Signals Project Report

Image Compression & Decompression

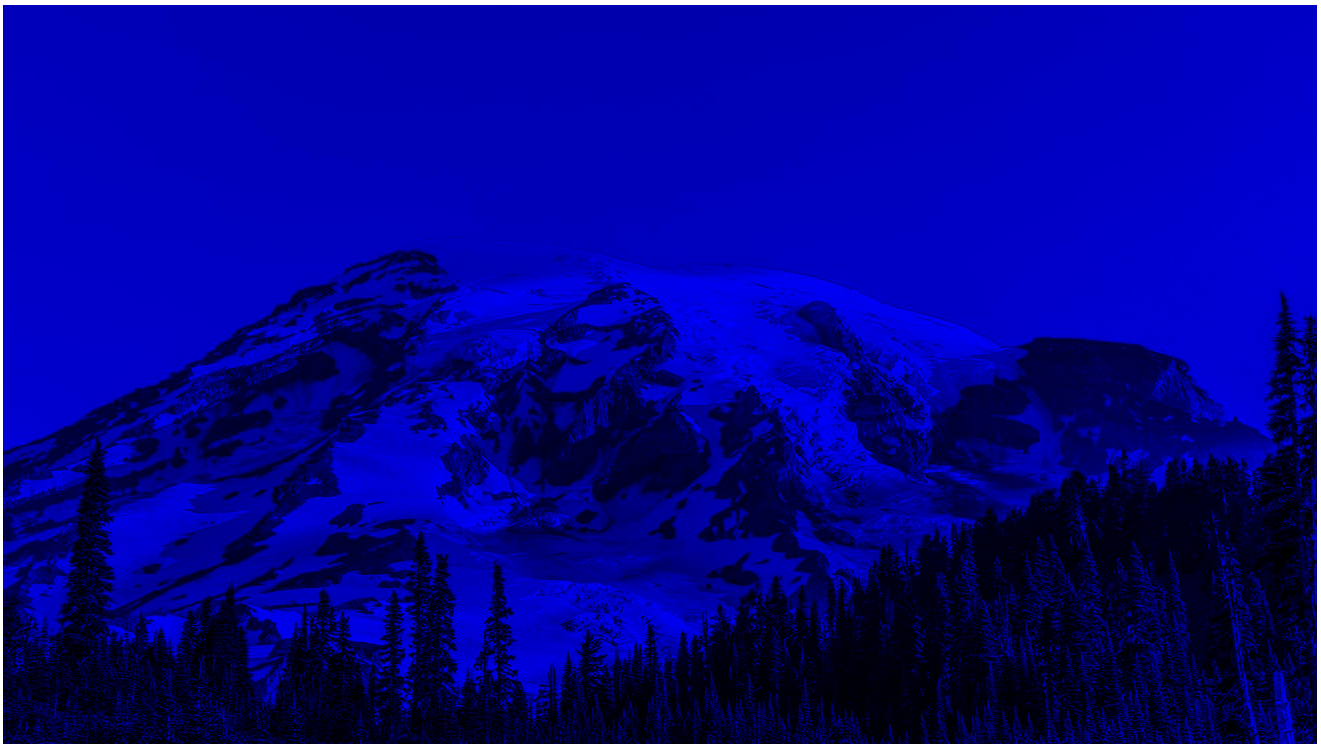
Team Members

Name	Code	Sec
Abdelruhman Sami Mohamad	9220430	1
Ahmed Hamed Gaber	9220027	1

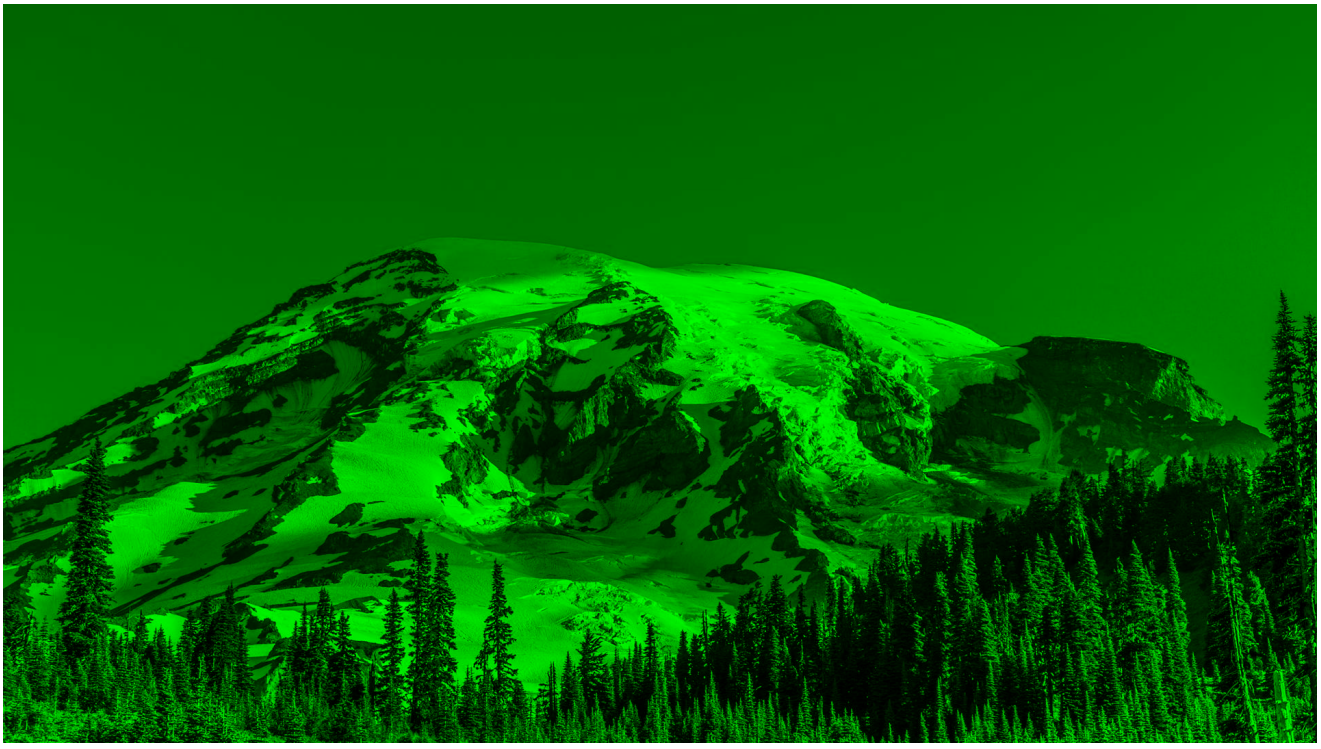
Original Image

Color Components

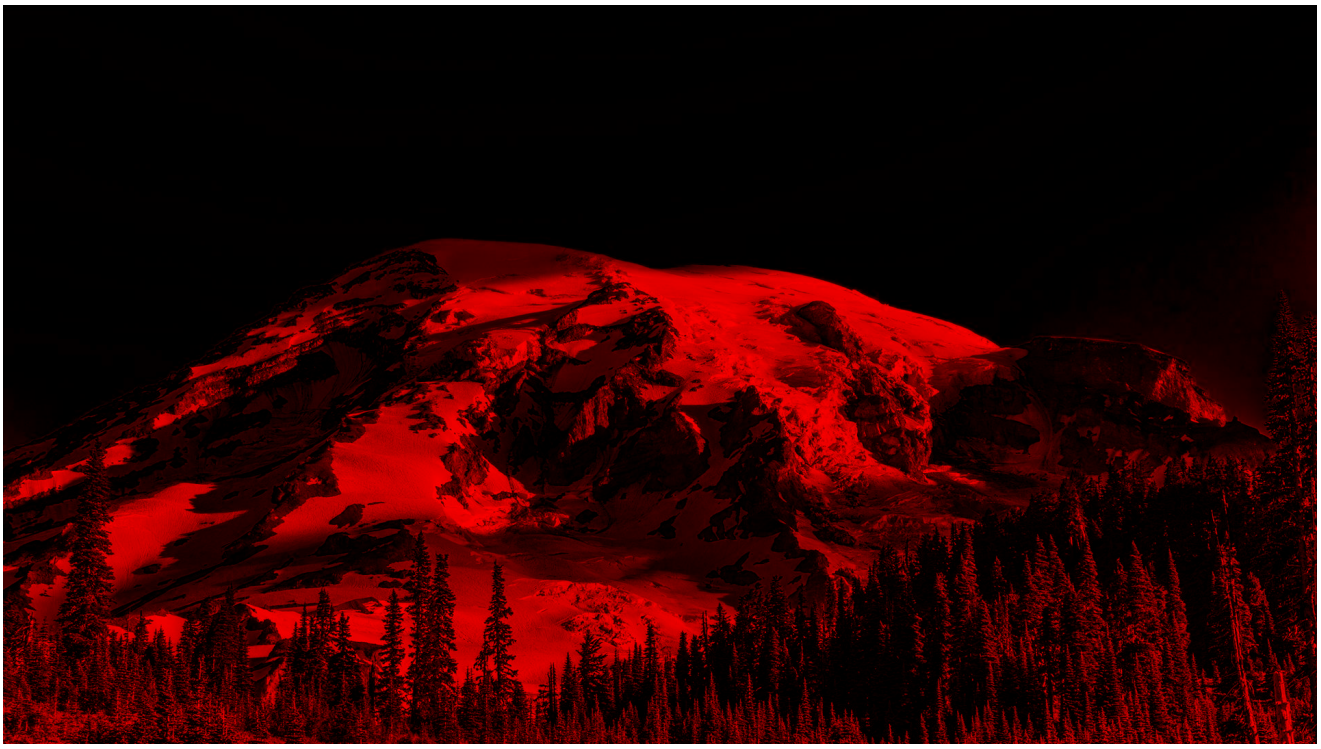
blue component



Green Component



Red Component



Decompressed Images

m = 1

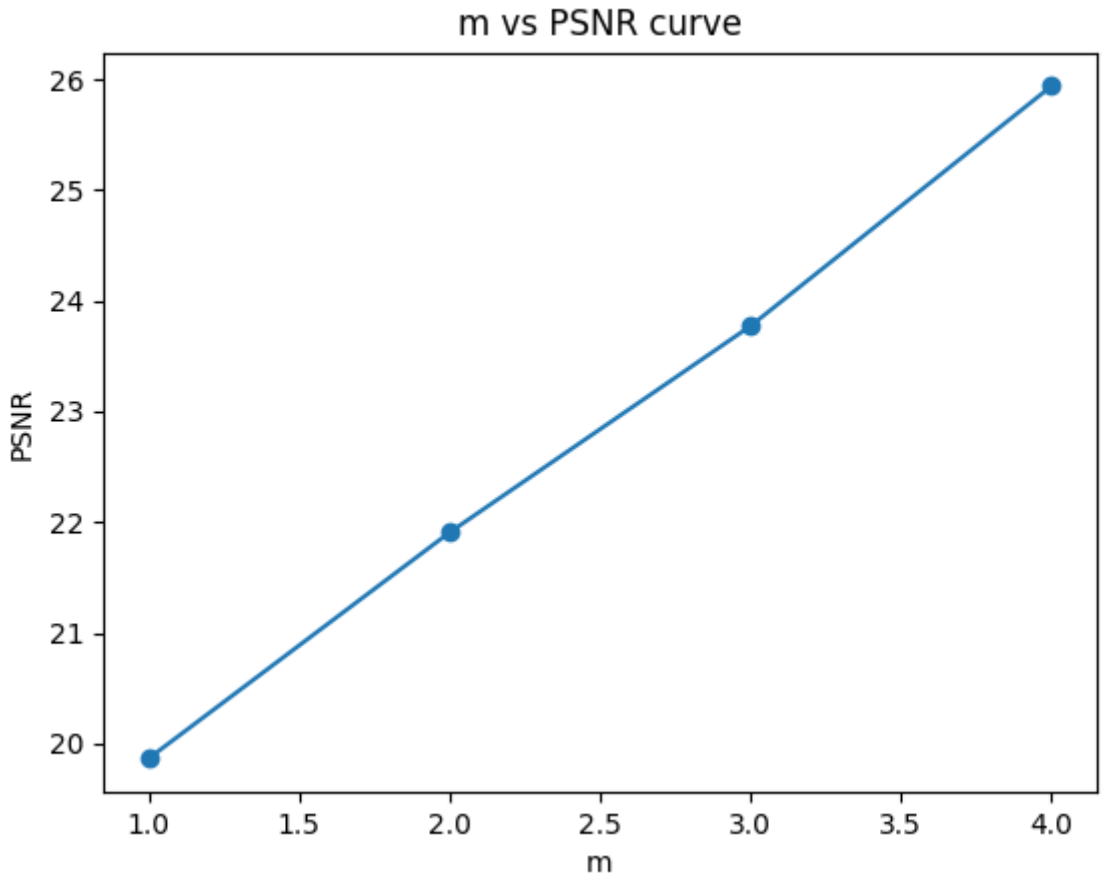
m = 2

m = 3

m = 4

PSNR VS M Curve

m value	PSNR (in db)
1	19.862
2	21.905
3	23.772
4	25.941



- From the results of the graph, it's obvious that as m increases, the value of the PSNR increases.
- The value *m* refers to the size of the block of frequency we return while performing 2D DCT analysis, so as m increases, this leads to increasing the number of frequency coefficients we take in consideration.
- This leads to getting a version of the image which is higher in quality (relative to that with less value of *m*)
- Finally, Increasing the quality leads to increasing the value of PSNR.

Comparing sizes

Image	Size in KB	Ratio to original image
original image	3482	1
compressed with m = 1	618	0.196
compressed with m = 2	2150	0.617
compressed with m = 3	2458	0.706
compressed with m = 4	2662	0.765

Code

```
#!/bin/env python3

import cv2

import numpy as np

import matplotlib.pyplot as plt

from scipy.fft import dctn, idctn



def printGraph(PSNR_array):

plt.plot(range(1, 5), PSNR_array, marker='o')

plt.xlabel('m')

plt.ylabel('PSNR')

plt.title('m vs PSNR curve')

plt.savefig('m_vs_PSNR.png')

plt.show()



def getInput():

imageName = input("Enter the name of the image file: ")

if imageName == "":

imageName = "image.png"

image = cv2.imread(imageName)

for color in range(3):

coloredImage = np.zeros_like(image)

coloredImage[:, :, color] = image[:, :, color]

cv2.imwrite(f"color{color}.png", coloredImage)

return image
```

```
def main(image, length, width, PSNR_array, m):
# Initialize the compressed and decompressed images
compressedImage = np.zeros_like(image)
decompressedImage = np.zeros_like(image)

# Compress the image
for color in range(3):
channel = image[:, :, color]
for i in range(0, length, 8):
for j in range(0, width, 8):
block = dctn(channel[i:i+8, j:j+8])
block[m:, :] = 0
block[:, m:] = 0
compressedImage[i:i+8, j:j+8, color] = block
decompressedBlock = idctn(block)
np.clip(decompressedBlock, 0, 255, out=decompressedBlock)
decompressedImage[i:i+8, j:j+8, color] = decompressedBlock

val = cv2.PSNR(image, decompressedImage)
PSNR_array.append(val)
print(val)
cv2.imwrite(f"compressed{m}.png", compressedImage)
cv2.imwrite(f"decompressed{m}.png", decompressedImage)
```

```
if __name__ == "__main__":

PSNR_array = []

image = getInput()

length = image.shape[0]

width = image.shape[1]

for m in range(1, 5):

main(image, length, width, PSNR_array, m)

printGraph(PSNR_array)
```