**CCIT 4076: Introduction to Engineering**

Final Report on Term Project

Project Title: Image and Video Processing

**Class Number:** CL02

**Group Number:** Group 2

**Contact Person:** Wang Shaolong

**Information of Group Members and Contribution**

|  |  |  |  |
| --- | --- | --- | --- |
| **Member** | **Student ID** | **Contact Email** | **Contribution Section(s)** |
| Siu Siu Ting | 20186469 | 20186469@learner.hkuspace.hku.hk |  |
| Fung Kwok Chun | 20187073 | 20187073@learner.hkupace.hku.hk |  |
| Wang Shaolong | 20182865 | 20182865@learner.hkuspace.hku.hk |  |
| Sonam Sherpa | 20191745 | 20191745@learner.hkuspace.hku.hk |  |
|  |  |  |  |

Content Page

[Content Page 2](#_Toc90561413)

[1. Motivation and Introduction 3](#_Toc90561414)

[2. Image Processing 5](#_Toc90561415)

[**2.1 Color coding 5**](#_Toc90561416)

[**2.2 Discrete cosine transform 6**](#_Toc90561417)

[**2.3 Simulation 11**](#_Toc90561418)

[3. Video compression 16](#_Toc90561419)

[4. Conclusion 19](#_Toc90561420)

1. Motivation and Introduction

Video compression is an extremely important technology nowadays. Without it, movies, TV shows, live streaming on YouTube and Twitch, video calls and so many more will no longer exist since the transmitting bandwidth cannot handle such a large amount of data. Using a photo as an example, assume there is a photo with only three-color red, green, and blue, and the photo size is 40003000 pixels, the file size of this photo will be 400030003 = 36000000 bit 4.292 MB. Down below is a 40003000 pixels wallpaper in jpeg(compressed) format we found online. Obviously, it contains for more than three colors, while the file size of it is only around 416 KB.

A picture containing outdoor, outdoor object, night

Description automatically generated

A screenshot of a video game

Description automatically generated with medium confidence

This shows that photo after compress can make a huge difference in file size. Taking a step further, if a video was made of the non-compressed three-color photo that plays for 5 minutes in 25 fps (fps = frame per second, means the number of photos play in a series in one second. 25 fps is the lowest fps needed to form a consistent video) with no audio, the file size of the video will be 3600000025605 = 270000000000 bit 31.432 GB. For the general public, this is a file size that exceed the monthly data’s quota they have in their subscribed cellular service plan. In other words, it is insufficient to support livestreaming that video. With the help of video compression, it will be possible. Although compressing a video can hugely reduce its file size, video after compression will lost a bit of its original quality, and the quality lost may not be recoverable (lossy compression). Therefore, other than compressing the video, making it look as good as original are also very important. In the following, we will explain how video compression works when balancing file size and quality during compression.

1. **Image Processing**

There are many kinds of video compression technology, all of which are designed to deal with various kinds of redundant video information to improve the compression ratio of a video. In this project, we will focus on two kinds of video compression technology: Transform coding and Predictive coding.

**2.1 Color coding**

It is required to compress identical photos in the video(fps) before compressing the whole video. Similar to video compression, before compressing the photos, it is essential to change the photos from RGB format to YCbCr format in order to do the compression process on both the brightness channel (luma channel) and the color channel (red/blue chroma difference component). By doing so, we can do the sampling on both channels and down sample the luma channel compared to chroma channel since human eyes are more sensitive to brightness changes than color changes.

By using this formula:

+

Where (R, G, B

(Y, Cb, Cr

we can convert a RGB photo to YCbCr photo.

The above formula was come from these three equations:

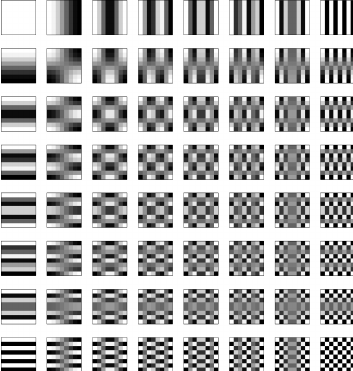
Where , , ,

and combine them into a matrix.

Since both RGB and YCbCr color format admit values ranging from 0 - 255 to represent the color on each pixel, the YCbCr value on each pixel can be obtained by putting the R, G, and B values into the above formula. After that, Discrete Cosine Transformation can be applied to the image.

**2.2 Discrete cosine transform**

Discrete Cosine Transform is referred to as DCT. It can transform the image blocks of size LL from the spatial domain to the frequency domain. Therefore, in the process of image compression and coding based on DCT, the image needs to be divided into non-overlapping image blocks first. Assuming that the size of an image is 1280720, Firstly, it is divided into 16090 image blocks of 88 size without overlapping with each other in the form of grid. Then DCT transformation can be performed on each image block. After segmentation, each 8x8 point image block is sent into the DCT encoder to transform the 8x8 image blocks from the spatial domain to the frequency domain. DCT is a method using cosine wave to represent all the possible spatial frequency on an 8x8 block. Each row of the 8x8 block will be consider as a cosine wave with different frequency, the peak of the cosine wave will be demonstrated as white and trough will be demonstrated as black, between the peak and trough are different density of gray depending on the amplitude of the spatial cosine wave. The figure below visualizes all 64 possible spatial frequency of an 88 block, it is also very clear to see that with higher frequency of the cosine wave, it results in a higher spatial frequency block.



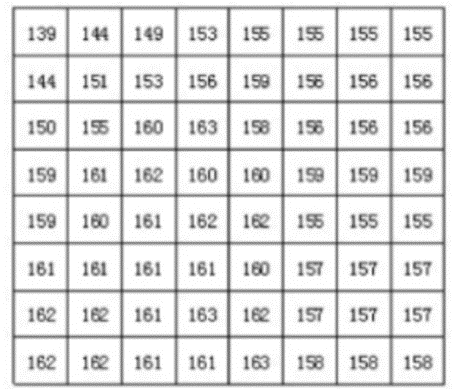
Each of these blocks can be obtain by using following formula:

= cos(2π()(m+))cos(2π()(n+))

Where m = 0, 1, 2, …, M−1 (In the 88 block case, M=8) n = 0, 1, 2, …, N−1

p = 0, 1, 2, …, M−1 (In the 88 block case, M=8) q = 0, 1, 2, …, N−1

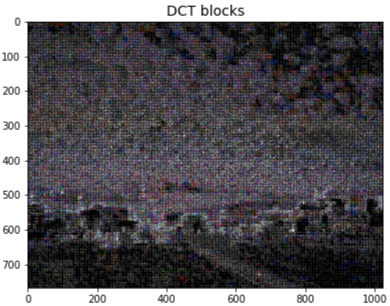
The original 8x8 image block can be form using all these blocks by stacking them together with different density, by storing the coefficient(density) multiply onto these blocks in an 88 matrix. The figure below shows an example of an actual 88 image block, where the numbers represent the brightness value of each pixel. As can be seen from the figure, the brightness values of each pixel in this image block are relatively uniform, especially the brightness values of adjacent pixels do not change very much, indicating that the image signals have a strong correlation.

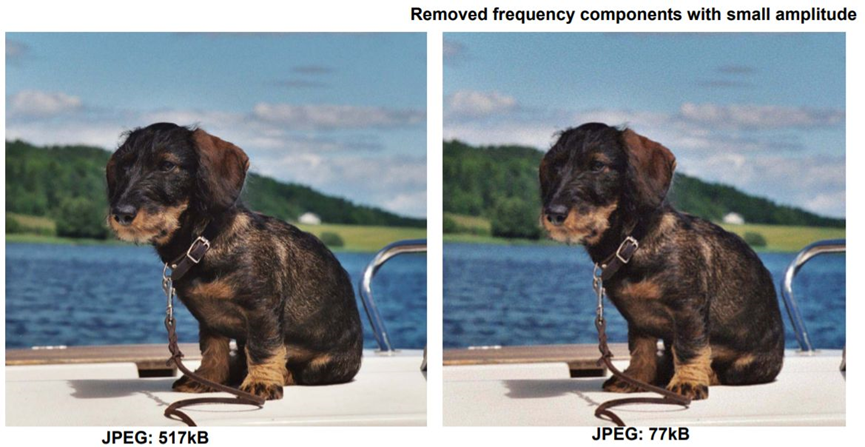
一張含有 文字 的圖片

自動產生的描述

The following figure is the result of quantizing the DCT transformation of the image block in the figure above using a quantizing table. It can be seen from the figure that after quantization, the low spatial frequency coefficient in the upper left corner concentrates a lot of energy, while the high spatial frequency coefficient in the lower right corner has very little energy.

一張含有 桌 的圖片

自動產生的描述

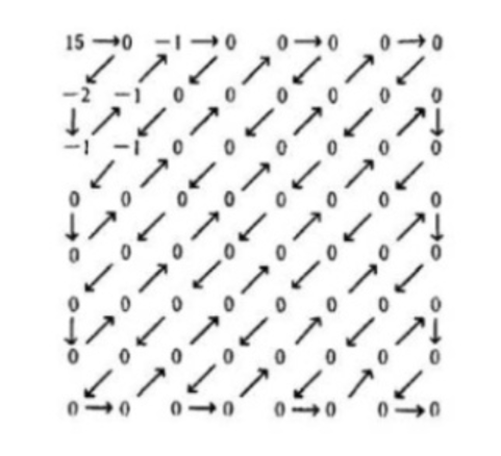
The signal needs to be quantized after DCT transformation. Human eyes are sensitive to the low spatial frequency characteristics of the image, such as the overall brightness of the object, and are insensitive to the high spatial frequency details in the image. As a result, they can transmit less or zero high spatial frequency information in the transmission process, and only transmit the low spatial frequency part. The example below shows that after removing the smaller high spatial frequency components, the image's bit rate is greatly compressed, but the subjective quality of the image is not affected by the same proportion.

In the quantization process, the coefficients in the low spatial frequency regions are finely quantized and the coefficients in the high spatial frequency regions are coarsely quantized to remove the high frequency information that is insensitive for human eyes, thus reducing the amount of information transmitted. The quantization result of the transformed image block is shown in the figure after a reasonable selection of quantization coefficient.

The process of quantization can be expressed by the following formula:

FQ (u,v) represents the DCT coefficient after quantization. F (u,v) represents the DCT coefficient before quantization; Q (u,v) represents the quantization weighting matrix; Q represents quantization step; Round means consolidation, which takes the output value to the nearest integer value.

The quantization result of the transformed image block is shown in the figure after reasonable selection of quantization coefficient.



Most of the DCT coefficients become 0 after quantization, and only a small part of the coefficients is non-zero values. At this point, these non-zero values only need to be coded in the zig-zag method to minimize the storage space needed. Zig-zag method is the number linking pattern shown in the above photo which used to link all the numbers in the matrix into a series. Then, Huffman coding can be applied to code all those numbers in the (run, length) format. For example, eighteen “0” s in a row, will be represent as (18, 0). Instead of storing all the numbers in the matrix individually, using the (run, length) format can hugely reduce the storge space needed to store those numbers.

* 1. **Simulation**

**2.3.1 Python**

We had tried to make a program for image compression. However, we still have not been able to make a workable program. Down below was the progress we had made so far.

Text

Description automatically generated

First, we import the image and change the image from RGB to YCbCr format, using the cv2 package. Then we try to reshape the image to 88 blocks on all three of the channels, so we divided the image height and width by 8 and get the quotient of it. Even though this act will delete some rows and columns, but compared to the size of the original photo, the effect would be relatively small. Then we apply DCT on each block and quantize it using a quantizing table, all the number after quantizing will be presented in(as) integer type. However, we are not able to create our own quantizing table. Furthermore, we have tried to use quantizing tables that were found online. It is still not possible for us to obtain a compressed image. An error that we cannot fix was encountered. Additionally, we cannot apply the zig-zag method on the quantized matrix.

* + 1. **Octave**

We also try to make a program to do the image compression by using Octave. The image down below is showing the program of RGB transform to YCbCr.一張含有 文字 的圖片

自動產生的描述

At the beginning, we need to load the image package to do the function ‘rgb2ycbcr’, which converts the red, green and blue values of RGB images to the brightness (Y) and chroma (Cb and Cr) values of YCbCr images. Then we read the RGB image to convert it to YCbCr. Next, we display both RGB image and YCbCr image to do comparison.

一張含有 文字, 室外, 鐵路 的圖片

自動產生的描述

We also try to make the program to do the process of DCT. The image down below is showing the program of DCT process.

一張含有 文字 的圖片

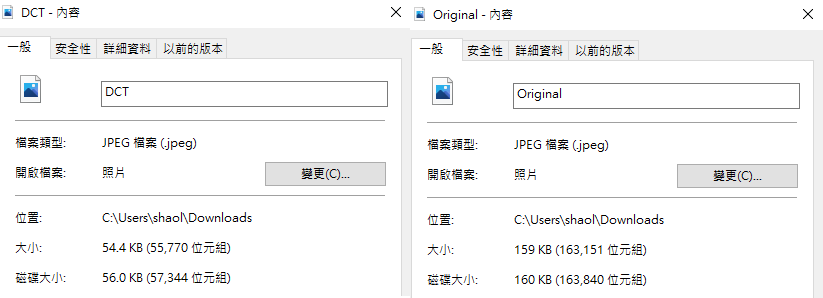
自動產生的描述

Before we run the program, we need to convert the JPEG file into a TIF file which can store high-quality bitmaps and load the signal package to do the function ‘dctmtx’ and load the image package to do the function ‘blockproc’. Read the image and convert it to the double class. The function im2double converts the image to double precision and rescales the output of the integer data type to the range [0, 1]. Then we calculate the 2D DCT of 88 blocks in the image. The function ‘dctmtx’ returns the N×N DCT transformation matrix. The function ‘blockproc’ process images with file name “I”, reading and processing one image block at a time. Then we discard most of the 64 DCT coefficients in each data block, leaving only 5. Finally, the two-dimensional inverse DCT of each data block is used to reconstruct the image. Here is the output of the original image and the image after DCT.

一張含有 天空, 跑道, 火車, 室外 的圖片

自動產生的描述

(Image after DCT) (Original image)



(File size after DCT) (Original file size)

After the process of DCT, the file size has been greatly reduced, from 159KB to 54.4KB.

1. Video compression

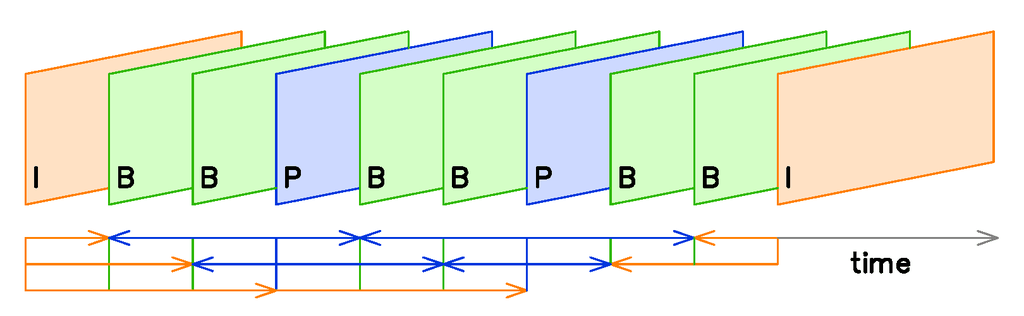
In video compression, the term "prediction" refers to finding or constructing the closest "approximation" of the current block of pixels among several surrounding blocks. Predictive encoding can be used to deal with redundancy in the temporal and spatial domains of video. Prediction coding in video processing can be divided into two main categories: intra-frame prediction and inter-frame prediction.

**Intra-frame prediction:** the predicted value and the actual value are in the same frame to eliminate the spatial redundancy of the image; The characteristic of intra-frame prediction is that the compression rate is relatively low, but it can be decoded independently and does not depend on the data of other frames. In general, the key frames in video use intra - frame prediction.

**Inter-frame prediction:** the actual value of inter-frame prediction is in the current frame, and the predicted value is located inside the reference frame, which is used to eliminate temporal redundancy of the image. The compression rate of inter-frame prediction is higher than that of intra-frame prediction, but it cannot be independently decoded. The current frame can be reconstructed only after the reference frame data is obtained.

**Motion estimation** and **motion compensation** are effective methods to eliminate temporal correlation of image sequences. For example, for digital video with static background and small motion of the main body, such as news broadcasting, the difference between each picture is very small and the correlation between pictures is very large. In this case, we do not need to encode each frame individually, but can encode only the changing part of the adjacent video frames, so as to further reduce the amount of data, which is realized by motion estimation and motion compensation.

Motion estimation technology generally divides the current input image into several small image sub-blocks that do not overlap each other. For example, if the size of an image frame is 1280x720, it is first divided into 40x45 image blocks with the size of 16x16 that do not overlap each other in the form of grid. Then search for a most similar image block for each image block within the scope of a search window of the previous image or the following image. This search process is called motion estimation. A motion vector can be obtained by calculating the position information between the most similar image block and the image block. In this way, in the process of coding, the block in the current image can be subtracted from the most similar image block pointed to by the reference image motion vector to obtain a residual image block. Since each pixel value in the residual image block is small, a higher compression ratio can be obtained in the compression coding. This subtraction process is called motion compensation.

The selection of reference image is very important because a reference image is needed for motion estimation and compensation in coding process. In general, the encoder divides each input frame into three different types according to different reference images: I (Intra) frame, B (Bidirection Prediction) frame, and P (Prediction) frame. 

As shown in the figure, frame I only uses the data in this frame to encode, and it does not need motion estimation and motion compensation in the process of encoding. Obviously, the compression ratio is relatively low because i-frames do not eliminate the correlation of time direction. In the process of coding, p-frame uses a previous I frame or P-frame as the reference image for motion compensation, actually encoding the difference between the current image and the reference image. The encoding method of B frame is similar to that of P frame, the only difference is that it uses a preceding I frame or P frame and a following I frame or P frame for prediction during the encoding process. Thus, the encoding of each P frame needs to use one frame as the reference image, while B frame needs two frames as the reference image. In contrast, B frames have a higher compression ratio than P frames.

1. **Conclusion**

In this report, we have covered how to compress a video and elements that will affect in the compression process. Video compression is a process of removing useless information to human eyes and storing that useful information in an effective way. Even though what we have mentioned is just one of the possible methods, all other methods are also doing the same thing as above. Using these compression methods, we can compress a video to a reasonable file size for people to watch in live and download to their devices. Now, if we use the black and white photo we have compressed (P.15) to generate a 5 min video again with the same presupposition (25 fps, no audio), the file size for the video will be 54.4 25605=408000 bit 0.051 MB. Although the height and width are different from the photo that we presumed at first (P.3, 40003000 pixels non-compress photo), you can already see a huge difference in terms of file size.

In the future, we will want to further study, to understand and master more methods of video compression, such as mixed coding. In addition, we also want to solve the procedures that cannot be done in this report.