İTÜ

10.12.2024

Istanbul Technical University
Computer Engineering Department
BLG 453E - Computer Vision - Fall 24/25
Assignment III
Due: 05.01.2025, 23.59

Notes

- You should do your own work! . Cheating is highly discouraged.
- You **cannot** use built-in Python, Numpy, and OpenCV functions until otherwise stated.
- You should write your codes in Python.
- **A well-written report is required, and the codes will be expected to be well-commented**.
- For your questions, do not hesitate to reach Res. Asst. Ziya Ata Yazıcı (yaziciz21@itu.edu.tr) and Res. Asst. Tuğçe Temel (temel21@itu.edu.tr).

Q1 (40 pts) - Where are Luigi and Wally?

In this task, you will perform the template matching technique to find the specific characters in the given scenes. The template matching technique can be applied to different functions, such as cross-correlation, sum of absolute differences (SAD), and sum of squared differences (SSD)[1]. However, the methods that implement the given functions are not robust to occlusions. Therefore, **you will experiment with the template-based (cross-correlation, SAD, SSD) and feature-based (SIFT) techniques** and compare the results to indicate the main differences between the methods. Mention the benefits of each function to the others.

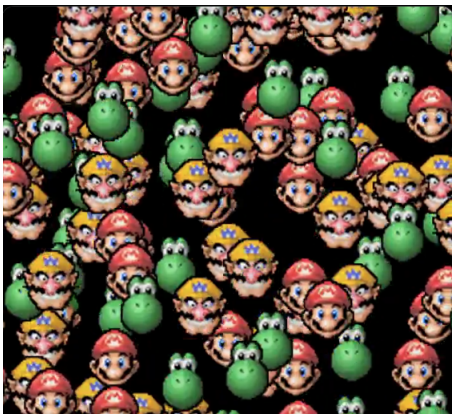

Figure 1: The image in which Luigi will be searched.

1) You just got Luigi'd!

Your first task is to find Luigi among the many of Mario's, Wario's, and Yoshi's faces (Figure 1). You are given two template images of Luigi's face. One is in a normal position, while the other is rotated. By using both templates and both crowded images, find and annotate Luigi. Experiment

[1] https://w.wiki/9KZY

with template-based and feature-based techniques and compare the results. Mention the benefits of each method to the others. The results should be visualized and shown in the Jupyter Notebook and your report.

2) Where's Wally?

Where's Wally?[2] (or known as Where's Waldo?) is a popular puzzle book series created by Martin Handford (Figure 2). In the puzzle, the challenge is to find Wally—a man in a red-and-white striped shirt, glasses, and a bobble hat—hidden in the crowd.

Figure 2: A sample from the Where's Wally book.

In this task, you will be given three images of Wally. It was reported that the pictures are of how he was last seen at the scenes, and due to the low-resolution cameras, the given images are relatively low in quality. The authorities said that it was tough to find him by manually comparing the low-resolution images. Thus, they requested that you, as an expert in the computer vision field, devise a technique to detect Wally automatically. Similar to the first task, **you are expected to use template- and feature-based techniques** to detect him at the scenes and compare the results. Each of the templates should be searched in each of the scenes. The results should be visualized and shown in the Jupyter Notebook and your report.

The operations should be implemented from scratch. No Python libraries are allowed! You may use the OpenCV library only for the feature-based approach.

Q2 (30 pts) - Noise Inclusion and Image Derivatives

In this task, you will first add noises—sampled from a normal distribution—to your selected images (any image you prefer) as shown in Eq. 1, then filter the noise from your images, and lastly, take the first-order derivatives and gradient magnitudes via Sobel filters, of the images **with and without Gaussian noise**.

For your gradient calculations, you should use the 2D Sobel operator (Eq. 2) in the x and y axes **and** the Sobel filters with the separated 1D kernels (Eq. 3). In the equations, **A** represents the input image. Finally, the gradient magnitude can be calculated as Eq. 4.

The operations (convolution, derivative calculation, etc.) should be implemented from scratch. No Python libraries are allowed!

$$I_{\text{noisy}}(x, y) = I(x, y) + N(x, y) \tag{1}$$

[2] https://w.wiki/CMto

$$\mathbf{G}_y = \mathbf{A} * \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{G}_x = \mathbf{A} * \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

$$\mathbf{G}_y = (\mathbf{A} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}) * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T \quad \text{and} \quad \mathbf{G}_x = (\mathbf{A} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}) * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T \tag{3}$$

$$||\mathbf{G}||_\mathbf{2} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2} \tag{4}$$

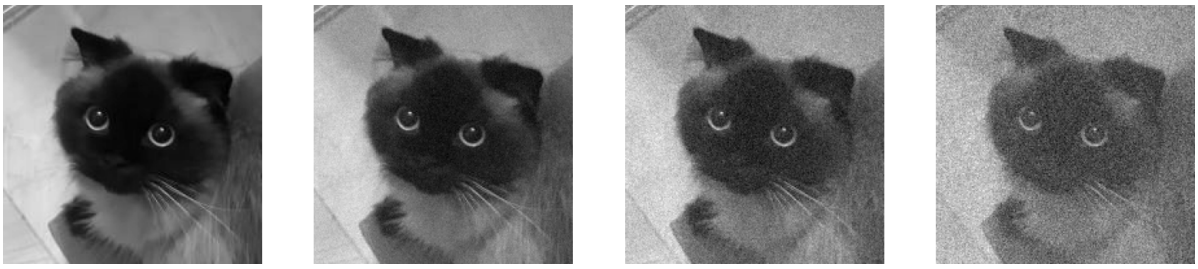

Figure 3: A sample noise inclusion process. (First) A sample grayscale image, (Second) the noise-added output ($\mu = 0, \sigma = 5$), (Third) the noise-added output ($\mu = 0, \sigma = 10$), (Fourth) the noise-added output ($\mu = 0, \sigma = 20$).

Figure 4: A sample noise removal and gradient magnitude comparison process. (First) The noise added $N(\mu = 0, \sigma = 5)$ image, (Second) the gradient magnitude of the noisy image, (Third) the gradient magnitude of the noise removed image with the 5x5 Gaussian kernel.

Q3 (30 pts) - Hough Transform for Line and Circle Detection

In this task, you will implement the Hough Transform **from scratch** to detect straight lines and circles in an image. You will then compare your results with OpenCV's built-in functions, *cv2.HoughLines* and *cv2.HoughCircles*. Select an image that contains both straight lines and circles to effectively demonstrate the performance of your implementation. A sample image is shown in Figure 5.

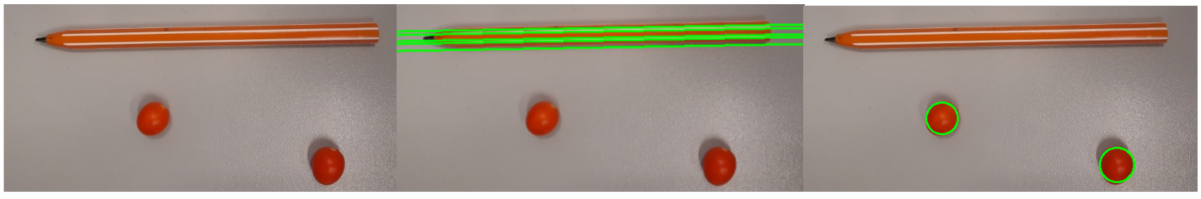

Figure 5: The Hough Transform results of an sample image for line and circle detection.

In you report, make sure to explain the steps clearly and showcase the outputs of the steps separately. **You may use the Canny Edge Detection Algorithm from the OpenCV library.**