TTK4255: Robotic Vision

Final project: Visual localization

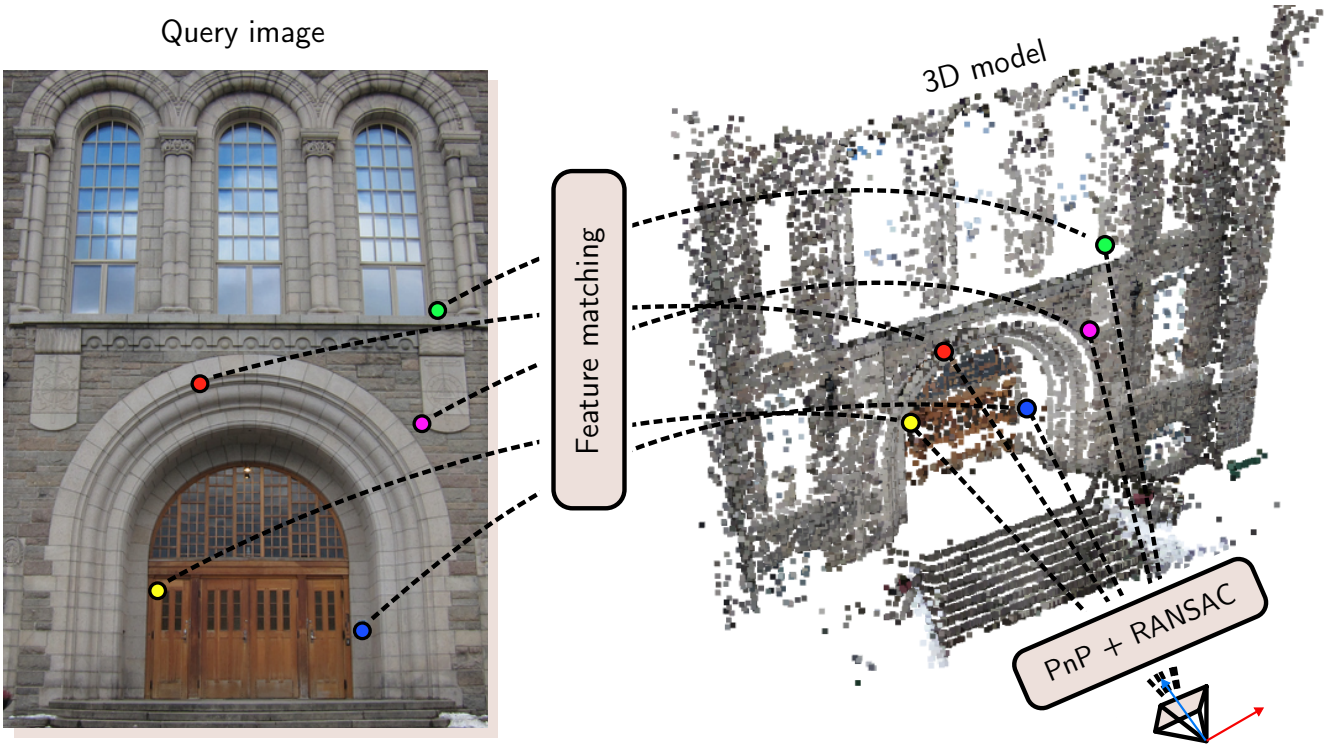

Figure 1: Operating principle behind the visual localization algorithm you will implement, here shown for a model of Hovedbygget, Gløshaugen.

Instructions

- The project can be done alone or in groups of two.
- Your project is approved/not approved. Your project is approved if you complete 60 percent of the tasks. The grading is explained in the "About" document on Blackboard in Course work.
- Upload a single PDF containing requested answers and program outputs within the deadline. You must also upload a zip containing your code.
- You are not permitted to collaborate with other groups or copy solutions from previous years.
- You may ask for help on Piazza and in the help sessions.

About the project

Visual localization is the problem of determining the 3D position and orientation from which a photo was taken. The output is the *camera pose* — the transformation between the coordinate system of the camera that took the photo and the coordinate system of a model of the environment. Here you will implement a visual localization system based on the following strategy:

1. Done once up front: Create a point cloud model of the environment of interest, where each 3D point has an associated image feature descriptor, e.g. SIFT.
2. Given a *query image* (image to localize): Establish 2D-3D correspondences between the query image and the model via descriptor matching, and estimate the camera pose via minimizing reprojection errors, after using PnP and RANSAC to eliminate outliers.

This strategy is called structure-based localization. It is similar to SLAM, except that the mapping is done up front, which is a good risk-reducing alternative if the environment has features that are persistent in appearance and position during operation. You will implement a simple version of this strategy where the model is created from only two images. In practice, you would likely need more images to fully cover the environment, but the above steps still form the basis for larger-scale systems.

Dataset and recommended software libraries

In all the tasks you will work with an extended version of the dataset from Homework 5. Besides including more images of the scene, this dataset also includes the images used to calibrate the camera, which you will be asked to do in Part 1. Additionally, in Homework 5, the point correspondences were given to you, but in Part 2 you will be asked to obtain correspondences yourself via feature matching.

To calibrate the camera, and to extract and match features, we recommend that Matlab users install the Computer Vision Toolbox, and that Python users install OpenCV (link) with the option `pip install opencv-contrib-python`. The tasks often provide tips and pointers for using these libraries.

Part 1 Camera calibration (15%)

To create a 3D model using the algorithm from Homework 5, you need a calibrated pinhole camera model and the images must be corrected for lens distortion. Both of these can be done using Matlab's Single Camera Calibrator App (link) or OpenCV's Calibration Module. OpenCV users may use the provided `calibrate_camera.py` script, which is largely based on the official tutorial (link).

The dataset also includes the author's undistorted images and associated intrinsic matrix. You are free to use these in Part 2 and Part 3.

Task 1.1 (5%)

Calibrate the camera using three radial and two tangential distortion coefficients. Run the provided `show_calibration_results` script and include the generated figure. Reflect on the following:

(a) The generated figure shows the mean reprojection error per image. Some images will have nearly twice the error as the image with the smallest error. What could be causing this? Describe an experiment you could do to test your hypothesis (you don't need to actually do the experiment).

(b) The generated figure also shows the detected checkerboard points from all the images together. This plot can be particularly helpful when assessing the accuracy of the lens distortion estimate. Does the plot suggest anything about this calibration, e.g. should more/other images be taken?

Task 1.2 (10%)

The above script also prints the estimated intrinsics and their standard deviations. While these can be analyzed numerically as in HW6, it can still be difficult to understand their uncertainty intuitively. A more visual understanding can be gotten by generating a few (e.g. ten) hypothetical outcomes from the distribution defined by the standard deviations, and undistorting an image with each set of hypothetical parameters. The resulting images can then be compared, to get a visual impression of the uncertainty.

Write a script to assess the uncertainty via this approach. You may keep the intrinsic matrix fixed, and only sample from the distributions of the distortion coefficients. You may further assume that these are independent normal distributions, with mean and standard deviation equal to the corresponding values printed by the above script. Matlab users may use `undistortImage` (link), and should also read how to create a `cameraParameters` object (link). OpenCV users may use `cv.undistort` (link), and should consult the usage example in the calibration tutorial. Reflect on the following:

(a) Should the image you choose to undistort for this analysis be one of the calibration images, or should it be an image of a different scene? Does it need to be an actual image from the camera?

(b) The undistortion functions in Matlab and OpenCV let you configure how the output should be scaled or cropped (if at all), and whether the output is allowed to contain invalid pixels. What configuration do you find gives a good indication of the uncertainty in the distortion coefficients?

(c) Is the uncertainty in the distortion coefficients visually noticeable? Is it more noticeable in some parts of the image? If so, can you relate that to your answer in Task 1.1b?

Part 2 Model creation (35%)

In this part you will create the 3D model using the two-view reconstruction algorithm from HW5. Having obtained the intrinsic matrix and undistorted images, the remaining preliminary step is to obtain some point correspondences and feature descriptors. Matlab users may follow the examples for `matchFeatures` (link). In Matlab 2021b or later you can use SIFT, otherwise SURF is a good alternative. Python users may follow OpenCV's tutorial (link). However, since OpenCV's interface tends to cause more trouble for students than Matlab, we also provide a Matlab-inspired interface to the relevant functionality (see `example_match_features.py`).

Task 2.1 (20%)

Pick a pair of images and create a two-view reconstruction. Extracting around 30,000 features per image should give a workable number of inlier correspondences after RANSAC, but a smaller number can be useful for testing initially. Report the following:

(a) ... a figure of the best fifty correspondences (before RANSAC), sorted by descriptor distance.

(b) ... a figure of eight inlier correspondences and their epipolar lines in both images.

(c) ... what options you tried for the feature extraction and matching.

(d) ... how you estimated the relative pose and the 3D point coordinates.

(e) ... the number of points in your reconstruction.

You may use code from HW5's solution. To simplify Part 3, you should save your reconstruction as one or more files. You will need to save the 3D point coordinates and a feature descriptor associated with each 3D point. Helpful functions are `writematrix` (link) and `np.savetxt` (link).

Task 2.2 (10%)

The algorithm from HW5 does not aim to minimize reprojection errors and can therefore be said to be suboptimal. Explain how you can refine the two-view reconstruction via bundle adjustment. What is the objective function? What are the parameters and how may you initialize them? What does the Jacobian look like? (You do not need to actually perform the bundle adjustment, but it can be useful for controlling your answer.)

Task 2.3 (5%)

Your reconstruction is only unique up to a rigid body transformation (the model coordinate frame) and a scaling factor. While you can arbitrarily define the model coordinate frame to coincide with one of the two viewpoints, the scale is still meaningless; e.g. the model coordinates cannot be interpreted as meters. Explain how you can transform the model coordinates to be in meter units, given the real distance in meters between two points in the scene. Optionally, you may measure (or make an educated guess of) the metric distance between two points of your choice and perform this transformation.

Part 3 Localization and uncertainty analysis (50%)

In this part you will finally use your model to localize other images. The key idea is that your model lets you establish 2D-3D correspondences (between the query image and the model) with the same descriptor matching technique as in Part 2. You can then estimate the camera pose via minimizing reprojection errors. The estimation can be done with non-linear least squares as in the midterm project, but due to potentially bad correspondences you should first find the inlier set with RANSAC.

Task 3.1 (15%)

Write a script that can localize an arbitrary image in the dataset. You may solve part of the task using `solvePnPRansac` (link) or `estimateWorldCameraPose` (link), but to minimize reprojection errors you should use the non-linear least squares solver you used in the midterm project.

Matlab users should note that the Computer Vision Toolbox follows a convention in which points are represented by row vectors; see e.g. the descriptions of the input arguments to `worldToImage` (link). This means that you must transpose all vectors/matrices passed into, and returned from, the toolbox.

Task 3.2 (5%)

Modify `show_localization_results` to visualize the model and the pose for at least three query images of your choice. Include the figures and discuss if your algorithm is working correctly. Does the pose appear where you expect based on the image contents (e.g. left or right side of door, looking left or right)? What happens if you try to localize the images that were used to create the model?

Task 3.3 (5%)

Before analyzing the uncertainty of the localization (the topic of the next tasks), can you suggest other visualizations, or things that you could compute, that would help you assess the correctness or quality of the localization? A text description is enough, but a figure or two can help clarify what you mean.

Task 3.4 (5%)

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 uncertainty in the 2D feature keypoints, which may be caused by imprecision in the keypoint detector. This requires you to estimate not only the optimal pose, but also its distribution. Due to the non-linearities involved, this distribution does not have a nice analytical form. However, if the measurements are normal distributed, then the estimated pose is to a first order approximation also normal distributed, with covariance matrix

$$\mathbf{\Sigma_p} = (\mathbf{J}^T \mathbf{\Sigma_r}^{-1} \mathbf{J})^{-1}, \tag{1}$$

where $\mathbf{p}$ is the pose parameter vector, $\mathbf{J}$ is the Jacobian of the residuals at the solution, and $\mathbf{\Sigma_r}$ is the covariance matrix of all the measurements. This result is due to *propagation of covariance*, which is explained in Hartley and Zisserman 5.2 (available on BB), in particular 5.2.2.

Compute the covariance $\mathbf{\Sigma_p}$ for each of the three queries you chose in Task 3.2. Assume that the measurements are independently normal distributed with standard deviation of 1 pixel, so that $\mathbf{\Sigma_r}$ is an identity matrix. Report the standard deviations of the pose parameters in millimeters and degrees, for the translational and rotational parameters respectively.

Tip: With a local pose parameterization (as in the midterm project), $\mathbf{p}$ has three translational and three rotational parameters, and $\mathbf{\Sigma_p}$ is a 6×6 matrix. With n correspondences, $\mathbf{\Sigma_r}$ is a $2n \times 2n$ matrix. The standard deviations are the square roots of the diagonal entries of the covariance matrix.

Task 3.5 (10%)

Hartley and Zisserman's derivation of Eq. (1) assumes that the estimated pose is actually the solution to a *weighted least squares* problem, where the objective function to be minimized is

$$E(\mathbf{p}) = \mathbf{r}(\mathbf{p})^T \mathbf{\Sigma_r}^{-1} \mathbf{r}(\mathbf{p}). \tag{2}$$

Note that if $\mathbf{\Sigma_r}$ is the identity, then you get the familiar sum of squared residuals of the unweighted problem. Note also that if the keypoint noise is uncorrelated from point to point (which seems like a reasonable assumption of a keypoint detection process), then $\mathbf{\Sigma_r}$ becomes block-diagonal, and the objective function simplifies to

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

where $\mathbf{\Sigma}_i$ is the 2×2 covariance matrix of keypoint i. This can be seen to downweigh the influence of uncertain keypoints, as their inverse covariance will be small (see also 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, and the better estimate is the one that includes information about the measurements' uncertainty.

Reestimate the camera poses for the three query images, this time with weighting. Assume that the keypoints are contaminated independently by zero-mean normal distributed noise, with covariance

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

For the sake of experimentation, let σ_u be much greater than σ_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.

Tip: Modify your residuals function to return the weighted residuals $\mathbf{\Sigma_r}^{-1/2}\mathbf{r}$. The matrix $\mathbf{\Sigma_r}^{-1/2}$ can in general be obtained from the Cholesky decomposition $\mathbf{\Sigma_r} = \mathbf{L}\mathbf{L}^T$ 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 $\mathbf{\Sigma_r} = \mathrm{diag}(\sigma_u^2, ..., \sigma_u^2, \sigma_v^2, ..., \sigma_v^2)$.

Task 3.6 (10%)

Besides uncertainty in the keypoint detector, you should also consider the effects of uncertainty in the calibration. Specifically, given a probability distribution over the camera parameters, what is the resulting distribution over the localized pose? Again we will simplify and use normal distribution approximations. However, instead of deriving an analytical approximation of the covariance matrix $\mathbf{\Sigma_p}$, here you should estimate it by simulation (Hartley and Zisserman 5.3):

Estimate covariance via Monte Carlo simulation:

- Over m trials:
  - Randomly sample camera parameters from their probability distribution.
  - Estimate the camera pose using the sampled camera parameters. (For simplicity you can use the same inlier set in each trial; e.g. the set obtained using the optimal parameter values.)
  - Append the resulting 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.

This method requires only enough patience to run sufficiently many trials, and can be used as ground-truth to validate any analytical approximation. But, for the sake of teaching the principle, we will make some simplifications. Ignore uncertainty in the distortion coefficients, and let the uncertainty in the remaining camera parameters be described by three normally distributed random errors $(\eta_f, \eta_{c_x}, \eta_{c_y})$, 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 optimal intrinsic matrix estimated in the calibration from Part 1. Assume further that the errors are uncorrelated, such that $\mathbf{\Sigma_K} = \mathrm{diag}(\sigma_f^2, \sigma_{c_x}^2, \sigma_{c_y}^2)$.

As a further simplification, you should estimate the pose *without* weighting. This is suboptimal if you know the calibration uncertainty (which you do here), but incorporating the calibration uncertainty is not as straightforward as in Task 3.5. Finally, although we have the actual calibration uncertainty from Part 1, you will consider three hypothetical scenarios, in order to better understand the effects of uncertainty in the different parameters.

Estimate $\mathbf{\Sigma_p}$ 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, but you may use 50 as a large value and 0.1 as a small value in each case. $m = 500$ trials should be sufficient. Report the standard deviations of the pose parameters and discuss which parameters you expect and observe to be most affected (i.e. have the highest standard deviations) in the three scenarios.