RWTH AACHEN UNIVERSITY

Computer Vision
Wintersemester 08/09

Prof. B. Leibe
<leibe@umic.rwth-aachen.de>

Exercise 6: Fundamental Matrix, RANSAC, Triangulation

due February 3rd 2008, 23:59

Question 1: Fundamental Matrix Estimation

In this exercise, we will use the Eight-point algorithm presented in lecture 16 in order to estimate the fundamental matrix between a pair of images (we're using a slightly simpler version of the algorithm here than the one presented in the lecture). Download the archive `exercise6.zip` from the class web page and unpack it in your directory. This archive contains two pairs of stereo images on which we want to try out our implementation.

Let's first assume that we are given a list of perfect correspondences $\mathbf{x} = (u, v, 1)$ and $\mathbf{x}' = (u', v', 1)$, so that we don't have to deal with outliers. The fundamental matrix constraint states that each such correspondence must fulfill the equation

$$\mathbf{x'}^\top \mathbf{F}\mathbf{x} = 0. \tag{1}$$

We can reorder the entries of the matrix to transform this into the following equation

$$\begin{bmatrix} uu' & uv' & u & vu' & vv' & v & u' & v' & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0 \tag{2}$$

By stacking $N \geq 8$ of those equations in a matrix $\mathbf{A}$, we obtain the matrix equation

$$\mathbf{Af} = 0 \tag{3}$$

$$\begin{bmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_Nu'_N & u_Nv'_N & u_N & v_Nu'_N & v_Nv'_N & v_N & u'_N & v'_N & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ \vdots \\ F_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4}$$

which can be easily solved by Singular Value Decomposition (SVD), as shown in Exercise 5.5. Applying SVD to $\mathbf{A}$ yields the decomposition $\mathbf{A} = \mathbf{UDV}^\top$. The homogeneous least-squares solution corresponds to the least singular vector, which is given by the last column of $\mathbf{V}$.

In the presence of noise, the matrix $\mathbf{F}$ estimated this way will however not satisfy the rank-2 constraint. This means that there will be no real epipoles through which all epipolar lines pass, but the intersection will be spread out over a small region. In order to enforce the rank-2 constraint, we therefore again apply SVD to $\mathbf{F}$ and set the smallest singular value D_{33} to zero. The reconstructed matrix will now satisfy the rank-2 constraint, and we can obtain the epipoles as

$$\mathbf{Fe_1} = 0 \quad \text{and} \tag{5}$$

$$\mathbf{F}^\top \mathbf{e_2} = 0 \tag{6}$$

by again applying SVD and setting $\mathbf{e_1} = \frac{\begin{bmatrix} V_{13} & V_{23} & V_{33} \end{bmatrix}}{V_{33}}$ and $\mathbf{e_2} = \frac{\begin{bmatrix} U_{13} & U_{23} & U_{33} \end{bmatrix}}{U_{33}}$.

Similarly, for the points $\mathbf{x}, \mathbf{x}'$, we can obtain the epipolar lines $\mathbf{l}' = \mathbf{F}\mathbf{x}$, $\mathbf{l} = \mathbf{F}^\top\mathbf{x}'$ in the other image. Note that in projective geometry, a line is also defined by a single 3D vector. This can be easily seen by starting with the standard Euclidean formula for a line

$$ax + by + c = 0 \tag{7}$$

and using the fact that the equation is unaffected by scaling to apply it to the homogeneous point $\mathbf{x} = (X, Y, W)$. Thus, we arrive at

$$\begin{aligned} aX + bY + cW &= 0 & (8)\\ \mathbf{l}^\top\mathbf{x} = \mathbf{x}^\top\mathbf{l} &= 0. & (9) \end{aligned}$$

The parameters of the line are easily interpreted: $-a/b$ is the slope, $-c/a$ is the x-intercept, and $-c/b$ is the y-intercept.

a) Write a function that implements the above algorithm to compute the fundamental matrix and the epipoles from a set of (at least 8) perfect correspondences given in the vectors `x1` and `x2`.

```
function [F e1 e2] = fundmatrix(x1, x2)
% Input:
%   x1,x2 : 3xN arrays of N homogenous points in 2D
% Output:
%   F     : The 3x3 fundamental matrix such that x2'*F*x1 = 0
%   e1    : The epipole in image 1 such that F*e1  = 0
%   e2    : The epipole in image 2 such that F'*e2 = 0
...
...
end
```

b) As explained in the lecture, we need to take care to normalize the points in order to make sure the estimation problem is well conditioned. Write a function `normalize2dpts` that normalizes the given list of 2D points `pts` by first shifting their origin to the centroid and then scaling them such that their mean distance from the origin is sqrt(2). The function should return both the transformed points `newpts` and the 3×3 transformation matrix `T`.

```
function [newpts T] = normalize2dpts(pts,)
% Input:
%   pts   : 3xN array of N homogenous points in 2D
% Output:
%   newpts: 3xN matrix of transformed points
%   T     : the 3x3 transformation matrix, newpts = T*pts
...
...
end
```

c) Now write an adapted function `normfundmatrix` that first normalizes the input points, computes the fundamental matrix based on the normalized points, and then undoes the transformation by applying $\mathbf{F} = \mathbf{T_2}^\top\mathbf{F}\mathbf{T_1}$ before computing the epipoles.

```
function [F e1 e2] = normfundmatrix(x1, x2)
% Input:
%   x1,x2 : 3xN arrays of N homogenous points in 2D
% Output:
%   F     : The 3x3 fundamental matrix such that x2'*F*x1 = 0
%   e1    : The epipole in image 1 such that F*e1  = 0
%   e2    : The epipole in image 2 such that F'*e2 = 0
...
...
end
```

d) Next we want to verify that our implementation is indeed correct. For this purpose, write a function `inputcorresp` that uses Matlab's `ginput` command to let the user specify a set of correspondences in the two input images. Save those correspondences for later reference.

```
function [x1 x2]=inputcorresp(img1, img2)
...
...
end
```

e) Apply the above algorithms to the provided test image pairs `house1/2` and `library1/2` based on the manually selected correspondences. (Note: at least 12 correspondences will typically be needed in order to get a robust estimate). For each point $\mathbf{x}$ in one image, compute the corresponding epipolar line $\mathbf{l}'$ in the other image and visualize both to see if the correspondence point $\mathbf{x}'$ indeed lies on the correct line. Compare the results of the normalized and the unnormalized Eight-point algorithm. What do you observe? What happens when your correspondence set contains outliers?

f) In order to get a quantitative estimate for the accuracy of your results, write a function `res_error` that computes the residual error of your estimation (which in this case should be defined as the mean-squared distance between points in one image and their corresponding epipolar lines). How big are the residual errors for the normalized and the unnormalized Eight-point algorithms?

```
function [x1 x2]=res_error(F, x1, x2)
...
...
end
```

Question 2: RANSAC

In practice, the correspondence set will always contain noise and outliers. We therefore apply RANSAC in order to get a robust estimate. As introduced in lecture 12, RANSAC proceeds along the following steps:

1. Randomly select a (minimal) seed group of point correspondences on which to base the estimate.
2. Compute the fundamental matrix from this seed group.
3. Find inliers to this transformation.
4. If the number of inliers is sufficiently large ($\geq m$), recompute least-squares estimate of the fundamental matrix on all inliers.
5. Else, repeat for a maximum of k iterations.

The parameter k can be chosen automatically. Suppose w is the fraction of inlier correspondences and $n = 8$ correspondences are needed to define a hypothesis. Then the probability that a single sample of n correspondences is correct is w^n, and the probability that all samples fail is $(1 - w^n)^k$. The standard strategy is thus, given an estimate for w, to choose k high enough that this value is kept below our desired failure rate.

In the following, we will implement the different steps of the RANSAC procedure and apply it for robust estimation of the fundamental matrix.

a) Write a function `get_inliers` which takes as input an estimated fundamental matrix and the full set of correspondence candidates and which returns the subset of correspondences that are inliers to this transformation. A point pair $\mathbf{x}, \mathbf{x}'$ is defined to be an inlier if the distance of $\mathbf{x}$ to the epipolar line $\mathbf{l} = \mathbf{F}^\top \mathbf{x}'$, as well as the opposite distance, are both less than some threshold ε.

```
function [x1 x2] = get_inliers(F, x1, x2, eps)
...
...
end
```

b) Write a function `ransac_fundmatrix` which implements the RANSAC procedure to estimate a fundamental matrix using the normalized Eight-point algorithm. For randomly sampling matches, you can use the `randperm` function. Here, we want to use a simple version of the algorithm that just runs for a fixed number of k iterations and returns the solution with the largest inlier set.

```
function [F e1 e2 x1 x2 r] = ransac_fundmatrix(x1, x2, eps, k)
% Input:
%   x1,x2 : 3xN arrays of N homogenous points in 2D
%   eps   : inlier threshold
%   k     : number of iterations
% Output:
%   F     : The 3x3 fundamental matrix such that x2'*F*x1 = 0
%   e1    : The epipole in image 1 such that F*e1  = 0
%   e2    : The epipole in image 2 such that F'*e2 = 0
%   x1,x2 : 3xNi arrays of Ni homogenous inlier points in 2D
...
...
end
```

c) Apply your RANSAC implementation to the correspondence sets from Question 1. Experiment a bit with the parameter ε. What do you observe? What fraction of outliers can your function deal with?

d) Next, we want to automate the estimation procedure by using point correspondences obtained with the Harris detector we developed in Exercise 5. You can either use your own implementation or take the one provided in the archive `exercise6.zip` from the class web page. Take the file `apply_homest.m` as a starting point and adapt it to extract Harris points together with e.g. `maglap` descriptors on fixed-size patches for both images, then match the extracted points to find correspondences. Experiment with different thresholds for accepting putative matches. Based on the obtained correspondences, apply your RANSAC fundamental matrix estimation algorithm. Is your implementation able to estimate the correct transformation between the image pairs?

Question 3: Triangulation

As a final step, we want to reconstruct the observed points in 3D by triangulation. Note that just using two images, this is not possible without a calibration. You can therefore find camera matrices for each image provided in the archive `exercise6.zip`. They are stored as simple text files containing a single 3×4 matrix and can be read in with the `load` command, i.e.

```
P1 = load('house1_camera.txt');
```

For triangulation, we use the linear algebraic approach from the lecture. Given a 2D point correspondence $\mathbf{x_1}, \mathbf{x_2}$ in homogeneous coordinates, the 3D point location $\mathbf{X}$ is given as follows

$$\lambda_1 \mathbf{x_1} = \mathbf{P_1 X} \tag{10}$$
$$\lambda_2 \mathbf{x_2} = \mathbf{P_2 X}. \tag{11}$$

We can now build the cross-product of each point with both sides of the equation and obtain

$$\begin{array}{rcl} \mathbf{x_1} \times \mathbf{P_1 X} = [\mathbf{x_{1\times}}]\mathbf{P_1 X} = 0 \\ \mathbf{x_2} \times \mathbf{P_2 X} = [\mathbf{x_{2\times}}]\mathbf{P_2 X} = 0, \end{array} \tag{12}$$

where we used the skew-symmetrix matrices $[\mathbf{x_{i\times}}]$ to replace the cross products

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}_\times]\mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \mathbf{b}. \tag{13}$$

Each 2D point provides 2 independent equations for a total of 3 unknowns. We can therefore solve the overconstrained system by stacking the first two equations for each point in a matrix $\mathbf{A}$ and computing the least-squares solution for $\mathbf{AX} = 0$.

a) Why is it not possible to obtain a metric reconstruction in this case without a calibration?

b) Find the centers of both cameras. Recall from the lecture that the camera centers are given by the null space of the camera matrices. They can thus be found by taking the SVD of the camera matrix and taking the last column of $\mathbf{V}$.

c) Write a function `triangulate` that uses linear least-squares to triangulate the position of a matching point pair in 3D, as described above.

```
function X = triangulate( P1, x1, P2, x2 )
...
...
end
```

d) Apply the triangulation to all inliers from your result of Question 2 and collect the resulting 3D points. Display the two camera centers and the scene matches in 3D using Matlab's `plot3` command. Use the `axis equal` option to avoid automatic nonuniform scaling of the 3D space. Also compute the residuals between the observed 2D points and the projected 3D points in the two images.

Please turn in your solutions by sending an email to Bastian Leibe <leibe@umic.rwth-aachen.de> *including all relevant m-files by Tuesday, February 3, 23:59h*