CSE166_WI23_assignment_5

February 22, 2023

1 CSE 166: Image Processing, Winter 2023 – Assignment 5

- Instructor: Ben Ochoa
- Due: Wednesday, March 1, 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 (11 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 6.1 (5 points)

Your answer here

2.2 b) Problem 6.3 (3 points)

Your answer here

2.3 c) Problem 6.17 (1 point)

Your answer here

2.4 d) Problem 6.30 (2 points)

Your answer here

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

3.1 PyWavelets package installation

You need to have the PyWavelets package installed 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

3.2 Part 1: The wavelet transform (10 points)

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

1) Create a 256×256 array where every pixel value is 1. Then, use the function `pywt.wavedec2` to obtain the approximation coefficients for decomposition levels 1, 2, 3, 4, and 5 for a 5-scale Haar wavelet decomposition of the image.

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

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

Your answer here

3) Complete the function `dwt` that performs the following.

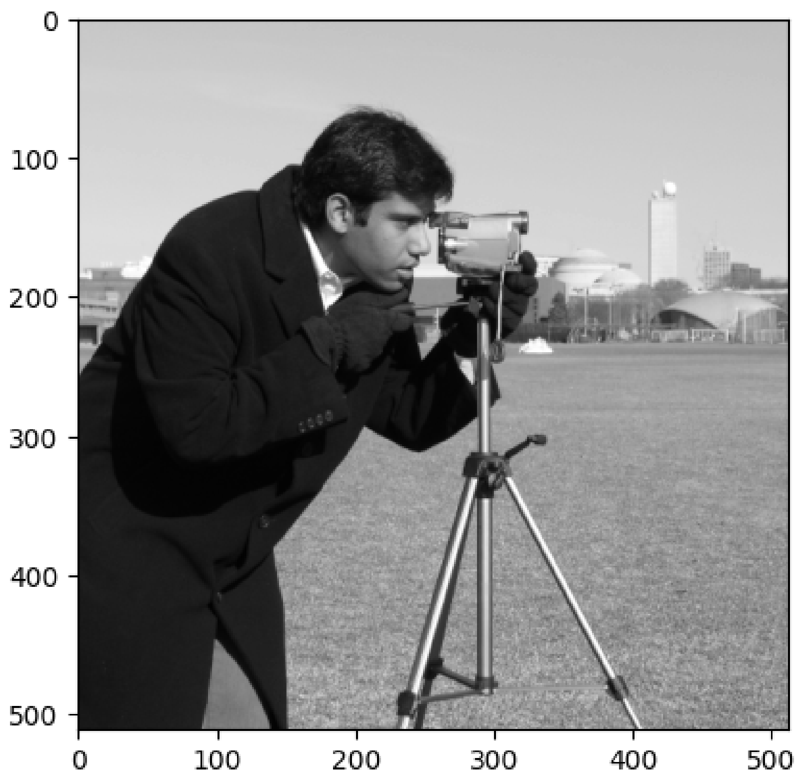
- Re-scales the image to $[0, 1]$. Then, uses the function `pywt.wavedec2` to compute a 3-scale Haar discrete wavelet decomposition of the image.
- Calculates the approximate and detail coefficients to create a figure similar to the one shown in the lower right of lecture 11, slide 43 (hint: scale each 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 decomposition levels 2 and 1, and then scale the approximation coefficients correctly.

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

         returns:
         res: figure similar to the one shown in the lower right of lecture 11, slide␣
      ↪43 (N,N)
         A1: decomposition level 1 approximate coefficient (N/2, N/2)
         A2: decomposition level 2 approximate coefficient (N/4, N/4)
         """
         # your code here

         return res, A1, A2
```

4) Call the function `dwt` with the camera image. Display the figure similar to the one shown in the lower right of lecture 11, slide 43, and also display the approximation coefficients images.

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

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

```
[ ]: <matplotlib.image.AxesImage at 0x2423969b5e0>
```



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

3.3 Part 2: Wavelet-based "Edge" Detection (10 points)

1) 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 decomposition level 0), again using Symlets 4 wavelet filters.

- **Hint:** it may be 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

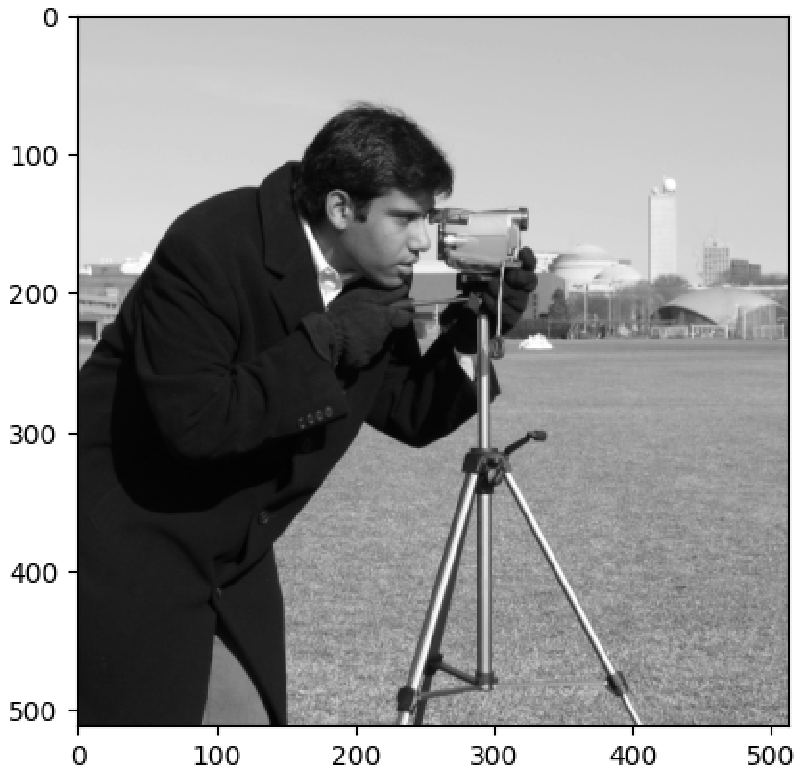
[ ]:
```
# function that detects edges using Symlets 4 wavelet decomposition
def edge_detection(img):
  """
  img: input image (N,N)
```

```
    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
```

2) Call the `edge_detection` function with the camera image. Display the reconstructed "edge" images for all scales.

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

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

[ ]:
```
<matplotlib.image.AxesImage at 0x242396df130>
```



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

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

Your answer here

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

1) Complete the function `noise_removal` that takes an image and a threshold as input, and returns the image after noise removal. The function

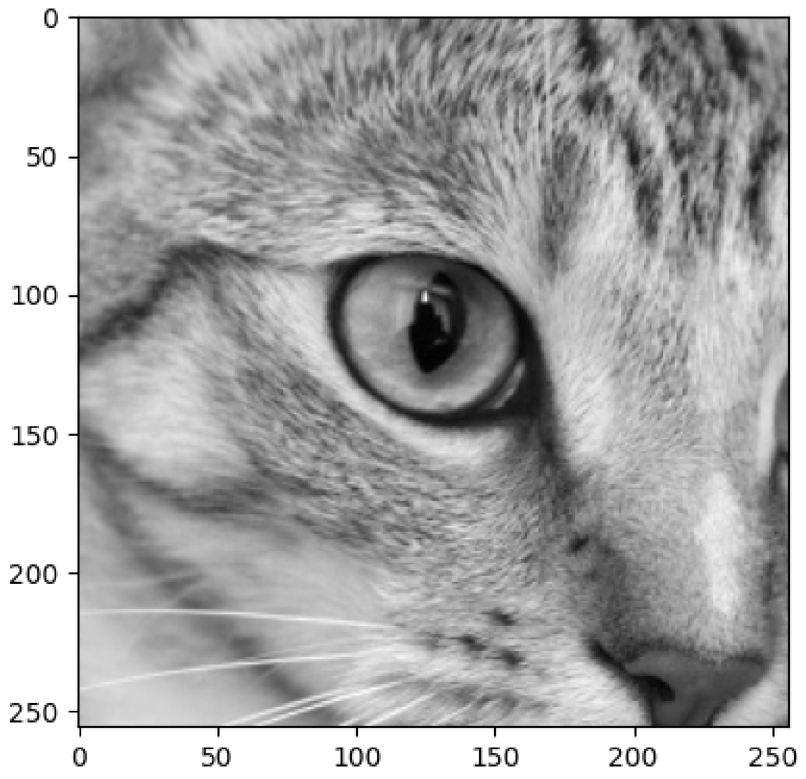
- Computes a 2-scale Symlets 4 wavelet decomposition of the input image
- Sets decomposition level 1 and level 2 detail coefficients with absolute value less than threshold to 0
- Performs wavelet reconstruction (back up to decomposition level 0), again using Symlets 4 wavelet filters
- **Hint:** It may be 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

[ ]:
```
# 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 this threshold to 0

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

    return out
```

2) Call the `noise_removal` function with the cat image and threshold = 40. Display the input image and the image after removing noise.

[ ]:
```
# 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')
```

[ ]:
```
<matplotlib.image.AxesImage at 0x2423975a520>
```



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

3) Briefly discuss the results.

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>
```