LUT Engineering Science
Data-Centric Engineering
Tuomas Eerola, Joona Kareinen, Samuel Repka
Spring term 2025

BM40A0902 3D Computer Vision
Exercise 4
Image features: corners and local features.

1. Corner detection (2 points).

   The task is to implement the Harris corner detector (see lecture slide 22 for the algorithm).

   Implement function `find_corners(img, N, t, k=0.04)`, which performs corner detection based on the Harris method described in the lectures. The function takes a gray level image `img` as input, the size `N` of local neighborhood `Q` and a threshold value `t` (as defined in the lecture slides). The function should output a vector with the `(x,y)` coordinates of all detected corners.

   Then use your function with the gray-scale block image (blocks_bw.png). Display the image and the detected corner points on the same figure.

   Does the Harris cornerness measure detect the same corners as Tomasi-Kanade corner detector?

   Note: For Harris corner detection, you do not actually need to calculate eigenvalues and can just use structure tensor $T = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$ to compute Harris cornerness measure R as follows:

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(T) - k \ \mathrm{tr}(T) = (ab - c^2) - k(a+b)^2$$

2. Scale invariant interest points (2 points)

   The goal of this task is to detect local scale invariant features from an image. Use the grayscale Lena image (lena_bw.png).

   The local features are found using the difference of Gaussian (DoG). The difference of Gaussian is defined as

$$D_i = I * G_{\sigma_i} - I * G_{\sigma_{i+1}}$$

   where I is the original image, G_{σ_n} is the Gaussian, and $*$ is convolution. When several values for σ are used, then a stack of images is constructed as

$$D(x, y, i) = D_i(x, y).$$

   Dimensions 1 and 2 are height and width, and dimension 3 corresponds to the scale level.

   (a) Use various σ_i to construct the stack $D(x, y, i)$. You can use values $\sigma_{i+1} = \sqrt{2}\sigma_i$ with $\sigma_1 = 1.6$ and 5 scale levels. That means that you will generate 4 DoG images D_i.

   (b) The local features can be found as local extrema (maximum or minimum) in D. For 3D matrices local extrema means that the sample (x, y, i) is the largest or smallest within its neighbourhood, i.e. it is larger/smaller than its 26 neighbours (8 neighbours in level i, 9 neighbours in level $i+1$ and 9 neighbours in level $i-1$).

   Find local extrema from the image (locations (x, y, i) which locally have the largest or smallest values).

(c) We are mostly interested in high contrast features, the number of local features can be limited with thresholding. Use the threshold $t = 0.03$ on the **absolute** values of difference of Gaussians after finding local features.

Note: Make sure that the scale of pixel values is $[0, 1]$, i.e. apply `im2double` to the original image if you are using MATLAB. In Python, you can cast the image to `np.float64` using numpy's `astype` and divide by 255.

(d) Plot the locations of local extrema on the original image. Do those points look like good features? How can they be improved?

(e) Now load lena_bw_transformed.png. This is a rotated and rescaled image of Lena with added Gaussian noise. Repeat steps (a)-(d) for this image. Compare plots from step (d) for the original image and transformed image by plotting them side by side as subplots. Are features extracted from the transformed image different from the original features?

Figure 1: Images in the experiments. Left: Blocks. Middle: Lena. Right: Transformed Lena.