İTÜ 250 YIL

08.11.2023

Istanbul Technical University
Computer Engineering Department
BLG 453E - Computer Vision - Fall 23/24
Assignment III
Due: 22.11.2023, 23.59

Introduction & Note

In this assignment, you will be performing Gaussian filtering, getting image derivatives, adding noise to the images, and detecting the edges.

- 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. The other functions you are required to code should be written from scratch.
- 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.
- **In all filtering operations**, you can consider a zero-padding is performed to keep the resolution the same, and the stride of the convolutions is set to 1.

Q1 - Gaussian Filtering

In Question 1, you will be filtering Grayscale images with the Gaussian filters in the shape of 3x3 and 5x5. The Gaussian filters take the weighted mean around a pixel with a given shape of region, by emphasizing the center the most. A sample figure with input and output images is given in Figure 1.

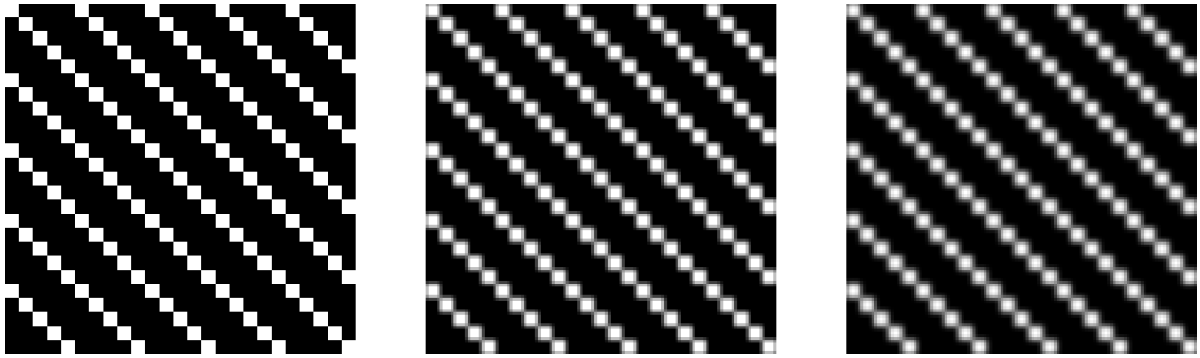

Figure 1: (First) The sample grayscale image, (Second) 3x3 and (Third) 5x5 Gaussian kernel filtered outputs.

$$G_{3x3} = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \text{and} \qquad G_{5x5} = \frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{1}$$

For this task,

- Read an RGB image (try to choose a low-resolution image, the example image has a resolution of 500x500 px), and convert it to Grayscale. (Use the *OpenCV* library to **read** the image.)
- Filter the Grayscale image by using G_{3x3} and G_{5x5} filters in Eq. 1.
- Save the original RGB image and the filtered output images.
- Add the input and output images to your report.

  *Comment on the effects of the kernels with different shapes, in your own words.
- Upload the zipped folder that has the input image, output images and the code file (*.py* or *.ipynb*).

Q2 - Image Derivatives

In Question 2, you will be implementing image derivative operations by using the Sobel filter. You will use the 2D Sobel operator (Eq. 2) in the x and y axes and the seperability of Sobel masks to convolve with the 1D kernels in Eq. 3. In the equations, **A** represents the input image.

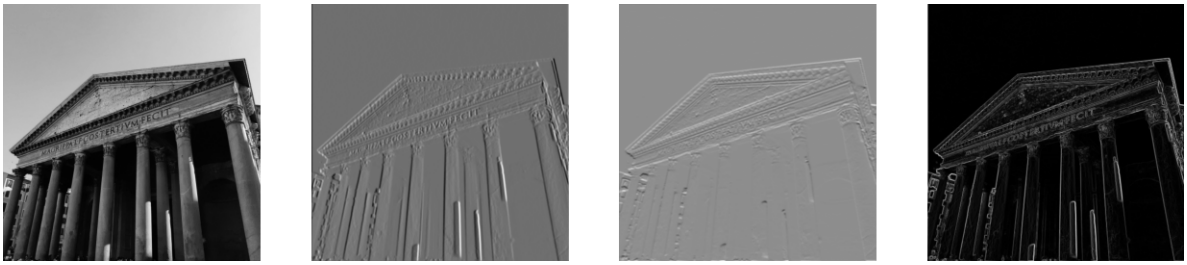

Figure 2: A sample derivative process. (First) A sample input image in grayscale format, (Second) the x-derivative output, (Third) the y-derivative output, and (Fourth) the gradient magnitude output.

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

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

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

For this task,

- Choose an RGB image and read it. (Use the *OpenCV* library to **read** the image.)

  NOTE:** To observe the gradients better, **do not convert the data type into *uint8 but keep it in *float64*, as default for this task.

- Convert the RGB image into grayscale with the NTSC formula as in Homework 1.

- First, get the x and y-axis derivatives of your image by convolving with the 2D Sobel filters in Eq. 2. Next, by utilizing the separability of the Sobel filters, perform the operations in Eq. 3. (You can convert 1D filters into 2D, by vertically or horizontally repeating them.)

- Take the gradient magnitude (L2 Norm, Eq. 4) of the outputs from Eq. 2 and Eq. 3, separately.

- Save the original RGB image and the gradient magnitude images.

- Add the input and output images to your report. Explain the steps and comment on the results, in your own words.

  *With which filters do we observe the vertical and horizontal lines? Why is it that in some edge locations, the intensity is darker/brighter than the other locations? Would the filters perform well when there is Gaussian noise in the image, why? Why do we need to observe the gradient magnitude?

- Upload the zipped folder that has the input image, output images, and the code file (*.py* or *.ipynb*).

Q3 - Noise Inclusion

In Question 3, you will be adding noises - sampled from a normal distribution - on your image and take the first-order derivative.

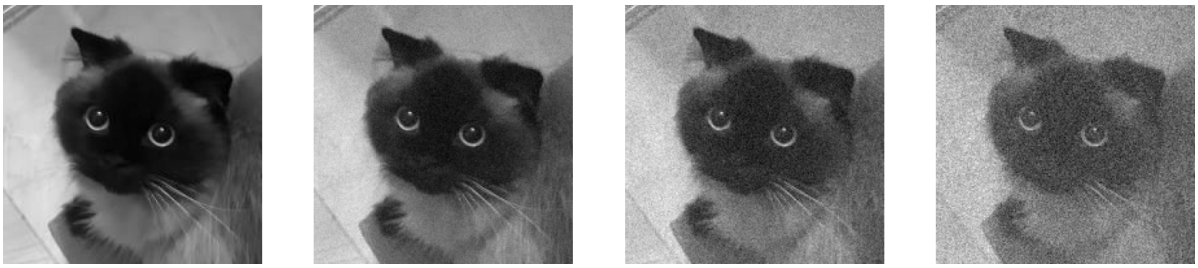

Figure 3: A sample noise inclusion process. (First) A sample grayscale image, (Second) the noise-added output ($\mu = 0, \sigma = 5$), (Third) the noise-added output ($\mu = 0, \sigma = 10$), (Fourth) the noise-added output ($\mu = 0, \sigma = 20$).

$$I_{\text{noisy}}(x, y) = I(x, y) + N(x, y) \tag{5}$$

For this task,

- Choose an RGB image and read it. (Use the *OpenCV* library to **read** the image.)
- Convert the RGB image into grayscale with the NTSC formula as in Homework 1.
- Add sampled noises from a Gaussian distribution ($\mu = 0$) in three σ values and use Eq 5, where $I_{\text{noisy}}(x, y)$ is the noisy image, $I(x, y)$ is the original image, and $N(x, y)$ is the Gaussian noise. (The sampled noises should be in **int8** format. You **can** use np.random.normal()[1] for the task.)
- By using the 2D Sobel filters, find the gradients of the images and get the magnitude, separately.
- Save the original RGB image and the result images.
- Add the input and output images to your report. Explain the noise effect on the images.

  ***Why does the impact of the noise increase as the σ increases? What would have changed in the image if we had a smaller/larger mean for the distribution? How do the gradients of the noisy images change as the σ increases?
- Upload the zipped folder that has the input image, output images, and the code file (*.py* or *.ipynb*).

[1] https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Q4 - Derivatives of the Noisy Images

In Question 4, you will remove the noise of the images you created in Question 3, and visualize the gradient magnitudes.

Figure 4: A sample noise removal and gradient magnitude comparison process. (First) The noise added $N(\mu = 0, \sigma = 5)$ image from Question 3, (Second) the magnitude of the noisy image, (Third) the magnitude of the noise removed image with the 5x5 Gaussian kernel from Question 1.

For this task,

- Use the three noise-added images in Question 3 to process.
- Remove the noise in the images by filtering them with the 5x5 Gaussian filter from Question 1.
- Get the gradient magnitudes of the noise-removed images after processing with the Sobel operator.
- Save the noise-removed images, gradients in the x and y axes, and the gradient magnitudes as a separate image.
- Add the input and output images to your report.

  *Comment on the changes in the gradient magnitudes of the noisy and noise-removed images. Give a specific location in your own image that you observe a change. Why do we observe these changes? What would happen if we used a Gaussian filter with a smaller/larger σ value?
- Upload the zipped folder that has the input image, output images, and the code file (*.py* or *.ipynb*).

Q5 - Edge Detection

In Question 5, you will compare the edge detection performance of the Sobel operator and the Canny edge detection algorithm.

Figure 5: A sample edge detection process. (First) A sample input image in grayscale format, (Second) the gradient magnitude of the input image, (Third) the result of the Canny edge detection algorithm.

For this task,

- Choose an RGB image and read it. (Use the OpenCV library to read the image.)
- Convert the RGB image into grayscale with the NTSC formula as in Homework 1.
- Get the gradient magnitude of the input image.
- To the same input image, perform the Canny[2] edge detection algorithm. Use two different parameters for low and high threshold values. (Use **cv2.Canny()** function from *OpenCV*.)
- Add the input and output images to your report.

  *Comment on the performance of the edge detection between the Sobel operator and the Canny algorithm. What is better/worse in which algorithm? Comment on the results with different threshold parameters for the Canny algorithm.
- Upload the zipped folder that has the input image, output images, and the code file (*.py* or *.ipynb*).

[2] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html