Exercise 1 : Basic Image Processing

One of the most commonly used steps in many computer vision tasks is feature extraction. In this lesson, you will study two algorithms for extracting low-level features, i.e., corners and edges.

For grading, each group needs to submit the following documents to *xuan-son.nguyen@ensea.fr* :

1. A report (english is the preferred language, although its usage does not imply better grades) with problem formulation, pseudocode of algorithms, experimental evaluation and a detailed analysis of the obtained results (hint : use figures if needed to make your report clear and concise).
2. Codes with comments. Details of important parts of the codes can be given in the report.

1 Harris Corner Detector

The Harris corner detector is used to locate corners in an image. Intuitively, if one moves a window over an image patch, and a corner is present, then a strong intensity change can be observed regardless of the moving direction.

Notations. Denote by I the input image, I_x and I_y the partial derivatives of I, I_x^2, I_y^2, and $I_x.I_y$ the second-order moments of partial intensity derivatives, $\bar{I_x^2}$, $\bar{I_y^2}$, and $\bar{I_x.I_y}$ the smooth versions of I_x^2, I_y^2, and $I_x.I_y$, respectively.

Algorithm.

— **Step 1 :** Compute approximations of I_x and I_y. Two common methods can be used. The first method consists in employing a two-dimensional Gaussian filter G_{σ_d} and convolving its closed-form partial derivatives with I. The second method applies the horizontal and vertical Sobel masks to I.

— **Step 2 :** Compute I_x^2, I_y^2, and $I_x.I_y$.

— **Step 3 :** Compute smooth versions of I_x^2, I_y^2, and $I_x.I_y$ using another two-dimensional Gaussian filter G_{σ_w}, where $\sigma_w > \sigma_d$.

The auto-correlation matrix $A_{i,j}$ at every pixel (i, j) is defined as

$$\mathbf{A}_{i,j} = \begin{bmatrix} \bar{I_x^2}(i,j) & \bar{I_x.I_y}(i,j) \\ \bar{I_x.I_y}(i,j) & \bar{I_y^2}(i,j) \end{bmatrix}.$$

— **Step 4 :** Compute the Harris response at pixel (i, j) as

$$H_{i,j} = \det(\mathbf{A}_{i,j}) - k(\text{trace}(\mathbf{A}_{i,j}))^2,$$

where k is a positive constant.

The Harris response $H_{i,j}$ can be seen as a measure of "cornerness" and can be interpreted as follows : negative values indicating edges, close-to-zero values indicating flat regions, and large positive values indicating corners.

— **Step 5 :** Apply non-maximum suppression to the Harris response H. A pixel (i, j) is identified as a corner location if $H_{i,j} \geq \theta$ and $H_{i,j} = \max_{(m,n)\in\mathcal{N}(i,j)}\{H_{m,n}\}$, where θ is a positive threshold and $\mathcal{N}(i, j)$ is the set of neighboring pixels to (i, j), including pixel (i, j).

1.1 Questions

— Is the Harris corner detector robust with respect to intensity shifts and intensity scalings ?

— Is the Harris corner detector robust with respect to translation ?

— Is the Harris corner detector robust with respect to rotation ?

1.2 Practical Work

Implement a Harris corner detector for grayscale images. Test your implementation on the provided images CircleLineRect.png and zurlim.png. Visualize the following images :

— The computed Harris response H as a grayscale image.

— The detected corner pixels on I.

Test the robustness of the Harris corner detector with respect to different transformations mentioned in Section 1.1.

Hint : In order to compute image derivatives, you might need to import the following functions at the beginning of your codes :

from scipy.ndimage.filters import gaussian_filter1d, gaussian_filter, convolve

The scipy library provides useful functions for geometric transformations of images :

from scipy.ndimage.interpolation import rotate

Results can be plotted using matplotlib :

import matplotlib.pyplot as plt

2 Canny Edge Detection

The Canny filter finds edges by searching for local minima or maxima in the approximated first-order derivative of an image.

Notations. The same notations are used as in Section 1.

Algorithm.

— **Step 1 :** Smooth image I by filtering with a Gaussian.

— **Step 2 :** Compute approximations of I_x and I_y at each point in the image. Apply the Sobel masks to the Gaussian-smoothed image to approximate partial intensity derivatives.

— **Step 3 :** At each point in the image, compute the direction of the gradient and the magnitude of the gradient.

$$\text{direction} = \arctan(I_y/I_x), \qquad \text{magnitude} = \sqrt{I_x^2 + I_y^2}.$$

Edges can be detected by thresholding the gradient magnitude. This often results in thick contours that you will refine in the next step.

— **Step 4 :** Perform non-maximum suppression to identify candidate edges. A pixel in the contour is kept as edge pixel only if its gradient magnitude is greater than that of its two neighboring pixels in the direction orthogonal to the edge. Note that this step is not the same as the one used in the Harris corner detector.

2.1 Questions

Consider a single row or column of a grayscale image. Figure 1 plots intensity as a function of position that gives a signal. Sketch the derivatives of the given functions and show the edge positions.

FIGURE 1 – Example of edge detection.

2.2 Practical Work

Implement a Canny edge detector for grayscale images. Test your implementation on the provided images zurlim.png, cubeleft.pgm, and cuberight.pgm. Visualize the following images :

— Gaussian-smoothed image.

— Gradient magnitude and direction images.

— Edge image obtained by thresholding the gradient magnitude (Step 3). The value of this threshold should be adjusted according to the distribution of gradient magnitude values in the image.

— Edge image obtained after the non-maximum suppression.