of the translational and rotational pose parameters in millimeters and degrees. (The standard deviations are the square roots of the diagonal entries of $\boldsymbol{\Sigma}_{\mathbf{p}}$.)

Task 3.3 (2 points)

The derivation of Eq. (1) assumes that the estimated pose is the solution to a *weighted* least squares problem, where the objective function to be minimized is

$$||\mathbf{r}||^2_{\mathbf{\Sigma_r}} = \mathbf{r}^T \mathbf{\Sigma}_\mathbf{r}^{-1} \mathbf{r}. \tag{2}$$

A common special case is when the measurement noise is uncorrelated from point to point, which is a reasonable model of e.g. keypoint detector noise. Then the weighted objective function simplifies to

$$\sum_{i=1}^{n} (\hat{\mathbf{u}}_i - \mathbf{u}_i)^T \mathbf{\Sigma}_i^{-1} (\hat{\mathbf{u}}_i - \mathbf{u}_i), \tag{3}$$

where $\mathbf{\Sigma}_i$ is the 2×2 covariance matrix of point i. This can be seen as downweighting the influence of uncertain points, as their inverse covariance will be small (see Szeliski B.1). If $\mathbf{\Sigma_r}$ is a scalar multiple of the identity matrix, as in the previous task, then the solution to the weighted problem is identical to that of the unweighted problem, but in general these are not equal.

Re-estimate the camera poses for the three query images, this time *with* weighting. Assume that the point measurements are corrupted independently by zero-mean Gaussian noise, with covariance

$$\mathbf{\Sigma}_i = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}, \tag{4}$$

where $\sigma_u \gg \sigma_v$ (e.g. $\sigma_u = 50$ pixels and $\sigma_v = 0.1$ pixels). Report the standard deviations of the pose parameters and discuss which parameters are most uncertain in this case.

If you are using your LM implementation from the midterm project, then a simple solution is to modify your residuals function to return the weighted residuals $\mathbf{\Sigma}_\mathbf{r}^{-1/2}\mathbf{r}$. The matrix $\mathbf{\Sigma}_\mathbf{r}^{-1/2}$ can in general be computed from the Cholesky decomposition of $\mathbf{\Sigma_r} = \mathbf{L}\mathbf{L}^T$ as $\mathbf{L}^{-1}$. Keep in mind that $\mathbf{\Sigma_r}$ must match the order of the residuals in $\mathbf{r}$. If the first N residuals are the horizontal differences and the last N residuals are the vertical differences, then $\mathbf{\Sigma_r} = \text{diag}(\sigma_u^2, ..., \sigma_u^2, \sigma_v^2, ..., \sigma_v^2)$.

Task 3.4 (2 points)

Besides imprecision in the keypoint detector, you should also consider the effects of an imprecise calibration. Instead of deriving an analytical approximation of the resulting covariance $\mathbf{\Sigma_p}$, here you should estimate the covariance by Monte Carlo simulation (Hartley and Zisserman 5.3):

Monte Carlo estimation of covariance:

- Over m trials:
  - Randomly sample an intrinsic matrix $\mathbf{K}$ under the expected calibration error distribution.
  - Estimate the query camera pose using $\mathbf{K}$. (For simplicity you can use the same inlier set; i.e. you don't need to repeat RANSAC.)
  - Append the vector $\mathbf{p}$ of estimated pose parameters to a list
- Estimate $\mathbf{\Sigma_p}$ as the covariance of the m parameter vectors, using e.g. `cov` in Numpy/Matlab.

Ignore the possibility of errors in the undistortion and assume that the remaining calibration errors are given by three normally distributed random variables, $(\eta_f, \eta_{c_x}, \eta_{c_y})$, with zero mean and covariance matrix $\mathbf{\Sigma_K}$, such that

$$\mathbf{K} = \bar{\mathbf{K}} + \begin{bmatrix} \eta_f & 0 & \eta_{c_x} \\ 0 & \eta_f & \eta_{c_y} \\ 0 & 0 & 0 \end{bmatrix}, \tag{5}$$

where $\bar{\mathbf{K}}$ is the intrinsic matrix produced by the calibration. Assume further that the calibration errors are uncorrelated, such that $\mathbf{\Sigma_K} = \text{diag}(\sigma_f^2, \sigma_{c_x}^2, \sigma_{c_y}^2)$. Ignore any other sources of noise and estimate the pose *without* weighting. Compute a Monte Carlo estimate of $\mathbf{\Sigma_p}$ for the following scenarios, for a single query image of your choice:

(a) Relatively uncertain focal length (σ_f much larger than σ_{c_x} and σ_{c_y}).

(b) Relatively uncertain horizontal principal point (σ_{c_x} much larger than σ_f and σ_{c_y}).

(c) Relatively uncertain vertical principal point (σ_{c_y} much larger than σ_f and σ_{c_x}).

The specific values for σ_* are not important; you may use e.g. 50 pixels as a "large" value and 0.1 pixels as a "small" value. $m = 500$ trials should be sufficient. Report the standard deviations of the pose parameters and discuss which parameters are most affected in (a), (b) and (c).

Part 4 Extras (11 points)

The scoring for this part will be adjusted based on everyone's achievements. You most likely do not have to do all the tasks here to get a full score for this part, but doing just one may not be enough.

Task 4.1

The covariance matrix you obtained in Task 3.4 represents the uncertainty of an estimate made *without* knowledge of the calibration error distribution. If the calibration error covariance is known, then we should be able to improve the estimate. Suggest a way to incorporate $\mathbf{\Sigma_K}$ into the pose optimization and re-estimate the poses (a), (b) and (c) in Task 3.4. Compare the resulting standard deviations of the pose parameters with and without weighting.

Task 4.2

Create a larger model involving at least five viewpoints. You are free to use any external code. Include a description of how you created the model, and three figures showing localization in the larger model.

Task 4.3

Compare the use of different feature descriptors. Some points of comparison could for example be the number of inliers in the localization and/or model creation stage, and the reprojection error and pose uncertainty at the localization optimum.

Task 4.4

RootSIFT is a simple modification of SIFT proposed by Arandjelović and Zisserman [1]. They claim that *"using a square root (Hellinger) kernel instead of the standard Euclidean distance to measure the similarity between SIFT descriptors leads to a dramatic performance boost in all stages of the pipeline."* Investigate the validity of their claim in your localization and/or model creation stage.

Task 4.5

Compare different RANSAC methods for the localization and/or model creation stage. See the recent survey by Jin et al. [2] and the associated video recording (link) for some state-of-the-art RANSAC methods.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE, 2012.

[2] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, pages 1–31, 2020.