CSCI 3280 Introduction to Multimedia Systems - Spring 2020

Assignment 4 – Discrete Cosine Transformation

Deadline: Apr. 29, 2020 (11:59pm)

Submission via Blackboard.cuhk.edu.hk

Late submission penalty: 10% deduction per day (max: 30% deduction; hard deadline: May 15)

PLAGIARISM penalty: whole course failed

Introduction

As the key to the JPEG baseline compression process, the Discrete Cosine Transform (DCT) is a mathematical transformation that transforms a signal from spatial representation into frequency representation. In an image, most of the energy will be concentrated in the lower frequencies. So, once an image is transformed into its frequency components, we can treat them selectively, e.g. retaining lower frequency coefficients with high accuracy while squeezing the size of high frequency coefficients, so as to reduce data amount needed to describe the image without sacrificing too much image quality. In practice, a specially configured quantization table will be used to realize such selective operation. In this assignment, you are required to implement a program that performs DCT on a 2D image and quantize its frequency coefficients.

Implementation Guideline

1. Discrete Cosine Transformation

Given an image $\mathbf{S}$ in the spatial domain, the pixel at coordinates (x, y) is denoted as $\mathbf{S}(x, y)$. To transform $\mathbf{S}$ into an image in the frequency domain $\mathbf{F}$, the most straightforward formula is:

$$\mathbf{F}(u,v) = \frac{c(u)c(v)}{\sqrt{2N}} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \mathbf{S}(x,y) \cos\left(u\pi \frac{2x+1}{2N}\right) \cos\left(v\pi \frac{2y+1}{2N}\right),$$

$$with\ c(x) = \begin{cases} \frac{1}{\sqrt{2}} & if\ x = 0 \\ 1 & else \end{cases}.$$

However, the implementation above uses four nested loops and has complexity $O(n^4)$ for a 2D DCT of size $N \times M$. **Here, you are required to implement a more efficient version.** First, the input image $\mathbf{S}$ (with pixel values ranging from 0~255) is preprocessed by shifting pixel value by 128 and get the shifted image $\bar{\mathbf{S}}$ (with pixel values ranging from -128~127). Then, it

is cut up into blocks of 8×8 pixels (e.g. an image of resolution 256×256 will be divided into 32×32 blocks), and the DCT is applied to each block individually by using row-column decomposition: build a 2D DCT by running a 1D DCT over every row and then every column. Specifically, for each block $\bar{\mathbf{S}}_P$ with 8×8 pixels, the 2D DCT is performed through the following two steps:

- Apply 1D DCT to row v of $\bar{\mathbf{S}}_P$: $\mathbf{R}_P(u,v) = \frac{c(u)}{2}\sum_{x=0}^{7} \bar{\mathbf{S}}_P(x,v)\cos\left(u\pi\frac{2x+1}{16}\right), \forall u \in \{0,1,\dots,7\}$, and repeat it over every row of $\bar{\mathbf{S}}_P$.
- Apply 1D DCT to column u of $\mathbf{R}_P$: $\mathbf{F}_P(\mathrm{u},\mathrm{v}) = \frac{c(v)}{2}\sum_{y=0}^{7} \mathbf{R}_P(u,y)\cos\left(v\pi\frac{2y+1}{16}\right), \forall v \in \{0,1,\dots,7\}$, and repeat it over every column of $\mathbf{R}_P$.

Then, the 8×8 frequency coefficient array $\mathbf{F}_P$ is obtained. By computing the frequency coefficients for every block, you can get the final DCT result, i.e. a $N \times M$ coefficient array $\mathbf{F}$.

2. **Coefficient Quantization**

The frequency coefficients generated by DCT are float numbers and many of them have very tiny values. To convert these coefficients into integers while cause less information loss, **you are required to quantize the frequency coefficients of every blocks by using the provided quantization matrix**. Specifically, for every element in the coefficient array $\mathbf{F}_P$, the actual formula for quantization is: $\hat{\mathbf{F}}_P(u,v) = \left[\!\left[\frac{\mathbf{F}_P(u,v)}{\mathbf{Q}(u,v)}\right]\!\right]$, where $[\![\cdot]\!]$ means rounding a float number to the nearest integer, and the 8×8 quantization matrix $\mathbf{Q}$ is defined as:

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Finally, a $N \times M$ quantized coefficient array $\hat{\mathbf{F}}$ will be achieved.

Basic Requirements: No quantization (80 point)

1. <u>You are required to implement the Discrete Cosine Transform (DCT) as described in Section 1 and apply it to the provided testing image</u>.

2. Program must be coded in ANSI C/C++ and uses standard libraries only.
3. The compiled program must run in Windows 10 command prompt as a console program and accepts source bitmap (.bmp format) with the following syntax and save generated images to the current directory.

```
C:\> dct <img_path> <apply_quantization>
```

dct is the executable file of your program, e.g. the command: **`dct linda.bmp 0`** is to transform the input image from spatial domain into frequency domain and output the frequency coefficient array without quantization applied.
<img_path> is the path where the input image is located.
<apply_quantization> specifies whether the obtained frequency coefficients need to be quantized, where 1 means TRUE while 0 means FALSE. Note that basic requirement does NOT require the implementation of the coefficient quantization part.

4. You are required to submit source code only. We will use Visual Studio 2015 C++ compiler and have your program compiled via Visual Studio command prompt (Tools Command Prompt) with the command line: **`C:\> cl dct.cpp bmp.cpp`** (**Please ensure your source code gets compiled well with it**).

Enhanced Part: With quantization (20 point)

You are further required to implement the coefficient quantization as described in Section 2, which will make the output coefficients be compression-efficient integers.

Submission

We expect the following files zipped into a file named by your SID (e.g. s1234567890.zip) and have it uploaded to the Blackboard by due date: **Apr. 29, 2020 (11:59pm)**

- **README.txt** (Tell us anything that we should pay attention to)
- **bmp.h & bmp.cpp** *(No need to change)*
- **dct.cpp** (write your code in this file only)