Portland State
UNIVERSITY

Maseeh College of Engineering & Computer Science

COMPUTER SCIENCE DEPARTMENT

COMPUTATIONAL PHOTOGRAPHY(CS510)

Color2Gray : Salience-Preserving Color Removal

Supervised By:

Dr. Feng Liu

Associate Professor
Department of Computer Science

Maseeh College of Engineering and Computer Science

Presented By:

Rahul Kumar

MS Computer Science

ACKNOWLEDGEMENT

I would like to express our deep gratitude to my supervisor, **Dr. Feng Liu** for his effort and encouragement throughout the preparation of this project. His Knowledge and endless support was a great asset from which I learned plenty. He was very enthusiastic about the project, which gave me a motive to work harder and harder.

ABSTRACT

A Grayscale image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel.

The grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive gray levels is significantly better than the gray level resolving power of the human eye.

When Color Image are converted to Grayscale, important image feature get visually lost. In this project, I am implementing a new Algorithm which reduces such losses and preserve the salience of the image. This new Algorithm has been taken from the paper "**Color2Gray: Salience-Preserving Color Removal**" written by **Amy A. Gooch and his team**.

TABLE OF CONTENTS

1.Introduction

As a Viewer, we want digital image to preserve a meaning visual experience in grayscale conversion also. While converting color image to grayscale important visual features get lost. The algorithms used in project preserves such losses.

Fig.1. A color image (Left) often reveals important visual details missing from a luminance-only image (Right).

On applying New Algorithms:

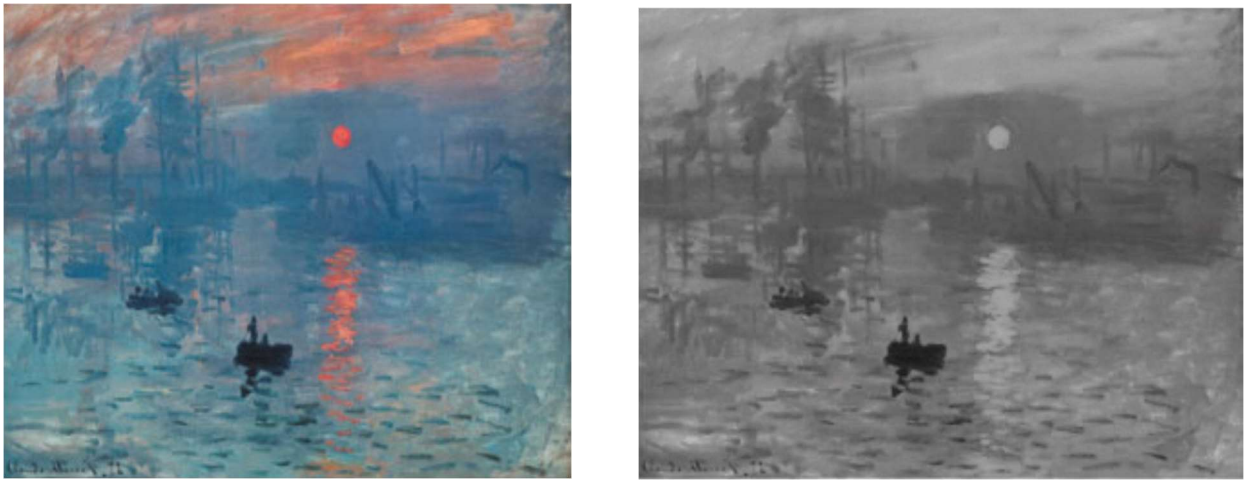


Fig.2. Color2Gray algorithm maps visible color changes(Left) to grayscale changes (Right).

2.Previous Work

- Earlier method for converting RGB images to grayscale used dot products or weighted sums to map a three dimensional color space to single dimension.

- Other method used by program like Adobe Photoshop devised custom non-linear projections and required users to set image-dependent parameters by trial and error.

- But these methods were ineffective to preserve luminance and chrominance.

(a) CIECAM97 Lum

(b) L*a*b* Lum.

(c) XYZ Lum.

(d) YCrCb Lum.

(e) Auto Contrast

(f) Color2Gray

3. Color2Gray Algorithm:

The Color2Gray Algorithm is a 3 Step process:

1) Convert RGB inputs to a perceptually uniform CIE $L^*a^*b^*$ color space.

2)Compute target difference between nearby image pixels using chrominance and luminance differences.

3)Use a squares optimization technique to selectively modulate the source luminance differences in order to reflect changes in the source image's chrominance.

3.1 Parameters:

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

i) φ: divides the chrominance plane and determines whether a chromatic difference will darken or lighten the source luminance difference.

ii) α: controls the amount of chromatic variation applied to the source luminance values.

iii) μ: controls the size of the neighborhood, indicating whether a user is more interested in preserving local or global changes.

Original Image

Grayscale Image

Color2Gray Image

-Here Parameters are φ=180; α=8; μ=entire image.

3.2 Computing Target Difference

-The color differences between pixels in the color image is then expressed as a set of signed scalar values and then build a grayscale version of image with those values.

-For each pixel *i* and neighbor pixel *j*, we find a signed distance scalar δ_{ij} based on chrominance and luminance difference between *i* and *j*.

$$\begin{aligned} \text{Given:} \quad & \text{crunch}(x) = \alpha * \tanh(x/\alpha) \\ & \overrightarrow{v_\theta} = (\cos\theta, \sin\theta) \end{aligned}$$

then:

$$\delta(\alpha,\theta)_{ij} = \begin{cases} \Delta L_{ij} & \text{if } |\Delta L_{ij}| > \text{crunch}(||\overrightarrow{\Delta C_{ij}}||) \\ \text{crunch}(||\overrightarrow{\Delta C_{ij}}||) & \text{if } \overrightarrow{\Delta C_{ij}} \cdot \overrightarrow{v_\theta} \geq 0 \\ \text{crunch}(-||\overrightarrow{\Delta C_{ij}}||) & \text{otherwise.} \end{cases}$$

-

crunch() : a function that allows us to compress large-amplitude values into the valid range.

ΔL_{ij} : Luminance Difference

ΔC_{ij} : Chrominance Difference

Since ΔL_{ij} is a scalar and ΔC_{ij} is a 2D vector, we first map ΔC_{ij} onto a single dimension using the Euclidean norm, $||\Delta C_{ij}||$.

3.3 Solving the optimization :

- Using above values , value for grayscale image *g* is derived between pixel *i* and a neighboring pixel *j* as ($g_i - g_j$).

$$f(g) = \sum_{(i,j)\in\mathcal{K}} ((g_i - g_j) - \delta_{ij})^2.$$

- Grayscale image g minimizes the given function *f(g)* in above equation
- Here, *K* is set of ordered pixel pair*(I,j).*

4.Implementation

- Following are the tools and technology I have used for the implementation of Color2Gray Algorithm in my project.

i) Code Blocks 13.12 used as IDE.

ii) C++

iii) OpenCV 2.4.9

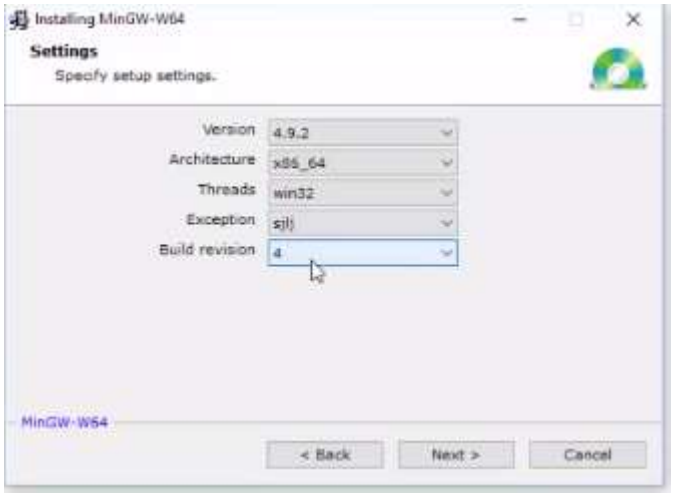
iv) MinGW 64bit

v) CMake 3.9.0

Setting the project workspace:

i) Download and install MinGW 64 bit. : It is a c++ compiler used to integrate in Code Blocks.

It requires following setup during installation.



ii) Download and install CMake 3.9.0 : it is used to build OpenCV library in the System.
iii) Download and install Code Blocks 13.12 : it is opensource IDE and lighter than Visual Studio.
iv) Download and install OpenCV 2.4.9 .
v) Set the system path for bin folder of MinGW
vi) Configure CMake with OpenCV source and build folder.
vii) Set the system path for bin folder of OpenCV.
viii) Configure compiler and linker settings of CodeBlocks with OpenCV libraries.
ix) Open CodeBlocks IDE and create a new image project.

-In this project I am loading an image file and converting into color space and then calculating the color differences between pixels in the color image as a set of signed scalar values.

-I have used some of the following functions in my project to perform different steps:

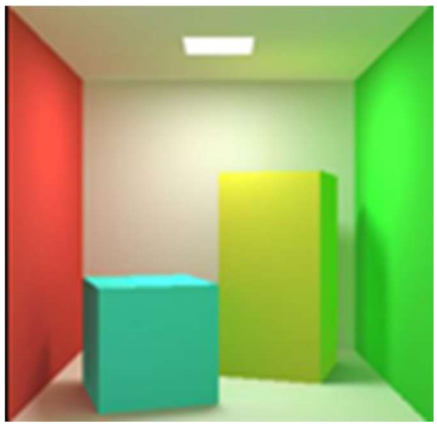
void ColorImage::load(const char * fname) : it read the image file mentioned in the main function.

*readPPM() : this function reads the image using magic number and also reads the rows ,column and color depths .

calc_d() : it is used to calculate the color differences between pixels.

r_solve() : deriving the value for grayscale image *g* and solve the optimization.

5. Results of Color2Gray :



Input Image

Color2Gray

Input Image

Color2Gray Image

6. Color2Gray+Chrominance:

-It makes color images more comprehensible .

-Add the source image's chrominance channels to Color2gray results and then convert image into RGB.

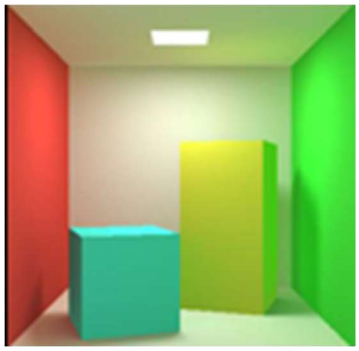
-Adding color to Color2Gray result provides visual adjustments made to Color2Gray Algorithms.

7.Results of Color2Gray+Color :





Input Color2Gray Color2Gray+Color





Input Color2Gray Color2Gray+Color

8.Conclusion & Future Works:

- The extension of this algorithm can be removal of dependency φ.

-we can use an optimization method to match both signed and unsigned difference.

-we can implement multiscale methods which would allow faster solutions on higher resolutions image.

References:

- Color2Gray: Salience-Preserving Color Removal ,Amy A. Gooch,Sven C. Olsen Jack Tumblin, Bruce Gooch, Northwestern University

-Heckbert, P. S. 1982. Color Image Quantization for Frame Buffer Display. In Proceedings of SIGGRAPH, 297–307.

- Levin, A., Lischinski, D., and Weiss, Y. 2004. Colorizationusing optimization. In ACM Transactions on Graphics

- Livingstone, M., Ed. 2002. Vision and Art: The Biology of Seeing. Harry N. Abrams, Inc., Publishers.

- Lotto, R. B., and Purves, D. 1999. The effects of color on brightness. Nature Neurosci. 2, 11, 1010–1014.

- Power, J. L., West, B. S., Stollnitz, E. J., and Salesin, D. H. 1996. Reproducing color images as duotones. In Proceedings of SIGGRAPH, ACM Press, 237–248.

- Rasche, K., Geist, R., and Westall, J. 2005. Detail preserving reproduction of color images for monochromats and dichromats. IEEE Comput. Graph. Appl. 25, 3.

-Adobe Photoshop, 2004. Photoshop: Converting color images to black and white. http://www.adobe.com/tips/phs8colorbw/ main.html.

-Brown, R., 2004. Photoshop tips: Seeing in black & white. http://www.russellbrown.com/tips_tech.html.

-Fattal, R., Lischinski, D., and Werman, M. 2002. Gradient domain high dynamic range compression. ACM Transactions on Graphics 21, 3 (July), 249–256.
-Healey, C. G., and Enns, J. T. 1996. A perceptual colour segmentation algorithm. Tech. Rep. TR-96-09, UBC CS.