CSE166_WI23_assignment_6

March 1, 2023

1 CSE 166: Image Processing, Winter 2023 – Assignment 6

- Instructor: Ben Ochoa
- Due: Wednesday, March 8, 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 (15 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 8.5(a) (1 point)

Your answer here

2.2 b) Problem 8.9(a) (1 point)

Your answer here

2.3 c)Problem 9.8 (3 points)

Your answer here

2.4 d) Problem 9.9 (3 points)

Your answer here

2.5 e) Problem 9.10 (2 points)

Your answer here

2.6 f) Problem 9.11 (2 points)

Your answer here

2.7 g) Problem 9.23 (3 points)

Your answer here

3 Problem 2: Programming (40 points)

3.0.1 PyWavelets package installation

You would need to install PyWavelets package for this assignment.

If you are using pip then, `pip install PyWavelets`

If you are using conda then, `conda install pywavelets`

Check out the following link for more details: https://pywavelets.readthedocs.io/en/latest/install.html

3.1 Part 1: The Discrete Cosine Transform and Lossy Block Processing (15 points)

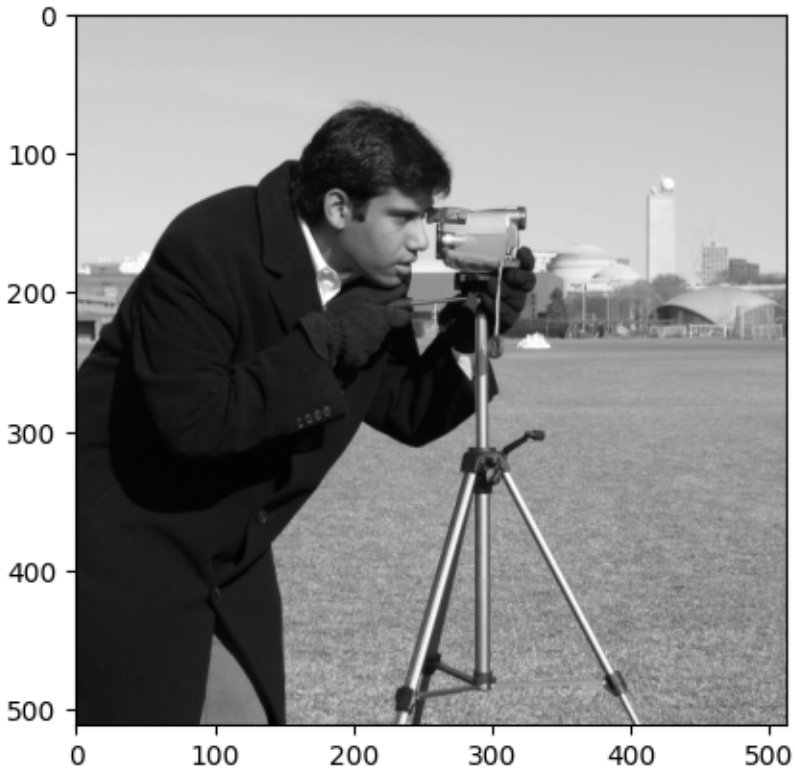
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
import skimage
import math
```

```
import pywt
import cv2
```

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

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

[ ]:
```
<matplotlib.image.AxesImage at 0x1a47d23d760>
```



1) Complete the function `get_basis_images` that computes the basis images for the Discrete Cosine Transform. The function takes a tuple (M, M) as input representing the size of basis image and returns M^2 basis images, each of size (M, M).

[ ]:
```
# function to compute the basis images for the Discrete Cosine Transform
def get_basis_images(size):
  """
  size: size of the basis image (M,M)

  returns:
  basis_images: basis images of DCT (M,M,M,M)
  """
```

```
  # your code here

  return basis_images
```

2) Call the function `get_basis_images` with size $(8, 8)$. Display all the 64 basis images using the `matplotlib.subplots` function.

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

3) Complete the function `lossy_dct_transform` that takes an input image, a set of basis images and an integer n as inputs, and independently processes non-overlapping blocks (i.e., subimages) of size 8×8 as follows. For each block, compute the discrete cosine transform (DCT), retain the n greatest magnitude DCT coefficients (i.e., set the $64 - n$ remaining DCT coefficients to zero), and compute the inverse DCT. The 2D DCT and inverse DCT must be computed using the basis images given to the function as input.

[ ]:
```
# see description above for complete function description
def lossy_dct_transform(img, basis_images, n):
  """
  img: input image (N,N)
  basis_images : set of basis images (M,M,M,M)

  returns:
  out: output image after processing (N,N)
  """
  # your code here

  return out
```

4) Call the function `lossy_dct_transform` with the cameraman image, the set of basis images computed earlier and for each $n = 64, 32, 16, 8, 4, 2$. Display the input image and the resulting output image for each value of n. (Note: you should get a total of 6 processed images)

- Hint: Normalize your output images by setting all values less than 0 as 0 and all values greater than 255 as 255

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

5) Display the error image and the root-mean-square error associated with each output image.

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

6) Briefly discuss the resulting images, including any differences between them.

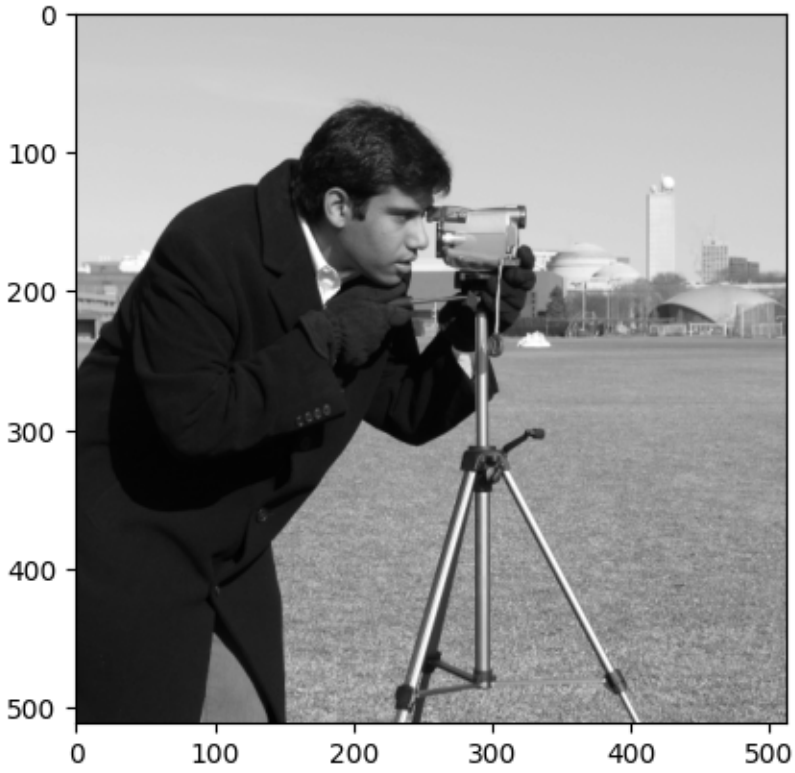
Your answer here

3.2 Part 2: The Discrete Wavelet Transform and Lossy Processing (15 points)

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

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

[ ]:
```
<matplotlib.image.AxesImage at 0x1a47d286040>
```



1) Complete the function `lossy_wavelet_transform` that computes the 2-level Discrete Wavelet Transform (DWT), sets p percent of the smallest magnitude detail coefficients to zero, and computes the inverse DWT. The function takes an image, the wavelet type and a floating-point number p as input, and returns the output image after processing.

[ ]:
```
# see description above for full function description
def lossy_wavelet_transform(img, wavelet, p):
  """
  img: input image (N,N)
  wavelet: wavelet algorithm to use ('haar' or 'db4' or 'bior4.4')
  p: percent of smallest magnitude detail coefficients that need to be set to␣
 ↪zero (floating point between 0 and 1)
```

```
    returns:
    out: output image after processing

    """
    # your code here

    return out
```

2) Call the function `lossy_wavelet_transform` with the cameraman image, for each wavelet type {Haar, Daubechies, and biorthogonal} and for each $p = 25\%, 50\%, 90\%, 95\%$. Display the output images.

- Hint: You can specify the wavelet types as 'haar', 'db4', and 'bior4.4' to the `pywt` function for the Haar, Daubechies, and biorthogonal wavelets, respectively.
- Hint: Normalize your output images from `lossy_wavelet_transform` function by setting all values less than 0 as 0 and all values greater than 255 as 255
- Note: you should get a total of 12 output images

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

3) Display the error images (12 images) and the root-mean-square error associated with each output image.

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

4) Briefly discuss the resulting images, including how the choice of wavelet and p affects compression quality. Also, discuss any differences you observe with compression using DCT.

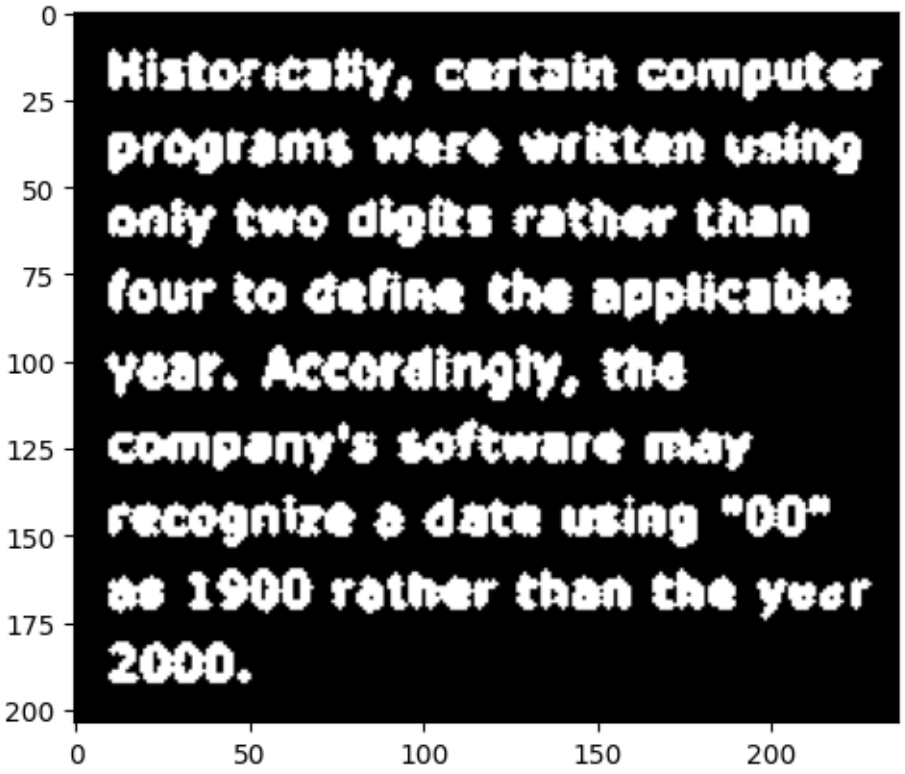
Your answer here

3.3 Part 3: Improving Text Recognition Using Morphological Image Processing (10 points)

[ ]:
```
# Read and display image
img = plt.imread('text.tif')

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

[ ]:
```
<matplotlib.image.AxesImage at 0x1a47d2f8820>
```



1) Complete the function `morphological_processing` that performs morphological image processing on a given image to remove noise in the text.

- Your results should visibly improve the quality of the letters.
- You may use the erosion, dilation, opening and closing functions provided with OpenCV library. Documentation: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_op

```
[ ]: # function that performs morphological image processing on the image to remove␣
      ↪noise in the text.
     def morphological_processing(img):
       """
       img: input image (N,N)

       returns:
       out: output image after morphological processing (N,N)
       """
       # your code here

       return out
```

2) Call the function `morphological_processing` with the text.tif image (provided to you in the assignment folder). Display the original and resulting images.

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

3) Briefly describe your methodology.

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>
```