Exercise: Interest Points, Feature Extraction and Matching

Local features result in a representation that allows to efficiently match local structures between image. This is usually done by the following pipeline:

1. Find a set of distinctive keypoints.
2. Define a region around each keypoint.
3. Extract and normalize the region content.
4. Compute a descriptor from the normalized region.
5. Match the local descriptors.

Question 1: Harris Detector

We will implement a Harris detector. This detector searches for corner-like structures by looking for points p = (x, y) where the autocorrelation matrix C around p has two large eigenvalues. The matrix C can be computed from the first derivatives in a window around p, weighted by a Gaussian G(x, y; σ) where "$\star$" denotes the convolution operator.

$$\mathbf{C}(x, y; \sigma, \tilde{\sigma}) = G(x, y; \tilde{\sigma}) \star \sigma \begin{bmatrix} D_x^2(x, y; \sigma) & D_x D_y(x, y; \sigma) \\ D_x D_y(x, y; \sigma) & D_y^2(x, y; \sigma) \end{bmatrix},$$

a) Do you have any idea what eigenvalues of C for each pixel say? Based on the magnitudes of the eigenvalues, can you inference following cases?

   $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$

   $\lambda_1 \approx 0$ and λ_2 has some large positive value

   λ_1 has some large positive value and $\lambda_2 \approx 0$

Note that exact computation of the eigenvalues is computationally expensive (it should be done for every pixel!), and instead suggest the following function

$$\det(\mathbf{C}) - \alpha \cdot \mathrm{trace}^2(\mathbf{C})$$

Do you know why? Because eigenvalues are related to det and trace with following equations:

$$\begin{aligned} \det(\mathbf{C}) &= \lambda_1 \lambda_2 \\ \mathrm{trace}(\mathbf{C}) &= \lambda_1 + \lambda_2 \end{aligned}$$

In practice, the parameters are usually set to the following values: σ=1.6,α=0.06

b) Write a function compute_harris which computes the matrix C for each pixel of a given image, calculates its trace and determinant. Then apply non-maximum suppression and return the coordinates of all points that pass the threshold.

def compute_harris(img1, sigma1, sigma2);
…
…
return ImageRes

c) Load the images graf.png and compute Harris interest points for it.

Question2: Matching

After finding the key points, the next step will be to extract the discriminative features that will allow us to perform matching. In matching, we will try to find point correspondences between two images.
load the two example images graff5/img1.ppm and graff5/img2.ppm and perform the following steps:

a) Compute Harris interest points for both images. (Use a high threshold to limit the number of interest points to less than 1000).

b) Compute r/g color histogram descriptors for all interest points: In order to find correspondences between interest points, we need to have region descriptors. They describe each interest point as a vector. we provided two simple descriptors based on the histogram representations namely r/g color histogram and dx/dy histogram descriptor. They take an input image and a list of interest points and computes an r, g color (or dx,dy) histogram over the m × m sub-windows around each interest point.

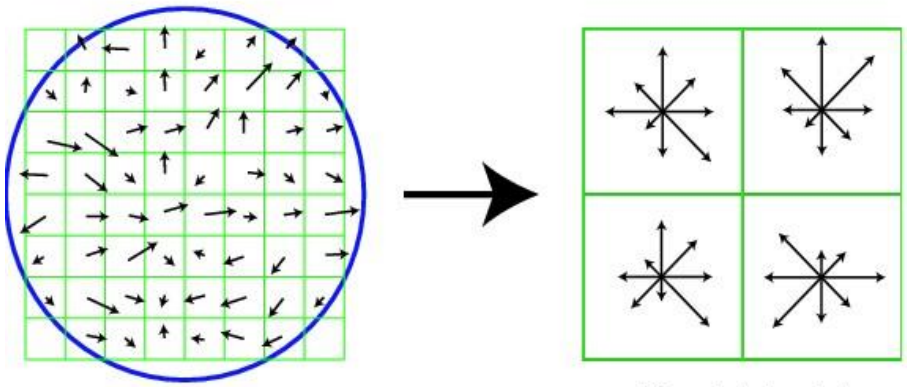
c) Find the best matches using the function findnn: Write the function findnn which takes two sets of region descriptors D1 and D2 and tries to find for each descriptor in D1 the nearest neighbor in D2 using the Euclidean distance (np.linalg.norm function). The function should return the indices Idx and distances Dist of the nearest neighbors.

d) Use the following function displaymatches to visualize the N best matches. What do you observe?

Robust features need to be scale invariant, able to handle viewpoint changes, illumination changes, etc. In next we will work with an efficient and effective feature extractor.

Question3: SIFT Features and Matching

Now, we will extract, visualize and match SIFT descriptors in two images. The SIFT extraction should be done with the cv2. SIFT_create().



Find the 50 best correspondences between two images and visualize them. Use the Euclidean distance as a metric between two descriptors.