Student Name: Amirreza Rajabi
Student ID: 9831126

Multimedia Systems
Assignment 4

Theory Questions

Question 1:

Lossy and lossless compression are two different methods used to reduce the size of data

Lossless compression techniques reduce file size
without sacrificing any data or quality. Lossless compression algorithms
exploit redundancies and patterns within the data to encode it more efficiently.
PNG - FLAC - ZIP

Lossy compression is a technique that achieves high compression rates by
permanently sacrificing some information from the original file.
JPEG - MPEG - MP3

Question 2:

Color Space Conversion: The image is converted from the RGB color space to
the YCbCr color space.
luminance (Y) - chrominance (Cb and Cr)

Downsampling: The chrominance components (Cb and Cr) are downsampled by
reducing their resolution.

Block Division: The image is divided into smaller blocks typically 8x8 pixels.

Discrete Cosine Transform (DCT): A two-dimensional DCT is applied to each 8x8
block of pixel values and converts spatial domain to frequency domain.

Quantization: The DCT coefficients are quantized. This step involves dividing each coefficient by a corresponding
value from a quantization matrix.

Coding DC/AC: AD coefficients and DC coefficients encoded in two different ways. It uses DPCM for DCs and RLC for ACs.
It uses ZigZag method on 8*8 patch for encoding turn.
RLC (Run-Length Coding) and DPCM (Differential Pulse-Code Modulation)

Entropy Coding: After Coding AC/DC step, It uses Huffman coding on first part of (SIZE, AMPlITUDE) that are output of last step for more compression.
It uses pre-defined table for Huffman Coding.

Saving: Save these data in a pre-defined structure (Headers, Tables, Data).

Question 3:

Baseline JPEG: This is the first type of jpeg and I describe it in the last question.

Progressive JPEG: delivers low quality versions of the image quickly
followed by higher quality passes.
It has two type:
1-Spectral selection:Takeing advantage of the spectrals. saving AC/DC coefficients in a hierreichal form.
2-Successive approximation: Instead of gradually encoding spectral bands, all DCT coefficients are encoded simultane ously but with their most significant bits (MSBs) first.

Hierarchical JPEG: delivers low-resolution of image first and after that delivers resident images in a hierarchical way.

Lossless JPEG: Unlike others, which use lossy compression, lossless JPEG compression preserves all the original image data without any loss in quality. but this kind is so larger compare to others.

Question 4

Pros:

Compression Efficiency

Lossless

Simple

Fast Decding

Cons:

Fixed codebook - Adaptive huffman solves it

Preproccesing Overhead - find freqs

Encoding Overhead

It is not good for small inputs

Can't access to the middle of output - Have to decode all codded data to access one middle part

Question 5

Image compression refers to the procedure of decreasing the file size of an image while making an effort to preserve its visual appearance. Image compression is important for several reasons:
1-Reduced storage space because it is smaller
2-Faster transmission because it has less bits
3-Improved user experience because user can receive images faster
4-Bandwidth optimization because it has less bits

Question 6

Spatial compression works directly on pixel values or samples without considering their relationships. It removes redundancy within the same image or signal. It's simple to implement but may not achieve the highest compression ratios.
Transform-based compression transforms data from the spatial domain to a frequency domain using mathematical transforms like DCT or DWT. It takes advantage of concentration of significant information in fewer coefficients. It can achieve higher compression ratios but is more computationally complex.
comparison: spatial compression is simpler and faster but may not achieve high compression ratios, while transform-based compression offers higher compression ratios with better quality preservation. The choice between the two depends on the specific requirements, constraints, and types of data being compressed.

Question 7

We can see image compression usage in our life for example sharing image through social media or the time we browse the web we request for images and servers are sending us compressed image in formats
like JPEG, PNG, WebP and etc. or even more when you are downloading video you actually download the compressed video in formats like MP4, MPEG and etc.
JPEG is one of the most commonly used image compression formats. It utilizes transform-based compression, specifically the Discrete Cosine Transform (DCT).
NG is a lossless image compression format commonly used for images that require preservation of all details without any loss in quality.
GIF is primarily used for animated images but also supports static images. It uses LZW (Lempel-Ziv-Welch) compression, which is a dictionary-based algorithm.
ebP is a modern image compression format that aims to provide both high compression efficiency and good image quality. WebP uses both spatial and transform-based compression techniques.

Question 8

1. **Loss of quality:** Lossy compression methods discard certain image data to reduce file size, which can result in a loss of image quality.

2. **Artifacts:** Lossy compression can be cuase of compression artifacts such as blurring or color distortions or etc.

3. **Banding:** Banding refers to the visible change from one color to another instead of a smooth, fading gradient

4. **Computational complexity:** Many compression algorithms require significant computational resources, making them slower for encoding or decoding images.

5. **Compatibility:** Different image compression formats may have limited compatibility across different platforms or software.

It is important to know that different compression techniques exhibit unique strengths and weaknesses. Therefore, having a comprehensive understanding of the requirements and limitations specific to your use case is crucial for selecting the most suitable approach to image compression.

Report

Part 1

Summary of this part :

```
# Steps of JPEG compression

Color Space Conversion:
    -   using this function ‘change_color_map‘
        -   args:
                -   image
                -   mod : "rg2ycbcr" or "rgb2hsv"
            return:
                -   image in new color space

Discrete Cosine Transform (DCT):
    -   using this function ‘dct_img‘
        -   args:
                -   image
            return:
                -   dct of image (it runs on 8*8 blocks)

Quantization:
    -   using this function ‘quantize‘
        -   args:
                -   dct image
                -   color_space (I found that in each color what is best quantization
                    matrix)
            return:
                -   quantize of dct image

Coding:
    -   Using ‘huffman‘ or ‘arithmetic‘ coding
    -   These are implemented in files with same name


DeQuantization:
    -   using of this function ‘dequantize‘
        -   similar to Quantization
```

```
Inverse Discrete Cosine Transform (DCT):
    -   using this function ‘idct_freq_img‘
        -   args:
                -   dct image
            return:
                -   image (it runs on 8*8 blocks)

Color Space Conversion:
    -   back to RGB using this function ‘change_color_map‘
```

Table 1: Number of Bits

	Arithmetic	Huffman
RGB	52	3497982
YCbCr	53	5150824
HSV	53	6063182

This table shows the number of bits required to represent the compressed image data using Arithmetic and Huffman coding algorithms. Here's what we can observe :

1. The RGB color space requires 52 bits when compressed with Arithmetic coding and 3,497,982 bits when compressed with Huffman coding.
2. The YCbCr color space requires 53 bits with Arithmetic coding and 5,150,824 bits with Huffman coding.
3. The HSV color space also requires 53 bits with Arithmetic coding and 6,063,182 bits with Huffman coding.

Table 2: Time

	Arithmetic	Huffman
RGB	4.49	2.51
YCbCr	6.71	2.92
HSV	7.59	3.10

This table represents the time taken (in some unit, not specified) to compress the image data using Arithmetic and Huffman coding algorithms. Key observations include:

1. For the RGB color space, Arithmetic coding took 4.49 units of time, while Huffman coding took 2.51 units of time.
2. For the YCbCr color space, Arithmetic coding took 6.71 units of time, while Huffman coding took 2.92 units of time.
3. For the HSV color space, Arithmetic coding took 7.59 units of time, while Huffman coding took 3.10 units of time.

This table provides the PSNR values resulting from compressing the image data using Arithmetic and Huffman coding algorithms. Some key observations are as follows:

Table 3: PSNR (Peak Signal-to-Noise Ratio)

	Arithmetic	Huffman
RGB	28.17	28.59
YCbCr	28.15	31.12
HSV	28.13	34.40

1. For the RGB color space, Arithmetic coding yielded a PSNR of 28.17, while Huffman coding resulted in a PSNR of 28.59.
2. For the YCbCr color space, Arithmetic coding resulted in a PSNR of 28.15, while Huffman coding resulted in a PSNR of 31.12.
3. For the HSV color space, Arithmetic coding yielded a PSNR of 28.13, while Huffman coding resulted in a higher PSNR of 34.40.

combining the results from all three tables, we can determine the overall ranking from worst to best:

1. HSV
2. YCbCr
3. RGB

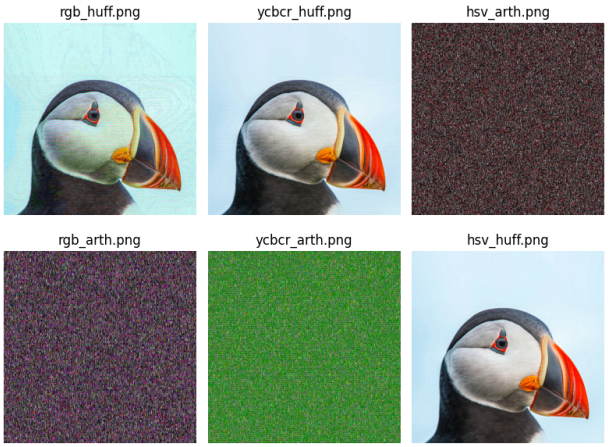

Figure 1: results

Based on the results presented, it appears that arithmetic coding is not effective for image compression tasks. Additionally, in terms of human vision ranking for Huffman coding, it is observed that converting images to different color spaces yields varying levels of compression performance.

In particular, when comparing the color spaces HSV, YCbCr, and RGB, it is found that compressing images in the HSV color space provides better results than both YCbCr and RGB. Similarly, YCbCr performs better than RGB in terms of compression effectiveness.

Part 2

K-means compression is considered a lossy compression technique because it involves quantization, which leads to a loss of information. In the context of image compression, for example, K-means clustering is applied to group similar colors together. The centroids of these color groups are then used to represent the image instead of the original pixel values.

The number of clusters in K-means compression has a direct relationship with compression loss, algorithm speed, and compression rate.

1. **Compression Loss:** The more clusters used in K-means compression, the better the representation of the original data. With a higher number of clusters, the reconstructed image or dataset can closely resemble the original input, resulting in lower compression loss.

2. **Algorithm Speed:** The computational complexity of the K-means algorithm is dependent on the number of clusters. Increasing the number of clusters typically increases the algorithm's computational time. This is because each iteration of K-means requires calculating distances between data points and cluster centroids, and more clusters mean more distance calculations.

3. **Compression Rate:** Using a larger number of clusters generally results in a lower compression rate because more centroids need to be stored, requiring additional bits for representation. On the other hand, a smaller number of clusters leads to a higher compression rate as fewer centroids are required.

    (a) K-means Algorithm:
        i. Initialize centroids using K-means++ initialization.
        ii. Perform K-means clustering on the image pixels.
        iii. Replace pixel values with corresponding centroid values.

    (b) Threshold Tuning:
        i. Apply a threshold value for convergence of kmeans

    (c) Vectorization Technique:
        i. Utilize vectorized operations to accelerate computation speed during clustering.

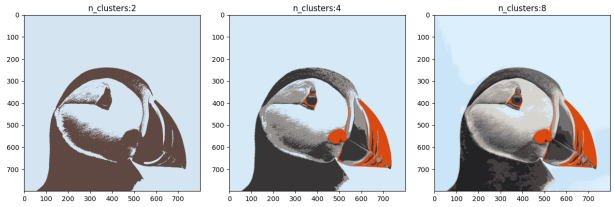

Figure 2: results

From the results, it can be observed that as the number of clusters increases, the compression time also increases. This is expected since more clusters require additional computations for encoding and decoding. Moreover, the compressed bits increase with larger cluster sizes,

Table 4: Performance Evaluation of Image Compression Algorithms

n_clusters	**Time (s)**	**Bits**	**PSNR (dB)**
2	8.35	640,000	28.39
4	15.57	1,280,000	29.88
8	45.04	1,920,000	31.60

indicating a higher level of detail and less aggressive compression. The PSNR values show an improvement as the number of clusters increases, suggesting better image quality at higher cluster sizes.