



Self-regulating Grayscale Video Colorization using MATLAB

Updesh Kumar Jaiswal, Navdeep Singh, Prakhar Sharma, Nikil Deshwani, Ayush

IMS Engineering College,
Ghaziabad, Uttar Pradesh, 201009, INDIA

Abstract: For a given grayscale video, this paper will be useful to produce a plausible colored version of the given video. This problem is insufficiently constrained, so previous methods were severely labor intensive and result was not more or less desaturated. Focusing on the result with color is more true to the nature of the objects in the video. We are implementing this using MATLAB with its several components such as Image Processing Toolbox, Color and Data extractors and so on. By this input video is divided into individual frames and accounting those frames into individual fragments. Each image is now converted in defined Black and White image. With use of Image Acquisition coloring, it is made sure that different coloring channel of each frames are obtained. With a collection of different color frames obtained, Image Merging Connector is used such that all frames are merged into the output content. We evaluate the method using a "Turing Test" asking different subjects to choose between the generated or substantive data (Acceptable option). This approach result in a notable difference from the method used in today's age and is adaptable for future modifications so it can be featured in several learning benchmarks.

Keywords: Data Connectors, Coloring Channels, Colorization, Fragments, Image Merging Connectors.

I. Introduction

The Process of Colorization adds color to black and white, sepia or other monochrome moving picture. It was done by hands in early times but now with because of modernization colorization has become common with the advent of Digital Image Processing. The first film colorization methods were hand done by individuals. Each frames for a video or a film is colored individually with a different role given to each individual for different colors in assembly style which will take huge amount of resources and time. Few decades later a new method came where color was added by tracing the original black-and-white frames onto new animation cell, sand then adding color to the new cells. This degrades the quality of the original animation to an extent where it might not be used. With the introduction to the age of computer technology it became possible to add color to black-and-white films by digitally tinting single objects in each frame of the film until it was fully colorized. In Computerized Colorization, a digitized copy of monochrome video is needed. We associate a range of gray levels to each object, and indicate to the computer any movement of the objects within a shot. Sensing variations in the light level from frame to frame and correcting it is necessary. The computer software then associates a variation of the basic color with each gray level in the object, while keeping intensity levels the same as in the monochrome original. The software then follows each object from frame to frame, applying the same color until the object leaves the frame. But this method was also labor intensive instead of the introduction of new technology. This is known as Segmentation process which carries the whole process. Though it was time consuming it can be used as a sub process for a larger project which can be more valuable to the maker or the user who wants to colorize the video.

New technology in sense of pattern recognition and background compositing was introduced which moves and morphs foreground and background masks from frame to frame [3]. In the process, backgrounds are colorized separately in a single composite frame which functions as a visual database of a cut, and includes all offset data on each camera movement [1, 2]. Once the foregrounds are colorized the background masks are applied frame to frame in a utility process. The process of colorization can be used at different levels of projects and in many different ways, either in the form of partial colorization used in early 20th century to major projects like Restoration and Integration.

II. Steps involved in our proposed approach

Main objective of this propose system is to change colorless or grayscale videos into color videos without having any loss in actual content. For converting a gray scale video into color video we refer following steps:

- **Video Selection**

In this step we select the source grayscale image and depending upon the video quality we will filter it using some technique to remove noise in video.

- **Video Splitting**

After video selection we need to split grayscale frames from it. Video splitting is needed for selecting each frames of video to be converted in appropriate color [6]. Splitting can be done in different ways in terms of frames per second to be converted

- **Pattern Recognition**

For applying color to any fragment of frames first we need to recognize pattern of that part. For example, if we take a human image the first section may be face then relative color should be applied to that portion. But to apply any color first we need to find out that what part of image it is? And then accordingly color can be selected.

- **Changing color of gray scale video**

After deciding color the different segmented parts of frames are first converted into single bit black and white framed and then final step of colorization is done.

- **Joining splitted color frames**

Now the newly obtain frames i.e. color frames will be in different segments so we need to join all parts properly to get complete colorful video.

III. Components used in our proposed approach

A. Image Segmentation

Image segmentation algorithms determine region boundaries in an image. It divides an image into multiple parts. This is typically used to identify objects or other relevant information in digital images. There are many different ways to perform image segmentation, some of the methods we perform includes:

- **Thresholding methods**

The most classic image segmentation method is based on gray threshold segmentation, which is accomplished by setting the threshold to split pixels into several classes at different gray levels to implement. Image binarization is the simplest example of threshold segmentation. In addition threshold method is also divided into artificial selection, automatic threshold method, and minimum error threshold and watershed algorithm.

- **Color-based Segmentation**

Three-color base is the red, green and blue, the human eyes to the red, green, and blue have the most sensitive, most of the color can be red, green, and blue color scale according to the type of synthetically produced. Also the majority of monochromatic light can be broken down into red, green and blue lights. There is also addition subtraction principle and subtract principles.

Image segmentation algorithm such as based on image segmentation of discontinuous change, recording to the gray-scale discontinuity between the search area boundaries, there are still algorithms based on image edge similarity segmentation, pursuant to guidelines established by the prior image segmentation for similar areas, such as thresholds, regional growth, and regional separation and polymerization.

B. Color Map Threshold

A color map is matrix of values between 0 and 1 that define the colors for graphics objects such as surface, image, and patch objects. Color maps can be any length, but must be three columns wide. Each row in the matrix defines one color using an RGB triplet. An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range [0, 1]. A value of 0 indicates no color and a value of 1 indicates full intensity.

To change the color scheme of visualization, call the color map function to change the color map of the containing axes or figure. For example, the following commands create a surface plot and set the color map of the figure to mymap.

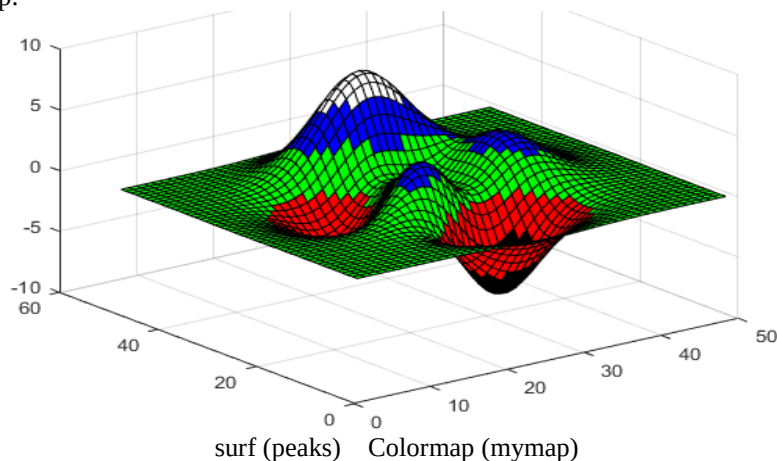


Figure 1

C. Composite and Blending of Images

When performing a series of arithmetic operations on a pair of images, you can achieve more accurate results if you use `imlincomb` to combine the operations, rather than nesting calls to the individual arithmetic functions, such as `imadd`. When you nest calls to the arithmetic functions, and the input arrays are of an integer class, each function truncates and rounds the result before passing it to the next function, thus losing accuracy in the final result [8]. `imlincomb` computes each element of the output Z individually, in double-precision floating point. If Z is an integer array, `imlincomb` truncates elements of Z that exceed the range of the integer type and rounds off fractional values.

$Z = \text{imadd}(X,Y)$ adds each element in array X with the corresponding element in array Y and returns the sum in the corresponding element of the output array Z . X and Y are real, non-sparse numeric arrays with the same size and class, or Y is a scalar double. Z has the same size and class as X , unless X is logical, in which case Z is double. If X and Y are integer arrays, elements in the output that exceed the range of the integer type are truncated, and fractional values are rounded.

Secondly, blending images seamlessly by solving for unknown pixel colors using discrete Poisson equations, the pixel colors we want to transplant into an image will be unknown initially, because they will be adjusted (e.g. for color constancy, edges) if we want the composite image to look right. We form a system of linear equations describing these unknowns, using our known data.

D. Color Channel

A color channel stores the color information for one of the primary color components of a color model. For example, the RGB color model has three separate color channels; one for red, one for green and one for blue. A color in the RGB color model is described by indicating how much of each of the red, green, and blue is included. The color is expressed as an RGB triplet (r,g,b) , each component of which can vary from zero to a defined maximum value. If all the components are at zero the result is black; if all are at maximum, the result is the brightest white.

Since colors are usually defined by three components, not only in the RGB model, but also in other color models such as CIIE LAB and Y'UV, among others, then a three-dimensional volume is described by treating the component values as ordinary Cartesian coordinate in a Euclidean space [7]. For the RGB model, this is represented by a cube using non-negative values a 0-1 range, assigning black to the origin at the vertex (0,0,0) and with increasing intensity values running along the three axes up to white at the vertex (1,1,1) diagonally opposite black. An RGB triplet (r,g,b) represents the three-dimensional coordinate of the point of the given color within the cube or its faces or along its edges.

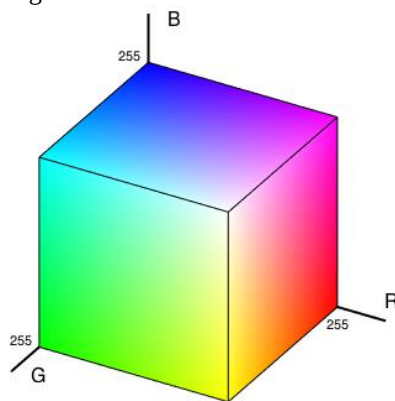


Figure 2

Each level is represented by the range of decimal numbers from 0 to 255 (256 levels for each color), equivalent to the range of binary numbers from 00000000 to 11111111, or hexadecimal 00 to FF. The total number of available colors is $256 \times 256 \times 256$, or 16,777,216 possible colors.

E. Color Map Extractor

In an 8-bit grey image there are 256 possible grey values. The values usually run from 0 (black) - 255 (white), so an intermediate grey will have a value of about 127. In grayscale there is one channel (i.e. one value for each pixel and that is a grey value). In RGB, you have three "grey" channels; each representing the red, blue, and green components of the color; so in an 8-bit RGB image you have $256 \times 256 \times 256$ possible colors = 16.7 million possible colors.

L.A.B Color also has three channels, but the channels are defined as such

L = lightness.

A = how red or green the color is.

B = how yellow or blue the color is.

The color channel values are from -128 to +127 so in the channel A value of -128 is completely green with no

red component.

A value of 127 is completely red with no green component and a value of 0 in the a channel has equal amounts green and red (so it looks neutral grey)

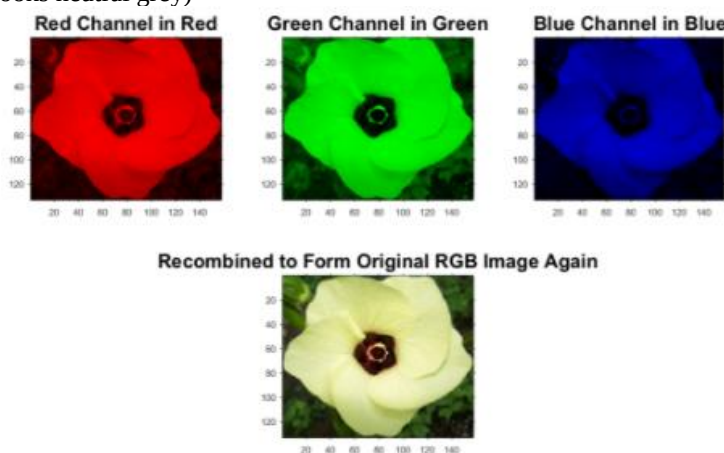


Figure 3

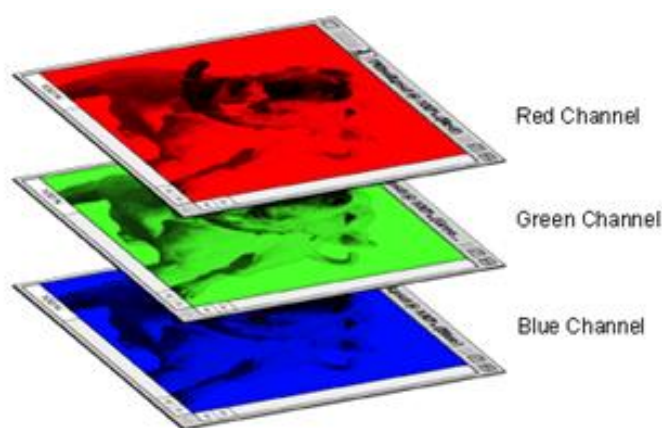


Figure 4

The same goes for the B channel where a value of -128 is completely blue with no yellow component and a value of 127 is completely yellow with no blue component [4,5].

The lightness channel runs from 0 to 100. So you can describe any possible color with three values:

- Lightness value from 0 to 100
- A value from -127 to 128
- B value from -127 to 128

Just probing the A and B values will tell you what color you are looking at, regardless of its brightness so, for example Lab of 20, -50, -50 is dark blue (the value 20 is towards the dark, zero value end of the lightness)

Lab of 50, -50, -50 is medium blue (the L value 50 is in the middle of the lightness)

Lab of 80, -50, -50 is bright blue (the L value of 80 is towards the top of the brightness scale)

Lab 0, 0, 0 is black Lab

100, 0, 0 is white and so on

There is not perfect way to convert an RGB value to LAB because the LAB gamut is much larger than RGB.

F. Image Acquisition

Image Acquisition Toolbox, a toolbox of MATLAB [9] provides an app, functions, and a programmatic interface to help you work with image acquisition hardware. The toolbox enables you to customize the acquisition process to include integrating image processing functionality to identify objects, enhance imagery, or construct mosaics and panoramic views as the data is acquired. By using this toolbox following steps can be performed:

(i) Connecting to Hardware

Image Acquisition Toolbox automatically detects compatible image and video acquisition devices. Each device connection is encapsulated as an object, providing an interface for configuration and acquisition. You can create multiple connection objects for simultaneous acquisition from as many devices as your PC and imaging

hardware support. Image Acquisition Toolbox can be used on Windows®, Linux®, and Macintosh systems, enabling you to reuse code when connecting to the same camera in different operating systems.

(ii) Acquiring Image Data

Image Acquisition Toolbox supports several modes, including background acquisition and continuous acquisition, while processing the acquired data. The toolbox automatically buffers data into memory, handles memory and buffer management, and enables acquisition from an ROI. The image acquisition engine is designed to acquire imagery as fast as your camera and computer can support, enabling analysis and processing of high-speed imaging applications.

Data can be acquired in a wide range of data types, including signed or unsigned 8-, 16-, and 32-bit integers and single- or double-precision floating points. The toolbox supports any color space provided by the image acquisition device including RGB, YUV, or grayscale. Raw sensor data in a Bayer pattern can be automatically converted into RGB data [4].

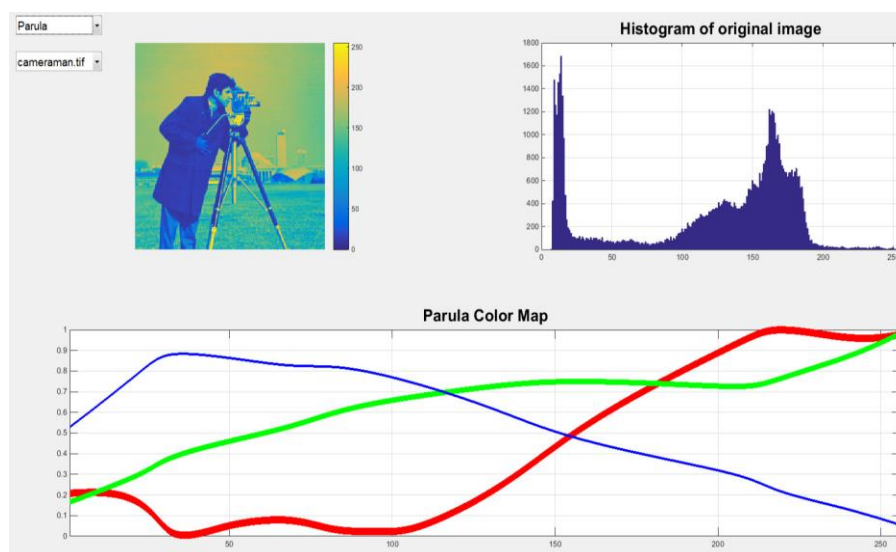


Figure 5

(iii) Triggering and Acquisition Customizations

Image Acquisition Toolbox supports multiple trigger types to customize the acquisition process: immediate, manual, and hardware. Initiating an acquisition with a manual trigger gives you extensive control over when to begin acquisition. In some applications, you may want to record at a high frame rate, perform acquisition only when an object is detected, or when a proximity warning is issued as a trigger. Code can be generated to activate a manual trigger that can be controlled by the toolbox. Hardware triggers, which are device-specific, let you synchronize your acquisition among several multimodal devices to an external signal. This enables you to record imagery along with other analog and digital input sources.

IV. Result

On applying our approach, frame of a given grayscale video as input is converted into frame of a colored video as an output by using RGB channels as shown in figures given below. Each frames converted is a combination of three different individual frames (R,G and B) thus creating a colored output frame. In similar manner all the frames are colored and finally output is obtained as a colored video.



Figure 6(a)



Figure 6(b)

V. Conclusions

We have formulated a general, fast, and user-friendly approach to the problem of colorizing grayscale video. While standard methods accomplish this task by assigning pixel colors via a global color palette, our technique empowers the user to first select a suitable color video and then transfer the color mood of this image to the gray level video at hand.

References

- [1] Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423
- [2] Dahl, R.: Automatic colorization. In: <http://tinyclouds.org/colorize/>(2016).
- [3] Charpiat, G., Hofmann, M., Schölkopf, B.: Automatic image (extracted frame) colorization via multimodal predictions. In: Computer Vision–ECCV 2008. Springer (2008) 126–139
- [4] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.: Context encoders: Feature learning by inpainting. In: CVPR. (2016)
- [5] Donahue, J., Krahenbuhl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)
- [6] Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image (extracted frame) analogies. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM (2001) 327–340
- [7] Sapiro, G., "Inpainting The Colors", IMA Preprint Series 1979 ,May 2004
- [8] Gupta R.K., Chia, A.Y.S., Rajan, D., Ng, E.S., Zhiyong, "Image colorization using similar images. In Proceedings", of the 20th ACM International Conference on Multimedia, pp. 369–378. 2012.
- [9] Stephen J. Chapman, Second Edition MATLAB Programming for Engineering,[2002].