1. Stereo Correspondences

You are allowed 1 hour 45 minutes for this graded exercise. You have to submit your files by 10:00 AM. You may use your notes, books and code from previous exercise sessions, but access to Internet resources is forbidden. Once you are done, make sure that your files are successfully uploaded to the system.

1.1 Introduction

In this exercise, you are asked to compute correspondences between corresponding pixels of stereo images. Retrieving corresponding pixels on each image of a stereo pair is useful, for instance, to compute their 3D position. Given a pixel $\mathbf{x}_L$ on the left image, the corresponding pixel on the right image lies on a line called the *epipolar line*; the equation of the epipolar lines of a stereo pair is described by the *fundamental matrix*. This matrix is generally unknown, and can be estimated starting from a reduced number of correspondences on the two images. Once the fundamental matrix has been computed, the problem of looking for the correspondence of a pixel on the left image is greatly simplified: we don't have to scan the whole right image, but only check the pixels lying on the epipolar line.

A common technique consists in using the fundamental matrix for computing two *rectified images* from the original images. The rectified images have the property that epipolar lines are horizontal at the same height.

1.2 Instructions

We are given a pair of stereo images I_l and I_r, and a small number (10) of hand-set correspondences. The first step is to estimate the fundamental matrix starting from these correspondences, so that matches for all other pixels can be automatically computed. In the second part of the exercise, you are asked to compute the disparity for a certain point on the image, which is closely related to the depth of this point.

1.3 Goals

In this exercise, you are asked to:

- Estimate the fundamental matrix relative to a pair of stereo images;
- Implement the Normalized Cross-Correlation algorithm (NCC).
- For a given point at the first image from the stereo, compute the best correspondence at the second image.

1.4 Exercise 1

Given a pair of stereo images I_L and I_R, all pixels (x_L, y_L) on I_L and their corresponding pixels (x_R, y_R) on I_R satisfy the following relationship:

$$[x_R,\ y_R,\ 1] \cdot \mathbf{F} \cdot \begin{bmatrix} x_L \\ y_L \\ 1 \end{bmatrix} = 0, \tag{1.1}$$

where $\mathbf{F} \in \mathbb{R}^{3\times3}$ is the so-called *fundamental matrix*. Being of rank two and determined only up to scale, the fundamental matrix can be estimated given at least seven point correspondences. We can estimate the fundamental matrix starting from a set of $N \geq 7$ corresponding pixels on the left and on the right image, employing the following algorithm:

1. Let (x_L, y_L) and (x_R, y_R) 2 corresponding pixels respectively on the left and the right image. Then the first step consists in re-writing the constraint (1.1) in a more useful way:

$$0 = \begin{bmatrix} x_R, & y_R, & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ 1 \end{bmatrix} =$$
$$x_R x_L f_{11} + x_R y_L f_{12} + x_R f_{13} + y_R x_L f_{21} + y_R y_L f_{22} + y_R f_{23} + x_L f_{31} + y_L f_{32} + f_{33}. \tag{1.2}$$

   In particular, let $\mathbf{f}$ the 1-D column array with the unknown coefficients of the matrix $\mathbf{F}$:

$$\mathbf{f} = \begin{bmatrix} f_{11}, & f_{12}, & f_{13}, & f_{21}, & f_{22}, & f_{23}, & f_{31}, & f_{32}, & f_{33} \end{bmatrix}^T. \tag{1.3}$$

   We can re-write the above equation as :

$$\mathbf{y}^T \cdot \mathbf{f} = 0, \tag{1.4}$$

   where:

$$\mathbf{y}^T = [x_R x_L, \; x_R y_L, \; x_R, \; y_R x_L, \; y_R y_L, \; y_R, \; x_L, \; y_L, \; 1]. \tag{1.5}$$

2. If we are given a set of $N \geq 7$ correspondences, we can produce N vectors $\mathbf{y}_n^T$ for each correspondence. So, the vector $\mathbf{f}$ of the coefficients of the fundamental matrix can be computed solving the rectangular linear system:

$$\mathbf{Y} \cdot \mathbf{f} = \mathbf{0}, \tag{1.6}$$

   where $\mathbf{Y}$ is a $[N \times 9]$ matrix whose rows are given by each $\mathbf{y}_n^T$.
3. Since correspondences are not exact, the linear system (1.4) usually can not be solved in an exact way. We can solve it in a least-squares sense and enforce the 2-rank constraint on $\mathbf{F}$ by mean of the singular value decomposition of $\mathbf{Y}$. Details are not covered here. [1]

   Point Normalization

   For solving numerical instabilities, it is better to normalize the pixels before estimating the fundamental matrix, so that the average magnitude of the points is 0 and their average norm is 1. Normalization is already implemented in the code, we explain it here for sake of completeness. We compute 2 transform matrices $\mathbf{T}_L$ and $\mathbf{T}_R$, so that the normalized pixels on the left (resp. right) image are computed as $\hat{\mathbf{x}}_L = \mathbf{T}_L \cdot [x_L, y_L, 1]^T$ and $\hat{\mathbf{x}}_R = \mathbf{T}_R \cdot [x_R, y_R, 1]^T$. A fundamental matrix $\tilde{\mathbf{F}}$ is computed using the normalized correspondences, and then de-normalized: $\mathbf{F} = (\mathbf{T}_R)^T \cdot \tilde{\mathbf{F}} \cdot \mathbf{T}_L$.

[1] For more details on the complete algorithm: http://en.wikipedia.org/wiki/Eight-point_algorithm

Exercise 1.1 Fundamental matrix estimation. You are asked to implement the function `Y = AssembleYMatrix(pixelsImageL, pixelsImageR)`, that takes as inputs 2 arrays of corresponding pixels on the left and right image, and builds the $N \times 9$ matrix $\mathbf{Y}$ described at step 2 of the above algorithm. ■

At this point, you should be able to visualize a pixel on the left image and its corresponding epipolar line on the right image as shown in Figure 1.1.

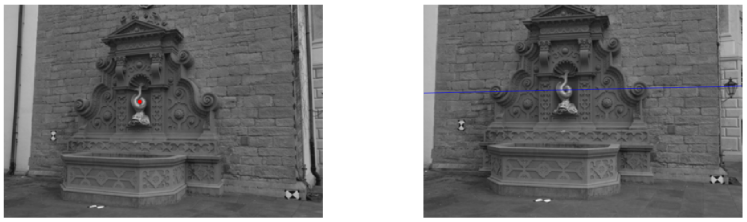

Figure 1.1: Sample output of the Exercise 1.1. A pixel is shown in red on the left image, and the corresponding epipolar line is shown in blue on the right image.

Now, we can compute the 2 rectified images `ImL` and `ImR` showed in Figure 1.2.

1.5 Exercise 2

NCC implementation

You can then verify that both 2D locations (the one in the reference image and the one in the second image) effectively correspond to the same point by computing the Normalized Cross Correlation (NCC) of small patches centered at the given locations.

Assume we are given a pair of gray-scale patches $t[i, j]$ and $s[i, j]$, of size $h \times w$, such that $1 \leq i \leq h$, and $1 \leq j \leq w$. NCC between these patches can then be calculated in the following way:

$$NCC(t, s) = \frac{1}{N} \left(\sum_{i,j} \frac{(t(i, j) - \mu(t))(s(i, j) - \mu(s))}{\sigma(t)\sigma(s)} \right), \tag{1.7}$$

where N is the number of pixels in the patch, and functions $\mu(\cdot)$ and $\sigma(\cdot)$ are mean and standard deviation of the patch respectively.

Exercise 1.2 You are asked to implement the function `sc = ncc(template,sample)`, which will get two patches as an input and output the normalized cross correlation score between them.

To do so you need to do the following steps:

- Check if the images have the same size, if it is not the case - then return -1
- Convert images to `double`
- Implement the Eq. 1.7.
  Hint: you can use standard MATLAB functions `mean()` and `std()` for estimation of mean and standard deviation of the array respectively.

■

Disparity estimation

Assume we are given a pair of rectified images $\texttt{ImL}(x, y)$ and $\texttt{ImR}(u, v)$, of size $W \times H$, such that $1 \leq x, u \leq W$, and $1 \leq y, v \leq H$. For a given point $p_1 = p_1(xp, yp)$ in ImL, we are asked to find the corresponding point $p_2 = p_2(up, vp)$ in ImR. As the images are rectified, vp is equal to yp, however up is still unknown. In order to find the best match you need to scan ImR along the line $v = yp$ and for every point estimate the NCC score. Selection of the best score along the line will give the best match for the point.

Figure 1.2: Sample output of the Exercise 1.3.

Exercise 1.3 You are asked to implement the function `[up,vp] = best_match(xp,yp,ImL,ImR,psz` where `psz` is size of the patch, extracted from the image in oder to estimate NCC. To do so you need to implement the following steps:

- from the ImL extract the patch t, surrounding the point (xp, yp) of size $2*psz+1$
- from the ImR extract a set of patches s_j, of the same size as t, around the points (j, yp), $\forall j : psz + 1 \leq j \leq W - psz$
- calculate the NCC scores, between s_j and t, $\forall s_j$
- find j, which corresponds to the maximum of these scores
  Hint: if there are several maximums, pick the first one
- return $up = j$, $vp = yp$

■

The disparity will be the difference between positions of the same point in different images and is calculated automatically. Fig. 1.2 illustrates the sample output of the program.

Consistency check

You might have seen that if the selected point appears in front of a uniform, or repetitive background, the corresponding point is not found correctly. In order to avoid these situations, we can apply the consistency check.

Provided you have a point p_1 in the ImL, you need first to estimate the corresponding point p_2 in the ImR. Then you need to repeat the process: starting from the p_2 in the ImR you need to estimate p_3 in the ImL. If the correspondence was found correctly p_3 will be equal to p_1.

Exercise 1.4 You are asked to edit your function `[up,vp] =best_match(xp,yp,ImL,ImR,psz)`, so that it applies the consistency check described above. To do so you need to implement the following:

- find (up, vp), using Exercise 1.3
- use Exercise 1.3 to find the point (nxp, nyp) in the `ImL`, which is a best match of the (up, vp) from the `ImR`
- compute the difference between points: $|xp - nxp|$ if it is lower then a threshold c (you can set $c = 2$), then the consistency test is passed, otherwise it is not passed and you should set $vp = -1$
- If you change the size of patch psz, how will it influence the performance of the method? **Write the explanation** of your point of view as a comment in the code of the `best_match` function.

■