CS 576 – Assignment 2
Instructor: Parag Havaldar

Assigned on Monday 02/12/2024,

Solutions due on Monday 03/04/24 by midday 12 pm noon

Late Policy: None.

PART 1: Theory Questions for practice, no need to submit (solutions will be provided)

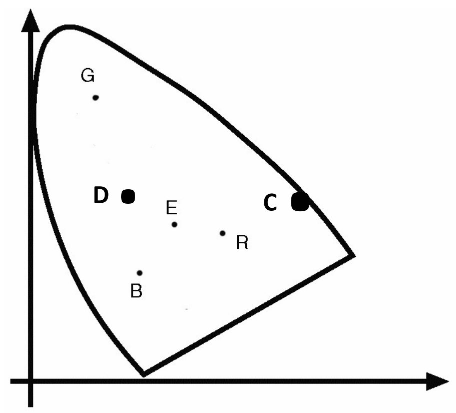
Question1: Color Theory

One of uses of chromaticity diagrams is to find the gamut of colors given the primaries. It can also be used to find dominant and complementary colors –

Dominant color of a given color D (or dominant wavelength in a color D) is defined as the spectral color which can be mixed with white light in order to reproduce the desired D color.

Complementary colors are those which when mixed in some proportion create the color white.

Using these definitions and the understanding of the chromaticity diagram that you have, answer the following.



- In the image alongside find the dominant wavelength of color D. Show this wavelength.
- Do all colors have a dominant wavelength? Explain your reasoning.
- Find the color which is complimentary to the color C. Show this color
- What colors in RGB color space map to the equiluminous point E upon projection into the chromaticity space

Question 2: Color Theory

- The chromaticity diagram in (x, y) represents the normalized color matching functions X, Y and Z. Prove that

  $Z = [\ (1-x-y)/y\ ]\ Y$

Here you are tasked with mapping the gamut of a printer to that of a color CRT monitor. Assume that gamuts are not the same, that is, there are colors in the printer's gamut that do not appear in the monitor's gamut and vice versa. So, in order to print a color seen on the monitor you choose the nearest color in the gamut of the printer. Answer the following questions.

- Comment (giving reasons) whether this algorithm will work effectively?
- You have two images – a cartoon image with constant color tones and a real image with varying color tones? Which image will this algorithm perform better – give reasons?
- Can you suggest improvements rather than just choosing the nearest color?

Entropy Coding –

Consider a communication system that gives out only two symbols X and Y. Assume that the parameterization followed by the probabilities are $P(X) = x^2$ and $P(Y) = (1-x^2)$.

Write down the entropy function and plot it as a function of x

From your plot, for what value of x does the Entropy become a minimum? At what values of x is the Entropy a maximum?

Although the plot visually gives you the value of x for which the entropy in maximum, can you now mathematically find out the value(s) for which the entropy is a maximum?

Can you do the same for the minimum, that is can you find mathematically prove the value(s) of x for which the entropy is a minimum?

Generic Compression –

The following sequence of real numbers has been obtained sampling a signal:
5.8, 6.2, 6.2, 7.2, 7.3, 7.3, 6.5, 6.8, 6.8, 6.8, 5.5, 5.0, 5.2, 5.2, 5.8, 6.2, 6.2, 6.2, 5.9, 6.3, 5.2, 4.2, 2.8, 2.8, 2.3, 2.9, 1.8, 2.5, 2.5, 3.3, 4.1, 4.9
This signal is then quantized using the interval [0,8] and dividing it into 32 uniformly distributed levels.

What does the quantized sequence look like? For ease of computation, assume that you placed the level 0 at 0.25, the level 1 at 0.5P, level 2 at 0.75, level 3 at 1.0 and so on. This should simplify your calculations. Round off any fractional value to the nearest integral levels

How many bits do you need to transmit the entire signal?

If you need to encode the quantized output using DPCM. Compute the successive differences between the values – what is the maximum and minimum value for the difference? Assuming that this is your range (ie, ignore first value), how many bits are required to encode the sequence now?

What is the compression ratio you have achieved (ignoring first value)?

Instead of transmitting the differences, you use Huffman coded values for the differences. How many bits do you need now to encode the sequence? Show all your work and how you arrived at the final answer

What is the compression ratio you have achieved now (ignoring first value)?

Programming Part (*200 points*)

This assignment will help you gain a practical understanding of color representation and quantization for displaying images and videos. For this assignment, you are only required to work with images. Since you have already worked on Assignment 1, you should be comfortable with reading and displaying images in Java/C++. You can use the same starter code provided previously for this assignment as well. Like assignment 1, you are not allowed to use environments which support libraries (matlab, python, javascript, etc.). Use of external libraries or API calls for algorithmic tasks are not allowed either.

Your task is to start with a colored RGB image of size 512x512, and then quantize the colors in the image. There are two types of quantization required to perform for full credit. Additionally, there is an extra credit opportunity for implementing the third type of quantization described later. The display size of your output will be just a 512x512 image while the content should change depending upon the 3 parameters described below:-

- The first parameter is the path to the image file, which will be 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 for this assignment of size 512 x 512. The .rgb file format stores the image in a particular order – first comes the red channel in scan line order, followed by the green and blue channels. Please refer to the reference code if needed.
- The second parameter would be the quantization mode. This parameter could take the following 3 values – 1, 2 or 3. For full credit, you can implement just 1 and 2. For extra credit, you must implement 3.
    - **1** indicates you must perform uniform quantization within the three channels. Each channel has the same number of buckets and are equally spaced.
    - **2** indicates you must perform non-uniform quantization within the three channels. Each channel here has the same number of buckets, but they are not equally spaced.
    - **3** indicates that you must perform non-uniform quantization within three channels where each channel can be assigned a different number of buckets. Each channel not only non-equally spaced buckets, but there may not be the same number of buckets in all channels.
- The third parameter would indicate the total number of buckets. This value would be a cubic value in the range [2^3, 255^3]. These buckets would indicate the number of total colors possible in your output image.

Expectations and Examples

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

YourProgram.exe Image.rgb Q B

where Q and B are the parameters as described above. Example invocations and their expected outputs are explained below which should give you a fair idea about what your input parameters do and how your program will be tested

1. *YourProgram.exe Image.rgb 1 8*

Here, the output image should comprise of at most 8 colors, where each channel has two buckets. Further, these bucket ranges are exactly half of the full range within each channel [0-127], [128-255].

2. *YourProgram.exe Image.rgb 2 27*

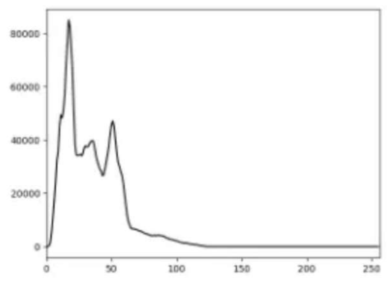
Here, the output image should comprise of at most 27 colors, where each channel has three buckets. However, unlike quantization mode 1, within each channel, the size of each of the 3 buckets need not be the same.

Along with your output, you must submit an analysis report, which includes graphs documenting how the sum of absolute error between the source image and your quantized image varies as you increase the number of buckets. The error must be calculated separately per channel and then added up. To be clear, given the input images that you will be analyzing, you will have two graph plots (one for mode1 and another for mode2) per image. In each plot, the X axis represents the number of buckets per channel (2-255) and the Y axis represents the error sum of all the channels. You are also required to draw some insights and conclusions based on your findings and briefly explain both plots eg:

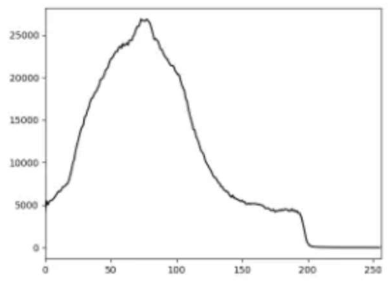
- Is the error is decreasing or increasing from mode 1 to mode 2? What is the reason for this change in error.
- Is the error converging faster or slower from mode 1 to mode 2? What is the reason for the change in convergence?
- Any other insights that you see!

Details of Implementation for modes 1 and 2

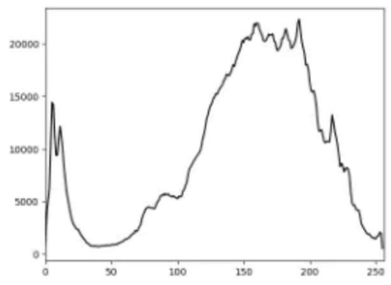
You will need to understand how to represent a histogram for a color channel. A histogram is a statistical data structure needed for this analysis and captures the number of pixels for each color. For a given image, the histograms for R, G and B may separately appear as follows when plotted:



Red channel



Green channel



Blue channel

Mode 1: Your goal is to display an image of dimensions 512x512 with colors quantized based on the provided parameters. To implement the simple uniform quantization for mode 1, all you need to do is to divide each channel into equally spaced $\sqrt[3]{B}$ number of buckets where B are the total number of buckets provided as parameter. You must take the mid-point of each range as the representative color for all the pixels that bucket. Doing this across all three channels and then displaying the image should give you the desired output.

Mode 2: To implement the non-uniform quantization for mode 2, you need to divide each channel into $\sqrt[3]{B}$ number buckets where B are the total number of buckets provided as parameter. However, these buckets in each channel need not be equally spaced. Instead, the spacing should be dynamic for each channel where you provide smaller ranged buckets to values that occur more often to reduce the overall error. To find these buckets, you can analyze the distribution of pixels in a channel and assign bucket ranges such that each bucket contains an equal number of pixels. Further, the representative color for each pixel can be set as the average value of each color in the bucket that it belongs to.

Please refer to lecture on 2/12/2024 for a descriptive discussion

Details of implementation for mode 3 (extra credit)

To receive extra credit, you must provide an implementation which takes **3** as the quantization mode parameter and performs non-uniform quantization across channels where each channel may not have the same number of buckets. Additionally, the error plot for mode 3 should show a a lower overall error score compared to the previous two modes, with faster convergence. This is an open-ended but applicable and interesting problem. You are encouraged to come up with your own approaches to meet the requirements that achieve a lower overall error. Some ideas may include:

- Working in color spaces other than RGB – Moving your representation to another color space and quantization there could help produce lesser error. However, to calculate the error plot and display the image, you must convert the image back to the RGB color space.

- Non uniform and varying number of buckets across different channels. For example, if an image is red dominant and you are provided with 125 buckets, you can assign 25 buckets to red, 5 to green and 1 to blue. This way you can reduce the overall error after quantization since you can more precisely quantize the red in the image. The number of buckets in each channel is an optimization process.

Please refer to lecture on 2/12/2024 for a descriptive discussion

What should you submit?

- Your source code, and your project file needed to compile the code. Please do not submit any binaries or images. We will compile your program and run our cases accordingly. If you need to include a readme.txt file with any special compilation instructions, that is fine too.
- Please submit your plots and analysis as a word/pdf document.