Intro to Image Understanding (CSC420)
Assignment 1

Due Date: October 4^{th}, 2021, 10:59:00 pm
Total: 150 marks

General Instructions:

- You are allowed to work directly with one other person to discuss the questions. However, you are still expected to write the solutions/code/report in your own words; i.e. no copying. If you choose to work with someone else, you must indicate this in your assignment submission. For example, on the first line of your report file (after your own name an information, and before starting your answer to Q1), you should have a sentence that says: *"In solving the questions in this assignment, I worked together with my classmate [name & student number]. I confirm that I have written the solutions/code/report in my own words".*
- Your submission should be in the form of an electronic report (PDF), with the answers to the specific questions (each question separately), and a presentation and discussion of your results. For this, please submit a file named **report.pdf** to MarkUs directly.
- Submit documented codes that you have written to generate your results separately. Please store all of those files in a folder called **assignment1**, zip the folder and then submit the file **assignment1.zip** to MarkUs. You should include a **README.txt** file (inside the folder) which details how to run the submitted codes.
- Do not worry if you realize you made a mistake after submitting your zip file; you can submit multiple times on MarkUs.

Part I: Theoretical Problems (70 marks)

[Question 1] LTI Systems (10 marks)

Linear time-invariant (LTI) systems are a class of systems that are both linear and time-invariant. In linear systems, the output for a linear combination of inputs is equal to the linear combination of individual responses to those inputs. In other words, for a system T, signals $x_1(n)$ and $x_2(n)$, and scalars a_1 and a_2, system T is linear if and only if:

$$T[a_1x_1(n) + a_2x_2(n)] = a_1T[x_1(n)] + a_2T[x_2(n)]$$

Also, a system is time-invariant if a shift in its input merely shifts the output; i.e. If $T[x(t)] = y(t)$, system T is time-invariant if and only if:

$$T[x(n - n_0)] = y(n - n_0)$$

Now, consider a linear time-invariant system T with input $x(n)$ and impulse response $h(n)$. Recall that the impulse response is defined as the output of the system when the input is an impulse function $\delta(n)$, i.e. $T[\delta(n)] = h(n)$, where:

$$\delta(n) = \begin{cases} 1, & \text{if } n = 0, \\ 0, & \text{else.} \end{cases}$$

Prove that $T[x(n)] = h(n) * x(n)$, where $*$ denotes convolution operation.

Hint: represent signal $x(n)$ as a function of $\delta(n)$.

[Question 2] Polynomial Multiplication and Convolution (15 marks)

Vectors can be used to represent polynomials. For example, 3^{rd}-degree polynomial $(a_3x^3 + a_2x^2 + a_1x + a_0)$ can by represented by vector $[a_3, a_2, a_1, a_0]$.
If $\mathbf{u}$ and $\mathbf{v}$ are vectors of polynomial coefficients, prove that convolving them is equivalent to multiplying the two polynomials they each represent.

Hint: You need to assume proper zero-padding to support the full-size convolution.

[Question 3] Image Pyramids (20 marks)

In Gaussian pyramids, the image at each level I_k is constructed by blurring the image at the previous level I_{k-1} and downsampling it by a factor of 2. A Laplacian pyramid, on the other hand, consists of the difference between the image at each level (I_k) and the upsampled version of the image in the next level of Gaussian pyramid (I_{k+1}).
Given an image of size $2^n \times 2^n$ denoted by I_0, and its Laplacian pyramid representation denoted by $L_0, ..., L_{n-1}$, show how we can reconstruct the original image, using the minimum information from the Gaussian pyramid. Specify the minimum information required from the Gaussian pyramid, and a closed-form expression for reconstructing I_0.

Hint: The reconstruction follows a recursive process; What is the base-case that contains the minimum information?

[Question 4] Laplacian Operator (25 marks)

The Laplace operator is a second-order differential operator in the "n"-dimensional Euclidean space, defined as the divergence $(\nabla.)$ of the gradient (∇f). Thus if f is a twice-differentiable real-valued function, then the Laplacian of f is defined by:

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

where the latter notations derive from formally writing:

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n} \right).$$

Now, consider a 2D image $I(x, y)$ and its Laplacian, given by $\Delta I = I_{xx} + I_{yy}$. Here the second partial derivatives are taken with respect to the directions of the variables x, y associated with the image grid for convenience. Show that the Laplacian is in fact rotation invariant. In other words, show that $\Delta I = I_{rr} + I_{r'r'}$, where r and r' are any two orthogonal directions.

Hint: Start by using polar coordinates to describe a chosen location (x, y). Then use the chain rule.

Part II: Implementation Tasks (80 marks)

[Question 4] Edge Detection

In this question, the goal is to implement a rudimentary edge detection process which uses derivative of Gaussian, through a series of steps. For each step (excluding step 1) you are supposed to test your implementation on the provided image, and also on one image of your own choice. Include the results in your report.

Step I - Gaussian Blurring (20 marks): Implement a function that returns a 2D Gaussian matrix for input size and scale σ. Please note that you should not use **any** of the existing libraries to create the filter, e.g. cv2.getGaussianKernel(). Moreover, visualize this 2D Gaussian matrix for two choices of σ with appropriate filter sizes. For the visualization, you may consider a 2D image with colormap, or a 3D graph. Make sure to include the color bar or axis values.

Step II - Gradient Magnitude (20 marks): In the lectures, we discussed how partial derivatives of an image is computed. We know that the edges in an image are from the sudden changes of intensity and one way to capture that sudden change is to calculate the gradient magnitude at each pixel. The edge strength or gradient magnitude is defined as:

$$g(x, y) = |\nabla f(x, y)| = \sqrt{g_x^2 + g_y^2}$$

where g_x and g_y are the gradients of image $f(x, y)$ along x and y-axis direction respectively. Using the Sobel operator, g_x and g_y can be computed as:

$$g_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * f(x, y) \text{ and } g_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * f(x, y)$$

Implement a function that receives an image $f(x, y)$ as input and returns its gradient $g(x, y)$ magnitude as output using the Sobel operator. You are supposed to implement the convolution required for this task from scratch, without using any existing libraries.

Step III - Threshold Algorithm (30 marks): After finding the image gradient, the next step is to automatically find a threshold value so that edges can be determined. One algorithm to automatically determine image-dependent threshold is as follows:

1. Let the initial threshold τ_0 be equal to the average intensity of gradient image $g(x, y)$, as defined below:

   $$\tau_0 = \frac{\sum_{j=1}^{h} \sum_{i=1}^{w} g(i, j)}{h \times w}$$

   where, h and w are height and width of the image under consideration.

2. Set iteration index $i = 0$, and categorize the pixels into two classes, where the lower class consists of the pixels whose gradient magnitudes are less than τ_0, and the upper class contains the rest of the pixels.

3. Compute the average gradient magnitudes m_L and m_H of lower and upper classes, respectively.

4. Set iteration $i = i + 1$ and update threshold value as:

   $$\tau_i = \frac{m_L + m_H}{2}$$

5. Repeat steps 2 to 4 until $|\tau_i - \tau_{i-1}| \leq \epsilon$ is satisfied, where $\epsilon \to 0$; take τ_i as final threshold and denote it by τ.

   Once the final threshold is obtained, each pixel of gradient image $g(x, y)$ is compared with τ. The pixels with gradient higher than τ are considered as edge point and is represented as a white pixel; otherwise it is designated as black. The edge-mapped image $E(x, y)$, thus obtained is:

   $$E(x, y) = \begin{cases} 255, & \text{if } g(x, y) \geq \tau \\ 0, & \text{otherwise} \end{cases}$$

Implement the aforementioned threshold algorithm. The input to this algorithm is the gradient image $g(x, y)$ obtained from step II, and the output is a black and white edge-mapped image $E(x, y)$.

Step IV - Test (10 marks): Use the image provided along with this assignment, and also one image of your choice to test all the previous steps (I to III) and to visualize your results in the report. Convert the images to grayscale first. Please note that the input to each step is the output of the previous step. In a brief paragraph, discuss how the algorithm works for these two examples and highlight its strengths and/or its weaknesses.