Color Image Processing

Digital Image Processing

by Trần Minh Hiếu, Nguyễn Gia Phong, Nguyễn Văn Tùng,
Nguyễn An Thiết and Nguyễn Thành Vinh

July 8, 2020

Contents

1 Introduction

1.1 Brief Description

Color images existed long before the rise of computing and digital image processing. While most techniques of monochrome image processing such as blur and edge detection can be directly applied to color images, others require modification. Furthermore, there exists procedures specific only to color images. In this project, we try to implement some of these techniques and note down our findings.

1.2 Authors and Credits

The work has been undertaken by group number 8, whose members are listed in the following table.

Full name	Student ID
Trần Minh Hiếu	BI9-101
Nguyễn Gia Phong	BI9-184
Nguyễn Văn Tùng	BI9-229
Nguyễn An Thiết	BI8-174
Nguyễn Thành Vinh	BI8-187

We would like to express our special thanks to Dr. Nghiêm Thị Phương, whose lectures gave us basic understanding on the key principles of digital image processing. The color image processing lecture notes from the UMSL's CS 5420 course [1] also help us gain initial intuition on the matter.

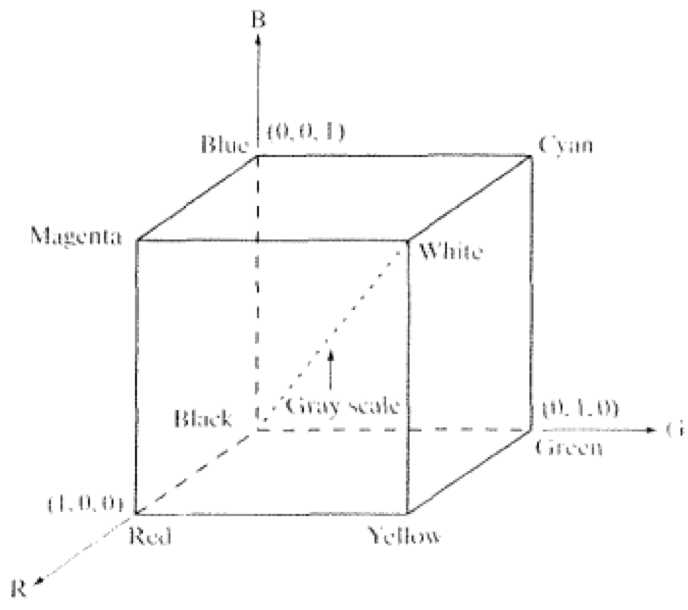
*`https://github.com/McSinyx/recipe`

2 Color Spaces

The color spaces in image processing aim to facilitate the specifications of colors in some standard way. Different types of color spaces are used in multiple fields like in hardware, in multiple applications of creating animation, etc.

2.1 The RGB Color Model

In the RGB model each color appears in its primary spectral components of red, green and blue. The color subspace if interest is the cube shown in the figure below. In which RGB values are at three corners:cyan, magenta and yellow are at three other corner: black is at the origin and white is at the corner farthest from the origin. In this model, the gray scale(point of equal RGB values) extends from black to white along the line joining these two point. The different colors in this model are point on or inside the cube and are defined by vectors extending from the origin. For convenience the assumption that all color values have been normalized so that the cube shown in the figure is the unit cube. That is all values R, G and B are assumed to be in the range [0,1]



2.2 The CMY Color Model

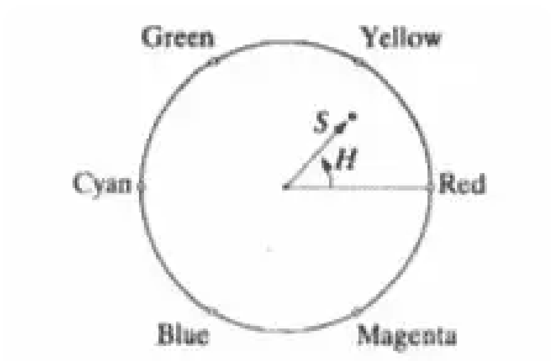
Cyan, magenta and yellow are the secondary colors of light. That is cyan subtracts red light from reflected white light, which itself is composed of equal amounts of red, green and blue light. Most devices that deposit colored pigments on paper such as color printers and copies, require CMY data input or perform an RGB to CMY conversion internally. This conversion is performed using this simple operation:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2.3 The HSI Color Model

As we have seen, creating colors in the RGB and CMY models and changing from one model to the other is a straightforward process. The RGB system matches nicely with the fact that the human eye is strongly perceptive to red, green and blue primaries. Unfortunately, the RGB, CMY and other similar color models are not well suited for describing colors in terms that are practical for human interpretation.

When humans view a color object we describe it by its hue, saturation and brightness while hue is a color attribute that describes a pure color. Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion ò intensity and is one of the key factors in describing color sensation.The HSI(hue, saturation, intensity) color model decouples the intensity component from the color carrying information in a color image. The HSI model is an ideal tool for developing image processing algorithms based on color descriptions that are natural and intuitive to humans, who after all are the developers and users of these algorithms.



The angle from the red axis gives the hue, the length of the vector is the saturation and the intensity is given by the position of the plane on the vertical intensity axis

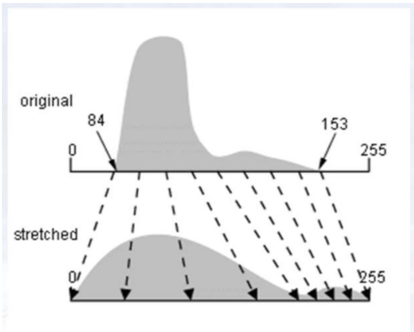
3 Color Image Enhancements

3.1 Contrast Stretching

Contrast Stretching is a linear image enhancement technique that tries to improve the contrast by stretching the intensity values of an image to fill the entire dynamic range.

An image of low contrast has a small difference between its dark and light pixel values. The histogram of a low contrast image is usually bends to the left (mostly light) or to the right (mostly dark), or located around the middle (mostly gray). Contrast Stretching computes the highest and the lowest pixel intensity values, sets them to 255 and 0 respectively, and scales all other pixel intensities accordingly.

$$\text{contrast} = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$



Both Contrast Stretching or Histogram Equalization can be used to adjust image intensities to enhance contrast of pictures. While a non-linear Histogram Equalization technique is more reliable, in contrast stretching, there exists a one-to-one relationship of the intensity values between the source image and the target image i.e, the original image can be restored from the contrast-stretched image. This cannot be done with Histogram Equalization.

3.2 Component Stretching

3.3 Local Contrast Enhacement

4 Pseudo Color Rendering

By mapping each intensity level to a color, one may derive a pseudo color image from a greyscale images. Typical usage of such technique is in thermal imaging, elevation and medical imaging to help the human visual system pick out detail, estimate quantitative values, and notice patterns in data in a more intuitive fashion [2].

In OpenCV, this is available under `applyColorMap`. It is also trival to reimplement this function only using NumPy:

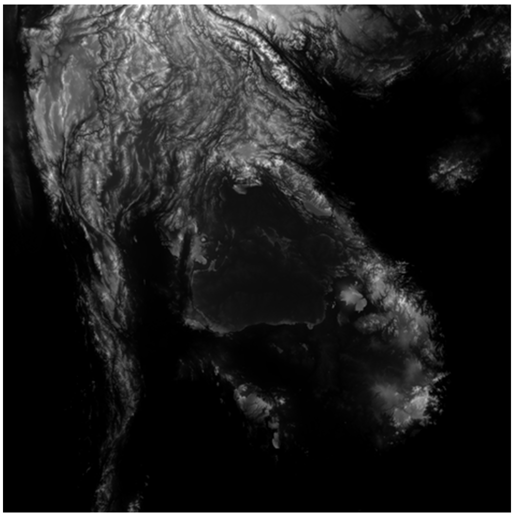
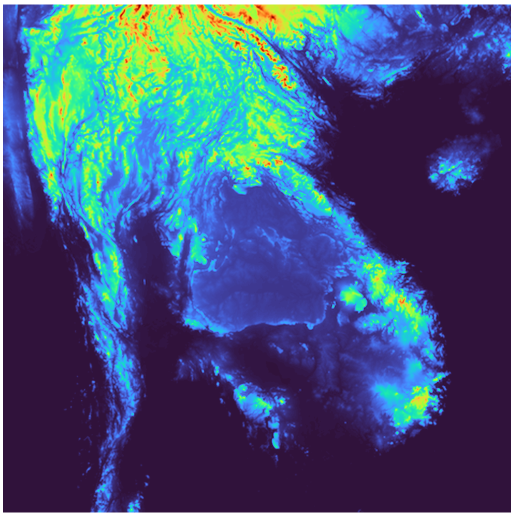
```
import numpy as np

def map_color(grey, mapping):
    r = np.vectorize(lambda i: mapping[i][0])
    g = np.vectorize(lambda i: mapping[i][1])
    b = np.vectorize(lambda i: mapping[i][2])
    # OpenCV uses BGR by default for whatever reason.
    return np.stack((b(grey), g(grey), r(grey)), axis=-1)
```

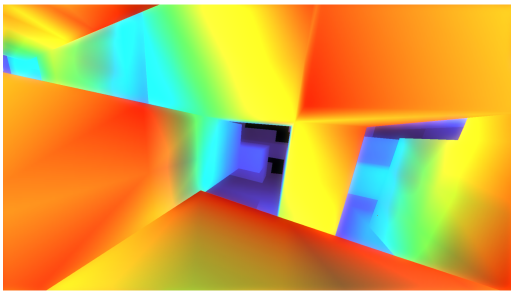
For demonstration, we are going to use the Turbo colormap [2]. We initially considered solely changing the hue based on intensity, which is also known as the rainbow map, however it is not a visually intuitive mapping [3].

Firstly, we tried to apply the mapping on the heightmap of mainland Vietnam and its neighboring regions[†]. As seen in the side-by-side comparison, the colormapped image allows the human eyes to notice more details, especially the high mountains in the north and the Khorat Plateau (center of the image).

[†]The original images are taken from heightmapper: `https://tangrams.github.io/heightmapper/`



In addition, we experimented with pseudo lighting, that is, use colormaps in place of normal greyscale lighting. The experiment was carried out on Phong's game Axuy, where colormapping proved to be an enhancement on helping players detecting shooting range (Axuy is a first person shooter game). The video where the game is in action is available on YouTube[‡].



5 Conclusion

6 References

[1] Sanjiv K. Bhatia. "Color Image Processing". *CS 5420: Digital Image Processing*. University Of Missouri—St. Louis, Fall 2018.

[2] Anton Mikhailov. *Turbo, An Improved Rainbow Colormap for Visualization*. Google AI Blog, August 20, 2019.

[‡] `https://www.youtube.com/watch?v=QVGAaoordpk`

[3] Noeska Smit. *Rainbow Colormaps—What are they good for? Absolutely nothing!* medvis.org, August 21, 2012.