İTÜ

04.12.2023

Istanbul Technical University
Computer Engineering Department
BLG 453E - Computer Vision - Fall 23/24
Assignment IV
Due: 18.12.2023, 23.59

Introduction & Note

In this assignment, you will be detecting lines and circles in the given images. Moreover, you will be working on the corner detection methods.

- You should do your own work! GPT is easy to detect ☺. The evaluation of students who use GPT will be more strict! .
- You **can use** the NumPy library **only** to create arrays, change the data type, round the floating values in your array, etc.
- You can work as a group of **up to 2 students** for the assignments. Include your name, surname, and student IDs in your report.
- All the questions asked in the handout should be included in the report. Please indicate the answers in **bold**.
- It is sufficient that only a single student from a group submits the files. The other group member will get the same score even if they do not submit.
- Due to the high disk size of the .npy files, **you are not required to submit .npy files to the homework**.
- **No late submission or correction from e-mail is allowed.** Make sure that you submitted the correct files in time, and check the files you submitted by downloading them from Ninova.

Q1 - Line Detection

In this task, you will perform line regression using the Mean Square Error criterion for images.

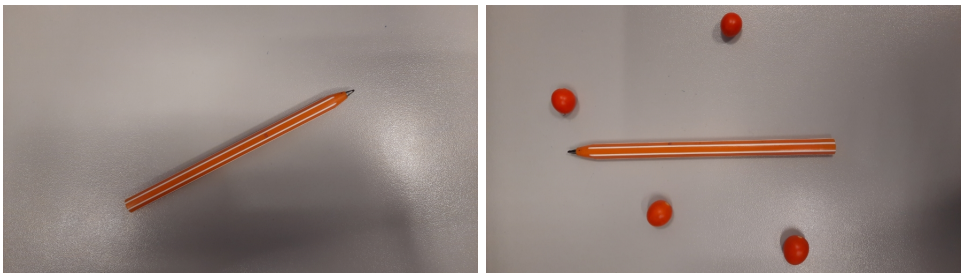

Figure 1: Images to be processed in the task. Left: A clear image with a pencil. Right: A pencil image with outliers.

$$\text{Slope} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{1}$$

- Read the given images in the homework, named Q1.jpg and Q1_2.jpg, via OpenCV.
- Utilize the Canny edge detection algorithm to find the borders of the foreground objects (**Show three results with different thresholds**).

  NOTE: For the **image without outliers**, try to get the cleanest output for edge detection. There might be some noise visible in your output, think about how to remove the noise and keep the borders of the pencil only.)
- Fit a **linear equation** on the pencil pixels. Consider each white pixel of edges as a sample, and try to estimate the linear equation that best fits the distribution.

  Using Equation 1, calculate the slope of the best-fitting line. In the equation; x is the horizontal coordinate, y is the vertical coordinate, $\bar{x}$ is the mean of x coordinates of the white pixels (detected edge pixels), and n is the number of edge pixels. After calculating the slope, create the linear equation by finding the missing coefficients. You can consider that the fitted line will pass from the mean coordinate of the edge pixels.
- Plot the line on the image. The line should start from the first row of the image and be drawn up until the last row of the image.
- Repeat **all the steps** for the image with outlier objects. **Keep the detected edges of the outlier objects as it is**, and fit the line.
- Save the edge detection results and the line plotted image outputs for both tasks.
- Add the input and output images to your report. Explain the steps and comment on the results, in your own words. Answer the following questions.

  ***How do the mis-detected edges influence the fitted line? Which image processing methods did you perform to remove potential noises in the first pencil image (without the outliers)? How do the threshold parameters of the Canny edge detection algorithm impact the resulting edge-detected output?
- Upload the zipped folder that has the output images, and the code file (*.py* or *.ipynb*).

Q2 - Hough Line Transform

In this task, you will perform line regression for images as in Q1; however, the process will be applied with the Hough Line Transform algorithm, using the Normal Form of a line.

- Read the images given in the homework, named Q1.jpg and Q1_2.jpg, via OpenCV.
- Utilize the Canny edge detection algorithm to find the borders of the foreground objects.

  NOTE: For the **image without outliers**, try to get the cleanest output for edge detection. There might be some noise visible in your output, think about how to remove the noise and keep the borders of the pencil only.
- Utilize the Hough Line Transform algorithm in OpenCV (**Show three results with different thresholds for the algorithm.** You can check the documentation of the method from here).
- Plot the line(s) on the image. The line(s) should start from the first row of the image and be drawn up until the last row of the image.
- Repeat **all the steps** for the image with outlier objects. **Keep the detected edges of the outliers as it is**, and perform the algorithm.
- Save the edge detection results and the plotted image outputs for both tasks.
- Add the input and output images to your report. Explain the steps and comment on the results, in your own words. Answer the following question.

  *How does the threshold parameter of the Hough Line Transform algorithm impact the output?
- Upload the zipped folder that has the output images, and the code file (*.py* or *.ipynb*).

Q3 - Hough Circle Transform

In this task, you will perform circle detection on images with coins using the Hough Circle Transform algorithm.

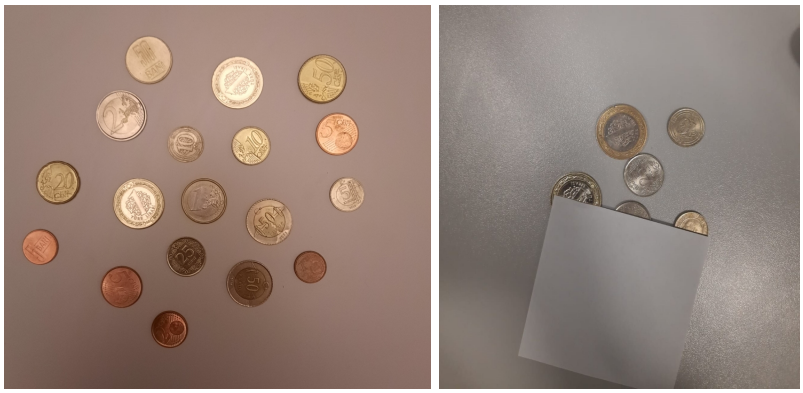

Figure 2: Images to be processed in the task. Left: An image of coins from different sizes. Right: An image with occlusion on the coins.

- Read the images given in the homework, named Q3.jpg and Q3_2.jpg, via OpenCV.
- Utilize the Hough Circle Transform algorithm in OpenCV. (You will be defining two threshold values, minimum radius and maximum radius values for the algorithm parameters. **Show five different parameter combinations and their results.** You can check the documentation of the method from here).

  For the image without occlusion, you should detect all the coins in the scene, where the coins have a radius from **20px** to **60px**.
- Plot the circles on the image.
- Repeat **all the steps** for the image with the occluded coins.
- Save the outputs for both tasks.
- Add the input and output images to your report. Explain the steps and comment on the results, in your own words. Answer the following question.

  *How do the algorithm parameters influence the correct coin detection? Are you able to detect the occluded coins? Explain the possible reasons if you could detect or could not.
- Upload the zipped folder that has the output images, and the code file (*.py* or *.ipynb*).

Q4 - Corner Detection

In this task, you will implement the cornerness measures: Harris & Stephens (1988), which is expressed in Eq. 3, Tomasi-Kanade (1994), which is given in Eq. 4, and Nobel (1998), which is written in Eq. 5. Here, the matrix M is called the **gradient covariance matrix** expressed in Eq. 2.

$$M = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2}$$

$$R = \det(M) - \alpha \cdot \text{trace}^2(M) \tag{3}$$

$$R = \min(\lambda_1, \lambda_2) \tag{4}$$

$$R = \frac{\det(M)}{\text{trace}(M) + \epsilon} \tag{5}$$

- Read an image you select via OpenCV and convert it into the Grayscale format using the NTSC formula. (Try to have objects with sharp edges and corners in your images.)
- Compute image gradients I_x and I_y using Sobel filters in x and y directions.
- Generate M, patch-wise gradient covarince matrix as shown in Equation 2.

  For each 3x3 patch W on the image, you should create a new 2x2 M matrix by performing gradient multiplications, where $I_x^2 = I_x \cdot I_x$, $I_y^2 = I_y \cdot I_y$, $I_x I_y = I_x \cdot I_y$, and w is a weighting matrix for 3x3 image patches with an equal value of 1 for each pixel.
- Compute the cornerness scores via Harris & Stephens (Eq. 3), Tomasi-Kanade (Eq. 4) and Nobel (Eq. 5) methods. In the equations, α is a weighting constant (should be fine-tuned); λ_1 and λ_2 are the eigenvalues of the M matrix; ϵ is a very small constant to prevent zero by zero division.

  The cornerness measure, R, should be calculated on each M matrix of an image.
- On the cornerness measure, R, apply a threshold to detect the locations with high cornerness scores. Optimize the threshold values and the hyperparameters based on the method you are applying, and show your results.
- Assume that you are using the Tomasi-Kanade algorithm. Determine the direction of **minimum** and **maximum** changes. For each M matrix, check the eigenvalues. Count the changes (horizontal, vertical, or diagonal) for all patches, and comment on the overall change trend of the image. Support your comments with images.
- Add the input and output images to your report. Explain the steps and comment on the results, in your own words.
- Upload the zipped folder that has the output images, and the code file (*.py* or *.ipynb*).