CSE166_FA23_assignment_4

November 7, 2023

1 CSE 166 Image Processing, Fall 2023 - Assignment 4

Instructor: Ben Ochoa

Assignment due: Mon, Nov 13, 11:59 PM

Name:

PID:

1.1 Instructions

1. All solutions to the problems below must be written in this notebook. Programming aspects of the assignment must be completed using Python (preferably 3.10).
2. This assignment must be completed **individually**. For more details, please review the Academic Integrity Policy and Collaboration Policy on Canvas.
3. The packages you need to complete the assignment will be indicated in the problem descriptions.You are not allowed to use the packages that directly solve the problems. Feel free to ask the instructor and TAs if you are unsure about the packages to use.
4. It is highly recommended that you begin working on this assignment early.
5. After finishing the assignment in the notebook, please export the notebook as a PDF and a python source file. You need to **submit 3 files: the Notebook, the PDF and the python file** (i.e. the `.ipynb`, the `.pdf` and the `.py` files) on Gradescope.

- To convert the notebook to PDF, you can choose one way below:
  - You may first export the notebook as HTML, and then print the web page as PDF
    - e.g., in Chrome: File → Save and Export Notebook as → "HTML"; or in VS-code: Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS), search for Jupyter: Export to HTML
    - Open the saved web page and right click → Print... → Choose "Destination: Save as PDF" and click "Save")
  - If you have XeTex installed on your machine, you may directly export the notebook as PDF: e.g., in Chrome, File → Save and Export Notebook as → "PDF"
  - You may use nbconvert to convert the ipynb file to pdf using the following command `jupyter nbconvert --allow-chromium-download --to webpdf filename.ipynb`

- To convert the notebook to python file, you can choose one way below:
  - You may directly export the notebook as py: e.g., in Chrome, File → Save and Export Notebook as → "Executable script"; or in VScode: Open the Command Palette and search for Jupyter: Export to Python Script
  - You may use nbconvert to convert the ipynb file to python file using the following command `jupyter nbconvert --to script filename.ipynb`

6. Please make sure the content in each cell (e.g. code, output images, printed results, etc.) are clearly visible and are not cut-out or partially cropped in your final PDF file.
7. While submitting on gradescope, please make sure to assign the relevant pages in your PDF submission for each problem.

Late Policy: Assignments submitted late will receive a 15% grade reduction for each 12 hours late (i.e., 30% per day). Assignments will not be accepted 72 hours after the due date. If you require an extension (for personal reasons only) to a due date, you must request one as far in advance as possible. Extensions requested close to or after the due date will only be granted for clear emergencies or clearly unforeseeable circumstances.

2 Problem 1: Textbook Problems (5 points)

2.1 a) Problem 5.10 (2 points)

Your answer here:

2.2 b) Problem 5.12 (2 points)

Your answer here:

2.3 c) Problem 7.7 (1 point)

Your answer here

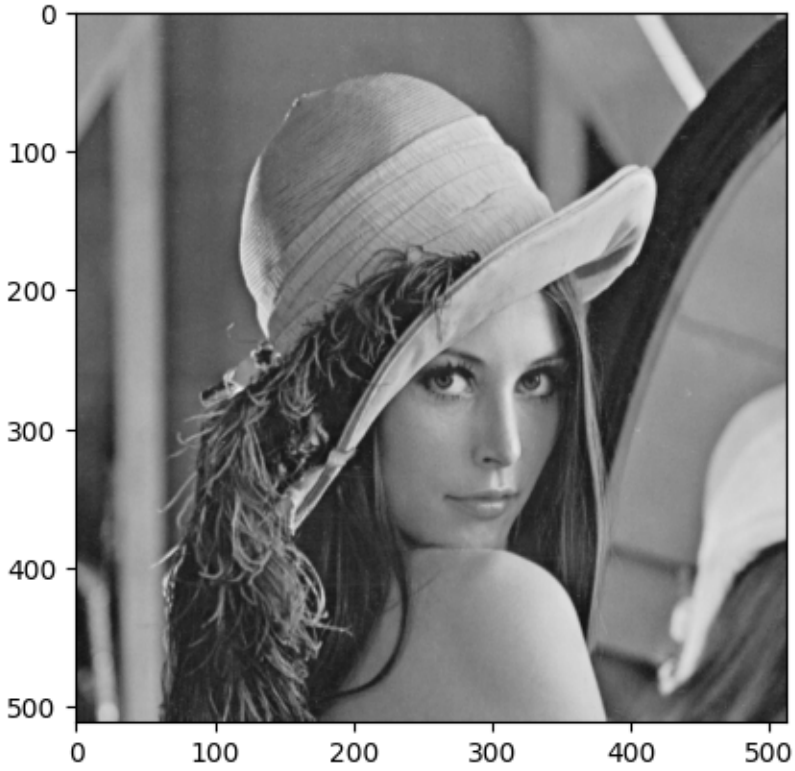
3 Problem 2: Programming: Image Restoration and Color Image Processing (30 points)

3.1 Part 1: Spatial Image Restoration (10 points)

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
import skimage
from skimage.color import rgb2gray
import math
```

[ ]:
```
# Read and display image
img = rgb2gray(plt.imread('Lenna.png'))
plt.imshow(img, cmap='gray', vmin=0, vmax=1)
```

```
plt.show()
```



1) Complete the function **adaptive_local_noise_reduction** that computes an estimate of the original image $\hat{f}(x, y)$ given a noisy image $g(x, y)$, the overall noise variance σ_η^2, and the filter window size using the adaptive expression

$$\hat{f}(x, y) = g(x, y) - \frac{\sigma_\eta^2}{\sigma_{S_{xy}}^2}\left(g(x, y) - \bar{z}_{S_{xy}}\right)$$

and enforce the assumption $\sigma_\eta^2 \leq \sigma_{S_{xy}}^2$, where the local mean $\bar{z}_{S_{xy}}$ and local variance $\sigma_{S_{xy}}^2$ are calculated over the filter window centered at (x, y).

```
[ ]: # function that computes an estimate of the original image for a given noisy␣
     ↪image
def adaptive_local_noise_reduction(noisy_img, var, window_size):
    """
    noisy_img: image with noise added (H,W)
    var: overall noise variance
    window_size: filter window (n,n)

    returns:
    fhat_img: estimate of the image after noise removal (H,W)
    """
```

```
    # your code here
    fhat_img = []

    return fhat_img
```

2) Apply Gaussian noise with variance 0.01 using the function **skimage.util.random_noise** for the Lenna image. Then, call the **adaptive_local_noise_reduction** with the noisy image and a window size of $(5, 5)$ and $(9, 9)$.

Note: Clamp the restored image such that all values less than zero are set to zero and all values greater than 1 are set to 1.

[ ]:
```
"""
Apply Gaussian noise with a variance of 0.01 for the Lenna image.
Then, call the adaptive_local_noise_reduction with the noisy image and window␣
 ↪sizes of (5,5) and (9,9).
"""
# your code here
```

3) Display the input image, noisy image, and two restored images. Set the value range to [0, 1].

[ ]:
```
"""
Display the input image, noisy image, and restored image.
"""
# your code here
```

4) Briefly discuss how well the restored images approximates the original image and how the window size affect the results. Is a local noise reduction filter suitable for this type of noise? What are the trade offs of using adaptive local noise reduction versus other noise reduction techniques such as an arithmetic mean filter? What are the advantages and disadvantages of using adaptive local noise reduction?

Your answer here:

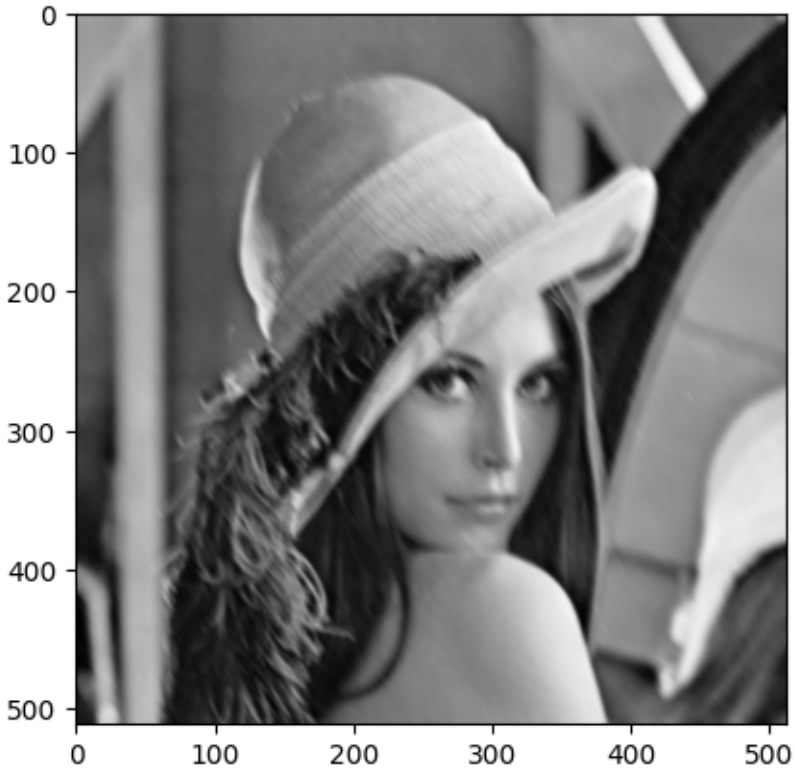
3.2 Part 2: Inverse Filtering (10 points)

The degraded image $d(x, y)$ is the result of convolving an input image with the degradation function

$$h(x,y) = \frac{1}{89}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 7 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 7 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 8 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

```
[ ]: # Read and display the image
degraded_img = 255*plt.imread('Lenna_degrade.png')[:,:,0]

plt.imshow(degraded_img, cmap='gray', vmin=0, vmax=255)
plt.show()
```



1. Complete the function **inverse_filter** that performs inverse filtering in the frequency domain. The function takes as input the degraded image $d(x, y)$, degradation function $h(x, y)$ and a threhold value t, and returns the estimated image after inverse filtering.
   - Once you obtain the estimated function $\hat{F}$ in the frequency domain, set values in $\hat{F}$ at coordinates (u, v) to 0, if the magnitude of H at the same coordinates (u, v) is less than the threshold value t. Then, map $\hat{F}$ to the estimated image $\hat{f}$ in the spatial domain.
   - You can use numpy functions for performing 2D Fourier transform related operations.

```
[ ]: # function that performs inversing filtering in the frequency domain
def inverse_filter(degraded_img, h, t):
  """
  degraded_img: degraded image (H,W)
  h: degradation function (kH, kW)
  t: threshold of H to determine corresponding coordinates in Fhat to set to␣
 ↪zero
```

```
    returns:
    fhat_img: estimated image after inverse filtering (H,W)
    """
    # your code here
    f_hat = []

    return f_hat
```

2) Call the function **inverse_filter** with the degraded image, the degradation function $h(x, y)$ described above and a threhold $t = 0.01$.

Note: Clamp the restored image such that all values less than zero are set to zero and all values greater than 255 are set to 255.

[ ]:
```
"""
Call the function inverse_filter with the image and the degradation function␣
 ↪h(x,y) desribed above.
"""
h =  1/89 *np.array([[  1, 1, 1, 0, 0, 0, 0, 0, 0],
                       [0, 4, 7, 4, 0, 0, 0, 0, 0],
                       [0, 0, 6, 7, 4, 0, 0, 0, 0],
                       [0, 0, 1, 5, 7, 0, 0, 0, 0],
                       [0, 0, 0, 3, 8, 3, 0, 0, 0],
                       [0, 0, 0, 0, 5, 2, 2, 0, 0],
                       [0, 0, 0, 0, 0, 3, 7, 0, 0],
                       [0, 0, 0, 0, 0, 0, 3, 3, 0],
                       [0, 0, 0, 0, 0, 0, 0, 1, 1]])
# your code here
```

3) Display the degraded image and the estimated image after inverse filtering. Set the value range to [0, 255] on all display calls.

[ ]:
```
"""
Display the degraded image and the estimated image after inverse filtering.
"""
# your code here
```

4) Discuss how well the inverse filtering restored the original image. How well would inverse filtering work if noise was also applied to the image?

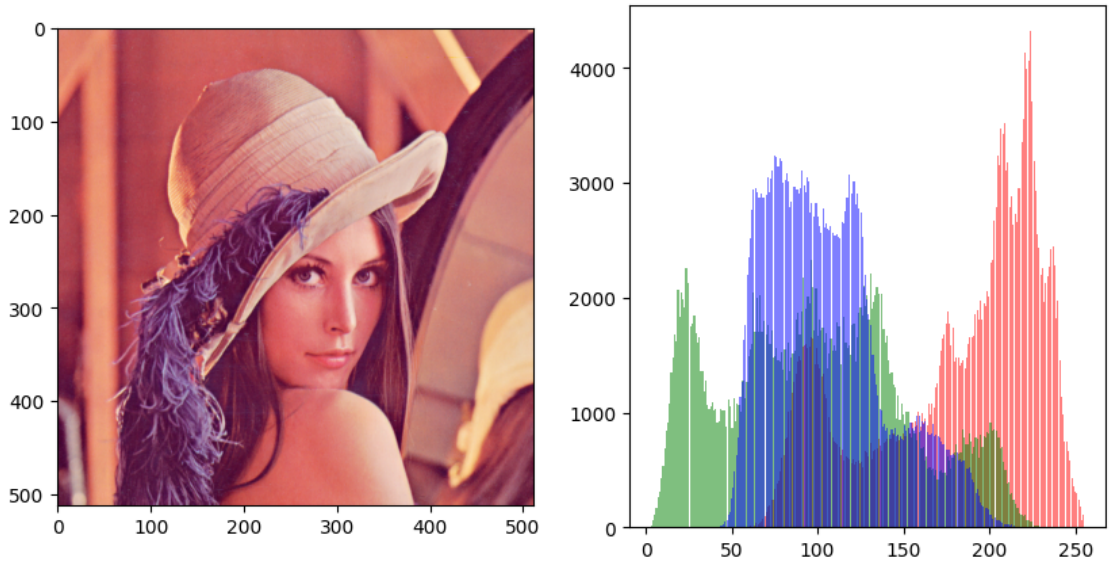
your answer here:

3.3 Part 3: Color Histogram Equalization (10 points)

[ ]:
```
# Read and display image along with its histogram
img = (255 * plt.imread('Lenna.png')).astype(np.uint8)
fig, ax = plt.subplots(1,2, figsize=(10,5))

ax[0].imshow(img, vmin=0, vmax=255)
```

```
ax[1].hist(img[:,:,0].flatten(), color='r', alpha=0.5, bins=256)
ax[1].hist(img[:,:,1].flatten(), color='g', alpha=0.5, bins=256)
ax[1].hist(img[:,:,2].flatten(), color='b', alpha=0.5, bins=256)
plt.show()
```



1) Complete the function **histeq_1** that histogram equalizes each color channel **independently** (i.e., independent of the other channels). The function takes an RGB image as the input and returns the histogram equalized image.

Note: You may use the skimage.exposure.equalize_hist function for histogram equalization. The output from this function will be in the range of [0,1]. Scale the output such that for each of the 3 channels, the maximum intensity value is 255. Convert the output image to uint8.

[ ]:
```
# function that histogram equalizes each color channel independently
def histeq_1(img):
  """
  img: input image in RGB format (H,W,3)

  returns:
  out: output image after histogram equalization (H,W,3)
  """
  # your code here
  out = []

  return out
```

2) Complete the function **rgb2hsi** that converts an RGB image to an HSI image. The function should take an RGB image as the input and output the corresponding HSI image.

```
[ ]: # function that conver rgb image to hsi image
def rgb2hsi(rgb):
  """
  rgb: image in RGB format (H,W,3)

  returns:
  hsi: image in HSI format (H,W,3)
  """
  # your code here
  hsi = []

  return hsi
```

3) Complete the function **hsi2rgb** that converts an HSI image to an RGB image. The function should take an HSI image as the input and output the corresponding RGB image.

```
[ ]: # function that converts hsi image to rgb image
def hsi2rgb(hsi):
  """
  hsi: image in HSI format (H,W,3)

  returns:
  rgb: image in RGB format (H,W,3)
  """
  # your code here
  rgb = []

  return rgb
```

4) Complete the function **histeq_2** that

- Takes an RGB image as input
- Calls your function **rgb2hsi** to convert the image from RGB to HSI
- Uses the function skimage.exposure.equalize_hist to histogram equalize the **I channel only**
- Calls your function **hsi2rgb** to convert the image from HSI to RGB
- Returns the output image

Notes:

- You may use the skimage.exposure.equalize_hist function for histogram equalization. The output from this function will be in the range of [0,1]. Scale the output such that the maximum intensity is 255.
- After converting the image from HSI to RGB, the output image may contain values larger than 255. Scale the output image such that the maximum value in the $512 \times 512 \times 3$ array is 255. Convert the output image to uint8.

```
[ ]: # see above for full function description
def histeq_2(img):
  """
```

```
    img: image in RGB format (H,W,3)

    returns:
    out: output image in RGB format after histogram equalization (H,W,3)
    """
    # your code here
    out = []

    return out
```

5) Call the function **histeq_1** and **histeq_2** with the Lenna image. Display the original image, output image of histeq_1 and output image of histeq_2. Set the value range to [0, 255] on all display calls.

[ ]:
```
"""
Call the function histeq_1 and histeq_2 with the Lenna image
Display the original image, output image of histeq_1 and output image of␣
 ↪histeq_2.
"""
# your code here
```

6) Display the **histograms** for the original image, output image of histeq_1 and output image of histeq_2. In each histogram, the color of each channel should be drawn with its corresponding color and transparency value of 0.5 (Check the histogram of the input image for reference).

[ ]:
```
"""
Display the **histograms** for the original image, output image of histeq_1 and␣
 ↪output image of histeq_2.
In each histogram, the color of each channel should be drawn with its␣
 ↪corresponding color and transparency value of 0.5
(Check the histogram for the input image for reference)
"""
# your code here
```

7) Briefly discuss the qualitative differences between the output images. Briefly discuss the quantitative differences, namely the range of intensities in each channel, between the output images.

Your answer here: