CSE166_WI23_assignment_2

January 23, 2023

1 CSE 166: Image Processing, Winter 2023 – Assignment 2

- Instructor: Ben Ochoa
- Due: Monday, January 30, 2023, 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 (10 points)

2.1 a) Problem 3.5 (2 points)

Your answer here

2.2 b) Problem 3.7 (2 points)

Your answer here

2.3 c) Problem 3.11a (1 point)

Your answer here

2.4 d) Problem 3.12a (1 point)

Your answer here

2.5 e) Problem 3.24 (1 point)

Your answer here

2.6 f) Problem 3.29 (2 points)

Your answer here

2.7 g) Problem 3.44 (1 point)

Your answer here

3 Problem 2: Programming: Intensity Transformations and Spatial Filtering (30 points)

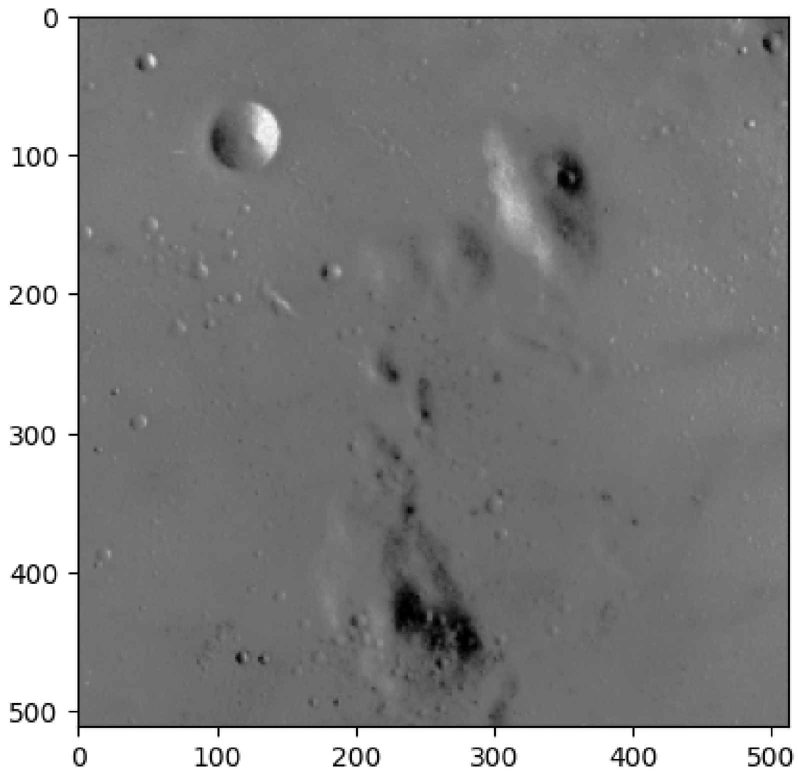
3.1 Part 1: Piecewise Linear Transformation (5 points)

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

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

plt.imshow(img, cmap='gray')
```

```
[ ]: <matplotlib.image.AxesImage at 0x14c4f025670>
```



1. Complete the function `piecewise_linear_transform` that computes an intensity transformed image of a given image. The function parameters are comprised of the input image to be transformed, a set of input intensity values, and a corresponding set of output intensity values. In both sets, the first intensity value is 0 and the last intensity value is 255. The resulting intensity values in the output image are calculated from a piecewise linear transformation determined from the parameters.

```
[ ]: # function that transforms intensities of the image using piece-wise linear␣
      ↪transform
     def piecewise_linear_transform(img, ip_intensities, op_intensities):
         """
         img: input image (H,W)
         ip_intensities: input intensities (N,)
         op_intensities: output intensities (N,)

         returns:
         tr_img: resulting image after piecewise linear transform (H,W)
         """
         # your code here

         return tr_img
```

Call the function `piecewise_linear_transform` for the moon image with input intensities

$\{0, 100, 130, 255\}$ and output intensities $\{0, 70, 200, 255\}$ to calculate the intensity transformed image. Display the input image and output image side by side.

```
[ ]: """
     Call piecewise_linear_transform function for the moon image with the given␣
      ↪input
     and output intensities.
     Display the input and output images.
     """
     ip_intensities = np.array([0, 100, 130, 255])
     op_intensities = np.array([0, 70, 200, 255])
     tr_img = piecewise_linear_transform(img, ip_intensities, op_intensities)

     fig, ax = plt.subplots(1, 2, figsize=(10,10))
     ax[0].imshow(img, cmap='gray')
     ax[1].imshow(tr_img, cmap='gray')
```

2. Briefly discuss the qualitative differences between the input and output images.

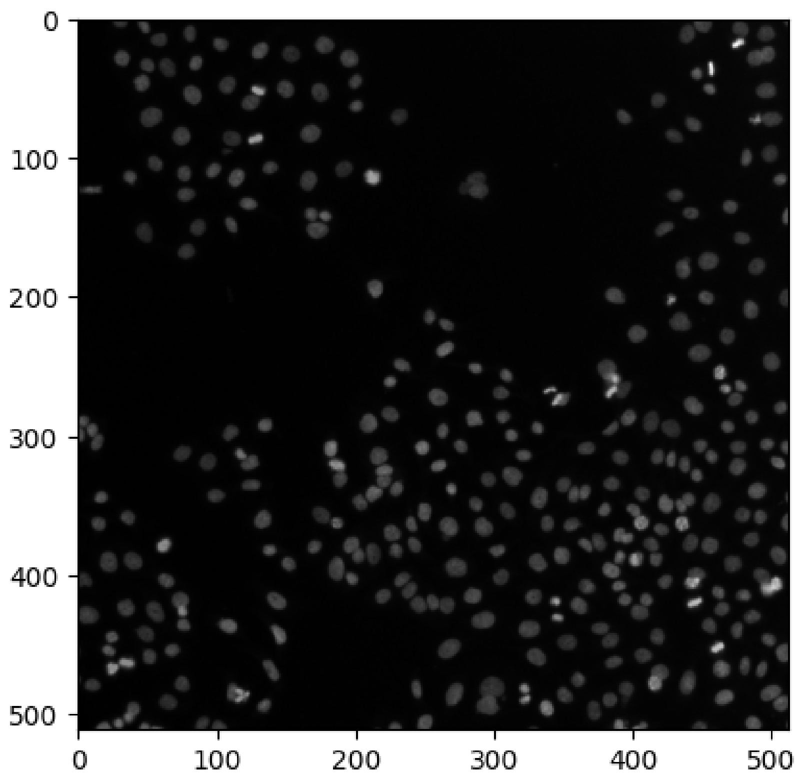
Your answer here

3.2 Part 2: Histogram Equalization (5 points)

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

     plt.imshow(img, cmap='gray')
```

```
[ ]: <matplotlib.image.AxesImage at 0x14c501eb550>
```



1. Complete the function `histogram_equalization` that calculates a histogram equalized image corresponding to a given input image. The function input is a grayscale image and the function output is the histogram equalized image corresponding to the input image.

[ ]:
```
# function that transforms the input image into histogram-equalized image
def histogram_equalization(img):
    """
    img: input image

    returns:
    heq_img: histogram-equalized image
    """
    # your code here

    return heq_img
```

Call the function `histogram_equalization` for the human-mitosis image and display the resulting output.

[ ]:
```
"""
Call histogram_equalization for the human-mitosis image.
Display the input and output images.
"""
```

```
heq_img = histogram_equalization(img)

fig, ax = plt.subplots(1,2, figsize=(10,10))
ax[0].imshow(img, cmap='gray')
ax[1].imshow(heq_img, cmap='gray')
```

2. Plot the histogram of the intensities of the input image and the histogram of the intensitives of the histogram-equalized image side-by-side.

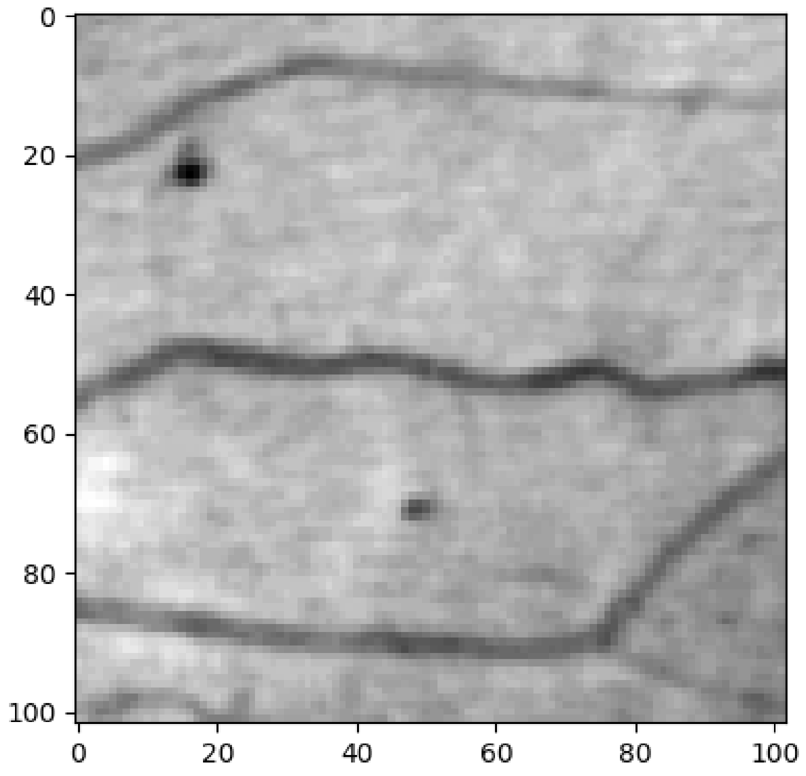
```
[ ]: """
plot the histogram of the intensities of the input image and the␣
 ↪histogram-equalized
image side-by-side
"""
# your code here
```

3.3 Part 3: Sharpening using the second derivative (10 points)

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

plt.imshow(img, cmap='gray')
```

```
[ ]: <matplotlib.image.AxesImage at 0x14c50224220>
```



1. Complete the function `sharpen_image` that computes a sharpened image corresponding to a given input image. The function input is a grayscale image and the function output is the sharpened image corresponding to the input image. The function must use the Laplacian filter (without diagonal directions). Note that $g(x, y) = f(x, y) - \nabla^2 f(x, y)$, where $f(x, y)$ is the input image and $g(x, y)$ is the output sharpened image. Filtering must be implemented **without** using Python in-built library functions like *scipy.ndimage.convolve*.

Important Note:
* While performing sharpening, the output pixels can have values outside the range [0,255]. You can handle this by working with `np.int32` data type matrices. * Clamp the output sharpened image such that all intensities less than zero is set to 0 and all intensities greater than 255 is set to 255.

[ ]:
```
# function that sharpens the given image
def sharpen_image(img):
    """
    img: input image (H,W)

    returns:
    sh_img: image after sharpening (H,W)
    """
    # your code here

    return sh_img
```

Call the function `sharpen_image` for the microaneurysms image. Display the input image and output image side by side.

[ ]:
```
"""
Call the sharpen_image function with the microaneurysms image.
Display the input and output images.
"""
sh_img = sharpen_image(img)

fig, ax = plt.subplots(1,2, figsize=(10,10))
ax[0].imshow(img, cmap='gray')
ax[1].imshow(sh_img, cmap='gray')
```

2. Briefly discuss your results.

Your answer here

3. **Optional**: Sharpen the image using the Laplacian filter with diagonal directions and briefly discuss the qualitative differences between these results and those obtained using the filter without diagonal directions.

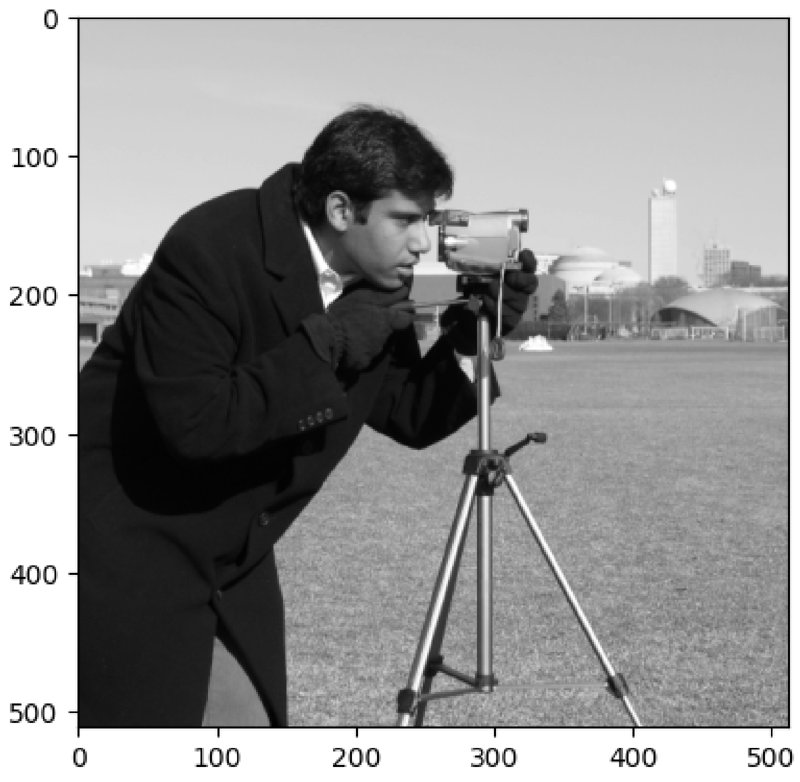
Your answer here

3.4 Part 4: Magnitude of gradient (10 points)

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

     plt.imshow(img, cmap='gray')
```

```
[ ]: <matplotlib.image.AxesImage at 0x14c50286c40>
```



1. Complete the function `gradient_magnitude` that calculates the magnitude of gradient vector image corresponding to a given input image. The function input is a grayscale image and the function output is the gradient magnitude image corresponding to the input image. The function must use the Sobel approximation to the derivative. Filtering must be implemented without using Python in-built functions like *scipy.ndimage.convolve*.

Important Note: * While computing magnitude of gradient, the output pixels can have values outside the [0,255] range and/or floating-point numbers. You can handle this by working with `np.float32` data type matrices. * Scale (only) the output image such that zero is black and the maximum gradient magnitude value in the image is white.

```
[ ]: # function that computes the magnitude of gradient for the given image
     def gradient_magnitude(img):
         """
```

```
    img: input image (H,W)

    returns:
    mg_img: image after applying magnitude of gradient (H,W)
    """
    # your code here

    return mg_img
```

Call the function `gradient_magnitude` with the camera image. Display the input image and output image side by side.

[ ]:
```
"""
Call the gradient_magnitude function with the camera image.
Display the input and output images.
"""
gm_img = gradient_magnitude(img)

fig, ax = plt.subplots(1,2, figsize=(10,10))
ax[0].imshow(img, cmap='gray')
ax[1].imshow(gm_img, cmap='gray')
```

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>
```