Line and point detection

1 Feature point detection and matching

Exercise 1. *Write a function* `corners(image: np.ndarray, w: int, kappa: float)` *that implements the Harris corner detector. Your function should:*

a. *Apply a $w \times w$ Gaussian filter to the image I to remove noise.*

b. *Compute the gradients I_x , I_y of the image in x and y directions.*

c. *Compute the Harris operator at each pixel (x, y)*

$$K(x,y) = \frac{det(H(x,y))}{trace(H(x,y))} \text{ where } H(x,y) = \sum_{(u,v)\in W(x,y)} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{(x+u,y+v)}$$

where $w(u, v)$ is the Gaussian window function and $W(x, y)$ is a $w \times w$ neighborhood of (x, y).

d. *Find the set of pixels $H = \{(x, y) : K(x, y) > \kappa\}$ such that the response of the Harris operator is above a threshold κ.*

e. *Apply non-maximum-suppression in a $w \times w$ pixel neighborhood of each $(x, y) \in H$. Store the pixel coordinates of the resulting keypoints.*

Write a script that tests your code on images "ames1.JPG" for different values of the window size w and the Harris corner threshold κ. For the best choice, plot the results of each step, as well as the Harris corners before and after non-maximum-suppression.

Exercise 2. *Write a function* `features(image: np.ndarray, keypoints: np.ndarray)` *that extracts a descriptor centered around each keypoint in keypoints $\in \mathbb{R}^{2\times P}$. Use a simple vectorized image patch of 9×9 pixels as your descriptor so that descriptors $\in \mathbb{R}^{81\times P}$.*

Exercise 3. *Write a function* `matching(descriptors1: np.ndarray, descriptors2: np.ndarray, tau: float)` *that matches two sets of feature descriptors using the sum of squared differences (SSD) as a matching score. Your function should return for each descriptor in image 1 the index of the best match in image 2 provided that the SSD score is above a threshold τ. If a descriptor in image 1 has no match, then the corresponding entry of matches should be set to -1. Write a script that tests your code on image pair: "ames1.JPG" and "ames2.JPG" and plots the resulting matches. Plot also the ROC curve and use it to select the threshold.*

2 Line detector

Exercise 4. *Write a script that does the following.*

a. *Apply the **Canny edge detector** to images: "ames1.JPG''.*

b. *Apply **Hough transform** to the detected edges. Plot the lines estimated by Hough transform superimposed on the image.*

c. *Show the original image, edge map (after step 1), and final image (after step 2).*