Introduction

Simplifying a color image to image with less complexity is one of the important base technique in the field of computer graphics.

For example, binarization of color image is often used in feature detection algorithms such as edge detection. Also, this technique is commonly used as a pre-processing algorithm in many Optical Character Recognition (OCR) algorithms.

In this posting, I will be introducing a method of converting a color image to a gray-scale image by transforming the image from RGB color space to the YCbCr color space. The converted gray-scale image will be further processed to a binary image using the threshold method.

The post will be utilizing Matlab for performing the conversion.

Background

Converting the RGB image to a binary image requires the RGB image to be encoding to YCbCr domain. The luminance (Y) channel of the YCbCr color space is often used for representing the image in grayscale.

Pixel values of grayscale image is within range of 0 to 255, with 0 typically taken to be black and 255 taken to be white.

By setting a threshold value as 128, and re-assigning pixels with value in range of 0-127 to 0 and 128-255 to 255, we can further process the grayscale image to a binary image.

Methodology

RGB Color Space to YCbCr Color Space

The RGB color space and YCbCr color space is related with following formula:

Y = ( 75R + 150G + 29B ) / 256

Cb = ( -44R – 87G + 131B ) / 256 + 128

Cr = ( 130R -110G - 21B ) / 256 -128

Where R, G, B represents the R-channel, G-channel, and B-channel of original image respectively.

When we load an image with imread() command, Matlab will represent the image as 3-dimensional array. Once you type whos command, you will find out that first two dimensions are same as the size (width and length) of the image, and each element of the arrays stores the pixel values.

The third dimension, which has value of 3, represents the channels of the image. The first array, stores the intensity value of red channel, next array stores information about green channel, and third array for the blue channel.

Using the code shown in Figure 2, we can separate each of the channel to separate variables R, G, B. These variables can be directly used in the formula presented above to find the values for Y, Cb, Cr channel.

Y Channel for Grayscale image

As it was mentioned in the Background section, the luminance (Y) channel can be used as a grayscale image.

Using the code shown in Figure 3, we can now display the original image in grayscale form. Note that the pixel values cannot contain decimal values, as only whole numbers are used to represent the intensity values. The unit8() function will make sure the pixel values can be represented with 8 bits (0 to 255). The uint8() function rounds off the decimal points, and sets any out-of-bound values to the closest bound number (0 or 255).

Figure 4 below shows the original and grayscale image.

Grayscale to Binary Image: Threshold Method

We can further simplify the grayscale image to a binary image. As analogy, we can think of the binarization like a quantizing an analog signal to a digital signal with single bit.

We will set the threshold as 128, which lies at the middle of the possible value range. Any pixel values in range of [0, 127] will be re-assigned to 0, and the pixels in range of [128, 255] will be re-assigned to 255.

Figure 5 below shows the result of the binarization, as well as the original image and the grayscale image.