CSE 166: Image Processing, Spring 2022 – Assignment 6

Instructor: Ben Ochoa

- Due On: **Monday, May 23, 2022, 11:59 PM**.

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 follow the Academic Integrity Policy and Collaboration Policy on Canvas.
- This assignment contains both math and programming problems.
- All the solutions must be written in this Jupyter notebook.
- 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. Please assign the pages of your PDF submission to the corresponding problems.
- 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.

Problem 1: Textbook problems (15 points)

a) Problem 8.5(a) (1 point)

your answer here

b) Problem 8.9(a) (1 point)

your answer here

c)Problem 9.8 (3 points)

your answer here

d) Problem 9.9 (3 points)

your answer here

e) Problem 9.10 (2 points)

your answer here

f) Problem 9.11 (2 points)

your answer here

g) Problem 9.23 (3 points)

your answer here

Problem 2: Programming (40 points)

Part 1: The discrete cosine transform and lossy block processing (15 points)

- Compelte the function **get_basis_images** that computes the basis images for the Discrete Cosine Transform. The function takes a (M,M) size as input and returns M^2 basis images, each of size M^2 .
- 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 that the function takes an input.
- Call the function **get_basis_images** with size (8,8). Display all the 64 basis images using the matplotlib.subplots function.
- Call the function **lossy_dct_transform** with the cameraman image, the set of basis images you 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 images)

    - Hint: Normalize your output images by setting all values less than 0 as 0 and all values greater than 255 as 255
- Additionally, display the error images and the root-mean-square error associated with each output image.
- Briefly discuss the resulting images, including any differences between them.

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`

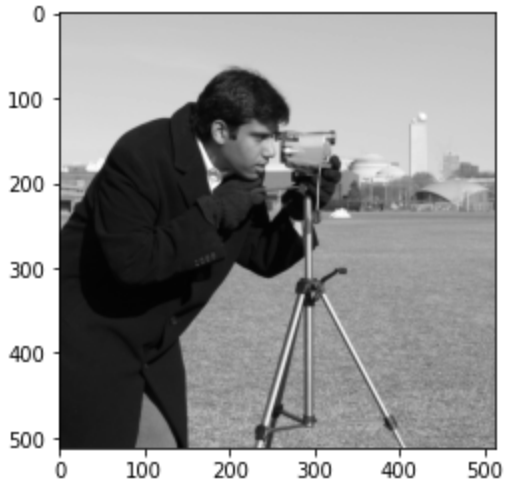
Check out the following link for more details: https://pywavelets.readthedocs.io/en/latest/install.html

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

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

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f1c16fc5e90>
```



In [ ]:
```
# 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
```

In [ ]:
```
"""
Call the function get_basis_images with size (8,8).
Display all the 64 basis images using the matplotlib.subplots function.
"""
# your code here
```

Out[ ]:
```
'\nCall the function get_basis_images with size (8,8). \nDisplay all the 64 basis images using the matplotlib.subplots function.\n'
```

In [ ]:
```
# see description above for complete function description
def lossy_dct_trasnform(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
```

In [ ]:
```
"""
Call the function lossy_dct_transform with the cameraman image, the set of basis
images you computed earlier and for each n = {32, 16, 8, 4, 2}.

Display the input image and the resulting output image for each value of n.
"""
# your code here
```

Out[ ]:
```
'\nCall the function lossy_dct_transform with the cameraman image, the set of basis \nimages you computed earlier and for each n = {32, 16, 8, 4, 2}. \n\nDisplay the input image and the resulting output image for each value of n.\n'
```

In [ ]:
```
"""
Display the error images and the root-mean-square error associated with each output imag
"""
# your code here
```

Out[ ]:
```
'\nDisplay the error images and the root-mean-square error associated with each output image. \n'
```

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

your answer here

Part 2: The discrete wavelet transform and lossy processing (15 points)

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

- 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 functions 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

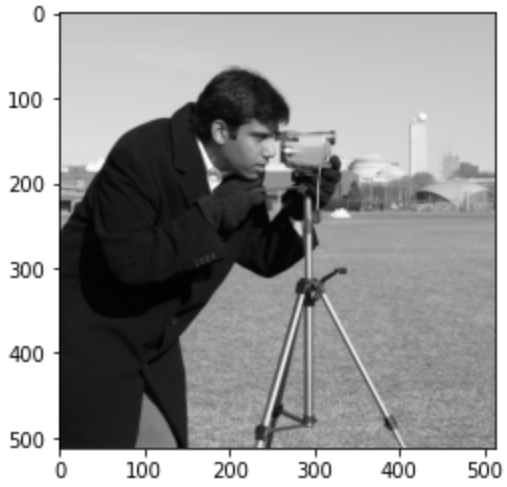
- Additionally, display the error images and the root-mean-square error associated with each output image.

- Briefly discuss the resulting images, including how the choice of wavelet and p effects compression quality. Also, discuss any differences you observe with compression based on the DCT.

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

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f1c14062cd0>
```



In [ ]:
```
# 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 (floa

  returns:
  out: output image after processing

  """
  # your code here

  return out
```

In [ ]:
```
"""
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 (12 images).
"""
# your code here
```

Out[ ]:
```
'\nCall the function lossy_wavelet_transform with the cameraman image, for each \nwavele
t type {Haar, Daubechies, and biorthogonal} and for each p = {25%, 50%, 90%, 95%}\n\nDis
play the output images (12 images).\n'
```

In [ ]:
```
"""
Display the error images along with its rms error (12 images)
"""
# your code here
```

Out[ ]:
```
'\nDisplay the error images along with its rms error (12 images)\n'
```

Briefly discuss the resulting images, including how the choice of wavelet and p effects compression quality. Also, discuss any differences you observe with compression based on the DCT.

your answer here

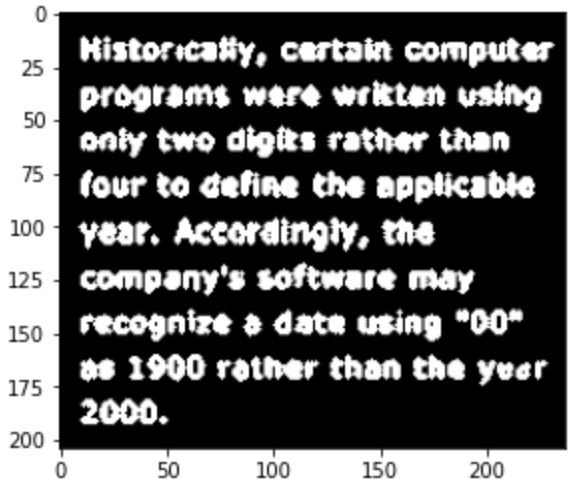
Part 3: Improving Text Recognition Using Morphological Image Processing (10 points)

- Complete the function **morphological_processing** that performs morphological image processing on the 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. Documentation: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphologica
- Call the function **morphological_processing** with the text.tif image (provided to you in the assignment folder). Display the original and resulting images.
- Briefly describe your methodology.

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

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f50105be310>
```



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

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

  return out
```

In [ ]:
```
"""
Call the function morphological_processing with the text.tif image (provided to you in t

Display the original and resulting images.
"""
# your code here
```

Out[ ]:
```
'\nCall the function morphological_processing with the text.tif image (provided to you i
n the assignment folder). \n\nDisplay the original and resulting images. \n'
```

Briefly describe your methodology.

your answer here