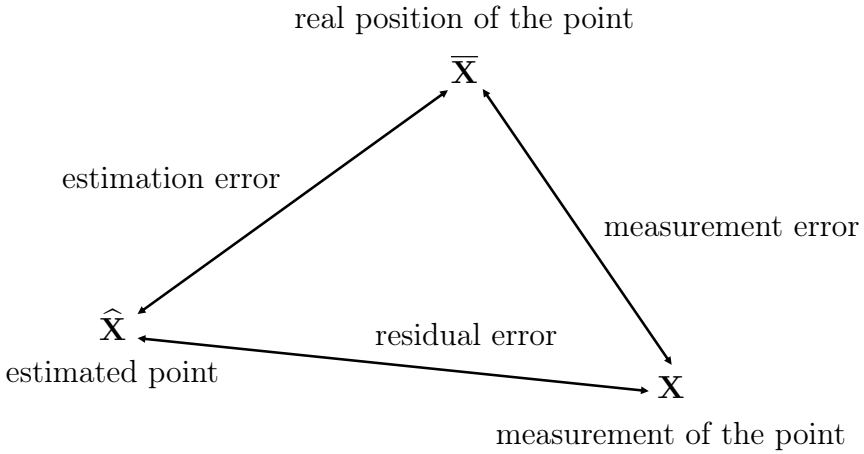
Technische Universität München
Fakultät für Informatik
Prof. Dr. N. Navab

SoSe 2011
Exercise Sheet 5
June 16, 2011

Exercises in 3D Computer Vision I

Exercise 1 Error Models in Estimation

For estimation of a homography $\mathbf{H}$, different errors with different properties can be considered. The error triangle shows the three different errors that occur in estimation problems.



a) Explain the estimation problem for homography estimation.

b) Referring to the error triangle, what error is minimized in estimation? Explain the different error models that can be used to describe this error.

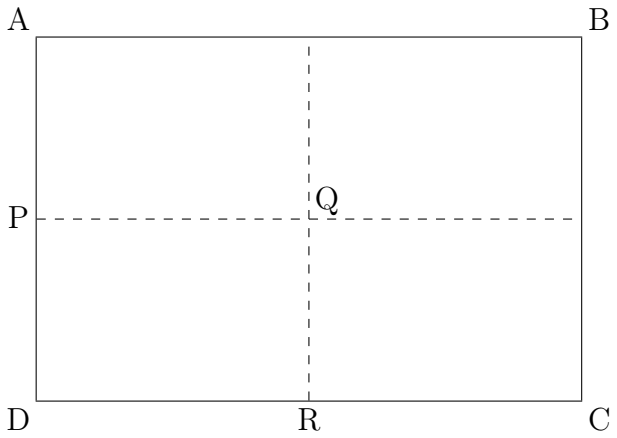
c) Explain the difference of algebraic error and geometric error (in one image). What is the reason for this difference?

d) When estimating a 2D homography, we usually have measurement errors in both images, not only in one. Considering this fact, explain symmetric transfer and reprojection error and their difference.

Exercise 2 Homographies

A surveillance camera views a rectangular parking lot to ensure that only authorized vehicles occupy spaces in the lower-left quadrant, defined by points P, Q, R, and D in the figure below. The four corners of the parking lot, A, B, C, and D, perspectively project into the image as corners of a quadriliteral with image coordinates $a = (1, 1)$, $b = (2, 1)$, $c = (3, 0)$, and $d = (0, 0)$, respectively. Assume the parking lot is flat, and the ratio of its "length" (i.e., distance from A to B) to its "width" (i.e., distance from B to C) is $3/2$. Using a plane projective camera model, compute the real-valued image coordinates of the scene points P, Q, and R.

Hint: Since the parking lot is planar, use 2D coordinates for A,B,C,D. Assume that $D = (0, 0)^\top$ and derive the remaining points (A,B,C) such that the ratio constraints are kept.



Exercise 3 (H) Automatic Image Stitching with RANSAC

In a previous exercise, you developed a program for image mosaicing. In order to stitch two images, the user was instructed to select at least 4 pairs $(\mathbf{x}_i, \mathbf{x}'_i)$ of corresponding points. This set S of matches was then used to create the matrix of the DLT algorithm, and a homography $\mathbf{H}$ was obtained using the SVD.

In this exercise, you will make this program fully automatic, not requiring any form of user interaction any more. In order to come up with the pairs of corresponding points required by DLT, you will use an implementation of SIFT (Scale Invariant Feature Transform) to extract meaningful spots from the images and match them. These correspondences S are estimated using statistical considerations, and they will contain a certain number of outliers. In order to get good results, you will need to identify these wrong matches and discard them during homography estimation.

One powerful method to do that is RANSAC (RANdom SAmple Consensus). It removes outliers by randomly selecting a sample of minimal possible size from S (4 correspondences in this case) and estimating the size of the consensus set. The latter consists of all matches $(\mathbf{x}_i, \mathbf{x}'_i)$ that satisfy $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ up to a small error. RANSAC stores the random sample with the largest consensus set, and after a sufficient number of evaluations considers this to be the outlier-free solution.

In "Multiple View Geometry in Computer Vision", pages 117ff., the generic RANSAC algorithm is specified as follows:

- Randomly select a minimal sample from S and instantiate the model from this subset.
- Determine the set of data points S_i that are within a distance threshold t of the model. The set S_i is the consensus set of the sample and defines the inliers within S.
- If the size of S_i (the number of inliers) is greater than some threshold T, re-estimate the model using all the points in S_i and terminate.
- If the size of S_i is less than T, select a new subset and repeat the above.
- After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i.

This algorithm uses two user-defined static numbers, a minimal number of inliers T and a maximal number of iterations N, to determine whether the result is good enough.

It is usually a good idea not to require users to select good values for such non-obvious values. This situation can be improved by using an adaptive termination condition:

- $N = \infty$, sample_count $= 0$.
- While $N >$ sample_count repeat
  - Choose a sample and count the number of inliers.
  - Set $\epsilon = 1 - (\text{number\_of\_inliers})/(\text{total\_number\_of\_points})$.
  - Set $N = \log(1-p)/\log(1-(1-\epsilon)^s)$.
  - Increment sample_count by 1.
- Terminate.

In order for you to test and implement your code, we provide some files to you. You find them on our website `http://campar.in.tum.de/Chair/TeachingSs11CV`.

- Test images `tum_mi_1.jpg` and `tum_mi_2.jpg`
- Manually chosen correspondences including outliers and ground truth in `tum_mi_matches.m`. Please read the comments in the file for details.

a) Write a function `robust_matching.m` of non-adaptive RANSAC using your previously implemented normalized DLT. The new method takes a set of correspondences including outliers and the parameters T and N, and returns the robustly estimated homography $\mathbf{H}$.

   Find useful values for T and N and test your code using the correspondences and ground truth provided by us.

b) Implement and test the adaptive version of RANSAC as `robust_matching_adaptive.m`. Compare the time consumption of the two versions using Matlab's `tic` and `toc` commands.

c) Get a Matlab SIFT implementation from the internet, for instance from `http://www.vlfeat.org/~vedaldi/code/sift.html` [1]. Use this code to automatically detect and match feature points in the images. Compute the homography from these correspondences and stitch the images. Properly visualize the original images including the feature points and the stitched image.

   You may also try this method with own images. Remember that you can rotate your camera only, you must not translate it!

[1] This package requires to compile native code. Please consult the `README` file for installation instructions. After installation, you can use the command `sift` to extract feature points and descriptors from an image, and `sift_match` to automatically estimate correspondences. Color images can be converted into grayscale using `rgb2gray`. The package also provides code for visualization.