



IMAGE COLORIZATION

Pankti Fadia, Jigna Shreasiya and Jyoti Pareek

Department of Computer Science, Gujarat University, Ahmedabad-380009, Gujarat, India.

Email - fpankti@gmail.com

ABSTRACT

Image Colorization is the process of adding colors to monochrome, grayscale or sepia images. There are several different methods of colorization. From that we have studied two algorithms which gives exceptionally promising outcomes. The goal of this paper is to implement existing algorithms to color a grayscale image. so, we are going to implement following algorithms in our paper:

(1) *optimization-based coloring, in which artist have to annotate image using color scribbles.it is proposed by Levin et al. [3]. It is based on simple idea “neighbouring pixels in space-time that have similar intensities should have similar colors”.*

(2) *Pseudo Coloring, a method of assigning arbitrary colors with the help of colormap(Look-Up table).*

Keywords: *Gray scale image, colormap, scribbles, Colorization*

INTRODUCTION

The earliest photo was taken in 1826 by Joseph Nicéphore Niépce which was hazy and black & white. When the artists discovered that it was possible to catch light with cameras, they wanted to harness all the colors associated with it. A part of the tests commenced from the mid-nineteenth century. The term colorization is presented by Wilson Markle in 1970 to depict his idea for coloring black and white pictures or videos using the computer [1]. But Colorization procedure is costly and tedious For an example, to colorize an image an artist first divides the image into several different regions, and afterwards continue to assign color to each region. Often, automatic segmentation algorithms do not recognize fuzzy or difficult region boundaries, such as the boundary between a subject's hair and her ears. Thus, the artist is often left with the undertaking of physically sketching complicated boundaries between regions.

Image colorization is the process of taking an input **Gray scale image** and then producing an output **colorized image** that represents the semantic colors and tones of the input. Objects should have their natural color. for example, Grass must be plausibly “Green” — it can't be colored as “Orange or Pink” by the algorithm. color image contains more information than gray scale image. So, it is more useful to extract information from color image than gray scale image and they are more realistic thus visually appealing to viewers. Gray scale image does not carry information of chromatic values. Basically, color image comprises of three-dimensional information about the color of image: **RED, GREEN, BLUE**. However, gray scale image comprises of **luminance** and hence it is one dimensional. Coloring gray scale images implies addition of this chromatic values to gray image by regeneration of chromatic values for pixels in colors. This regeneration of chromatic values for the gray image is a problem. Since it is difficult to generate precise information of chromatic values.

Converting color image to gray scale image implies we are dropping information about color. It is easy to convert color image to gray scale image but reverse is not easy. Since there can be numerous colors which represents one gray level, but in reverse when we want to color gray level we can't decide which color contributes to corresponding gray level.

1.1 Theoretical Background:

1.1.1 Grayscale Image

Grayscale image solely contains intensity value. Any particular intensity value defines the appearance of a certain pixel in image. In image, lowest intensity value represents black color

and highest intensity value represents white color. All values between this highest and lowest value represent the shades of gray. These different shades are dependent on possible values which a pixel may hold. If a pixel is represented by a bit, then it can hold two values 0 and 1.[6]

1.1.2 Color Image

A color image is represented by some color space. This color space is not dependent on only one value like gray scale image. Each pixel in a color image is represented by more than one value and the combined effect of these values gives the appearance of color. First, understand how our eyes sense color.[6] Our eyes consist of Rods and Cone cells. Rod cells are designed to feel the intensity of light where cone cells are designed to sense the color of light which will fall on the retina of the eye. Cone cells are usually of three types: short, medium, long. Short cones are sensitive to shorter wavelengths of light, which means they sense blue color. Medium cones are sensitive to medium wavelengths of light, which means they sense green color. Whereas long cones are sensitive to longer wavelengths of light, which means they sense red color. Based on this, a color space is designed -RGB.

1.1.3 YUV Color space

The first algorithm works in YUV color space, where Y is the monochromatic luminance channel, which is solely referred to as intensity, while U and V are the chrominance channels, encoding the color.[3]

1.2 Literature Survey:

In Markle's unique colorization process, a color mask was physically painted. Then, to automatically fill the other frames in regions, motion detection and tracking was applied. As for the colors in the locality of the moving edges, colors were assigned using optical flow, which often requires manual fixing by the operator. Earlier, BlackMagic [5], a commercial software for colorizing still images, was used to provide the user with different brushes and color palettes. Yet, the segmentation task was left altogether to the user.

In the following section, we have made an exhaustive study on different image colorization techniques. Efforts done to formulate a solution to this problem by researchers can be categorized below:

1.2.1 Manual coloring

In manual coloring, a huge human exertion is required. One has to color different parts of the image by their perception of the colors of the gray image. One has to select each pixel of the gray image to get the job done. Mostly, image editing software like Paint or Adobe Photoshop are utilized for converting a gray image to a colored image.[6]

Disadvantage: The principle downside of this strategy is that a segmentation task is done completely physically and this process is very tedious and monotonous. And furthermore, it requires human vision of proposed colors of a gray pixel.

1.2.2 Semi-automatic coloring

In semi-automatic coloring, the user provides some colored scribbles and then the color is propagated over the image based on an optimization framework. Though, this method is efficient but it is time-consuming. It is more useful in medical image applications like X-Ray, CT-Scan and MRI images.

Disadvantage: Such methods require users to provide an impressive number of scribbles on the grayscale image, which is time-consuming and requires ability. Additionally, it is practically difficult to add such markings to enormous pictures.

1.2.3 Advance semi-automatic coloring

Unlike manual coloring and semi-automatic coloring, advanced semi-automatic coloring techniques evacuate the human intercession to a huge extent. It is generally founded on a technique proposed by Welsh et al [2]. In this method, a similar reference image is considered and it transfers the color information from a similar reference image to the input grayscale image on the basis of mean intensity and standard deviation calculated on neighbourhood

pixels.

Disadvantage: Feature matching is critical to the quality of the results so; satisfactory results cannot be obtained if feature matching is not performed correctly. Besides, the strategy is exceptionally sensitive to image brightness and contrast, while genuine aeronautical pictures consistently incorporate huge territories of shadow and low contrast.

1.2.4 Fully Advance automatic coloring

An elective approach is to employ a large number of training images, which is a recent example of deep learning. A huge database of color images including wide range of objects is used for training the neural networks. The source image is retrieved automatically from a database of images by some matching procedure. The trained model can then be used to efficiently transfer the color information to the grayscale images. In these method user takes no exertion in source image selection as well as in color transfer process.

Disadvantage: This approach is computationally expensive, and the training is significantly slow.

For pseudo coloring two techniques can be used:

- Intensity Slicing
- Gray level to color transformations

For example, to colorize an image an artist first divide image into several different regions and afterwards continue to assign color to each region. Often, automatic segmentation algorithms don't recognize fuzzy or difficult region boundaries, such as the boundary between a subject's hair and her Ears. Thus, the artist is often left with the undertaking of physically sketching complicated boundaries between regions.

APPROACH

We had seen different methods of coloring a gray scale image. The colorization algorithms we have implemented are as follows:

2.1 Semi-automatic Coloring (Optimization - based Coloring):

Levin et al. [3] tackled an optimization problem that minimizes a quadratic cost function of the difference of color between a pixel and its weighted average neighbourhood. Their proposition comprises of the minimization of a free-parameter cost functional. The algorithm propagates the color information from pixels colored by hand (scribbles) or any software to the rest of the image. These least-squares optimization problem automatically propagates the scribbled colors, thus colorized an image. Levin et al. [3] assumed that neighbouring pixels in image that have similar intensities should have similar colors. This solution is accomplished in the color space YUV.

❖ Steps:

1) Convert both the original gray-scale image and the marked image which is scribbled (color marks) by the user in few pixels, from RGB to YUV color space. The algorithm works in YUV color space where the Y component represents brightness, and U, V components represent the chromatic value of pixel. The algorithm is given as input an intensity value $Y(x; y; t)$ and outputs two color values $U(x; y; t)$ and $V(x; y; t)$. Here letters r and s are used to denote $(x; y; t)$ triplets.

2) From the scribbled and gray-scale images compute the difference image. Figure 2 explains the constraint that two neighbouring pixels r, s should have similar colors if their intensities are similar. Assuming g a gray scale image set the luminance channel of the colored image as $Y = g$ and the chrominance components u and v are computed by minimizing equation (1):

$$J(U) = \sum_r (U(r) - \sum_{s \in N(s)} W_{rs} U(s))^2$$

$$(1) \quad J(V) = \sum_r (V(r) - \sum_{s \in N(s)} W_{rs} V(s))^2$$

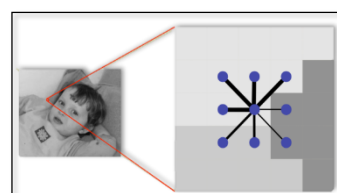


Figure 1 Levin, Colorization using Optimization,[11] (Pixels having similar intensity)

3) Then compute the **weight matrix** W- Figure 3, Which relies upon the similarities in the neighbour intensities for a pixel from the original gray-scale image. **W_{rs}** is a weighting function that sums to one. Its value is large when Y(r) has similar intensity to Y(s), and small when the two intensities are different. Y(r), Y(s) are luminance value of r and s, and **σ_r** represents the variance of the luminance in a window around r.

$$w = \exp\left(\frac{-(Y(r) - Y(s))^2}{2\sigma_r^2}\right)$$

(2)

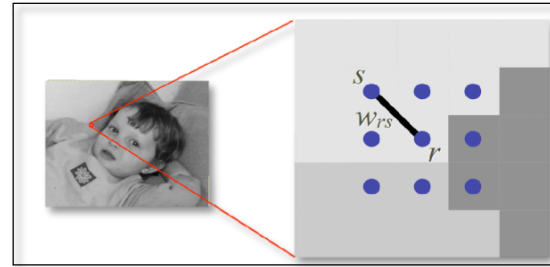


Figure 2 Levin, Colorization using Optimization, [11] (Compute weight matrix based on intensities)

4) The next step is to solve the **system of linear equations**: it assumes that the colour at a pixel U(r) is a linear function of the intensity: **Y(r): U(r) = (ai*Y(r)) +bi** and the linear coefficients ai, bi are the same for all pixels in a small neighbourhood around r.

5) However, the W matrix is going to be very huge and sparse, hence sparse-matrix based implementations are used to obtain an efficient solution.

6) Once W is computed obtain the least-square solution, by computing the pseudo-inverse.

7) Once the solution is obtained in YUV space, it needs to be converted back from YUV to RGB. The following formula is used for conversion.

From RGB to YIQ	
$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$	$= \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$
From YIQ to RGB	
$\begin{bmatrix} R \\ G \\ B \end{bmatrix}$	$= \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$

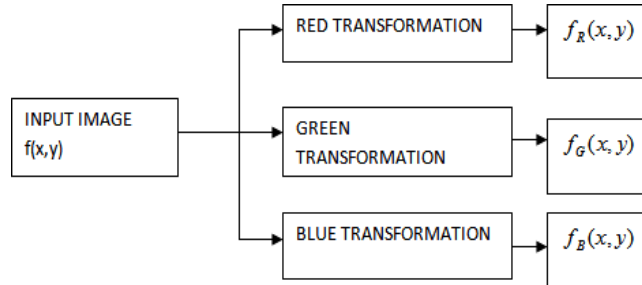
2.2 Pseudo coloring:

Figure 3 YUV to RGB color conversion formula

A pseudo color image is derived from a gray scale image by mapping each intensity value to a color according to a table or function. The idea is to perform three independent transformations (**RED, GREEN, BLUE**) on a particular gray level and map the corresponding intensity value in the image to obtain the result. The result is a composite image whose color content depends on the gray level to color transformations. In some cases, there is no color concept for gray scale image, but we can assign false color to image.

2.2.1 Colormap

Color mapping is a function that maps (transforms) the colors of one (source) image to the colors of another (target) image. A colormap is an m-by-3 matrix of real numbers between 0.0 and 1.0. Each row is an RGB vector that defines one color. The kth row of the colormap defines the kth color. Map the position of the values in colormap to the intensity or the pixel value of the gray scale image and replace it with the corresponding value from the colormap.



2.2.2 The mapping scheme

Figure 4 Three independent transformations

positional mapping: map values at any position in the map are generated only as a function of that position (index). There is no dependency on the properties of the pixel value, only on physical layout.

Steps:

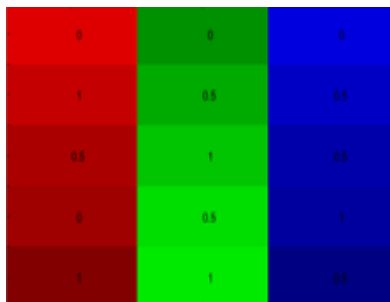
(1) Consider a Grayscale image of size M*N:[13]

4 3 1 2
A= [2 1 5 1]
4 3 2 3

2) Define colormap with 3 columns and number of rows depending on the maximum possible intensity of input image.

3) RED GREEN BLUE

RED GREEN BLUE



0 0 0
F 0 0 1
1 0.5 0.5
My_map= 0.5 1 0.5
I 0 0.5 1 I
[1 1 0.5]

Figure 5A C Johnsy, (2018), Gray Scale to Pseudo color transformation, image processing, [13] (Define Colormap)

(3) Pre allocate an output matrix with zeros of size M*N*3.

0 0 00
0 0 00
0 0 00

$B[:, :, 1] = [0 \ 0 \ 00]$ $B[:, :, 2] = [0 \ 0 \ 00]$ $B[:, :, 3] = [0 \ 0 \ 00]$
 0 0 00
 0 0 00
 0 0 00




Update Matrix

0 0 00 ○
 0.5 0 00 ○
 1 0 00 ○
 $B[:, :, 1] = [0 \ 0 \ 00]$ $B[:, :, 2] = [0 \ 0 \ 00]$ $B[:, :, 3] = [0 \ 0 \ 00]$
 0 0 00
 0 0 00
 0 0 00

(4) Map the position of values in colormap to pixel value of the gray scale image and replace it with corresponding value from color map:

0 0 0
 1 0.5 0.5
 My_map = 0.5 1 0.5
 I 0 0.5 1 I
 [1 1 0.5]



(5) Repeat the process for all pixel values in matrix A, and update the matrix B.

0 0.5 0 1
 0.5 1 0 0.5
 1 0.5 0 0.5
 $B[:, :, 1] = [1 \ 0 \ 1 \ 0]$ $B[:, :, 2] = [0.5 \ 0 \ 1 \ 0]$ $B[:, :, 3] = [0.5 \ 0 \ 0.5 \ 0]$
 0 0.5 1 0.5
 0.5 1 0.5 1
 1 0.5 0.5 0.5

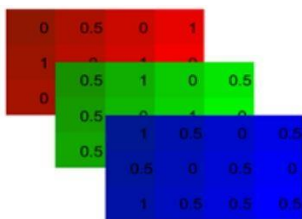


Figure 6 A C Johnsny, (2018), Gray Scale to Pseudo color transformation, image processing, [13] (updated Matrix representing new colormap)

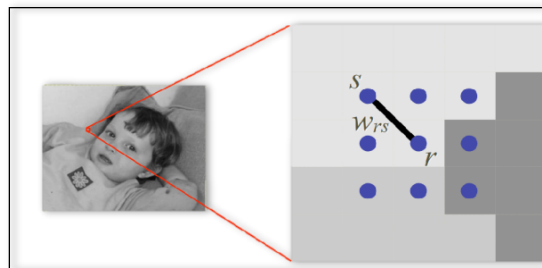
(6) Convert the matrix B into uint8 format and display it.

❖ **Application areas of colorization:**

- Airport System
- Old photos and films
- Satellite Imaging
- Medical Imaging

8) Then compute the **weight matrix W**- Figure 3,Which relies upon the similarities in the neighbour intensities for a pixel from the original gray-scale image. **W_{rs}** is a weighting function that sums to one.Its value is large when Y(r) has similar intensity to Y(s), and small when the two intensities are different. Y(r), Y(s) are luminance value of r and s, and **σ_r** represents the variance of the luminance in a window around r.

$$w = \exp\left(\frac{-(Y(r) - Y(s))^2}{2\sigma_r^2}\right)$$



(2)
Figure 2 Levin, Colorization using Optimization, [11] (Compute weight matrix based on intensities)

9) The next step is to solve the **system of linear equations**: it assumes that the colour at a pixel **U(r)** is a linear function of the intensity: **U(r) = (a_i*Y(r)) + b_i** and the linear coefficients a_i, b_i are the same for all pixels in a small neighbourhood around r.

10) However, the W matrix is going to be very huge and sparse, hence sparse-matrix based implementations are used to obtain an efficient solution.

11) Once W is computed obtain the least-square solution, by computing the pseudo-inverse.

12) Once the solution is obtained in YUV space, it needs to be converted back from YUV to RGB. The following formula is used for conversion.

From RGB to YIQ		
$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$	$=$	$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$
From YIQ to RGB		
$\begin{bmatrix} R \\ G \\ B \end{bmatrix}$	$=$	$\begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$

Figure 3 YUV to RGB color conversion formula

2.2 Pseudo coloring:

A pseudo color image is derived from a gray scale image by mapping each intensity value to a color according to a table or function. The idea is to perform three independent

transformations (**RED, GREEN, BLUE**) on a particular gray level and map the corresponding intensity value in the image to obtain the result. The result is a composite image whose color content depends on the gray level to color transformations. In some cases, there is no color concept for gray scale image, but we can assign false color to image.

2.2.1 Colormap

Color mapping is a function that maps (transforms) the colors of one (source) image to the colors of another (target) image. A colormap is an m-by-3 matrix of real numbers between 0.0 and 1.0. Each row is an RGB vector that defines one color. The kth row of the colormap defines the kth color. Map the position of the values in colormap to the intensity or the pixel value of the gray scale image and replace it with the corresponding value from the colormap.

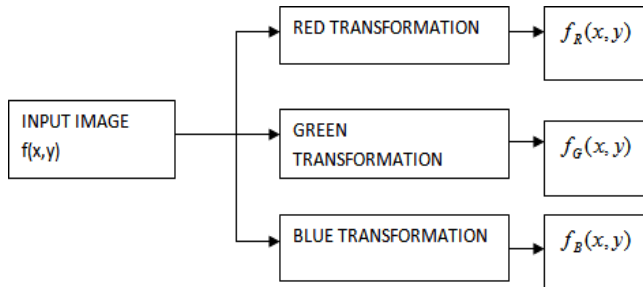


Figure 4 Three independent transformations

2.2.2 The mapping scheme

positional mapping: map values at any position in the map are generated only as a function of that position (index). There is no dependency on the properties of the pixel value, only on physical layout.

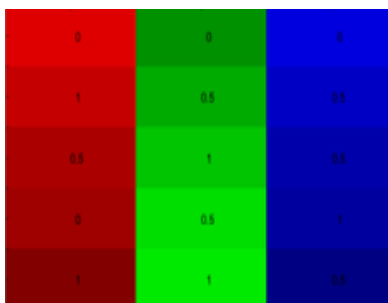
Steps:

(1) Consider a Grayscale image of size M*N:[13]

4 3 1 2
A= [2 1 5 1]
4 3 2 3

4) Define colormap with 3 columns and number of rows depending on the maximum possible intensity of input image.

RED GREEN BLUE



0 0 0
F 0 0 1
1 0.5 0.5
My_map= 0.5 1 0.5
I 0 0.5 1 I
[1 1 0.5]

Figure 5A C Johnsy, (2018), Gray Scale to Pseudo color transformation, image processing, [13] (Define Colormap)

(3) Pre allocate an output matrix with zeros of size M*N*3.

```
0 0 00
0 0 00
0 0 00
B[:, :, 1]=[0 0 00] B[:, :, 2]=[0 0 00]B[:, :, 3]=[0 0 00]
0 0 00
0 0 00
0 0 00
```



Update Matrix

```
0 0 00
0.5 0 00
1 0 00
B[:, :, 1]=[0 0 00] B[:, :, 2]=[ 0 0 00]B[:, :, 3]=[0 0 00]
0 0 00
0 0 00
0 0 00
```

(4) Map the position of values in colormap to pixel value of the gray scale image and replace it with corresponding value from color map:

```
0 0 0
1 0.5 0.5
My_map= 0.5 1 0.5
I 0 0.5 1 I
[ 1 1 0.5 ]
```



(5) Repeat the process for all pixel values in matrix A, and update the matrix B.

```
0 0.5 0 1
0.5 1 0 0.5
1 0.5 0 0.5
B[:, :, 1]=[1 0 1 0 ]B[:, :, 2]=[0.5 0 1 0 ]B[:, :, 3]=[0.5 0 0.5 0 ]
0 0.5 10.5
0.5 1 0.5 1
1 0.5 0.50.5
```

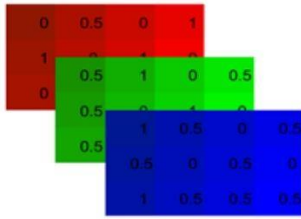


Figure 6 A C Johnsy, (2018), Gray Scale to Pseudo color transformation, image processing, [13] (updated Matrix representing new colormap)

(6) Convert the matrix B into uint8 format and display it.

❖ Application areas of colorization:

- Airport System
- Old photos and films
- Satellite Imaging
- Medical Imaging

DATASET

Images used are royalty free and publicly available. There is no constraint on the size of our input. Initially we have taken 12 gray scale images. These images can be of any categories. Then it is manually marked (scribbled) by using any photoshop editor. One can use any photoshop editing software, but then result of color may get changed based on the whatever type of tools (pencil, marker or pen) is used. In our project we have marked image using the website photopea [10] by the pencil tool. In order to adapt the datasets to the algorithms in Python, a short period of time was spent at the beginning of the project to make sure that every image has the correct format of BMP as well as the right name. These changes lead to a smooth running of the tests. For pseudo color method we have taken two classes: Satellite and Medical images.



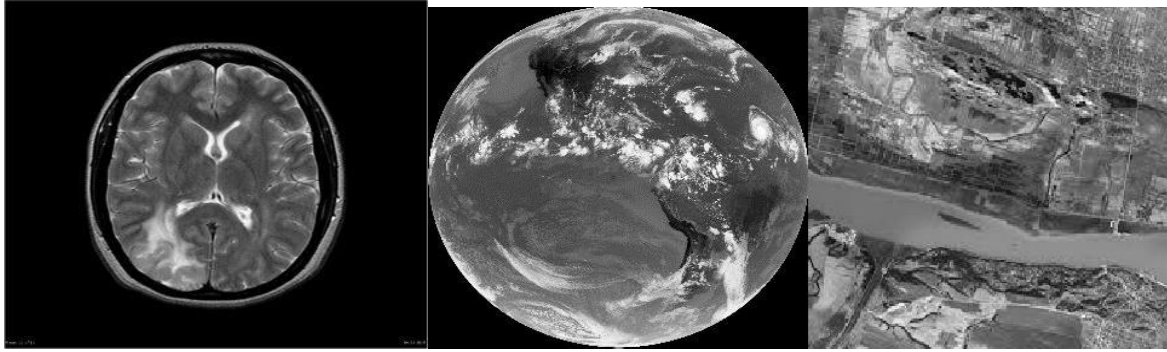


Figure 7 Dataset Samples of medical and satellite images

IMPLEMENTATION

Implementation is done using Python, version 3.7. Hardware configuration is windows 10 with 4 GB RAM. And in pseudo coloring we have implemented OpenCV's predefined colormap function to pseudocolor a grayscale image of Medical and Satellite.[4]

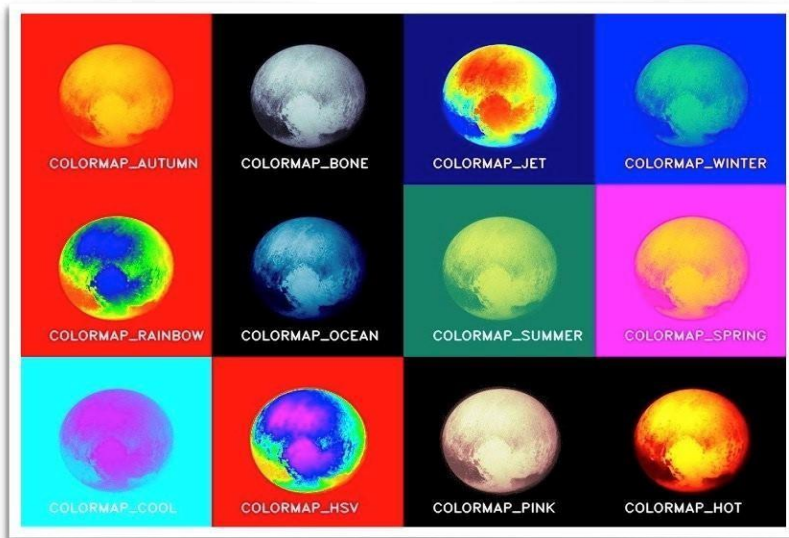


Figure 8 Mallick, S. (2015), applyColorMap for pseudocoloring in Opencv (c++/python). Learn OpenCV, [4]

RESULT

2.3 Semi-automatic Coloring (Optimization based Coloring):



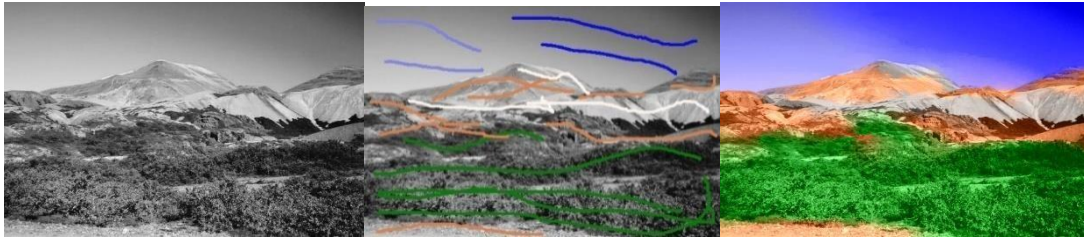


Figure 9 Result of semiautomatic coloring technique left: Gray image, Middle: marked image, Right: colored image

2.4 Pseudo coloring:

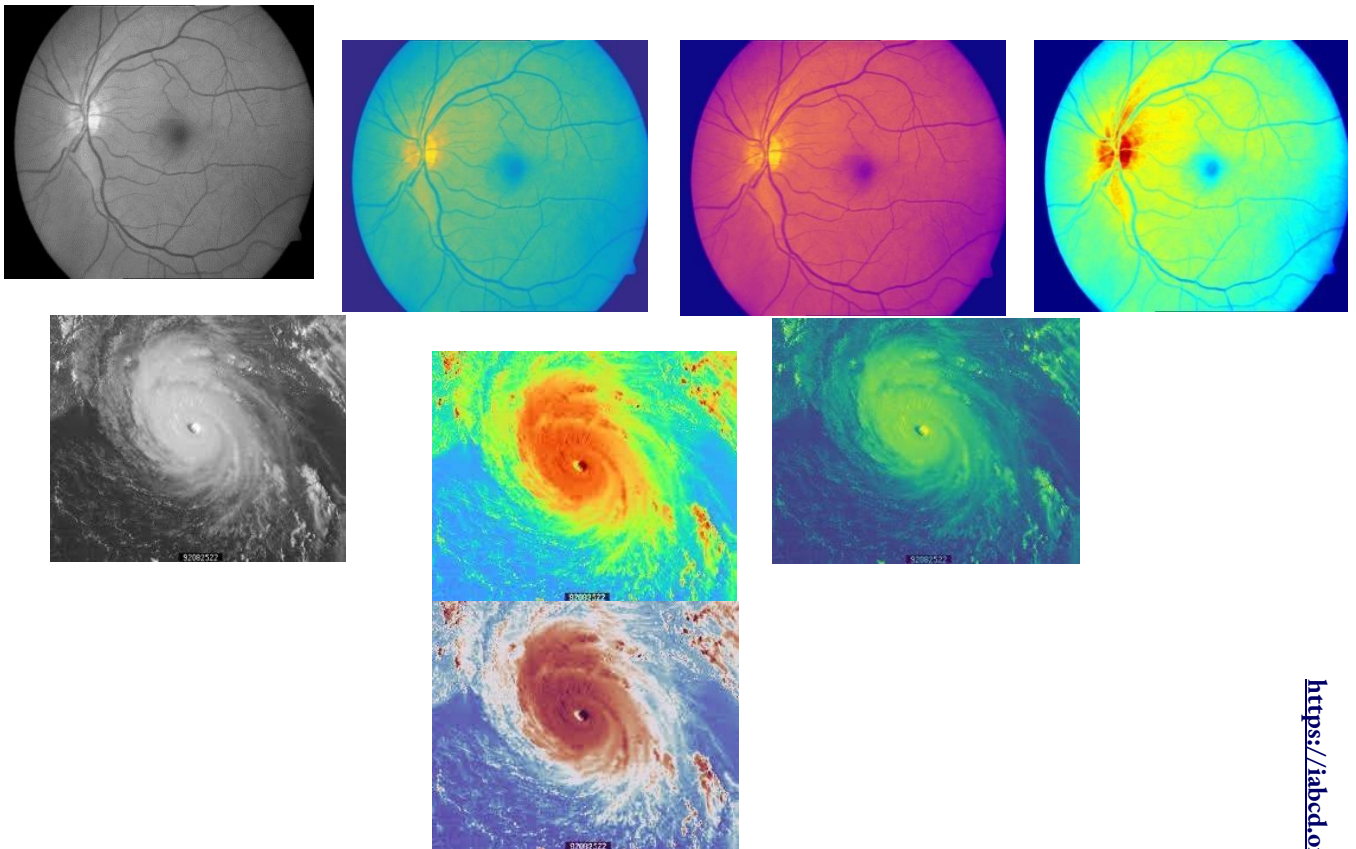


Figure 10 Result of pseudo coloring technique on medical images Left: Gray image Right different colormaps

3 CONCLUSION & FUTURE DISCUSSION

❖ Conclusion:

In this paper we have seen that Levin's algorithm requires only few pixels to be colored in image and then algorithm propagates these colors in whole image with respecting intensity boundaries. This method shows that excellent colorization can be obtained with a shockingly small amount of user effort. Although for large image it would be very time consuming. And sometimes result is not much accurate. Also, as human perception of color changes result of output colored image will also change. likewise, we have seen that pseudo color gives us very good representations of colors although false color sacrifices natural color interpretation in order to ease the detection of features that are not readily visible otherwise but for the images of medical and satellite where natural color is difficult to predict pseudo color provides good results.

❖ Future Directions:

In future work, we intend to explore other algorithms which gives even better outcomes with lessor none human intervention. Like Pseudo coloring with Histogram Interpolation, Color Transfer To Grayscale Images Using Texture Spectrum (CTTS), reference-based image colorization and Deep Convolutional Neural Networks to colorize gray scale images.

REFERENCES

- 1) W. Markle and B. Hunt, "Coloring a black and white signal using motion detection," Canadian patent no.1291260, December 1987.
- 2) T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, 2002, pp. 277–280.
- 3) A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," ACM Transactions on Graphics, vol. 23, pp. 289–694, 2004.
- 4) applyColorMap for pseudocoloring in OpenCV, Web Page: "<https://www.learnopencv.com/applycolormap-for-pseudocoloring-in-opencv-c-python/>"
- 5) NEURALTEK, 2003. BlackMagic photo colorization software, version 2.8. "<http://www.timebrush.com/blackmagic.>"
- 6) Muhammad, Imran. "Colorizing Grey Scale Images.", 2011." <https://www.diva-portal.org/smash/get/diva2:519159/fulltext01.pdf>
- 7) G. Qiu and J. Guan, "Color by linear neighborhood embedding," in IEEE International Conference on Image Processing (ICIP'05), Sept. 11–14, 2005, pp. III– 988–91.
- 8) Colorization Using Optimization, "<https://www.cs.huji.ac.il/~yweiss/Colorization/>"
- 9) Image coloring Techniques and Applications, Web Page: "<https://www.grin.com/document/205750>"
- 10) Photo Editor. Web Page: "<https://www.photopea.com>"
- 11) Colorization Slides, Web page: "<https://www.cs.tau.ac.il/~dcor/Graphics/pdf.slides/colorization.pdf>"
- 12) Image Colorization By Scribbling, My Space, Web Page: "<https://sites.google.com/site/fanyangspace111/image-colorization-by-scribbling>"
- 13) Image Processing. , "<https://www.imageprocessing.com/2016/03/gray-scale-to-pseudo-color.html>"
- 14) Overview of Scribbled-Based Colorization, Web Page: "<https://www.scirp.org/journal/paperinformation.aspx?paperid=87936#ref6>"
- 15) Pseudo Color Application, Web Page: "<https://www.ft.unicamp.br/docentes/magic/khoros/html-dip/c4/s9/front-page.html>"
- 16) sandipanweb. Web Page: "<https://sandipanweb.wordpress.com/2018/01/27/image-colorization-using-optimization-in-python/>"