VINUNIVERSITY

Lab Assignment 02

COMP3040 Computer Vision **October 21, 2024**

Due: October 21, 2024, 5:05pm

Objective: The goal of this assignment is provide students with hands-on experience in working with Canny Edge Detector and Harris Corner Detector

Evaluation Criteria:

- Correctness of the code.
- Proper comments and documentation.

Submission Instructions:

- Download **Lab02_code.zip** to your computer
- Write your Python code in the required files:
  - utils.py
  - canny_edge_detection.py
  - harris_corner_detection.py
  - (optional) matching.py
- Zip the folder and submit on Canvas

Note: Please install the following Python packages:

- numpy
- opencv-python (cv2)

Problem 0: Helper Functions (10 pts)

Information: We will use *utils.py* as helper functions to do the tasks in Problem 1 & 2.
Your Task: In file *utils.py*, write Python code to complete the following:

- *compute_gaussian_kernel()*: function to return the normalized gaussian kernel.

Problem 1: Canny Edge Detector (45 pts)

Information: Canny is one of the best edge detection algorithms that produces a black and white image with edges as the white pixels. It has five steps as follows:

- Convert image to the gray scale and remove the noise in the image with a 5x5 Gaussian filter. The formula for a Gaussian filter is given by:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

- The smoothened image is then filtered with a Sobel kernel in both the horizontal and vertical directions to get the first derivative in the horizontal direction (G_x) and the vertical direction (G_y).

- From these two images (G_x) and (G_y), we can find the edge gradient and direction for each pixel as follows:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- The image magnitude produced results in thick edges. Ideally, the final image should have thin edges. Thus, we must perform non maximum suppression (NMS) to thin out the edges.

- Use hysteresis to determine the final edges by converting weak edges into strong or non-edges.

Your Task: Follow the instruction in file *canny_edge_detection.py*, write Python code to implement Canny Edge Detection from scratch (without using the OpenCV library Canny function).

- Students should locate the sections marked as "TODO" within the provided codespace and complete the missing/incomplete code.

Problem 2: Harris Corner Detector (45 pts)

Information:The Harris Corner Detection involves the following steps:

1. Convert the image to grayscale

2. Compute image gradients: Compute the partial derivatives of the image intensity along the x and y directions, I_x and I_y, using Sobel filters.

3. Compute the products of derivatives: Compute the products of the gradients at each pixel:

$$I_{xx} = I_x^2$$
$$I_{yy} = I_y^2$$
$$I_{xy} = I_x \cdot I_y$$

4. Apply Gaussian smoothing: Apply a Gaussian filter to the derivative products I_{xx}, I_{yy}, and I_{xy} to reduce noise and smooth the values:

$$S_{xx} = G_\sigma * I_{xx}$$
$$S_{yy} = G_\sigma * I_{yy}$$
$$S_{xy} = G_\sigma * I_{xy}$$

   where G_σ is a Gaussian kernel with standard deviation σ.

5. Compute the Harris Response:

   - At each pixel, construct the structure tensor matrix M:

$$M = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix}$$

   - Calculate the corner response function R for each pixel using the determinant and trace of the matrix M:

$$R = \det(M) - k \cdot (\text{trace}(M))^2$$

     where k is an empirical constant (typically $k = 0.04$).

   - The determinant of M is given by:

$$\det(M) = S_{xx}S_{yy} - S_{xy}^2$$

     and the trace of M is:

$$\text{trace}(M) = S_{xx} + S_{yy}$$

6. Threshold the response: Apply a threshold to the Harris response R to identify corners. Pixels with R values above a certain threshold are classified as corners.

Your Task: Follow the instruction in file *harris_corner_detection.py* and theoretical implementation steps mentioned above, write a Python code solution that completes the Harris Corner Detection algorithm manually (without using the pre-built Harris Corner Detector function from OpenCV).

Problem 3: Template Matching (optional with extra 20 pts)

Information: Given 2 images file *matching_source.png* and *matching_template.png*. The objective is to accurately determine the spatial coordinates or region within the source image where the template image is located. Thus, we need to identify the exact position in the source image where the template matches.

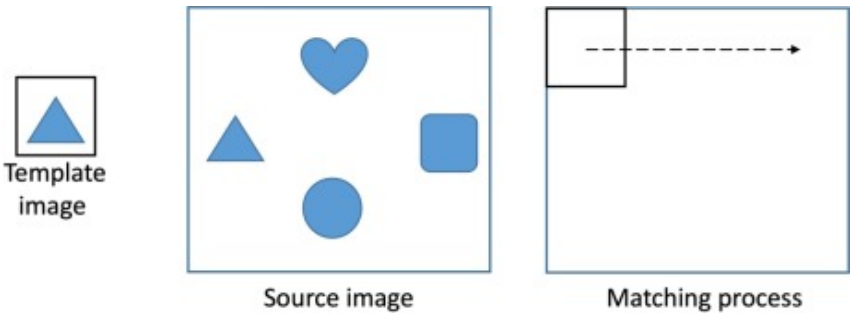

Figure 1: Example of template matching

Your Task: In the *matching.py* file, write Python code to implement the basic template matching function from scratch (without using OpenCV related template matching function). The function can identify and list the bounding boxes that meet the specified template matching, and display on the screen.