CS 576 – Assignment 1

Assigned on 09/04/08,
Solutions due on 09/19/08 (midnight)

This assignment will help you gain a practical understanding of Color spaces, Quantization and Sampling by analyzing how they affect images.

First, you will write a program to display images in the RGB format. We have provided a Microsoft Visual C++ project to display two images side by side. They are the original and the output of your operation. This source has been provided as a reference for students who may not know how to read and display images. You are free to use this as a start, or write your own in C/C++ or any other programming language such as java.

Part 1. Color spaces. (20 points)

The input to your program will be two parameters where

- The first parameter is the name of the image, which is provided in an 8 bit per channel RGB format (Total 24 bits per pixel). You may assume that all images will be of the same size (512x512), more information on the image format will be placed on the class website.
- The next parameter controls what kind of color space is used. The parameter can be 1, 2 or 3.
  - If the parameter is 1, then you need to display images of the R, G, and B channels separately in a single window.
  - If the parameter is 2, then you need to convert from RGB to YUV format, and display images of each channel (Y, U, and V) separately in a single window.
  - If the parameter is 3, then you need to convert RGB format input image to HSV format, and display image of each channel (H, S, and V) separately in a single window. Note: H is an angle!

To invoke your program we will compile it and run it at the command line as

    YourProgram.exe C:/myDir/myImage.rgb M

where M is the parameter as described above.
Part 2. Quantization and Sampling (40 points each)

Input to your program will be four parameters where

- The first parameter is the name of the image, which is provided in an 8 bit per channel RGB format (Total 24 bits per pixel). You may assume that all images will be of size 512 x 512. You will only use the Y channel is the part of the assignment, so the first step is to convert the RGB image to a YUV one.
- The next parameter controls the subsampling of your image (both in row and column). The parameter can take on values from 0 to 8 for some \( n \), 0 suggesting no sub sampling and \( n \) suggesting a sub sampling by \( 2^n \).
- The third parameter is used to choose a subsampling method from 4 different subsampling methods.
- The last controls quantization of Y values. It is a number that specifies how many quantization levels the Y channel can take.

To invoke your program we will compile it and run it at the command line as

```
YourProgram.exe C:/myDir/myImage.rgb S M Q
```

where S, M and Q are the parameters as described above. Example inputs are shown below and this should give you a fair idea about what your input parameters do and how your program will be tested.

1. `YourProgram.exe image1.rgb 0 1 256`
   There are 256 values (8 bits) per Y channel, and no subsampling, which implies that the output is the same as the input.

2. `YourProgram.exe image1.rgb 0 1 64`
   There are 64 values (6 bits) per Y channel and no subsampling.

3. `YourProgram.exe image1.rgb 2 1 256`
   There are 256 values (8 bits) per Y channel (no additional quantization), but the output image is subsampled by 4 with subsampling method 1.

Now for the details - In order the display an image on a display device, the normal choice is an RGB representation. This is what the format of the input image is. However, since we are only interested in the Y channel, you will need to convert the image into YUV space, process your subsampling and reconvert it back to RGB space to show the output to display. Here is the dataflow pipeline that depicts this
1. Read Input Image

2. Convert to YUV space

3. Process subsampling of Y channel only

4. Display only Y channel of Image

5. Quantize Y channel only

6. Convert back to RGB space

Display Output Image

This code is already provided to you, if you choose to make use of it

The RGB to YUV with the conversion matrix is given below

Sub sample Y according to the input parameter

Quantize the Y channel according to the input parameter

Apply the inverse matrix to get the RGB data and display
**Conversion of RGB to YUV**

Given R, G and B values the conversion from RGB to YUV is given by

\[
\begin{align*}
Y & = 0.299 & 0.587 & 0.114 & R \\
U & = -0.147 & -0.289 & 0.436 & G \\
V & = 0.615 & -0.515 & 0.312 & B
\end{align*}
\]

Remember that if RGB channels are represented by n bits each, then the YUV channels are also represented by the same number of bits.

RGB values are positive, but YUV can take negative values!

**Conversion of YUV to RGB**

Given R, G and B values the conversion from YUV to RGB is given by

\[
\begin{align*}
R & = 0.5303 & -0.9545 & 1.1401 & Y \\
G & = 1.2392 & 0.0915 & -0.5806 & U \\
B & = 1.0002 & 2.0324 & -0.0005 & V
\end{align*}
\]

Remember that if YUV channels are represented by n bits each, then the RGB channels are also represented by the same number of bits.

YUV channel can have negative values, but RGB is always positive!

**Sub sampling of Y channel & processing**

Sub sampling will reduce the number of samples for each Y U V channel. And, it effectively reduces your image size by \(1/2^n\) in each dimension. Because values in U and V channels are supposed to be all 0s in grayscale images, you need to fill up 0s in U and V channels of reduced size image instead of doing subsampling. For this assignment, you are supposed to implement hierarchical subsampling. It means that the original image needs to be subsampled by 2 in both dimensions, and then the resulting \(1/4^{th}\) sized image should be subsampled by 2 again in both dimensions, and so on until the size of resulting image is \(1/2^n\) of the original image.

In each subsampling step, you can choose one of 4 methods as explained below, and analyze the output of each method.

You will create the output image by

1) Taking upper left pixel from every \(2 \times 2\) pixel block of each channel.
2) Randomly choosing 1 pixel from every \(2 \times 2\) pixel block of each channel.
3) Copying the average of every \(2 \times 2\) pixel block of each channel.
4) Copying over the average of \(3 \times 3\) neighborhood around the pixel.
Y channel Quantization.
Assume that the quantization levels are uniformly distributed. Initially we have 8 bits per pixel of Y channel. This will take only values from 0 – 255, however if the Q parameter values is 64 then, when quantized, the intensity values will be 0, 3, 7, 11 … 255. You will have to define your quantization function accordingly depending on the input parameter given which specifies the number of intensity levels each channel can take. Assume that the Q parameter will be a power of 2. You don’t need to do quantization on U & V channels, because all the pixel values in U & V channels are 0s.

What should you submit?
- Your source code, and your project file or makefile, if any, using the submit program. Please do not submit any binaries. We will compile your program and execute our tests accordingly.
- Along with the program, also submit an electronic document (word, pdf, pagemaker etc format) using the submit program that explains compilation and execution of your program.

Extra Credit:
Implement better way to do quantization than the way provided in the assignment. And, explain which method you use in your electronic document.