CS 576 – Assignment 2
Instructor: Parag Havaldar

Assigned on Monday 09/18/2023,
Solutions due on Monday 10/09/2023 by midday 12 pm noon

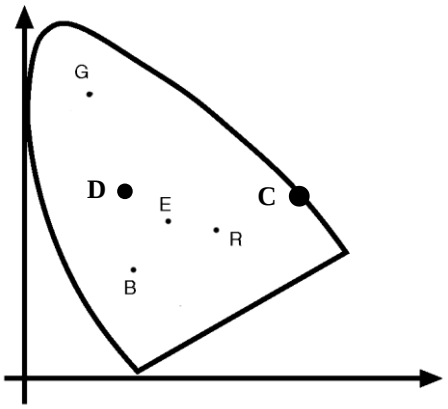
Late Policy: None, unless prior arrangement has been made.

PART 1: Theory Questions for practice, no need to submit, answers will be posted later.

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 (*140 points*)

This assignment will help you gain a practical understanding of color representations, color spaces and data structures normally used in color analysis. Detecting and localizing objects in an image is a challenging computational task. The overall problem space is complex and robust solutions in practical scenarios need many computational descriptors and vectors. In this assignment we will explore using color as one such data point to accomplish this. While color representation is only one vector (among others) to help address this problem, we are hopeful that you get a good understanding of color representation and color spaces which is the intent of this assignment.

You will be given a small library of object images where each image represents an object. All these images will be in a standard format representing the picture of one colorful object placed against a green background. You may process these images to make color representations of objects. You will be given an input image which contains a combination of objects some of which will be in your library. You need to display your processed output where you need to outline using a bounding box around the area of pixels that match any object detected from the library. You are free to extend the display code sample given of assignment 1 to implement this project. No external libraries or scripting environments are allowed.

You program will take a few parameters.

- InputImage.rgb – This is the input image that contains a one of more of colorful objects some of which (or none of which) will be in your library.
- A list of objects object1.rgb object2.rgb …objectn.rgb which will represent your library of objects.

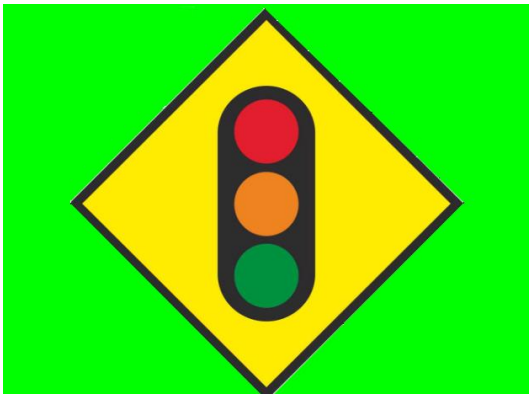
All the images will be of the same size 640x480.

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

YourProgram.exe InputImage.rgb object1.rgb object2.rgb …. objectn.rgb

where the number of objects (*objectn*.rgb) may vary between 1 to *n*. Here is an example data set of the input image and library images with an expected output.

Object1.rgb



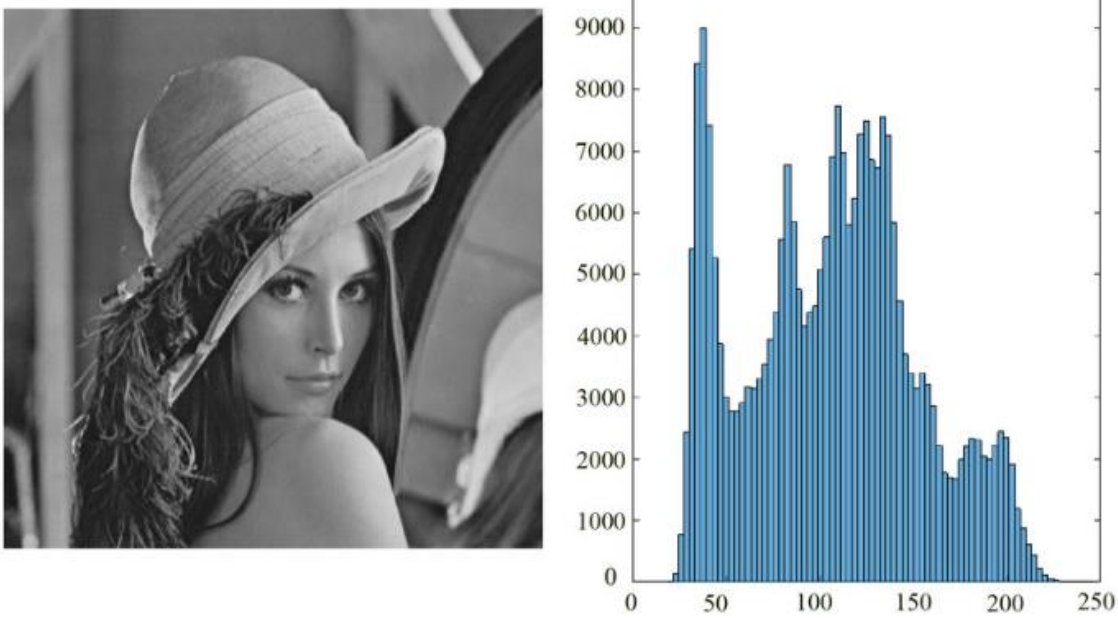
Object2.rgb

Inputimage.rgb

Processed output by program

Process and Execution:

The main data structure you will be using is a color histogram. A histogram is a representation of the distribution of data and the frequency of each type of data. For your understanding, given a gray scale single channel image, its associated histogram is illustrated by the example below, where the X axis describes all possible values of a pixel 0-255 and the Y axis gives the number of times the pixel value occurs in the image.



A color representation has three channels R, G and B. You may create three one dimensional histograms, one for each channel, or alternatively one three-dimensional histogram. Now that you have understood the histogram representation lets see how to make use of the this for detecting and localizing objects.

1. Creating histograms for object images.

The input parameters to your program has a list of objects (from a library). All these object images are given to you against a constant green chroma background. The green chroma background is given so you can conveniently choose pixels that belong to the object. Create multiple histograms for each object image given on the command line list.

2. Creating a histogram of the input image

Given the input image, you may assume that it contains object(s). Create a histogram of the image. At this point you should make note of the fact that the position/orientation/scale of an object in the input image will be very different from its proportions in the object image. This will result in the number of pixels that represent the object in the two images be vastly different, though there is a pattern to the distribution which will be similar in both histograms.

3. Object detection using histogram matching

Given object histograms and the input image histogram, implement a method that will allow you detect the presence of an object histogram in the input image histogram comparing all the channels. Given that color is the main purpose of comparison, you may want to implement this matching algorithm in a space that ignores luminance eg

- Choice 1 – Convent the image from RGB to YUV, and create histograms for U and V. Here you may ignore Y and compare the U&V histograms between the input image and object images. You can read about the color conversion matrix from the class lecture slides.
- Choice 2 - Convent the image from RGB to HSV, and create histograms for H. Here you may ignore S & V and compare the H (hue) histogram between the input image and library image objects. You can read about the color conversion here: https://en.wikipedia.org/wiki/HSL_and_HSV & https://www.cs.rit.edu/~ncs/color/

4. Object Localization

Once you have detected the presence of the object by comparing histograms, you need to highlight the pixels in the image that correspond to the object either by highlighting them or showing a bounding box around them. To achieve this you need to keep track of [x,y] locations of pixels when forming the histogram.

What should you submit?

- Your source code files, and your project file or makefile, if any – all combined into a zip archive with your name on the archive. ***Please do not submit any binaries or images***. We will compile your program and execute our tests accordingly.
- If you need to include a readme.txt file with any special instructions on compilation, that is fine too.

576_pictures - Google Drive