Project 1

Computer Vision(CSI4116-01)

Spring, 2022

Due 11th April, 23:55

Task1

Noise Add & Removal Filter

Task 1

- Digital images contain various types of noises.
- We are going to add and remove some noises.

Task 1-1. Noise Addition

- Implement 3 noise addition functions in **noise.py** file
  - add_gaussian_noise(image)
    - Take an image, and return an image with Gaussian noise
  - add_uniform_noise(image)
    - Take an image, and return an image with Uniform noise
  - add_impulse noise(image)
    - Take an image, and return an image with pepper noise
- You cannot modify the main function
- '**bird.jpg**' is the input image for this task

Task 1-2. Noise Estimation & Removal

- Implement 3 filters using convolution in **denoise.py** file
  - apply_median_filter(img, kernel_size)
  - apply_bilateral_filter(img, kernel_size, sigma_s, sigma_r)
  - apply_my_filter(img)
    - Design any filter excluding median and bilateral filter
- Complete task1(src_path, clean_path, dst_path) function so that your file
  - Finds an optimal filter from the above 3 filters and window size for each image that gives the minimum RMS error
  - Saves denoised image filtered with optimal solution to *dst_path*

  - Output image size should be the same as the input image

Task 1-2. Test

- Test images are included in “test_images/” directory.
- You may use {}_noisy.jpg as an input for your filter function.
- You can use {}_clean.jpg as an answer to calculate RMS.

- RMS calculation code for color image is provided in the file

Performance measure - RMS

- We will use RMS(Root Mean Square) to evaluate the performance of your code. If your denoising program exceeds the performance of our advanced algorithm, you will get extra score for this assignment.

- RMS Baseline
  - **gaussian.jpg** 29.81 < RMS < 30.01
  - **uniform.jpg** 16.80 < RMS < 16.89
  - **impulse.jpg** 51.74 < RMS < 52.06
  - **cat_clean.jpg** RMS < 8.0
  - **fox_clean.jpg** RMS < 11.1
  - **Snowman_clean.jpg** RMS < 10.2

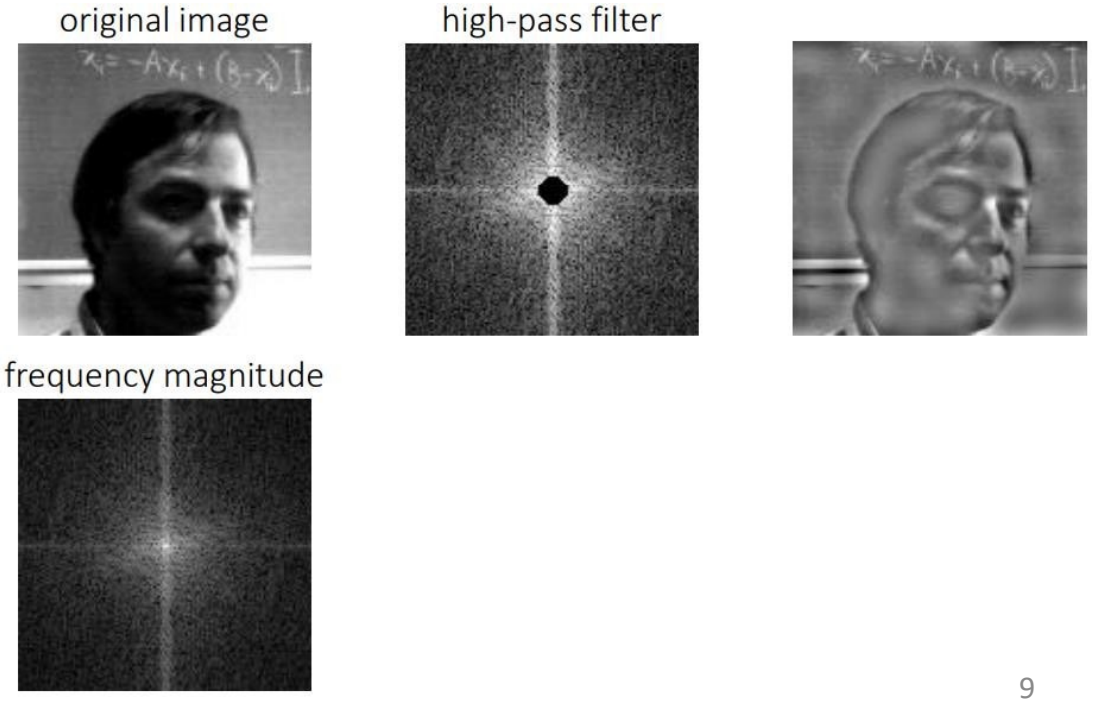
- RMS Advanced
  - **cat_clean.jpg** RMS < 7.3
  - **fox_clean.jpg** RMS < 10.1
  - **Snowman_clean.jpg** RMS < 8.4

Task2

Fourier Transform

Task2 - Introduction

- Fourier transform is a way to transfer image into frequency domains. We can apply frequency domain filtering to image processing.



Task2 - Fourier Transform (40pts)

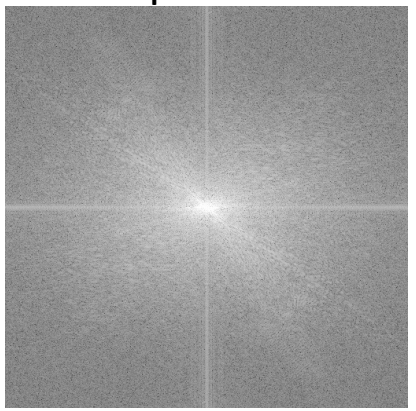
- **In this task, you will implement these functions**
  - fftshift & ifftshift (5pts)
  - fm_spectrum (5pts)
  - low_pass_filter (5pts)
  - high_pass_filter (5pts)
  - denoise1 (10pts)
  - denoise2 (10pts)

Task2 - Specification

- fm_spectrum
  - Get **frequency magnitude spectrum image** of the input image.
  - Spectrum image should be **shifted to the center (use fftshift function)**.
  - You may adjust intensity for recognizing the spectrum.

original

fm spectrum



Task2 - Specification

- low_pass_filter
  - Get filtered image that passes through **low-pass filter**.
  - Use fourier transform.
  - User can set the **frequency threshold (radius of the mask)**.

original

filtered

Task2 - Specification

- high_pass_filter
  - Get filtered image that passes through **high-pass filter**.
  - Use fourier transform.
  - User can set the **frequency threshold (radius of the mask)**.

original

filtered

Task2 - Specification

- denoise1
  - **Denoise checker effect** on the given image. (task2_noised1.png)
  - You don’t have to write a function for general purpose. It can work only for the given image.
  - Use fourier transform.

noised

Task2 - Specification

- denoise2
  - **Denoise wave effect** on the given image. (task2_noised2.png)
  - You don’t have to write a function for general purpose. It can work only for the given image.
  - Use fourier transform.
  - You may consider band-reject filter.

noised

Task2 - Specification

- The skeleton code will be provided. You should implement functions on the provided skeleton code.
- You can see the result images by running code.
- Submit the complete code.
  - task2
    - fourier.py
- Use **grayscale** to read the image.

Task2 - Extra Credit

If you implement the following fourier transforms by yourself, you can get extra credit.

- DFT and IDFT (5pts)
- FFT and IFFT (5pts)

*Fast Fourier Transform (O(nlogn)) is a much faster implementation of Discrete Fourier Transform ($O(n^2)$)

Report

- You should submit a report that explains **your implementation and intermediate/result images**.

- Task 1
  - Explain your implementation with screenshots of the code
  - For task 1-2, state the optimal solution for each image and the analysis of such result.
- Task 2
  - Explain your implementation with screenshots of the code
  - Make sure to include the results after going through high-pass filter, low-pass filter, denoise1, and denoise2 as well as fm spectrum of those images.

Grading Policy

- Total 100pts
  - Task1 (40pts)
  - Task2 (40pts)
  - Report (20pts)
  - Extra Credit (13pts)

- You cannot get more than 100pts for this assignment. Extra credit will only be added if you get deduction in Task1, 2 and the report.

- If there are some problems in grading(code error, wrong file name or structure), you may get some penalty.

Grading Policy

- Task1 (40pts+3pts)
  - Implementing gaussian, uniform, and impulse noise (10 pts)
  - Implementing median, bilateral, and your own filter (18 pts)
  - Your RMS error is in baseline RMS boundary (12 pts)
  - Your RMS error for noise removal is in advanced RMS boundary (3 pts)
- Task2 (40pts+10pts)
  - fftshift & ifftshift (5pts)
  - ft_spectrum (5pts)
  - low_pass_filter (5pts)
  - high_pass_filter (5pts)
  - denoise1 (10pts)
  - denoise2 (10pts)
  - DFT and FFT implementation (extra 10pts)
- Report (20pts)

ANY KIND OF PLAGIARISM WILL RECEIVE 0

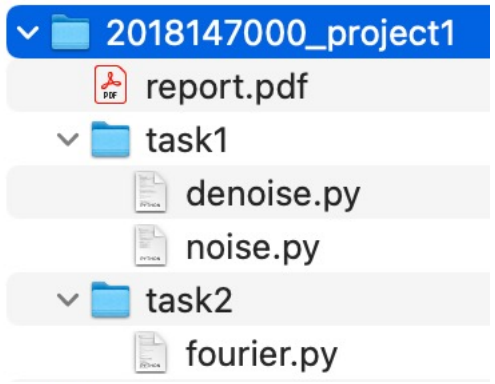
Caution

- Allowed library functions
  - cv2.imread
  - cv2.imwrite
  - numpy.fft.fft2
  - numpy.fft.ifft2
  - numpy.random
  - Other numpy function for basic calculation.
  - matplotlib
- Do not use any short-cut function(especially filters) in third-party packages except the allowed functions above.
- You can add your own .py file for modulization. But if so, you must write about it in the report.

Submission

- Submit the **zip** file in **LearnUs**. The file must have the structure below.

- [Student ID]_project1.zip ex) 2018147000_project1.zip
  - task1
    - denoise.py
    - noise.py
  - task2
    - fourier.py
  - report.pdf



Due Date

- 11th April, 23:55 KST
- Delay Policy
  - -50% pts for ~12th April, 23:55 KST