Homework 2: Image Features

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2024-09-20
Due: 2024-10-04, at 3:00 PM on Canvas

Instructions:

- Homework is on a "**rolling"** basis and more **questions will be added until 1 week** before the due date. There will be an announcement (on Discord or in class) when new questions are added.
- For all problems in this homework, you can convert your images to **grayscale** for simplicity. So, no need to work with RGB images (unless you want to).
- You can get help from your teammates (or others) for all problems and/or code. However, you will need to submit your report **individually** on Canvas. Reports that are **completely identical** will receive a grade of **zero**.
- The quiz will strongly resemble homework questions, and if you understood/coded the homework yourself, you will be able to answer the quiz questions immediately. Because the quiz will be closed-notes & no internet access, understanding the homework is crucial!
- **Deliverables**: You will submit a **single PDF** file to Canvas. The PDF must contain your **answers**, your **code** (copy-paste in the document), and any requested **outputs** (like images). For convenience, you may use Jupyter notebook and convert it to a PDF.
- **Grading:** This homework will be scaled to 10pts of your final grade. Grading rubric will be posted on Canvas.

Problem 1 (tbd pts): Image Compression using DFT.

- Compute the DFT of the "Mines" image provided in homework material
- Display both DFT magnitude and phase (as images)
- Compress the image by keeping the top 1000 DFT coefficients (in magnitude)
- Display the DFT magnitude image that contains the top 1000 DFT coefficients
- Reconstruct the original image by using *only* the top 1000 DFT coefficients (via inverse DFT)
- Display the resulting reconstructed image
- Answer: how many elements/pixels the original image has? How many elements the DFT compression has? What is the compression ratio?

Notes:

- For this problem, you can use any existing DFT function/library
- See the example in lecture "03_2_Fourier_Transform_II"

Problem 2 (tbd pts): Simple Canny Edge Detector

- Filter the Mines image with derivatives of Gaussian filters to find its x and y gradients I_x, I_y:
    - Create a Gaussian filter kernel of $\sigma = 1.4$ with appropriate size
    - Display the Gaussian filter kernel as an image
    - Convolve the Gaussian kernel with 1D derivate filter kernels along the x and y directions to obtain derivative of Gaussian kernels
    - Display both x and y derivative of Gaussian kernels as images
    - Filter the Mines image with derivative of Gaussian filters to obtain I_x, I_y
    - Display I_x, I_y
- Compute the magnitude and orientation (theta) of gradient
- Display both magnitude and orientation of gradient as images

Homework 2: Image Features

Kaveh Fathian, Email: fathian@ariarobotics.com

Handout: 2024-09-20
Due: 2024-10-04, at 3:00 PM on Canvas

- Remove pixels in the magnitude image that are below a certain threshold (pick the threshold appropriately to keep edges)
- Display the resulting edge image

Notes:

- The size of the Gaussian filter kernel should be large enough so that the pixels on the edge of the kernel image are close to zero
- See the examples in lectures “04_1_Edge_Detection” and “04_2_Edge_Detection_II”

Problem 3 (tbd pts): Harris Corner Detector

- Use the Mines image as input and compute its gradients I_x, I_y via convolving the image with the Sobel filters (of appropriate kernel size)
- Display I_x, I_y and their Hadamard product $I_x \circ I_y$ as images
- Compute the “cornerness” score C using only I_x, I_y, and $I_x \circ I_y$ and $\alpha = 0.04$ (see lecture material)
- Display C as an image
- Threshed C to pick high cornerness pixels AND display the results as an image
- Use non-maximum suppression (with appropriate threshold) to pick corners as individual pixels
- Display the corners as an image (with black background) AND display them overlapped on the original image

Notes:

- For this problem, you *cannot* use any existing Harris corner detector implementation
- For non-maximum suppression, you can search the neighboring pixels of a target pixel (e.g., pixels in a 5x5 box centered at the target pixel) and keep the target pixel only if it has an intensity larger than all of its neighbors.
- See the examples and algorithm in lecture “05_1_Features”