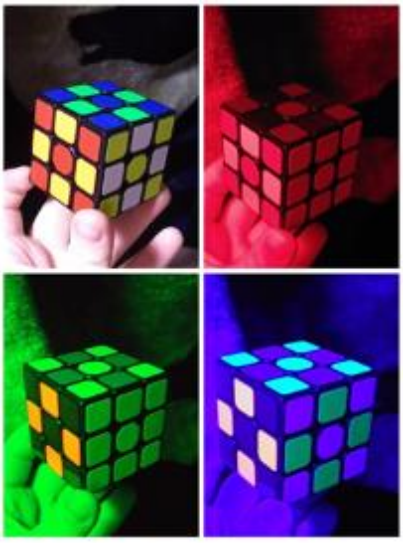
CS 576 – Assignment 2
Instructor: Parag Havaldar

Assigned on Mon 09/19/22,
Solutions due on Monday 10/10/22 by 4:00 pm afternoon

Theory Part (40 points)

Question1: Color Theory – 10 points (courtesy TA Jared Hwang)
A Rubik's cube is a cube-shaped puzzle with 6 different 3x3 colored tiled sides: white, green, red, blue, orange, and yellow. The goal of the puzzle is to rotate sides and make each face have 3x3 tiles of the same color. When held under different colored lights (white, red, green, blue) the cube looks very interesting and vivid, see below:



- Explain why this happened. Why do some tiles look bright, almost glowing, while others appear muted and devoid of their original color? *(4 points)*
- Assuming ideal conditions, you have the following lighting conditions to solve the puzzle – under pure yellow light or under red light. Which of these two light choices make it harder to solve? Give reasons for your choice of answer. *(6 points)*

Question 2: Color Theory (*10 points*)

- The chromaticity diagram in (*x*, *y*) represents the normalized color matching functions X, Y and Z. Prove that *(2 points)*

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

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? *(2 points)*
- 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? *(2 points)*
- Can you suggest improvements rather than just choosing the nearest color? *(4 points)*

Question 3: Entropy Coding *10 points* –
Consider a communication system that gives out only two symbols X and Y. Assume that the parameterization followed by the probabilities are $P(X) = x^k$ and $P(Y) = (1-x^k)$

- Write down the entropy function and plot it as a function of x for k=2. (*1 points)*
- From your plot, for what value of x with k=2 does H become a minimum? (*1 points)*
- Your plot visually gives you the minimum value of x for k=2, find out a generalized formula for x in terms of k for which H is a minimum *(3 points)*.
- From your plot, for what value of x with k=2 does H become a maximum? (*1 points)*
- Your plot visually gives you the maximum value of x for k=2, find out a generalized formula for x in terms of k for which H is a maximum *(4 points)*.

Question 4: Huffman Coding/Entropy *10 points – (courtesy to TA Jared Hwang)*
Bob has a pen pal, Alice, who has been learning about information theory and compression techniques. Alice decides from now on that they should exchange letters as encoded signals so they can save on ink. The following is a letter that Alice sends to Bob on her trip to Paris:

> Dear Bob,
> *Hello from Paris!*
> *I got this postcard from the Louvre. You would love Paris! I hope to hear from you.*
> Alice

- Find and show a Huffman code for the body of Alice's postcard (i.e. exclude "Dear Bob" and "Alice"). Treat each word as a symbol, and don't include punctuation. What is the average code length? (*3 points*)

Bob, having just learned about the telegram in history class, suggests to Alice that they can try writing their letters as telegram messages to shorten them even more. He sends Alice what her postcard might look like as a telegram:

IN PARIS POSTCARD FROM LOUVRE *STOP*
YOU WOULD LOVE *STOP*
HOPE HEAR FROM YOU *STOP*

- Find a Huffman code for the telegram message. What is the average code length? How does it compare to the original letter? (*3 points)*
- Which version of the message, postcard, or telegram, contains more information? Show quantitatively and explain qualitatively where the difference (if any) comes from. (*4 points)*

Programming Part (*160 points*)

This assignment will help you gain a practical understanding of analyzing color channels especially as it pertains to image segmentation. While image segmentation is a challenging task in the field of image processing and computer vision, the process has been made simpler via the use of green screen and chroma keying techniques. I am sure you are all too familiar with online video conferencing applications such as zoom, webex where you can change your background with or without a green screen. Here you will implement similar functionality and hopefully get an opportunity to explore color and color spaces.

You will be given two input videos in the same rgb format – each video will be a 640x480 video that plays at 24 fps for a total of 20 seconds (480 frames). The frames are named with a base name and indexed by frame number eg basename.0001.rgb, basename.0002.rgb … basename.0600.rgb. You are free to use extend the display code sample given of assignment 1 to display a sequence of images at the frame rate of display and implement the color algorithms needed in this video. (*no scripting languages such as MATLAB or python please!*).

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

YourProgram.exe C:/myDir/foreGroundVideo C:/myDir/backGroundVideo mode

Where,

- *foreGroundVideo* is the base name for a green screened foreground video, which has a foreground element (actor, object) captured in front of a green screen.
- *backGroundVideo* is any normal video
- *mode* is a mode that can take values 1 or 0. 1 indicating that the foreground video has a green screen, and 0 indicating there is no green screen.

Implementation details for mode 1:
In this mode you have a green screen used in the foreground video. While the color of screen used is intended to be constant, practically it never is and has slight variations that come across in the captured video. While a specific color might have been used, the actual RGB pixel values of the screen can vary depending on lighting conditions, shadows cast, noise and quantization in the capture process etc. Normally thresholds may be used to decide how to detect green screen pixels. In your implemented you need to arrive at these thresholds by analyzing your video.
For all frames write a process that will detect the green screen pixels in the foreground video and replace them with the corresponding background video pixels in all the frames.

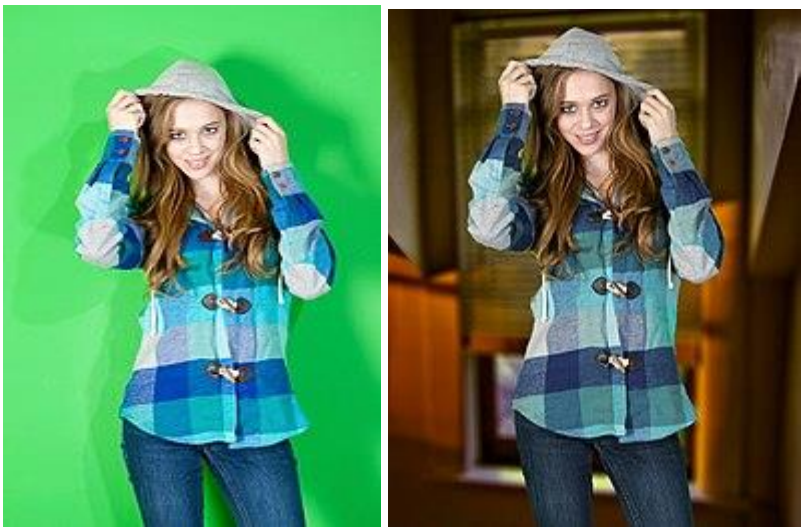


Image taken from https://en.wikipedia.org/wiki/Chroma_key

Some thoughts that you might want to consider

- How do you detect the thresholds to label a pixel as a green screen pixels given that the screen pixels might have noise, shadows etc.
- You can certainly make this work by processing in the RGB color space, but other color spaces might work better like HSV. Here are references that were shared in the class and should give you an understanding of these spaces.
  https://en.wikipedia.org/wiki/HSL_and_HSV
  https://www.cs.rit.edu/~ncs/color/
- To have good quality at the boundaries of the composition (where foreground regions and background video meet), can you think how you to blend boundary pixels correctly, so that the foreground element does not appear stuck on a background but blends in nicely

Implementation details for mode 0:
In this mode your foreground video does not have any constant colored green screen and while this is a hard problem to find automatically, the foreground videos we give you will have the foreground element (actor, object) *moving in every frame* while *the camera is static*. In other words, you should be able to arrive at your green screen pixels by comparing two frames where – pixels that are constant (not changing within some

threshold) can be assumed to be “green screen” pixels and hence can be replaced by the corresponding pixels in the background video. This algorithm is known as ***background subtraction***.

For example, shown below are two frames from a *static* camera. Comparing corresponding pixels in frame1 and frame2 (for each x,y), you should be able to assess pixels that have “not changed” and hence can serve as “green screen” pixels. Also, pixels that have changed and hence are foreground pixels. You may then proceed to composite the other video’s corresponding frame with this extracted green screen. ***Note*** – while the camera may be static, under changing conditions of lighting, motion etc, you might not get “perfect” results, especially in at the boundaries of the areas in motion. Also, if there is no motion, then you will not be able to extract this foreground.


Frame 1


Frame 2


Foreground

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 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.