# CSE166_WI23_assignment_4

February 6, 2023

## 1 CSE 166: Image Processing, Winter 2023 – Assignment 4

- Instructor: Ben Ochoa
- Due: Wednesday, February 15, 11:59 PM

### 1.1 Instructions

Please answer the questions below using Python in the attached Jupyter notebook and follow the guidelines below:

- This assignment must be completed **individually**. For more details, please review the Academic Integrity Policy and Collaboration Policy on Canvas.
- All the solutions must be written in the Jupyter notebook only.
- After finishing the assignment in the notebook, please export the notebook as a PDF and **submit both the Notebook and the PDF** (i.e. the `.ipynb` and the `.pdf` files) on Gradescope. While submitting the PDF on gradescope -
  - Make sure that the outputs generated are clearly visible and are not cut-off or partially cropped in the final PDF.
  - Make sure to assign the relevant pages in your PDF submission for each problem.
  - Do not export the jupyter notebook as a single page. Please see the detailed submission instructions at the end of this file.
- You may use basic algebra packages (e.g. `NumPy`, `SciPy`, etc) but you are not allowed to use the packages that directly solve the problems. Feel free to ask the instructor and the teaching assistants if you are unsure about the packages to use.
- It is highly recommended that you begin working on this assignment early.

**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)

**Note:** As stated in the syllabus, you must use the 4th edition of the textbook, ISBN 9780133356724. Answers to different problems will not be accepted.

- The textbook problems are conceptual and/or math problems, not programming problems. You are provided with a Markdown cell to answer each problem. Do not change this cell to a Code cell or create a new Code cell. Answers in the form of code will not be accepted.

### 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**

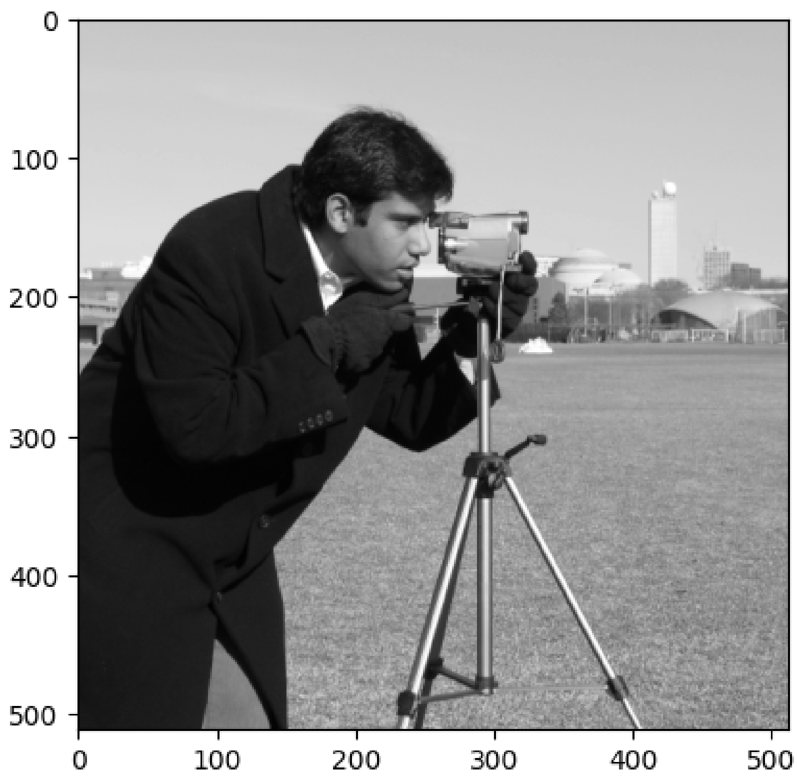
## 3 Problem 2: Programming: Image Restoration and Color Image Processing (30 points)

### 3.1 Part 1: Spatial Image Restoration (10 points)

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     from skimage import data
     import skimage
     import cv2
     import math
```

```
[ ]: # Read and display image
     img = data.camera()

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

```
[ ]: <matplotlib.image.AxesImage at 0x299ff7761f0>
```



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 $\sigma_\eta^2$, 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 $\sigma_\eta^2 \le \sigma_{S_{xy}}^2$, 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 to it (H,W)
         var: floating point value representing overall noise variance
         window_size: a tuple for filter window size (n,n)

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

```
    return fhat_img
```

2) Apply Gaussian noise with variance 0.1 for the camera image using the function `skimage.util.random_noise`. Then, call the function `adaptive_local_noise_reduction` with the noisy image and a window size of $(7, 7)$.

**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.

```
[ ]: # your code here
```

3) Display the input image, noisy image, and restored image.

```
[ ]: # your code here
```

4) Briefly discuss how well the restored image approximates the original image and whether or not a local noise reduction filter is suitable for applying to 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 technique?

**Your answer here**

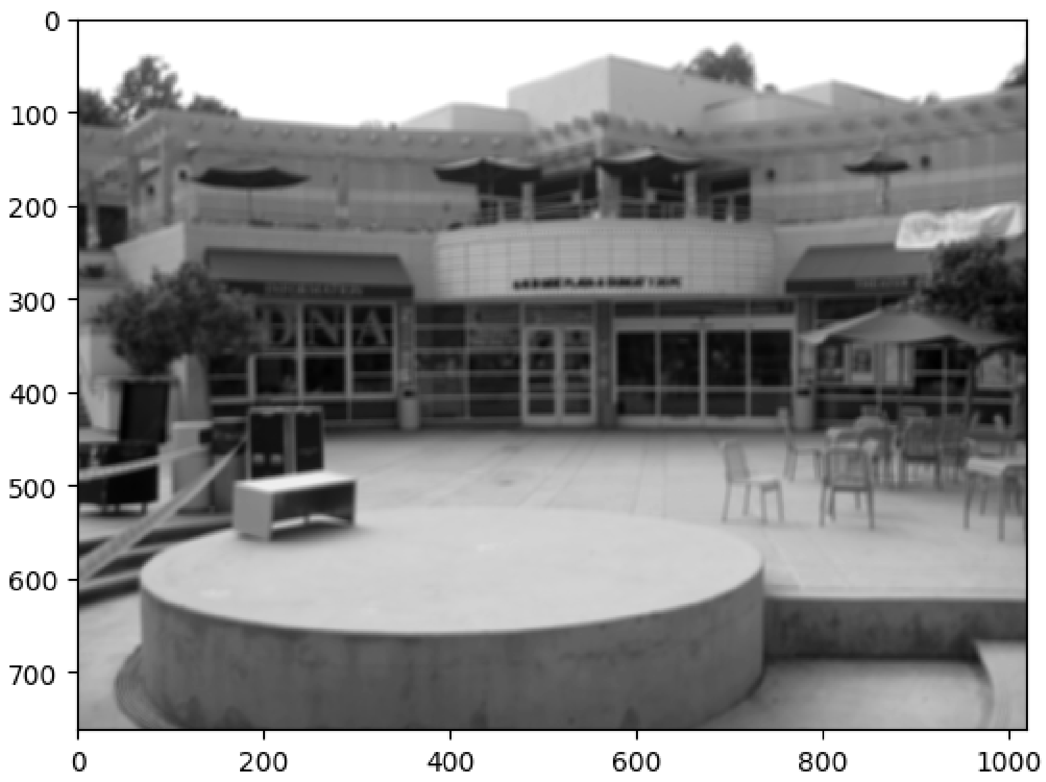
## 3.2 Part 2: Inverse Filtering (10 points)

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

$$h(x,y) = \frac{1}{239}\begin{bmatrix} 3 & 7 & 7 & 7 & 8 & 9 & 5 \\ 4 & 6 & 2 & 5 & 8 & 8 & 3 \\ 8 & 5 & 4 & 4 & 6 & 5 & 8 \\ 1 & 5 & 6 & 5 & 4 & 6 & 2 \\ 1 & 3 & 8 & 3 & 8 & 6 & 3 \\ 2 & 7 & 1 & 5 & 5 & 2 & 2 \\ 6 & 2 & 9 & 5 & 4 & 3 & 3 \end{bmatrix}$$

```
[ ]: # Read and display the image
     degraded_img = plt.imread('q2_input.png')[:,:,0]

     plt.imshow(degraded_img, cmap='gray', vmin=0, vmax=1)
```

```
[ ]: <matplotlib.image.AxesImage at 0x299ff7adc70>
```



1) Complete the function `inverse_filter` that performs inverse filtering in the frequency domain. The function takes as input the degraded image $g(x, y)$, degradation function $h(x, y)$ and a threhold value $t$, and returns the estimated image $\hat{f}(x, y)$ 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 `scipy.signal.convolve2d` for performing convolution and 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 which corresponding coordinates in Fhat to set␣
   ↪to zero

    returns:
    fhat_img: estimated image after inverse filtering (H,W)
    """
    # your code here
    return fhat_img
```

2) Call the function `inverse_filter` with the `q2_input.png` image (provided to you in the assignment folder) as the degraded image $g(x, y)$, the degradation function $h(x, y)$ described above and a threhold value of $t = 0.01$.

**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.

```
[ ]: # your code here
```

3) Display the degraded image and the estimated image after inverse filtering.

```
[ ]: # your code here
```

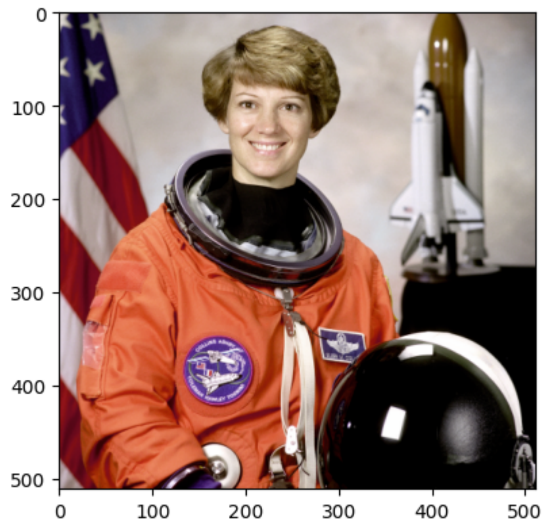
4) Discuss how well the inverse filtering method restored the original image. How well would inverse filtering work if noise was also applied to the image?

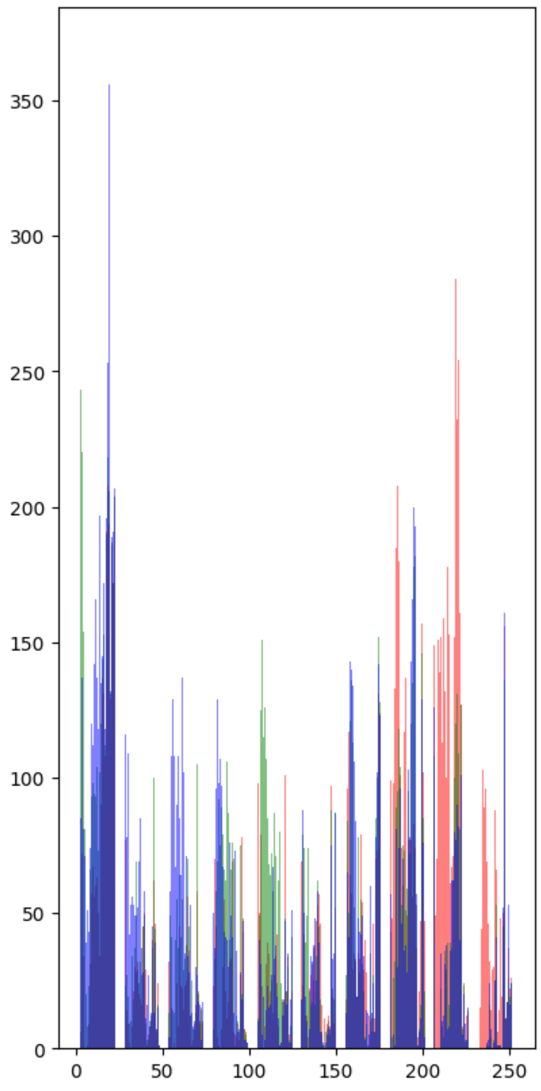
**your answer here**

### 3.3 Part 3: Color Histogram Equalization (10 points)

```
[ ]: # Read and display image along with its histogram
     img = data.astronaut()

     fig, ax = plt.subplots(1, 2, figsize=(10,10))

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

     ax[1].hist(img[:,:,0], color=['r']*512, alpha=0.5)
     ax[1].hist(img[:,:,1], color=['g']*512, alpha=0.5)
     ax[1].hist(img[:,:,2], color=['b']*512, alpha=0.5)
```

```
[ ]: (array([[127.,  37.,   3., …,  39., 144.,   0.],
             [108.,  55.,  15., …,  35., 147.,   0.],
             [ 87.,  78.,  27., …,  38., 142.,   0.],
             …,
             [217.,   1.,   2., …,  11.,   3.,   1.],
             [219.,   2.,   2., …,  11.,   4.,   0.],
             [222.,   3.,   1., …,   6.,   1.,   0.]]),
      array([  0. ,  25.5,  51. ,  76.5, 102. , 127.5, 153. , 178.5, 204. ,
             229.5, 255. ]),
      <a list of 512 BarContainer objects>)
```





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

**Note**: You may use the `skimage.exposure.equalize_hist` function for histogram equalization.

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

         returns:
         out_img: output image after histogram equalization (H,W,3)
         """
```

```
    # your code here

    return out_img
```

2) Complete the function `rgb2hsi` that converts an RGB image to an HSI image. The function input is an RGB image and the output is the corresponding HSI image.

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

         returns:
         hsi_img: image in HSI format (H,W,3)
         """
         # your code here

         return hsi_img
```

3) Complete the function `hsi2rgb` that converts an HSI image to an RGB image. The function input is an HSI image and the output is the corresponding RGB image.

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

         returns:
         rgb_img: image in RGB format (H,W,3)
         """
         # your code here

         return rgb_img * 255
```

4) Complete the function `histeq_2` that

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

```
[ ]: # see above for full function description
     def histeq_2(img):
         """
         img: image in RGB format (H,W,3)

         returns:
```

```
    out_img: output image in RGB format after histogram equalization (H,W,3)
    """
    # your code here

    return out_img
```

5) Call the function `histeq_1` and `histeq_2` with the astronaut image. Display the original image, output image of `histeq_1` function and output image of `histeq_2` function.

```
[ ]: # 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 must be drawn with its corresponding color and transparency value of 0.5 (Check the histogram of the input astronaut image for reference).

```
[ ]: # your code here
```

7) Briefly discuss the qualitative differences between the output images of `histeq_1` and `histeq_2` functions. Briefly discuss the quantitative differences, namely the range of intensities in each channel, between the output images.

**Your answer here**

## 4 Submission Instructions

1. Remember to submit **both** the Jupyter notebook file (`.ipynb`) and the PDF version (`.pdf`) of this notebook to Gradescope.
2. 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.
3. While submitting on gradescope, please make sure to assign the relevant pages in your PDF submission for each problem.
4. Do not export the jupyter notebook as a single page.

To convert the notebook to PDF, you can choose one way below:

1. You can print the web page and save as PDF (e.g. Chrome: Right click the web page → Print... → Choose "Destination: Save as PDF" and click "Save").
2. You can find the export option in the header: File → Download as → "PDF via LaTeX"
3. You can use nbconvert (https://nbconvert.readthedocs.io/en/latest/install.html) to convert the ipynb file to pdf using the following command

```
jupyter nbconvert --allow-chromium-download --to webpdf <ipynb filename>
```