Volume 2, No. 8, August 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

COLORIZATION OF GRAYSCALE IMAGES: AN OVERVIEW

Ambika Kalia[1], Balwinder Singh[2]
[1]Asst. Professor (Computer Engg), Bhai Gurdas Institute of Engg and Technology, Sangrur.
[1]ambika282004@yahoo.com
[2]Asst. Professor (Computer Engg.), Yadavindra college of engineering, Guru Kashi Campus,Talwandi Sabo.

Abstract: An efficient colorization scheme for images based on prioritized source propagation is proposed in this work. A user first scribbles colors on a set of source pixels in an image. The proposed algorithm then propagates those colors to the other non-source pixels and the subsequent frames. Specifically, the proposed algorithm identifies the non-source pixel with the highest priority, which can be most reliably colorized. Then, its color is interpolated from the neighboring pixels. This is repeated until the whole image or movie is colorized. Simulation results demonstrate that the proposed algorithm yields more reliable colorization performance than the conventional algorithms.

Keywords— Image colorization, priority, source pixels and propagation

INTRODUCTION

Colorization is the process of adding colors, which play an important role in the human perception of visual information, to monochrome images or videos [1]. Colorization, the task of coloring a grayscale image or video, involves assigning from the single dimension of intensity or luminance a quantity that varies in three dimensions, such as red, green, and blue channels. Mapping between intensity and color is, therefore, not unique, and colorization is ambiguous in nature and requires some amount of human interaction or external information.

The rapid progress in computer technology for multimedia system has led to a rapid increase in the use of digital images. Rich information is hidden in this data collection that is potentially useful in a wide range of applications like Crime Prevention, Military, Home Entertainment, Education, Cultural Heritage, Geographical Information System (GIS), remote sensing, Medical diagnosis, and World Wide Web [9, 10]. Rich information is hidden in these data collection that is potentially useful. A major challenge with these fields is how to make use of this useful information effectively and efficiently. Exploring and analyzing the vast volume of image data is becoming increasingly difficult. Since the human visual system can perceive color information more efficiently than monochrome information, the value of monochrome images, films and TV programs can be increased through the colorization process. However, manual colorization consumes a lot of time and labor.Based on the concepts of luminance-weighted chrominance blending and fast intrinsic distance computations, high-quality colorization results for still images and video are obtained at a fraction of the complexity and computational cost of previously reported techniques.

Possible extensions of the algorithm introduced here included the capability of changing the colors of an existing color image or video, as well as changing the underlying luminance. Adding color to a grayscale image is a neat little effect you see all over the place. Now, this isn't to be confused with taking a color image and removing its color, only to add some of it back in certain places. This technique is entirely different. It's a really simple technique that's fun to use; it's great for creating visual interest, and drawing attention to a certain portion of a photo.. Reinhard *et al.* [2] introduced an early colorization scheme, which transfers colors from a color source image to a gray target image by matching the luminance components of the two images.

Welsh *et al.* [3] improved the matching performance by exploiting the luminance values and textures of neighboring pixels. These color transferring schemes provide acceptable colorization performance, provided that an input image has distinct luminance values or textures across object boundaries. An alternative approach is to demand a user to assign colors to some pixels and propagate those colors to the other remaining pixels. Levin *et al.* [4] formulated the propagation problem as the minimization of a quadratic cost function, assuming that neighboring pixels with similar intensities should have similar colors. Yatziv and Sapiro [5] blended the colors of source pixels to paint a target pixel based on the geodesic distances from the source pixels to the target pixel. A geodesic distance measures the variation of luminances along the path from a source pixel to a target pixel. However, the propagation-based schemes may yield color blurring errors, and their performances are affected significantly by the locations of color sources.

In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges share the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a planar graph is just a vertex coloring of its planar dual. However, non-vertex coloring problems are often stated and studied *as is*. That is partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

Colors make images more vivid. They can now be recorded with a point-and-shoot camera easily. However, people often want to add colors to old monochrome photos, and pictures are sometimes shot with severely wrong white balance settings, in such a case, a possible remedy is to keep only the captured intensities and transfer colors from another source to it. The technique of adding colors is particularly useful when the image is taken with special sensors, such as X-ray, MRI, near infrared and so on. This is called "pseudo-coloring" . The difficulty of assigning colors to a monochrome image rises from the lack of deterministic relations between the luminance and the hue/saturation channel of an image – in an image, pixels of the same intensity may have different colors and vice versa. A human may intuitively guess the colors given a monochrome picture, because we know what are in the picture and we have prior knowledge on how their colors should be. However, assigning colors to each piece in an image is very tedious.

The characteristics generally used to distinguish one color from another are brightness, hue and saturation. Brightness refers to intensity. Hue is an attribute associated with the dominant wavelength in a mixture of light waves. Saturation refers to relative purity or the amount of white light mixed with a hue. The Hue and saturation taken together are called chromaticity, and therefore a color may be characterized by its brightness and chromaticity. The amounts of RGB needed to form any given color are called the tri-stimulus value. The chromaticity is useful for color mixing because a straight line segment joining any two points, and define all the different color variations that can be obtained by combining these two colors additively. Color can be added to gray-scale images in order to increase the visual appeal of images such as black and white photos, classic movies and scientific illustrations.

METHOD

Graph Coloring Algorithm:

The traditional optimistic graph coloring algorithm[6, 7, 8] consists of five main phases **Build** An interference graph is constructed using the results of data flow analysis. A node in the graph represents a variable. An edge connects two nodes if the variables represented by the nodes interfere and cannot be allocated to the same register. Restrictions on what registers a variable may be allocated to can be implemented by adding precolored nodes to the graph.

Simplify A heuristic is used to help color the graph. Any node with degree less than k, where k is the number of available registers, is removed from the graph and placed on a stack. This is repeated until all nodes are removed, in which case we skip to the Select phase, or no nodes can be simplified.

Potential Spill If only nodes with degree greater than k are left, we mark a node as a potential spill node, remove it from the graph, and optimistically push it onto the stack. We repeat this process until there exist nodes in the graph with degree less than k, at which point we return to the Simplify phase.

Select In this phase all of the nodes have been removed from the graph. We now pop the nodes off the stack. If the node was not marked as a potential spill node then there must be a color we can assign this node that does not conflict with any colors already assigned to this node's neighbors. If it is a potential spill node, then it still may be possible to assign it a color; if it is not possible to color the potential spill node, we mark it as an actual spill and leave it uncolored.

Actual Spill If any nodes are marked as actual spills, we generate spill code which loads and stores the variable represented by the node into new, short lived, temporary variables everywhere the variable is used and defined. Because new variables are created, it is necessary to rebuild the interference graph.

Note that the Simplify, Potential Spill, and Select phases together form a heuristic for graph coloring. If this heuristic is successful, there will be no actual spills. Otherwise, the graph is modified so that it is easier to color by spilling variables and the entire process is repeated.

The Graph Partitioning Problem (GPP):

Let G = (V, E) be any graph with an even number (2N) of vertices, V. The GPP involves partitioning V into two node sets V_1 and V_2 (i.e., $|V| = |V_1| + |V_2|$ and $V_1 \cap V_2 = \Phi$) such that the sum of the edge-cost having end-points in different sets is minimized. Indeed, if c_{ij} is the symmetric cost of the edge connecting nodes i and j, the GPP is the following nonlinear optimization problem :

$$\text{Minimize} \sum_{i\in V} \sum_{j\in V} c_{ij}\, x_i \cdot (1 - x_j) \quad (1)$$

$$\text{Subject to} \sum_{i\in V} x_i = N \qquad (2)$$

where $x_i \in \{0, 1\}$ for all $i \in V$; and $x_i = 1 \Rightarrow$ i is in set V_1 ; $x_i = 0 \Rightarrow$ i is in set V_2. (3)
The formulation of the GPP can be alternatively rewritten in an unconstrained form as:

$$\text{Minimize} \sum_{i\in V} \sum_{j\in V} c_{ij}\, x_i \cdot (1 - x_j) + \Pi \quad (4)$$

where $\prod$ is a penalty measure associated with (2). The latter formulation was explicitly utilized by Johnson *et. al.* in [9] and later in the genetic algorithm proposed by [10, 11]. In the absence of a systematic algorithm for tackling the problem the most obvious strategy is to resort to a brute force exhaustive search of the solution space. In this case, the solution space is prohibitively large for any meaningful problem; furthermore, this space grows exponentially with the number of nodes. Indeed, when |V| = 10, the solution space is of cardinality 126. When |V| = 100 the space has more than 10^{29} feasible solutions . Rather than attempt to obtain the optimal solution, most "approximation algorithms" try to produce near-optimal solutions. Indeed, a whole body of literature has gone into designing heuristic strategies that yield solutions that are "arbitrarily close" to the optimal one, and which are computable in a "reasonable" amount of time.

Given the luminance information of an image or video signal, the proposed algorithm attempts to generate color information, which looks natural and realistic. We work in the *YUV* space, where *Y* is the luminance channel, and *U* and *V* are the chrominance channels. Let *Y* (p) denote the luminance of pixel p, and C(p) = (*U*(p), *V* (p)) denote the chrominance vector of p.

Image Colorization:

Given a luminance image, a user puts color values onto a selected set of pixels with a few brush strokes. Then, the proposed algorithm propagates those color values to neighboring pixels to construct a color image. The color accuracy *a*(**p**) of pixel **p** is defined as a number between 0 and 1, which indicates the reliability of the color **C**(**p**). For example, *a*(**p**) = 1 when pixel **p** has the most accurate color **C**(**p**), whereas *a*(**p**) = 0 means that **C**(**p**) is not reliable at all and thus should be updated using the color information of neighboring pixels. Initially, *a*(**p**) is set to 1 if **p** is assigned a color vector by the user, or set to 0 otherwise.We call **p** a source pixel if *a*(**p**) = 1. The color information of source pixels is propagated to non-source pixels.To this end, the priorities of non-source pixels are defined, and non-source pixels are colorized in the decreasing order of their priorities. Specifically, the priority *π*(**p**) of a non-source pixel **p** is defined as

$$\pi(\mathbf{p}) = \sum_{\mathbf{q}\in\mathcal{N}_\mathbf{p}} a(\mathbf{q})e^{-|Y(\mathbf{p})-Y(\mathbf{q})|},$$

where *N***p** denotes the 4-neighbor set of **p**. In other words, if

p = (*x*, *y*),*N***p** = *{*(*x*−1, *y*), (*x*+1, *y*), (*x*, *y*−1), (*x*, *y*+1)*}*.

Notice that **p** is assigned a higher priority *π*(**p**), if a neighboring pixel **q** has a high accuracy *a*(**q**) and the luminances *Y* (**p**) and *Y* (**q**) are similar to each other. In other words, if a source pixel has a neighbor and they have similar luminances,the neighbor is assigned a high priority.

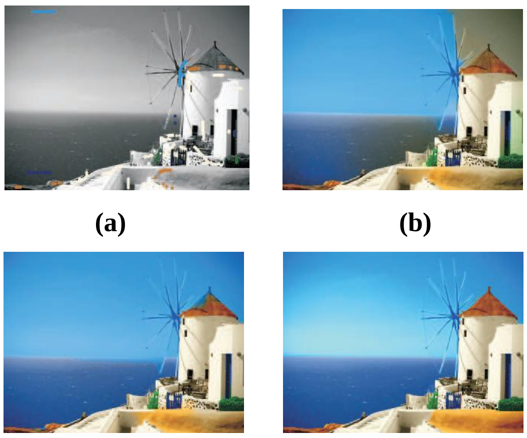

Fig. 1. Colorization of the "Ocean" image: (a) the input image with color sources, (b) the Levin *et al*.'s algorithm, (c)the Yatziv and Sapiro's algorithm, and (d) the proposed algorithm.

After assigning and sorting the priorities, we update the color vector **C**(**p**) of the non-source pixel **p** with the highest priority by

$$\mathbf{C}(\mathbf{p}) \leftarrow a(\mathbf{p})\cdot\mathbf{C}(\mathbf{p}) + (1-a(\mathbf{p}))\cdot\sum_{\mathbf{q}\in\mathcal{N}_\mathbf{p}} w_{\mathbf{p},\mathbf{q}}\mathbf{C}(\mathbf{q}), \quad (2)$$

where the weight *w***p**,**q** is defined as

$$w_{\mathbf{p},\mathbf{q}} = \frac{a(\mathbf{q})e^{-|Y(\mathbf{p})-Y(\mathbf{q})|}}{\sum_{\mathbf{r}\in\mathcal{N}_\mathbf{p}} a(\mathbf{r})e^{-|Y(\mathbf{p})-Y(\mathbf{r})|}}. \quad (3)$$

In Eq. (2), **C**(**p**) is set as the weighted sum of neighboring colors, since the accuracy *a*(**p**) of the non-source pixel **p** is 0.In the weighted summation, **C**(**q**) is given a high weight *w***p**,**q**

if **q** has a high accuracy and **p** and **q** have similar luminances.After coloring pixel **p**, its accuracy *a*(**p**) is updated to 1,*i.e.*, **p** becomes a source pixel. The priorities of the neighboring pixels in *N***p** are also updated by Eq. (1). Then, the priorities of all non-source pixels are sorted, and the non-source pixel with the highest priority is colorized by Eq. (2). This is repeated until all pixels are colorized.In the image colorization, the accuracy *a*(**p**) is binary: it is 1 if **p** is a source pixel, and 0 otherwise.

CONCLUSIONS

In this paper we have studied a new, general, fast, and user friendly approach to the problem of colorizing grayscale images. A simple but efficient colorization scheme for images and videos, which propagates the colors of source pixels to non-source pixels in a prioritized manner, was proposed in this work. The proposed algorithm derives the color accuracies and the colorization priorities, and colorizes the non-source pixels in the decreasing order of their priorities.

REFERENCES

[1]. G. Burns, "Colorization," Museum of broadcast communications:The Encyclopedia of Television, http://www.museum.tv/archives/etv/index.html

[2]. E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," IEEE Computer Graphics and Applications, vol. 21, no. 5, pp. 34–41, 2001.

[3]. T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to grayscale images," in Proc. ACM SIGGRAPH Conf., 2002,pp. 277–280.

[4]. A. Levin, D. Lischinski, and Y.Weiss, "Colorization using optimization," in Proc. ACM SIGGRAPH Conf., 2004, pp. 689–694.

[5]. L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," IEEE Trans. Image Processing,vol. 15, no. 5, pp. 1120–1129, 2006.

[6]. Preston Briggs. Register allocation via graph coloring. PhD thesis, Rice University, Houston, TX, USA, 1992.

[7]. Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. ACM Trans. Program. Lang. Syst., 16(3):428–455, 1994.

[8]. G. J. Chaitin. Register allocation & spilling via graph coloring. In Proceedings of the 1982 SIGPLAN symposium on Compiler construction, pages 98–101. ACM Press, 1982.

[9]. Johnson, D. S., Aragon, C.R., McGeoch, L.A. and Schevon, C., "Optimization by Simulated Annealing: An Experimental Evaluation; Part 1, Graph Partitioning", Operations Research, 37, 6, 1989.

[10]. Rolland, E., Abstract Heuristic Search methods for Graph Partitioning, Ph.D. dissertation, The Ohio State University, Columbus, Ohio. 1991.

[11]. Rolland, E. and Pirkul, H., "Heuristic Solution Procedures for the Graph Partitioning Problem", Proc. of the 1992 ORSA-CSTS Conf. on Computer Science and Operations Research : New Developments in Their Interfaces, Williamsburg, 1992, pp. 475-490.