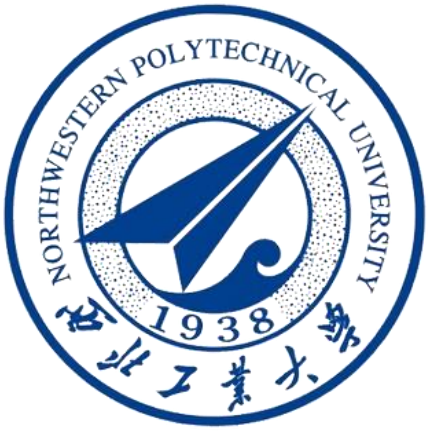
Experiment Report

Multimedia Technology



Experiment Information:

Number	2
Name	Discrete Cosine Transform (DCT)

Student Information:

Student ID:	2021380101
Student Name:	Rheina Trudy Williams

Northwestern Polytechnical University

School of Computer Science

Autumn 2023

I. Introduction

Image compression is one of the various methods that fall under image processing. The goal of image compression is to minimize data representation redundancies, which lowers the need for data storage and, consequently, transmission expenses. Lowering the storage need is equal to raising the storage medium's capacity and, thus, the transmission bandwidth. Discrete Cosine Transform (DCT) is an image compression technique works by breaking breaks down an image, into a collection of different-frequency cosine functions. Through eliminating less important frequency components, this representation enables effective compression, maintaining the crucial visual information while minimizing file size. The DCT is beneficial for image processing applications due to a number of important advantages. Firstly, it generates real-valued coefficients, which makes computations easier to understand and less difficult. Secondly, it is appropriate for lossy compression (compromise some of the image's finer details in order to save a little bit more bandwidth or storage space) due to its strong energy compaction property, which efficiently groups higher energy components towards lower frequencies. Thirdly, it can be quickly implemented and processed in real time due to its computational efficiency.

In this experiment, a grayscale image is first loaded and formatted. Next, the software does a DCT on the complete image and on each individual 8x8 block. The block-based method compresses the image by removing less important frequency information by setting a predetermined number of AC (alternating current) and DC (direct current) components to zero. After that, the altered coefficients are subjected to the inverse DCT (IDCT) in order to recreate the image. The original image, the DCT coefficients, the image following block-based DCT compression, and the restored image following IDCT are all shown in the visualization.

II. Goals

1. Understand the implementation of Discrete Cosine Transform (DCT).
2. Understand the process of image compression.
3. Comparing histogram equalization algorithm.
4. Write a program to implement the Discrete Cosine Transform (DCT).
5. Explore the frequency component and compression quality of Discrete Cosine Transform (DCT).

III. Tools

Pycharm community 2023.3.1

IV. Code

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def dct2(a):
    return np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(a)))
def idct2(a):
    return np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(a)))
```

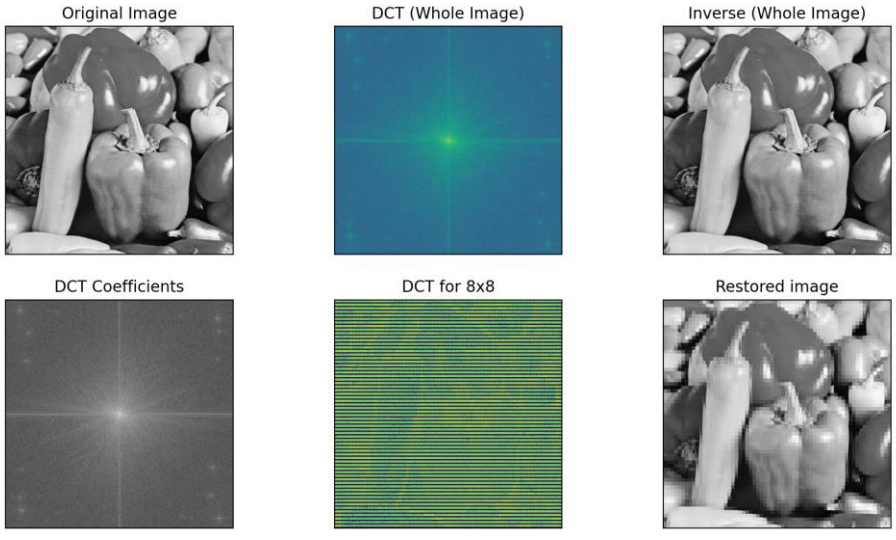
```
def whole_img_dct(img_f32):
    img_dct = dct2(img_f32)
    img_dct_log = np.log(abs(img_dct) + 1e-10)
    img_idct = idct2(img_dct)
    return img_dct, img_dct_log, img_idct

def block_img_dct(img_f32, f_ac, f_dc):
    height, width = img_f32.shape[:2]
    block_y = height // 8
    block_x = width // 8
    height_ = block_y * 8
    width_ = block_x * 8
    img_f32_cut = img_f32[:height_, :width_]
    img_dct = np.zeros((height_, width_), dtype=np.float32)
    new_img = img_dct.copy()

    for h in range(block_y):
        for w in range(block_x):
            img_block = img_f32_cut[8 * h: 8 * (h + 1), 8 * w: 8 *
(w + 1)]
            img_dct[8 * h: 8 * (h + 1), 8 * w: 8 * (w + 1)] =
dct2(img_block)

            # Change the number of preserved harmonics
            img_dct[8 * h: 8 * (h + 1), 8 * w + f_ac:] = 0
            # Zero out AC
            img_dct[8 * h + f_dc:, 8 * w: 8 * (w + 1)] = 0
            # Zero out DC
            dct_block = img_dct[8 * h: 8 * (h + 1), 8 * w: 8 * (w
+ 1)]
            img_block = idct2(dct_block)
            new_img[8 * h: 8 * (h + 1), 8 * w: 8 * (w + 1)] =
img_block

    img_dct_log = np.log(abs(img_dct) + 1e-10)
    return img_dct, img_dct_log, new_img

if __name__ == '__main__':
    img = cv2.imread(r"C:\Users\Rheina
Trudy\Documents\UNI\SEMESTER
5\Multimedia\Experiments\images\1.jpg")
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img_f32 = img_gray.astype(float)

    # (a) Perform DCT
    img_dct, img_dct_log, img_idct = whole_img_dct(img_f32)

    # (b) Change the number of preserved harmonics
    f_ac = 10
    f_dc = 5
    img_dct_block, img_dct_log_block, new_img =
block_img_dct(img_f32.copy(), f_ac, f_dc)

    # (c) Perform Inverse DCT
    I_DCT = dct2(img_f32)

    # Visualization
    plt.figure(figsize=(12, 8))
    plt.subplot(231)
    plt.imshow(img_gray, cmap='gray')
    plt.title('Original Image'), plt.xticks([]), plt.yticks([])
```

```
    plt.subplot(232)
    plt.imshow(img_dct_log)
    plt.title('DCT (Whole Image)'), plt.xticks([]), plt.yticks([])

    plt.subplot(233)
    plt.imshow(abs(img_idct), cmap='gray')
    plt.title('Inverse (Whole Image)'), plt.xticks([]),
plt.yticks([])


    plt.subplot(234)
    plt.imshow(np.log(abs(I_DCT) + 1e-10), cmap='gray')
    plt.title('DCT Coefficients'), plt.xticks([]), plt.yticks([])

    plt.subplot(235)
    plt.imshow(img_dct_log_block)
    plt.title('DCT for 8x8'), plt.xticks([]), plt.yticks([])

    plt.subplot(236)
    plt.imshow(abs(new_img), cmap='gray')
    plt.title('Restored image'), plt.xticks([]), plt.yticks([])

    plt.show()
```

Description:

First, the DCT process is applied to the entire image, and the logarithm of the absolute values of the DCT coefficients that are produced is calculated. After that, a block-wise DCT of the image is carried out, keeping or zeroing out a predetermined number of harmonics for the AC and DC components. Within 8x8 image blocks, localized frequency analysis is made possible by this block-wise DCT technique. The initial grayscale image, the block-wise DCT coefficients and restored image, the DCT coefficients and their logarithms for the full image, and the inverse DCT of the complete image are all seen by the code. The code demonstrates how changing the amounts of preserved harmonics during the block-wise DCT affects the image's compressed form.

V. Result and analysis



Explanation:

The original grayscale image is displayed in the first subplot of the generated plots. The logarithm of the absolute values of the DCT coefficients calculated for the full image is shown in the second subplot. The image's absolute values, produced by applying the inverse DCT (IDCT) to the whole set of DCT coefficients, are shown in the third subplot. The logarithm of the absolute values of the DCT coefficients computed for the full image is displayed in the fourth subplot, which highlights the frequency data. High-frequency coefficient (AC coefficient) is the image's texture and boundaries, it is well-preserved, while low-frequency information (DC coefficient) is retained, mostly contains information about the level region in the stored picture. The log-transformed DCT coefficients for each 8x8 block in the image are displayed in the fifth subplot, along with the number of AC and DC components that have been zeroed out. Finally, the absolute values of the image restored from the changed DCT coefficients are shown.

The reason why DCT is applied in 8x8 block rather than the whole image:

The global frequency information, which shows how intensity fluctuates throughout the image, is captured by the Discrete Cosine Transform (DCT) when it is applied to the full image. However, localized frequency analysis inside smaller regions is possible when DCT is applied to 8x8 blocks. We are able to deliberately alter and eliminate high-frequency elements in the spatial domain by individually transforming tiny blocks. While reducing perceptual loss, larger compression ratios are made possible by the selective removal of high-frequency information during compression. Furthermore, because various blocks may have distinct frequency characteristics, applying DCT to blocks offers some flexibility to image properties.

VI. Conclusion

In conclusion, the experiment of the Discrete Cosine Transform (DCT) for image compression gives me understanding on how this method is used in multimedia applications. The experiment illustrated how DCT can be used to an image as a whole as well as to individual 8x8 blocks, to record both localized and global frequency information in image compression. A comprehensive compression process was made possible by the visualization of the original grayscale image, the DCT coefficients, and the restored image. At first the DCT works by separating parts of image into different sections of importance through frequency decomposition, usually 8x8 blocks, each of which represents a different frequency component. The significance of this frequency decomposition is to treat image data selectively according to its relative quality. Finer details are captured by higher-frequency components, and wider patterns are represented by lower-frequency components. By compressing energy into fewer coefficients, the DCT effectively converts these sections into a set of coefficients, highlighting energy distribution and enabling efficient compression. The information can be prioritized during

compression thanks to this step-by-step breakdown, which also helps to optimize transmission bandwidth and lower storage needs.

During the experiment, I encountered some problems trying to figure out the process behind DCT without using the intrinsic function of OpenCV. I tried to manually implement it using manual mathematical methods, but it shows long process of loops thus creating long execution run error in python. Therefore, instead of mathematical manual implementation, I use fourier transformation to apply the DCT and process the image into energy distribution and the coefficient. After obtaining the necessary information, I successfully finish the experiment by compressing the image.