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 code the problems and submit your report **individually** on Canvas. Reports/code that are **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.
- Only use **images provided** in the homework material, as requested by each problem. Using any other image, will result in a **grade of zero**.
- **Grading:** This homework will be scaled to **10pts** of your final grade. Grading **rubric** will be posted on **Canvas**.

Problem 1: 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: 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

Homework 2: Image Features

Kaveh Fathian, Email: fathian@ariarobotics.com

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

- Compute the magnitude and orientation (theta) of gradient
- Display both magnitude and orientation of gradient as images
- 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: 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.
- You should **not** compute the matrix M and/or find its eigenvalues. Note that C can be computed directly from I_x, I_y:

$$C = \det(M) - \alpha\,\mathrm{trace}(M)^2 = g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2 - \alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

Reminder: $a \circ b$ is Hadamard product (element-wise multiplication)

- See the examples and algorithm in lecture "05_1_Features"

Problem 4: SIFT Features

In this problem, you will implement a SIFT-like algorithm (we will ignore scale). Use an existing Harris corner detector (e.g., from OpenCV or Scikit packages) and extract Harris corners in the image named "**image1**" provided in homework material. Adjust the corner detector parameters to get **at least 50** corners.

- Compute image gradients I_x, I_y via convolving the image with Sobel filters (of appropriate kernel size), and use them to compute magnitude and orientation of gradient for each pixel.
- For each corner:

Homework 2: Image Features

Kaveh Fathian, Email: fathian@ariarobotics.com

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

    - In a 16x16 window around the corner, compute gradient orientation histogram. To achieve this, use a histogram with 36 bins, each covering 10 degrees, to encompass 0 to 360 degrees.
    - Find the dominant orientation, and normalize orientations by rotating them so that the dominant orientation is in the first bin.
    - Create a SIFT descriptor using the (rotated) 16x16 window. That is, use 16 sub-blocks of 4x4 size. For each sub-block, create an 8-bin orientation histogram. Stack the histogram values of all sub-blocks so that a 128-element descriptor vector is created.
    - Normalized the descriptor (to the range 0-1). Clamp all vector values > 0.2 to 0.2, and re-normalize.
- For **only one** of the corners:
    - Display the gradient orientation histogram and print the dominant orientation
    - Re-compute & display the gradient orientation histogram after rotation
    - Display the 8-bin orientation histogram for each sub-block (we have 4x4 sub-blocks, so a total of 16 histograms)
    - Print out the 128-element descriptor vector constructed from the histograms
    - Print out the normalize descriptor, and re-normalized descriptor

Note:

- You cannot use any existing implementations of the SIFT algorithm.
- You can ignore the features around the edge of the image where the 16x16 window falls outside of the image.
- You can normalize a descriptor vector (to the range 0-1) by dividing the vector by its max element.
- See the details in lecture "06_1_Feature_Matching".

Problem 5: Feature Matching

In this problem, you will match the SIFT-like descriptors you found in Problem 4 across two images. Use an existing Harris corner detector (e.g., from OpenCV or Scikit packages) and extract Harris corners in the images named "**image1**" and "**image2**" provided in homework material. Recall, that your corner detector parameters should be set so that you get **at least 50** corners for each image. It is okay to ignore corners around the image borders, for which you don't get SIFT descriptors.

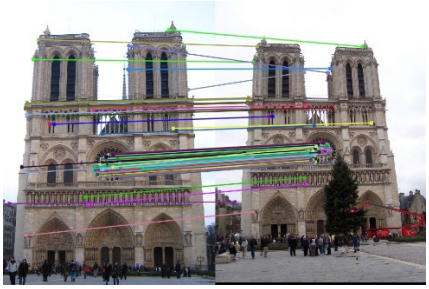
- Find corners & compute your SIFT-like descriptors using Problem 4.
- Calculate the (L2 norm) distance between all pairs of descriptors across the two images (much like `scipy.spatial.distance.cdist()`, but your own implementation).
- Match each keypoint in **image1** to a keypoint in image2 based on closest distance
- Match each keypoint in **image2** to a keypoint in image1 based on closest distance
- Explain: did you get the same matchings from image1 to image2 and vice versa? If yes, do you think that will always be the case? If no, explain why they can be different.
- Find keypoints that matched **bi-directionally** across image1 and image2

Homework 2: Image Features

Kaveh Fathian, Email: fathian@ariarobotics.com

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

- Display the results: show images side-by-side, display the keypoints on each image, and display the matchings with a line across images. Example:



- Explain: Are all keypoints matched correctly across the 2 images? If yes, will that always be the case? If no, why wrong matching happens?

- Now, repeat the matching from **image1** to image2 by computing the distance of closest (NN1) and second-closest (NN2) keypoints, and computing the Lowe's ratio (NN1/NN2).
- Discard matchings that have Lowe's ratio **> 0.7**
- Repeat above for **image2** to image1, and keep keypoints/matchings that are bi-directional
- Display the results: show images side-by-side, display the keypoints on each image, and display the matchings with a line across images.
- Explain: What is the difference between previous results and the results with Lowe's ratio?

Note:

- See the details in lecture "06_1_Feature_Matching".