Paper- Color2Gray: Salience-Preserving Color Removal

Report By- FNU Akanksha

Introduction:

Grayscale images can be considered as ones that needs to record light intensities using a flat spectral response. The current techniques of do the conversion but they are not preserving the meaningful visual experience, which the viewers expect, rather than the accuracy of the of light intensities. Sometimes while conversion from color to grayscale few colors are represented by the same gray shade. To reduce such losses and to prevent the salient features of the color images after conversion a new algorithm has been introduced which follows three steps explained later in the report.

Scientists have concluded that the human visual system perceives chrominance and luminance values are calculated based on the relative assessments, due to the organization of the cells, and they have observed that the relationships between the nearby pixels are more important as compared to the representation of the pixel values. This led to the construction of new algorithm for conversion and the chrominance distance calculation, which preserves the features of the images.

Background and Related work:

The old methods map dot product from 3-D to 1-D, or requires parameters to be set by trial and error and were ineffective to preserve chrominance difference between the pixels. Contrast enhancement techniques increases the dynamic range of the image but this leads to mapping to same gray value. Isoluminant changes are not preserved with traditional color to grayscale conversion. So, in the algorithm used here explores both linear as well as non-linear dimension using the reduction techniques and thus produces better results for most of the images.

Algorithm:

This algorithm has 3 steps-

-Convert color images to a perceptually uniform color space (CIE $L * a * b *$),

-Use chrominance and luminance differences to create grayscale target difference to create grayscale target differences between nearby pixels,

If, $\text{crunch}(x) = \alpha * \tanh(x/\alpha)$

And $v\theta = (\cos\theta, \sin\theta)$

then: $\delta(\alpha, \theta)ij = \Delta Lij$ if $|\Delta Lij| > \text{crunch}(||\Delta Cij||)$

$\text{crunch}(||\Delta Cij||)$if $\Delta Cij \cdot v\theta \geq 0$

$\text{crunch}(-||\Delta Cij||)$ otherwise.

-and solve the optimization problem designed to selectively modulate the grayscale representation as a function of the chroma variation of the source image.

$f(g) = \text{Summation } ((i,j)\in K) \text{ of } ((gi - gj) - \delta ij)^2$

The Color2Gray algorithm allows users to control the mapping of color differences to grayscale differences via three simple parameters:

θ: controls whether chromatic differences are mapped to increases or decreases in luminance value.

With angles between 90-270 the images features become darker as compared to 90-45 where it is brighter as shown in figure 3.

α: determines how much chromatic variation can change the source luminance value

Increasing alpha increases the contribution from chrominance differences but may cause dynamic range mapping problems

μ: sets the neighborhood size used for chrominance estimation and luminance gradients.

Implementation:

There is a **main** function which reads the input image using the **imread** and **rgb2lab** functions of the color class which is converted to the **gray_matrix**, and then calls a function to generate the delta matrix using the input generated using **rgb2lab** function of the color class. The delta is generated by considering the neighborhood of each pixel. Then we use this delta matrix to do the optimizations and generate the output which is saved it the same directory as the input directory.

Functions used for generating delta are – create_delta, neighbor_pixels, crunch, and dot_prod.

For optimization- do_optimization.

It requires- python libraries- scipy, numpy and skimage to be preinstalled before running.

To run the code- python color2gray.py - - input picture_name (without the png extension)

Performance and Results:

Color2Gray algorithm is better than Photoshop's grayscale mode on a wide variety of images. The Color2Gray algorithm does not provide large improvements for scenes with high dynamic range, especially natural scenes, which typically have a wide range of luminance changes but improve any image that contains large isoluminant regions with a small number of different chrominance values. It does not perform better than the traditional grayscale representation when trying to map widely spread chrominance gradations to something independently distinguishable.

Results obtained are as shown in below figures.

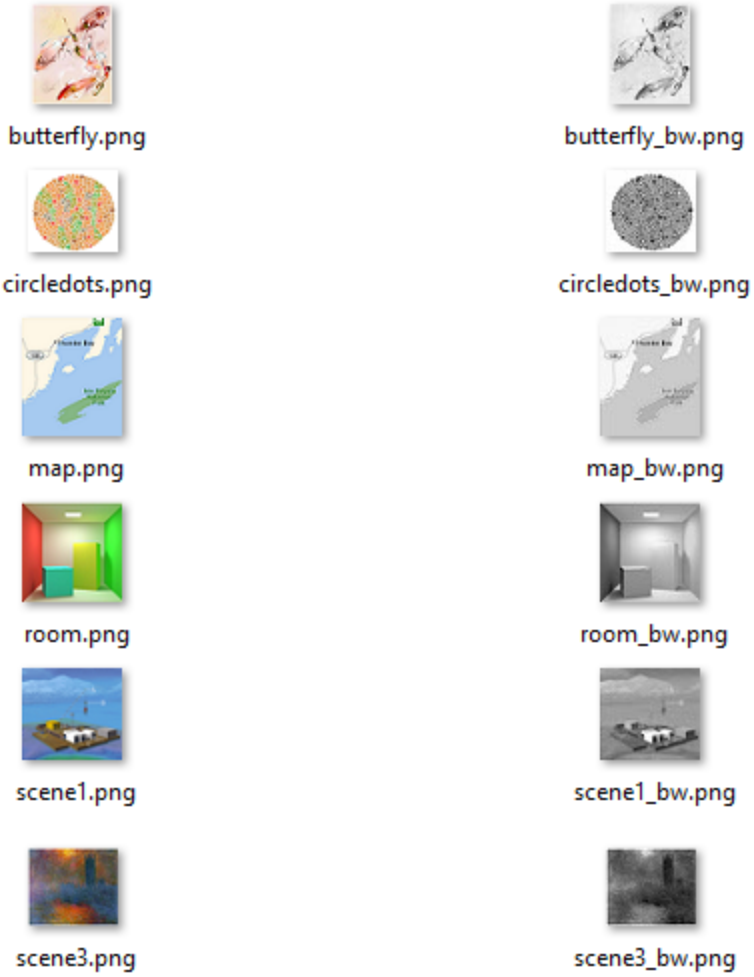

Figure 2: Color to Gray output images.

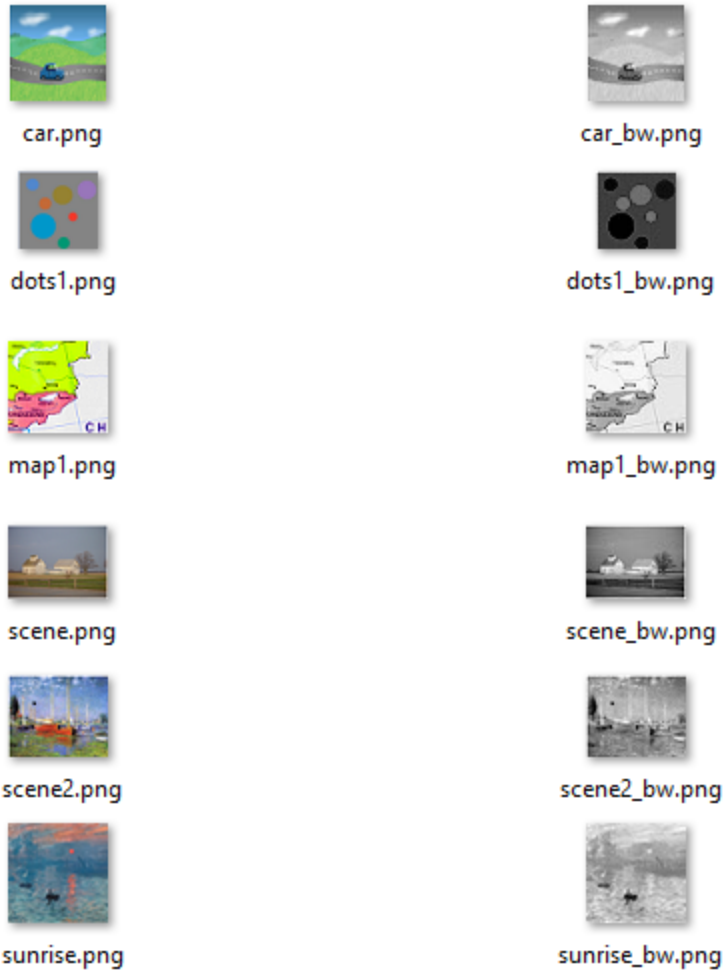

Figure 2 - Output with the color to gray code

Figure 3: Shows the difference on varying theta value as can be seen, there is difference in the contrast values the sun is brighter from 90 – 305 and darker from 135-270. Alpha is kept 8 and mu as 1.

Future Work and Summary:

A system which takes input as the color images and converts it into the gray scale image which preserves the salient features of the images, by using the mentioned algorithm. The results of the code using this algorithm are much better for the viewers and help viewers get the clarity of the features in grayscale, to the same level as the color image.

There are few limitations of the code I have written like it can only convert 50*50 image right now as I am using dense matrix instead of the sparse matrix so in future I want to scale my code to work for large images as well. Also, it takes only .png inputs, so in future I would like to use the code for other extensions as well. Also, the code takes lot of time to run, so need to optimize it in future. Also, I want to try to reconvert the images back to the color from the grayscale images, so will try to implement it in future.

References:

http://www.cs.northwestern.edu/~ago820/color2gray/color2gray.pdf