as16288-ivp-ca3

October 28, 2023

1 ECE-GY 6123 Image and Video Processing

1.1 Computer Assignment 3: Pyramid Transforms

###Abirami Sivakumar

###as16288

[1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

[2]:
```
imgRGB = cv2.imread("/content/color image.png")
img = cv2.cvtColor(imgRGB, cv2.COLOR_BGR2GRAY)
```

1.2 Problem 1 (Gaussian and Laplacian pyramids).

1.2.1 *(a) Write functions gaussian_pyramid and laplacian_pyramid that decompose an input grayscale image into a J-level Gaussian and Laplacian pyramid, respectively, where J is an input to each function. You can use cv2.resize for downsampling and upsampling with INTER_LINEAR and INTER_CUBIC filters, respectively.*

[3]:
```
def gaussian_pyramid(img, J):
    gaussianPyramid = [img]
    for _ in range(J-1):
        img = cv2.resize(img, (img.shape[1] // 2, img.shape[0] // 2),␣
 ↪interpolation=cv2.INTER_LINEAR)
        gaussianPyramid.append(img)
    return gaussianPyramid

def laplacian_pyramid(img, J):
    gaussianPyramid = gaussian_pyramid(img, J)
    laplacianPyramid = []

    for j in range(J-1):
        currentImage = gaussianPyramid[j]
        nextImage = gaussianPyramid[j+1]
```

```
        upsampledImage = cv2.resize(nextImage, (currentImage.shape[1],␣
↪currentImage.shape[0]), interpolation=cv2.INTER_CUBIC)
        laplacianPyramid.append(currentImage - upsampledImage)

    laplacianPyramid.append(gaussianPyramid[-1])
    return laplacianPyramid
```

1.2.2 *(b) Write a function reconstruct_laplacian that reconstructs the original image from a J-level Laplacian pyramid. Verify it works correctly on a test image. Display the Gaussian and Laplacian pyramid images for $J = 3$.*

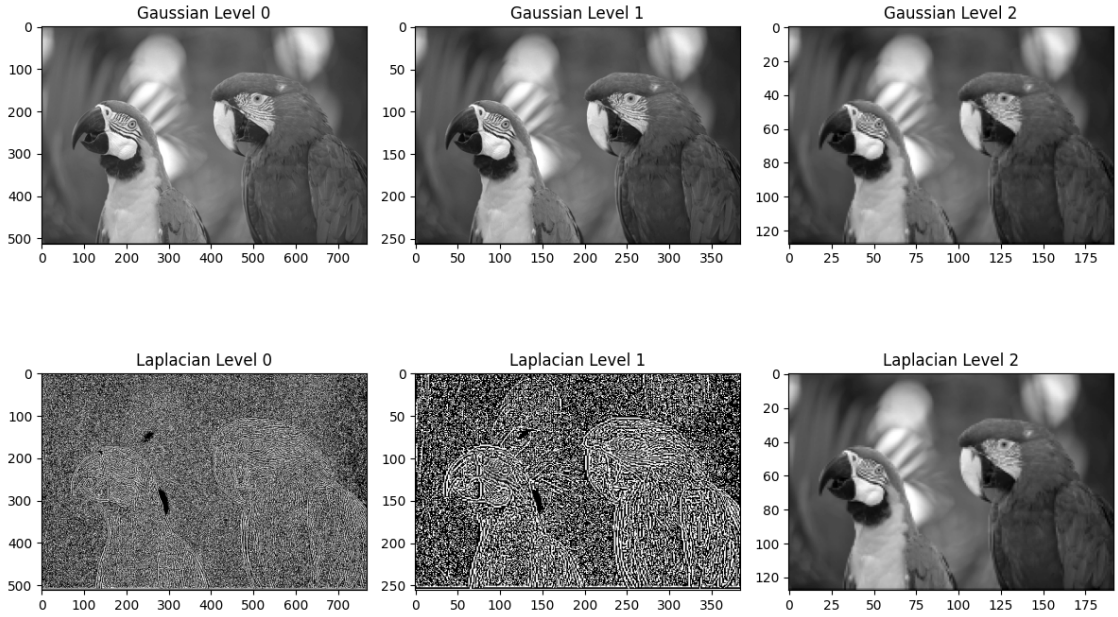
[4]:
```
def reconstruct_laplacian(laplacianPyramid):
    reconstructed = laplacianPyramid[-1]
    for i in range(len(laplacianPyramid) - 2, -1, -1):
        reconstructed = cv2.resize(reconstructed, (laplacianPyramid[i].
↪shape[1], laplacianPyramid[i].shape[0]), interpolation=cv2.INTER_CUBIC)
        reconstructed = (reconstructed + laplacianPyramid[i]).clip(0, 255)
    return reconstructed.astype(np.uint8)
```

[5]:
```
J = 3
gaussianPyramid = gaussian_pyramid(img, J)
laplacianPyramid = laplacian_pyramid(img, J)
```

[6]:
```
reconstructed_image = reconstruct_laplacian(laplacianPyramid)

if np.array_equal(img, reconstructed_image):
    print("Successfully reconstructed and verified")
else:
    print("Failed")
```

```
Successfully reconstructed and verified
```

[7]:
```
fig, axes = plt.subplots(2, J, figsize=(12, 8))

for j in range(J):
    axes[0, j].imshow(gaussianPyramid[j], cmap='gray')
    axes[0, j].set_title(f'Gaussian Level {j}')

    axes[1, j].imshow(laplacianPyramid[j], cmap='gray')
    axes[1, j].set_title(f'Laplacian Level {j}')

plt.tight_layout()
plt.show()
```

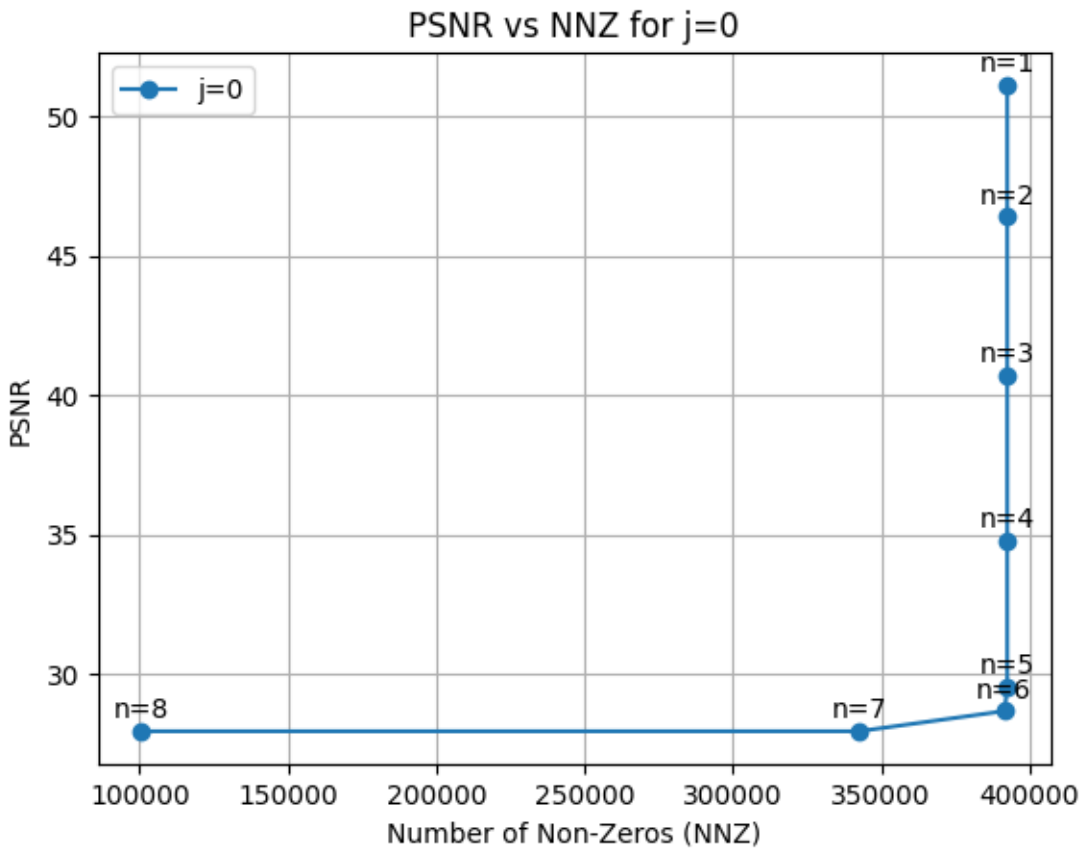


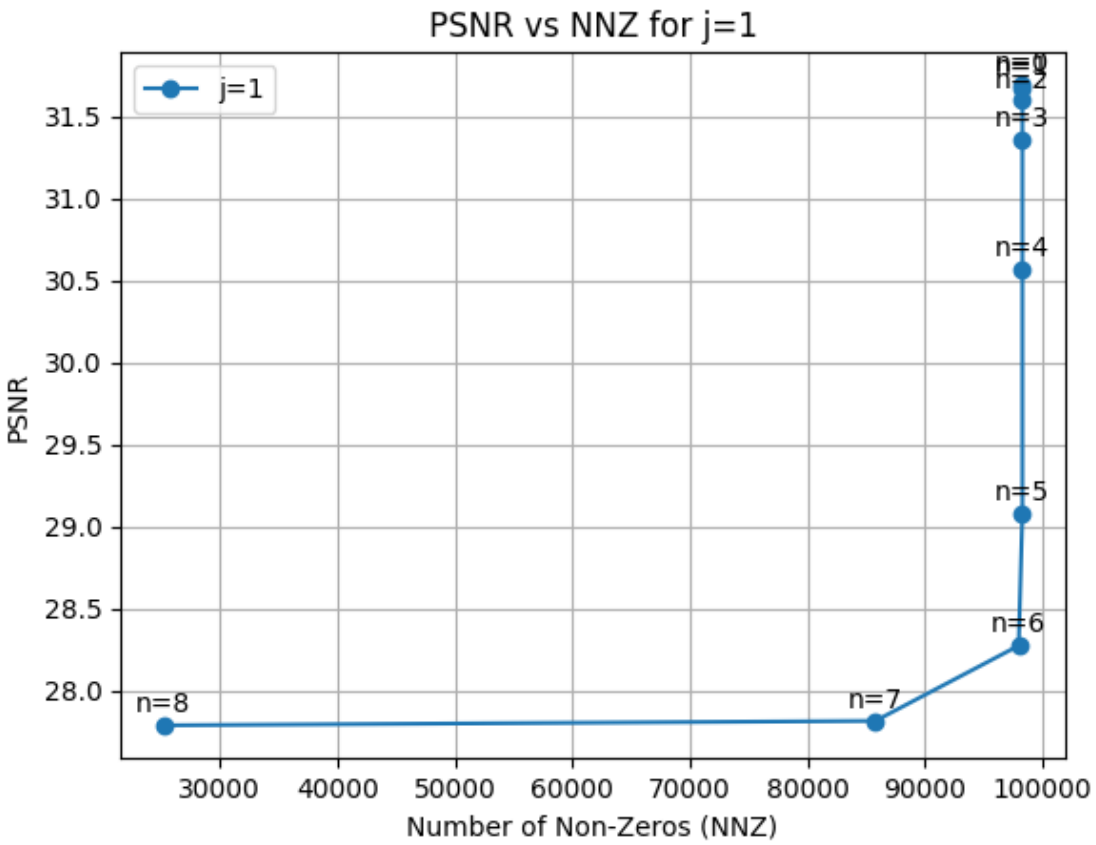
1.2.3 *(c) Write a function quantize_pyramid that takes in a Laplacian pyramid and quantizes the coefficients c with quantization step-size q as follows, Q(c, q) = q floor((c − μ)/q+1/2)+ μ where μ is the mean of the coefficient map, assumed to be μ = 0 for residual (Laplacian) images and μ = 128 otherwise (Gaussian images).*

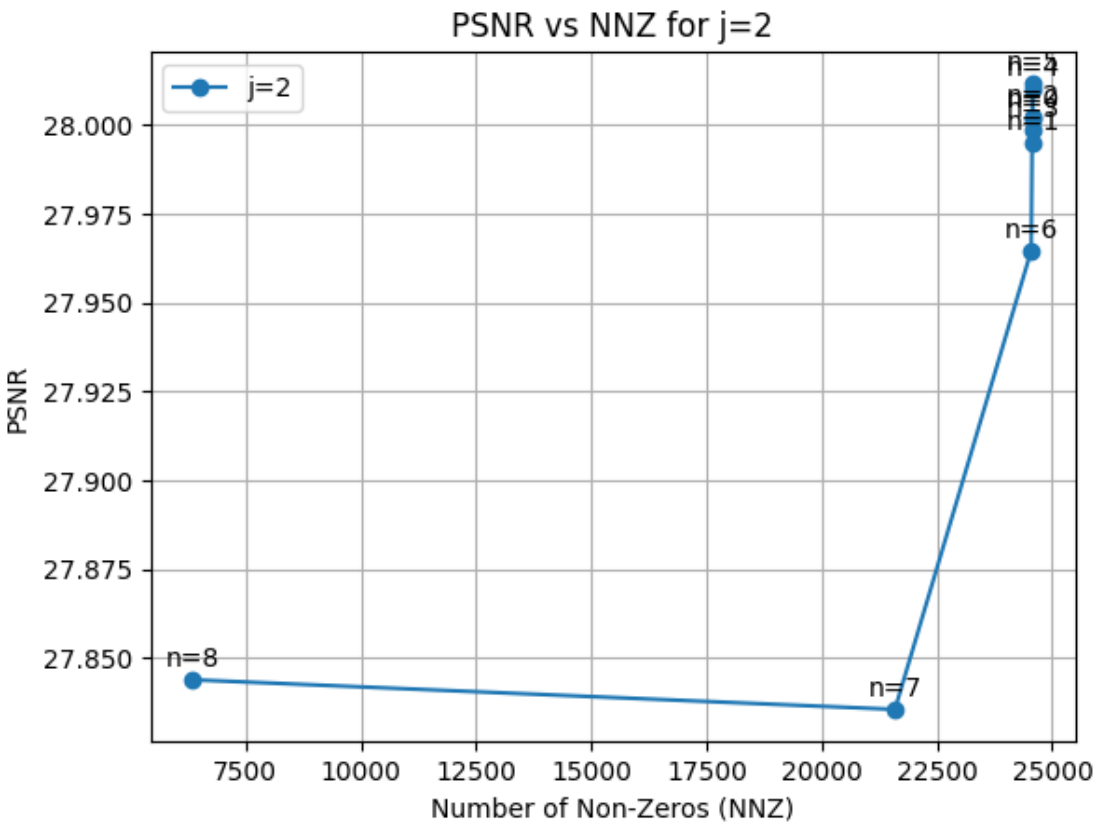
[8]:
```
def quantize_pyramid(laplacianPyramid, q):
    quantizedPyramid = []
    for c in laplacianPyramid:
        mu = 0
        currentQuantized = q * np.floor((c - mu) / q + 0.5) + mu
        quantizedPyramid.append(currentQuantized)

    return quantizedPyramid
```

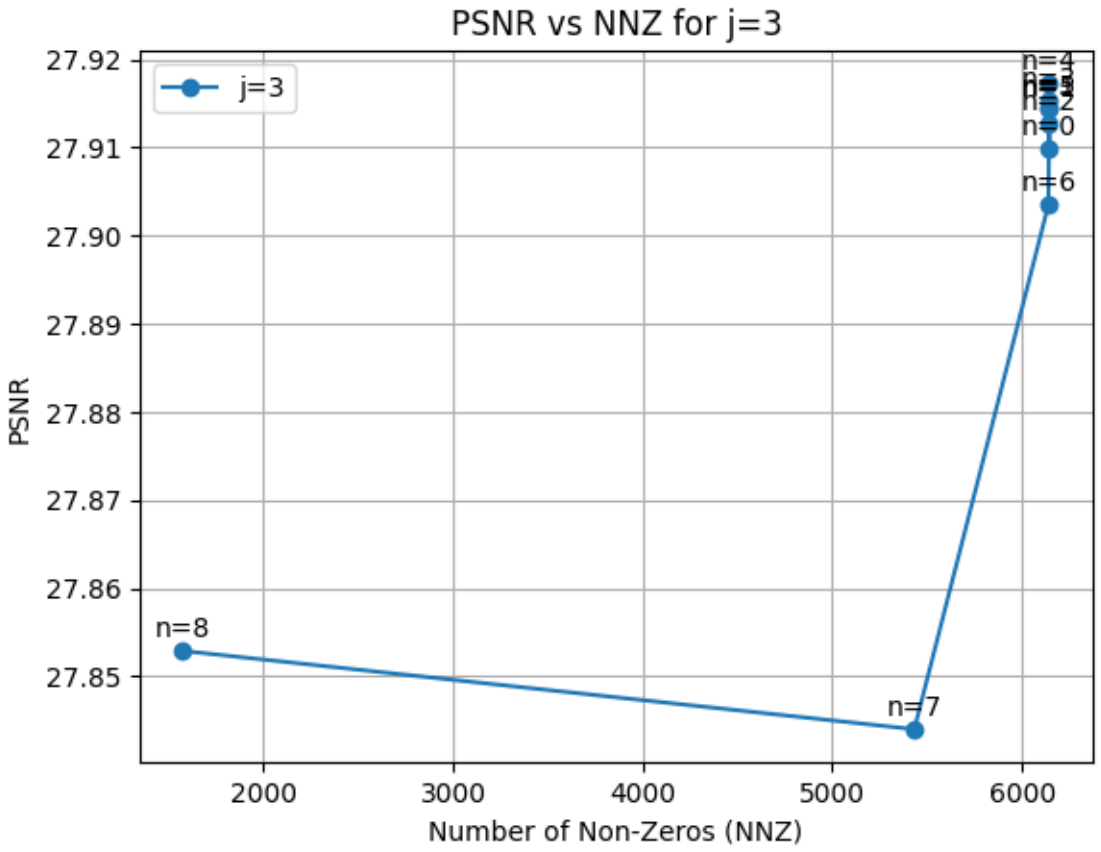
1.2.4 *(d) For pyramid levels J = 0, 1, 2, 3 (where J = 0 is simply the original image) plot the reconstruction PSNR, PSNR = 10 log10 (255^2/MSE), between the original and reconstructed image vs. the number of non-zeros in the representation for pyramids quantized at steps q = 2^n, n = 0, 1, . . . , 8. Plot PSNR on the y-axis and NNZ on the x-axis. What relationship do you observe between pyramid depth and representation efficiency for a desired reconstruction PSNR? Is this expected? Such a curve helps us evaluate the representation efficiency of the Laplacian pyramid. Note that we're using the number of non-zeros (NNZ) as a surrogate for the number of bits needed to describe the image.*

[9]:
```
def computePSNR(orig_img, reconstructed_img):
    mse_value = np.mean((orig_img - reconstructed_img) ** 2)
    return float('inf') if mse_value == 0 else 10 * np.log10(255.0**2 /␣
 ↪mse_value)

for j in range(4):
    plt.figure()

    x_values = []
    y_values = []

    for n in range(9):
        q = 2 ** n
        laplacianPyramid = laplacian_pyramid(img, j+1)
        quantizedLaplacianPyramid = quantize_pyramid(laplacianPyramid, q)
        reconstructed_img = reconstruct_laplacian(quantizedLaplacianPyramid)
        currentPSNR = computePSNR(img, reconstructed_img)

        nnz = np.count_nonzero(quantizedLaplacianPyramid[-1])

        x_values.append(nnz)
        y_values.append(currentPSNR)

    plt.plot(x_values, y_values, '-o', label=f'j={j}')
    for i, n in enumerate(range(9)):
        plt.annotate(f'n={n}', (x_values[i], y_values[i]), textcoords="offset␣
 ↪points", xytext=(0,5), ha='center')

    plt.xlabel('Number of Non-Zeros (NNZ)')
    plt.ylabel('PSNR')
    plt.title(f'PSNR vs NNZ for j={j}')
    plt.legend()
    plt.grid(True)
    plt.show()
```


PSNR vs NNZ for j=0
j=0
n=1
n=2
n=3
n=4
n=5
n=6
n=7
n=8
PSNR
Number of Non-Zeros (NNZ)
50
45
40
35
30
100000
150000
200000
250000
300000
350000
400000


PSNR vs NNZ for j=1
j=1
n=0
n=1
n=2
n=3
n=4
n=5
n=6
n=7
n=8
PSNR
31.5
31.0
30.5
30.0
29.5
29.0
28.5
28.0
30000
40000
50000
60000
70000
80000
90000
100000
Number of Non-Zeros (NNZ)


PSNR vs NNZ for j=2
j=2
28.000
27.975
27.950
27.925
27.900
27.875
27.850
PSNR
n=8
n=7
n=6
7500
10000
12500
15000
17500
20000
22500
25000
Number of Non-Zeros (NNZ)



1.2.5 *What relationship do you observe between pyramid depth and representation efficiency for a desired reconstruction PSNR? Is this expected?*

It can be observed that as the pyramid depth increases, the PSNR value decreases. This implies that as depth increases, the quality of reconstruction diminishes. This is an anticipated outcome since, as one delves deeper into the pyramid, the images are downsampled, containing fewer details and emphasizing the general structure of the original image

1.2.6 *(e) For J = 3, determine qualitatively at what point the quantization level is unnoticeable. How do the number of non-zeros compare to the original image?*

Looking at Figure J with n = 3, it becomes evident that the PSNR values for n = 0, n = 1, n = 2, and n = 3 are closely grouped together, indicating a subtle difference in the reconstructed images for these quantization levels. To elaborate further, the quantization values of n = 0, n = 1, and n = 2 are practically imperceptible.

Furthermore, as we extend the analysis from n = 0 to n = 5, we observe that the number of non-zero values closely resembles that of the original unquantized image, hovering around 296,500. However, starting from n = 6 through n = 8, the number of

non-zero values begins to decline. Notably, there is a significant decrease from n = 7 to n = 8. This decline occurs because higher values of n result in a larger q value. For n = 8, q equals 2^8, which is 256, causing most of the pixels in the image to fall below 1 after being divided by q and having 0.5 added. The floor operation then rounds these values down to 0, leading to a reduction in the number of non-zero pixels.

[ ]: