Assignment 1

Computer Vision (CSI4116-01)

Spring, 2021

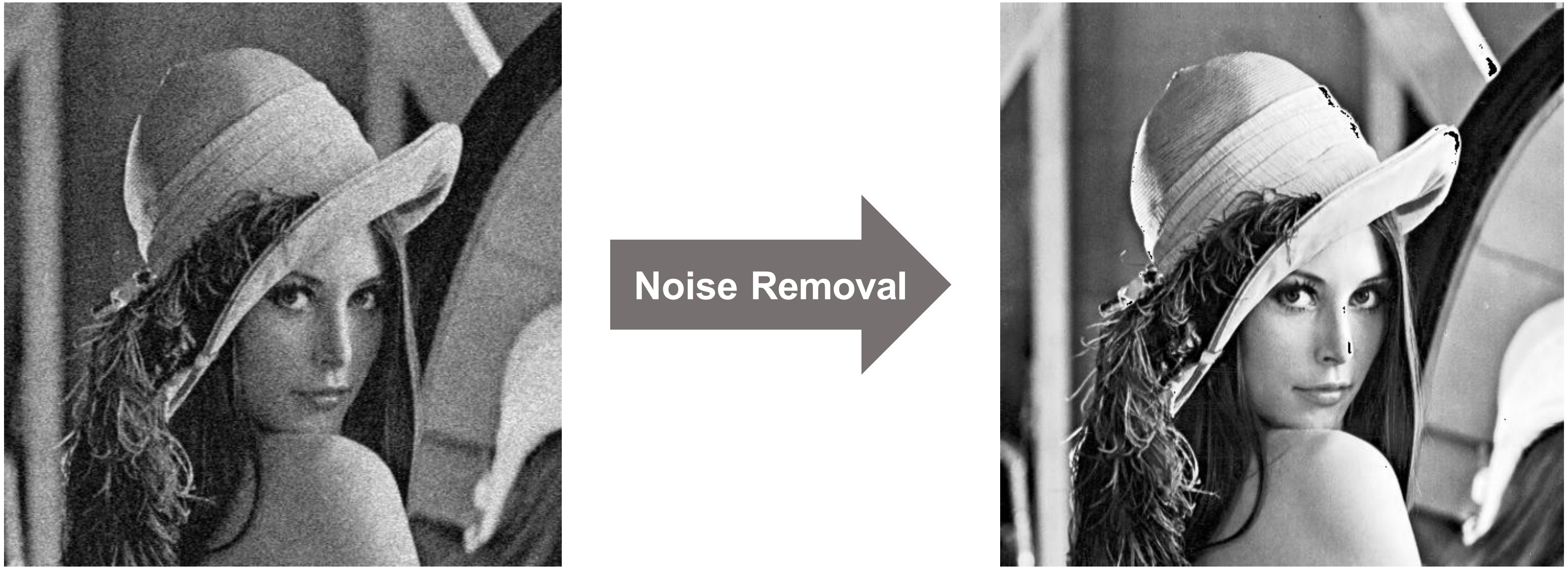
Due date 5th April, 23:55

Task1

Noise Removal Filter

Task1 - Introduction

- We are going to estimate and remove noise contained in images using convolution.



Task1 - Noise Estimation & Removal

- Images taken from real world contains various types of noise.
- By using convolution with various filters, we can reduce noise contained in images.
- In this assignment, we are going to do noise estimation and removal by using the ideas we learned in this course.

Task1 - RMS (Root Mean Square)

- RMS (Root Mean Square) is the square root of the mean square.
- RMS is commonly used as a performance measure in image restoration problem.
- We will use RMS as a performance measure to evaluate performance of your code.

Task1 - Assignment Details

- Your program should do convolution using various noise- removal filters with various window sizes to remove noise contained in input images.

- By doing so, your program should try to minimize RMS error between output images and clean image.

- It means that your program finds optimal filters and window sizes that can produce best output for each input images. (output that shows minimum RMS error)

Task1 - Specification

- Implements function that applies median filter to image.
- Implements function that applies average filter to image.
- Implements function that applies bilateral filter to image.
- Implements main function that finds optimal filter from above three filters and optimal window size for each input image, and then produce output images with minimum RMS error.
- Implements some advanced algorithms in your main function in order to improve performance of your program.
- Write a report about the optimal solution for each image and the analysis of such result.

Task1 – Skeleton code

- We provide skeleton code for this assignment.
- **In task1/utils.py**, there is a utility function for calculating RMS error between two images. You may use this function if you need.
- You should implement details of 4 functions in **task1/noise.py**
- You should include your output images in the "outputs/" directory, of which name should be same with the input.
  ex) input: "inputs/test1.jpg", output: "outputs/test1.jpg"

Task1 - Details (cont'd)

- apply_average_filter(img, kernel_size)
  - Function that **applies average filter** to given image.
  - It takes two arguments.
    - img (np.ndarray object): source image. It should be a np.ndarray object, which is a data type returned from cv2.imread function. You should apply convolution to this image.
    - kernel_size (int): filter size of average filter.
  - It should return output image (np.ndarray object), which is a result of convolution with average filter.

Task1 - Details (cont'd)

- apply_median_filter(img, kernel_size)
  - Function that **applies median filter** to given image.
  - It takes two arguments.
    - img (np.ndarray object): source image. It should be a np.ndarray object, which is a data type returned from cv2.imread function. You should apply convolution to this image.
    - kernel_size (int): filter size of median filter.
  - It should return output image (np.ndarray object), which is a result of convolution with median filter.

Task1 - Details (cont'd)

- apply_bilateral_filter(img, kernel_size, sigma_s, sigma_r)
  - Function that **applies bilateral filter** to given image.
  - It takes four arguments.
    - img (np.ndarray object): source image. It should be a np.ndarray object, which is a data type returned from cv2.imread function. You should apply convolution to this image.
    - kernel_size (int): filter size of bilateral filter.
    - sigma_s (int): sigma value for G_s(gaussian function for space)
    - sigma_r (int): sigma value for G_r (gaussian function for range)
  - It should return output image (np.ndarray object), which is a result of convolution with bilateral filter.

Task1 - Details (cont'd)

- task1(src_img_path, clean_img_path, dst_img_path)
  - **Main function** for task 1.
  - It takes three arguments.
    - src_img_path (string): image path of the input image. you should read image
      by using this path.
    - clean_img_path (string): image path of the clean image. you
    - dst_img_path (string): image path of the output image. you should save your output image by using this path.
  - Your main algorithms, finding optimal conditions for noise removal and producing output images with least RMS error should be in this function.
  - You may use other functions, including apply_median_filter, apply_average_filter, apply_bilateral_filter functions you implemented.

Task1 - Performance measure

- Given clean & noisy image pairs, your output images will be scored by comparing RMS with outputs produced by our programs (Baseline, Advanced).

- Test images are included in "inputs/" directory

- Baseline
  - It tries to find optimal option for noise removal.
  - It chooses one filter that shows best performance among three filters, and applies convolution only once using that filter.
  - It chooses best window size.

Task1 - Performance measure (cont'd)

- Advanced
  - Our advanced program uses further techniques in order to boost up the performance.
  - If your program exceeds performance of our advanced algorithm, you will get **extra score** for this assignment.

- Baseline, advanced RMS boundaries of each image will be announced soon.

Task1 - Grading Policy

- Total 40 points
  - Implementing average filter (5 points)
  - Implementing median filter (5 points)
  - Implementing bilateral filter (10 points)
  - Your RMS error is in our baseline RMS boundary (5 points)
  - Report of the analysis (15 points)

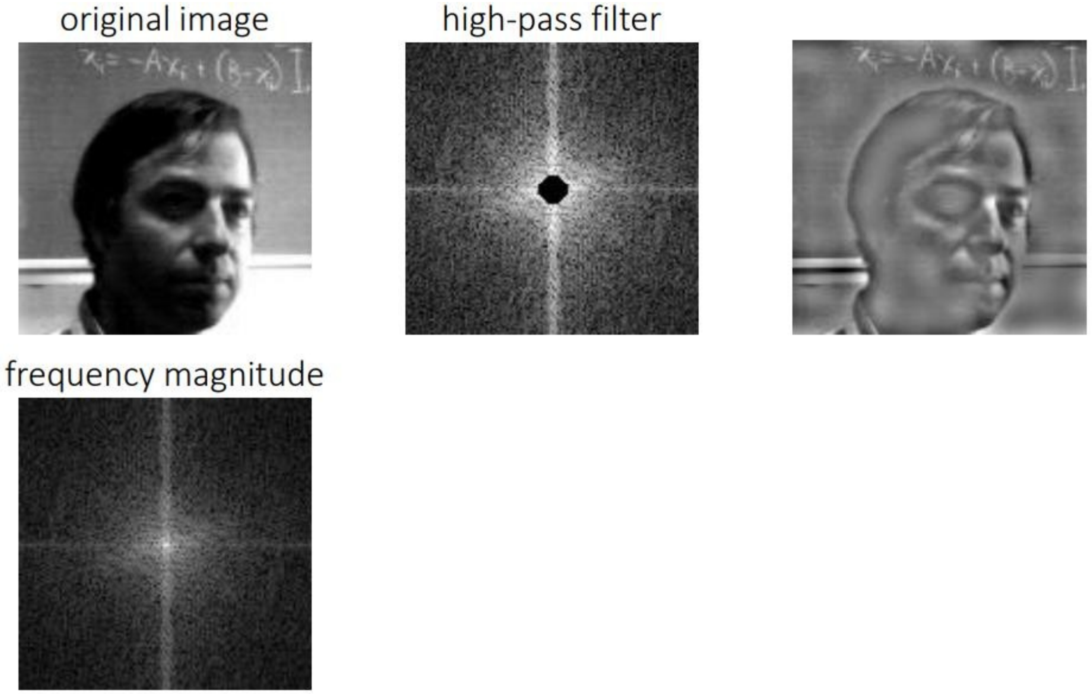
- And extra 10 points
  - Your RMS error is in our advanced RMS boundary (10 points)

Task2

Fourier Transform

Task2 - Introduction

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



Task2 - Fourier Transform (40pts)

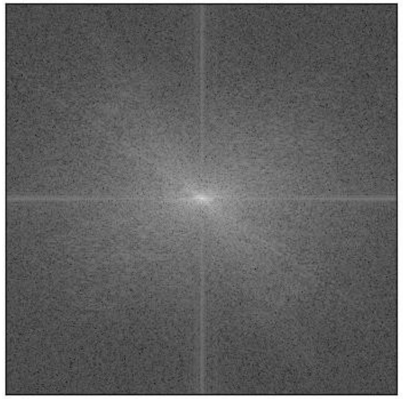
- In this task you will implement some functions
  - ft_spectrum (5pts)
  - low_pass_filter (5pts)
  - high_pass_filter (5pts)
  - denoise1 (10pts)
  - denoise2 (15pts)

Task2 - Specification

- fm_spectrum
  - Get **frequency magnitude spectrum image** of input Image.
  - Spectrum image should be **shifted to center**.
  - You may adjust intensity for recognizing spectrum.

Original

Spectrum



Task2 - Specification

- low_pass_filter
  - Get filtered image that pass through with **low-pass filter**.
  - Use fourier transform.
  - User could be set **frequency threshold**.

Task2 - Specification

- high_pass_filter
  - Get filtered image that pass through with high- pass filter.
  - Use fourier transform.
  - User could be set frequency threshold.

Original

High-pass

Task2 - Specification

- denoise1
  - **Denoise checker effect** with given sample image. (courrpted_1.png)
  - You don't have to write function for general purpose. Only for given image.
  - Use fourier transform.

Task2 - Specification

- denoise2
  - **Denoise wave effect** with given sample image. (courrpted_2.png)
  - You don't have to write function for general purpose. Only for given image.
  - Use fourier transform.
  - You may consider band-reject filter.

Task2 - Specification

- The Skeleton code will be provided. You should implement functions in there.
- You can see result images by running code.
- Submit complete code and two denoised image.
  - task2
    - fourier.py
    - denoised1.png
    - denoised2.png
- Use **grayscale** for read image.

Task2 - Extra Credit

- If you implement fft and ifft function for yourself, you can get extra credit. (10pts)
- You don't have to implement fast fourier transform. **Discrete fourier transform** will be ok.

Report

- You should submit report include **explanation for your implementation and intermediate/result images.**

Caution

- Allowed library functions
  - cv2.imread
  - cv2.imwrite
  - numpy.fft.fft2
  - numpy.fft.ifft2
  - Other numpy function for basic calculation.
- Do not use any short-cut function(especially filters) in third-party packages except above allowed functions.

- You can add your own .py files for modulization. But if that so, you have to write about it on report.

Submission

- Submit the zip file that has below structure to yscec.

- [ID]_assignment1.zip ex) 2015147000_assignment1.zip
    - task1
        - noise.py
        - utils.py
    - task2
        - fourier.py
        - denoised1.png
        - denoised2.png
    - report.pdf

Grading Policy

- Total 120pts
  - Task1 (40pts)
  - Task2 (40pts)
  - Report (20pts)
  - Extra Credit (20pts)

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

Grading Policy

- Task1 (40pts+10pts)
  - Implementing average filter (10pts)
  - Implementing median filter (10pts)
  - Implementing bilateral filter (10pts)
  - Your RMS error is in our baseline RMS boundary (10pts)
  - Your RMS error is in our advanced RMS boundary (extra 10pts)

- Task2 (40pts+10pts)
  - ft_spectrum (5pts)
  - low_pass_filter (5pts)
  - high_pass_filter (5pts)
  - denoise1 (10pts)
  - denoise2 (15pts)
  - DFT implementation (extra 10pts)