CSE 166: Image Processing, Spring 2022 – Assignment 5

Instructor: Ben Ochoa

- Due On: **Monday, May 16, 2022, 11:59 PM (Pacific Time)**.

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 (11 points)

a) Problem 6.1 (5 points)

your answer here

b) Problem 6.3 (3 points)

your answer here

c) Problem 6.17 (1 point)

your answer here

d) Problem 6.30 (2 points)

your answer here

Problem 2: Programming: The wavelet transform and wavelet-based image processing (30 points)

Part 1: The wavelet transform (10 points)

- Create a 256×256 array of where every pixel value is 1. Then, use the function pywt.wavedec2 to obtain the approximation coefficients for levels 1, 2, 3, 4, and 5 for a 5-scale Haar wavelet decomposition of the image. Determine the equation that scales pixel values in the original image (i.e., the image at level $n = 0$) to pixel values in the image at level n. Ensure this equation holds regardless of the pixel values in the original image.

- Complete the function **dwt** that performs the following.

    - Re-scales the image to $[0, 1]$. Then, uses the function pywt.wavedec2 to compute a 3-scale Haar discrete wavelet decomposition of the image.
    - Calculates the approximate and detail coefficients to create a figure similar to the one shown on slide 30 of lecture 12 (hint: scale each detail coefficients ''image'' by 1 over the quantity two times the maximum absolute value of the coefficients, then add $1/2$ to the result). Ensure the approximation coefficients are scaled correctly using the scale determined in the previous subproblem.
    - Additionally, reconstruct the approximation coefficients for levels 2 and 1, and then scale the approximation coefficients correctly.
- Call the function **dwt** with the cameraman image. Display the figure similar to the one shown on slide 30 of lecture 12, and also output the approximation coefficients images.

Note: You may use numpy functions for performing mathematical operations

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
```

In [ ]:
```
"""
Create a $256 \times 256$ array of where every pixel value is 1.
Then, use the function pywt.wavedec2 to obtain the approximation coefficients for
levels 1, 2, 3, 4, and 5 for a 5-scale Haar wavelet decomposition of the image.
"""
# your code here
```

Out[ ]:
```
'\nCreate a $256 \times 256$ array of where every pixel value is 1.  \nThen, use the fun
ction pywt.wavedec2 to obtain the approximation coefficients for \nlevels 1, 2, 3, 4, an
d 5 for a 5-scale Haar wavelet decomposition of the image.\n'
```

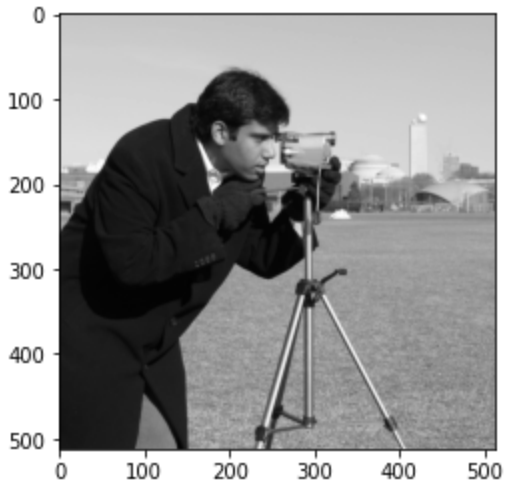
Determine the equation that scales pixel values in the original image (i.e., the image at level $n = 0$) to pixel values in the image at level n. Ensure this equation holds regardless of the pixel values in the original image.

your answer here

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

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fee9d9f9f90>
```



In [ ]:
```
# Read description above for full function description
def dwt(img):
  """
  img: input image (N,N)

  returns:
  res: figure similar to the one shown on slide 30 of lecture 12 (N,N)
  A1: level 1 approximate coefficient (N/2, N/2)
  A2: llevel 2 approximate coefficient (N/4, N/4)
  """
  # your code here


  return res, A1, A2
```

In [ ]:
```
"""
Call the function dwt with the cameraman image.

Display the figure similar to the one shown on slide 30 of lecture 12,
and also output the approximation coefficients images.
"""
# your code here
```

Out[ ]:
```
'\nCall the function dwt with the cameraman image. \n\nDisplay the figure similar to the one shown on slide 30 of lecture 12, \nand also output the approximation coefficients images.\n'
```

Part 2: Wavelet-based "edge" detection (10 points)

- Complete the function **edge_detection** that takes an image as input and computes a 1-, 2-, 3-, and 4-scale Symlets 4 wavelet decomposition of the input image. For each resulting decomposition, set the approximation coefficients to zero and perform wavelet reconstruction (back up to level 0), again using Symlets 4 wavelet filters.
  - For this problem (and the next one), it is easier to use the functions pywt.dwt2 and pywt.idwt2 than pywt.wavedec2 and pywt.waverec2
  - Ensure you set the DWT extension mode to 'periodization' (otherwise, you will encounter issues with different sized images)
  - Hint: You may have to rescale the final "edge" images to [0,255]
  - Hint: 'sym4' corresponds to the Symlets 4 wavelet decomposition in pywt.dwt2 and pywt.idwt2 functions
- Call the **edge_detection** function with the cameraman image. Display the the reconstructed ''edge'' images for all scales. Briefly discuss the resulting images, including any differences between them.

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

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

In [ ]:
```
# function that detects edges using Symlets 4 wavelet decomposition
def edge_detection(img):
  """
  img: input image (N,N)

  returns:
  E1: Reconstructed 'edge' image at scale 1 (N,N)
  E2: Reconstructed 'edge' image at scale 2 (N,N)
  E3: Reconstructed 'edge' image at scale 3 (N,N)
  E4: Reconstructed 'edge' image at scale 4 (N,N)
  """
  # your code here


  return E1, E2, E3, E4
```

In [ ]:
```
"""
Call the edge_detection function with the cameraman image.

Display the the reconstructed ''edge'' images for all scales.
"""
# your code here
```

Out[ ]: "\nCall the edge_detection function with the cameraman image. \n\nDisplay the the reconstructed ''edge'' images for all scales.\n"

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

your answer here

Part 3: Wavelet-based noise removal (10 points)

- Complete the function **noise_removal** that takes an image and a threshold as input, and returns the image after noise removal. The function
  - Computes a 2-scale Symlets 4 wavelet decomposition of the input image
  - Sets level 1 and level 2 detail coefficients with absolute value less than **threshold** to 0
  - Performs wavelet reconstruction (back up to level 0), again using Symlets 4 wavelet filters
  - Hint: It is easier to use the functions pywt.dwt2 and pywt.idwt2 than pywt.wavedec2 and pywt.waverec2
  - Hint: Set the DWT extension mode to 'periodization' (otherwise, you will encounter issues with different sized images)
  - Hint: 'sym4' corresponds to the Symlets 4 wavelet decomposition in pywt.dwt2 and pywt.idwt2 functions
- Call the **noise_removal** function with the cat image and $threshold = 40$. Display the input image and the image after removing noise. Briefly discuss the results.

In [ ]:
```
# Read and display image
img = skimage.color.rgb2gray(data.chelsea()[:256, 44:300])
img *= 255
img = img.astype(np.uint8)

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

In [ ]:
```
# function that removes noise using Symlets 4 wavelet decomposition
def noise_removal(img, threshold):
  """
  img: input image (N,N)
  threshold: threshold for setting detail coefficients with absolute value less than thi

  returns:
  out: image after noise removal (N,N)
  """
  # your code here

  return out
```

In [ ]:
```
"""
Call the noise_removal function with the cat image and threshold=40.

Display the input image and the image after removing noise.
"""
# your code here
```

Out[ ]: '\nCall the noise_removal function with the cat image and threshold=40. \n\nDisplay the input image and the image after removing noise.\n'

Briefly discuss the results.

your answer here