Fast Colorization Using Edge and Gradient Constrains

Yao Li
Digital Media Lab,
Shanghai Jiao Tong university
No.800 Dongchuan Road
200240, Shanghai, China
yaolily77@sjtu.edu.cn

Ma Lizhuang
Digital Media Lab,
Shanghai Jiao Tong university
No.800 Dongchuan Road
200240, Shanghai, China
ma-lz@cs.sjtu.edu.cn

Wu Di
Electrical Engineering Department
Shanghai Jiao Tong university
No.800 Dongchuan Road
200240, Shanghai, China
guanliao@sjtu.edu.cn

ABSTRACT

This paper proposes a fast but effective user-guided colorization algorithm. The main difficulty with colorization is its intensive computational cost. And the color sometimes diffuses from one region to others. We make full use of the information given by the gray-scale images, including the edge, gradient and gradient direction, to propagate color over regions from the user's scribbles. We introduce a novel local distance definition which reduces the color confusion between two regions obviously. Two-Dimensional Programming is used to get the minimum distance from the scribbles to every pixel, and every pixel is blended by the chrominance with top three minimum distances. This speeds up the colorization process. Our algorithm can also be extended to recolorization and movie colorization. Experimental results show that our algorithm outperforms many state-of-the-art algorithms from small amount of user scribbles. Besides, the implementation require less than 1 second on the image with 320*240 pixels.

Keywords

Colorization, recolorization, interactive

1. INTRODUCTION

Color can be added to gray-scale images in order to increase the visual appeal of images such as old black and white photos, classic movies or scientific illustrations. In addition, the information content of some scientific images can be perceptually enhanced with color by exploiting variations in chromaticity as well as luminance.

Colorization is a term introduced by Wilson Markle in 1970 to describe the computer assisted process he invented for adding color to black and white movies[1]. The term is generically used now to describe the process of adding color to monochrome still images and movies. Colorization is the problem of assigning three-dimensional pixel values to an image which varies along only one dimension. Since different colors may have the same luminance value but vary in hue or saturation, the problem of colorizing gray-scale images has no inherently "correct" solution.

The main difficulty with colorization is its intensive computational cost. The user cannot see the effect immediately and colorize the images interactively. Levin[2] propose a new user-guided colorization technique based on the premise that nearby pixels in space that have similar gray levels should also have similar colors. However, because no boundary or region information is considered in their colorization process, unseemly color sometimes diffuses from one region to others. The processing time of colorizing an image with 320*240 pixels costs ten or more seconds.

In this paper, we propose a user-guided colorization method inspired by Levin's. This paper makes several specific technical contributions. First, we introduce a novel local distance definition, which take the edge, gradient and gradient direction into accounts. With the edge and gradient constrains, the color confusion between two regions reduce obviously. Next, we use Two-Dimensional Programming to get the minimum distance from the scribble to every pixel. This make the process time

only one tenth of Levin's. We also extend our algorithm to movies.

2. RELATED WORK

Colorization has been extensively studied in the movie industry since 1970's. Various analogue techniques have been used to accomplish this challenging task[3]. In this section we focus on digital colorization only.

Semi-automatic Colorization

Welsh[4] exploited local luminance distribution as textural information and transfer the color between the similar texture regions in the source color image and the target gray-scale image. The idea is to transfer color from neighborhoods in the reference image that match the luminance in the target data. There is then an underlying assumption that different colored regions give rise to distinct luminance, and their approach works properly only when this is not violated, otherwise requiring significant user intervention. This approach is a special case of the more general image analogies framework[5]. The results reported by the authors are quite impressive, although the technique intrinsically depends on the user to find proper reference data.

Irony[6] presented a method to colorize grayscale images by transferring color from a segmented example image. Instead of relying on a series of independent pixel-level decisions, they developed a new strategy that attempts to account for the higher-level context of each pixel. Target pixels are classified using LDA and image space voting with DCT coefficients as feature vectors. The colorized results exhibit nice spatial consistency.

Sykora[7] introduced a colorization framework for old black-and-white cartoon video. The dynamic part of the scene is represented by a set of outlined homogeneous regions which superimpose the static background. They combined unsupervised image segmentation, background reconstruction, and structural prediction to reduce manual intervention.

User-guided Colorization

Levin[2] proposed a simple yet effective user-guided colorization method. In this method the user is required to scribble the desired colors in the interiors of the various regions. These constraints are formulated as a least-squares optimization problem that automatically propagates the scribbled colors to produce a completely colorized image. The basic premise is that neighboring pixels having similar intensities in the monochrome data should have similar colors in the chroma channels .Other algorithms based on color scribbles have subsequently been proposed [8][9]. These approaches have produced some impressive colorizations from a small amount of user input.

Manga[10] Colorization proposed a novel colorization technique that propagates color over regions exhibiting pattern-continuity as well as intensity continuity. It propagates colors over pattern-continuous regions containing hand drawn hatching and printed screening patterns. The user is free from manually segmenting the patterned regions. Both pattern-continuous and intensity-continuous regions can be segmented under the same mathematical framework.

3. ALGORITHM

Our colorization method is working in YUV color space(other color spaces could be used as well). The problem of colorization is to estimate the U, V component value of the input gray-scale image from the only known intensity value Y. That is an ill-posed problem in which two and more solutions exist, since different color may have the same intensity value.

Our algorithm is based on three assumptions:

- The colors have high correlativity between pixels in one local region and low correlativity in different regions;
- Pixels having similar intensities have similar colors;
- The color values in the input image change smoothly.

We start with the description of the proposed algorithm for still images. Let $Y(x,y):\Omega \rightarrow R^+$ be the given monochromatic image defined on the region. The given monochromatic image becomes the luminance *Y*. The goal is to compute $U(x,y):\Omega \rightarrow R^+$ and $V(x,y):\Omega \rightarrow R^+$. The colors are given in a region Ω_c, and $|\Omega| << |\Omega_c|$. This information is provided by the user via color strokes in editing type of applications, or automatically obtained for compression (selected compressed regions) or wireless (non lost and transmitted blocks) applications. The goal is from the knowledge of *Y* in Ω and *U*, *V* in Ω_c to get the color information *(U,V)* into the rest of Ω, Ω-Ω_c.

$P_{s,t}$ represent a path connecting *s* and *t*, where $s \in \Omega_c$ and $t \in \Omega$-Ω_c. Our objective is to find a path which have the minimum link cost from each scribble to *t*. We define the geodesic distance between *s* and *t* by:

$$dist(s,t) = \min_{P_{s,t}} \sum J(p,q),$$

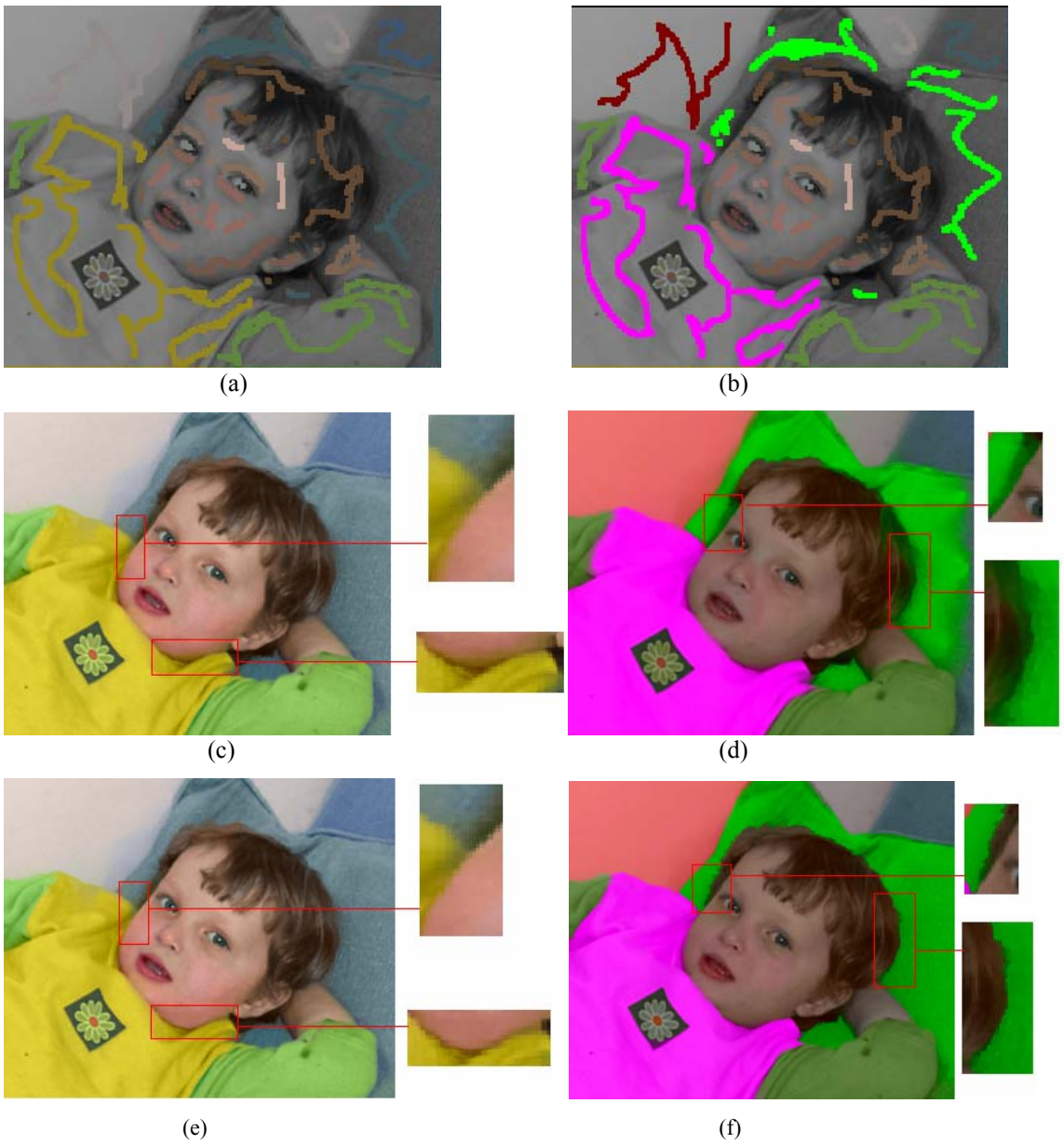
$p,q \in P_{s,t}$, $J(p,q)$ is the local link cost between two neighbor point.

Every scribble gives a uniform color *c*. To every point *t*, we define the geodesic distance $d_c(t)$ as the minimum distance *d(s,t)* from *t* to *s* of the same chrominance *c*:

$$dist_c(t) = \min_{\forall s \in \Omega_c} d(s,t)$$

Local cost

Figure 1 shows the result of Levin[2]. At first glance, the result is exciting. But there is color confusion near the edge between two images even after careful color choosing and scribbling(figure 1.c). If we change the color of the scribbles, we can see that there are obvious color confusion near the area between two regions(figure 1.d).



(a) (b) (c) (d) (e) (f)

Figure 1. Comparison of Levin's results[2] and ours. (a) The gray-scale image with color scribbles from [2], (b)The gray-scale image with color scribbles whose positions are same as(a), but color changes; c) and (d) are the results using Levin's algorithm; e)and f) are the results of our algorithm.

The edges, the gradient and its direction give important cue of the color, so we compute the local cost of two neighbor point as a weighted sum of three parts:

$$J(p,q) = w_E J_E(p,q) + w_G J_G(q) + w_D J_D(p,q),$$

where J_E is the edge cost, J_G is the gradient cost and J_D is the gradient direction cost.

We use Canny edge detector to extract the boundary of the input gray image I and restore it into the boundary image I_E, then

$$J_E(p,q)=\begin{cases}1; & if\ \ p\in I_E\ \ or\ \ q\in I_E\\ 0; & otherwise\end{cases}$$

Then, we calculate every pixel's gradient magnitude:

$$G=\sqrt{I_x^{\,2}+I_y^{\,2}}\,,$$

where I_x and I_y represent the partials of an image I in x and y respectively.

To keep the resulting maximum gradient at unity, we define:

$$J_G=\frac{G}{\max(G)}$$

Then $J_G(q)$ is scaled by 1 if q is a diagonal neighbor to p and by $\sqrt{2}$ if q is a horizontal or vertical neighbors.

Next, we define the gradient direction cost. For every pixel, the cost is computed in 8 directions respectively. The gradient in each direction is defined as:

$$J_i(p,q)=\begin{cases}\dfrac{q_{i+1}-q_{i-1}}{2}, & i \text{ is the diagonal direction}\\ \dfrac{q_{i+1}+q_{i+2}-q_{i-1}-q_{i-2}}{4}, & \text{otherwise}\end{cases},$$

where $q\in N(p)$, q_{i+j} means the neighbor in the j clock wise direction p, q_{i+j} means the neighbor in the j counter clockwise direction to p, $i\in(0,8)$, $j\in(-2,2)$.

Color blending

We compute the U, V component of every point $t\in\Omega-\Omega_c$ by blending the different chrominance in Ω_c:

$$C(t)=\frac{\sum_{\forall c\in\Omega_c} W(d_c(t))c}{\sum_{\forall c\in\Omega_c} W(d_c(t))}, \qquad (1)$$

where $W(.)$ is a blending weight function. Here we use $w(r)=e^{-r^2/2}$.

Obviously, the chrominance with lower intrinsic distance effect the point's color more. We found that we can get the satisfactory results using three closest chrominance to blend the color. So we only use the chrominance with the three closest distances to blend the point. This can fast the blending speed.

Two-Dimensional Programming

The pixels with link cost construct a graph. Dynamic programming can be formulated as a directed graph search for an optimal path. We utilizes an optimal graph search similar to that present by Dijkstra[11] and Nilsson[12].

Firstly, we construct a heap to store the nodes sorted by link cost. The user's scribbled colors are the seeds and we insert them into the heap. Then an expending process deals with all their neighbors and inserts them to the heap until the end point is reached.

The *2-D* dynamic programming (DP) graph search algorithm is follows:

Input :

I //Grayscale image

s //scribbles drawn by user

Output:

I' //colored image

Data Structures:

M; // a map structure which store the chrominance reached this point and the correspondence distance.

blend;//the color blend result of a point

P(p);// a structure store a point p's M and blend

Link; // a path whose nodes pointer the point/pixel

I_E; //The edge image;

Algorithm:

Using Canny edge detector to get edge image I_E;

For $t\in\Omega$

New *P(t)*;//Create an empty *M* for every point;

End for

New *Link_pool*;

For $p\in\Omega_c$

P(p).M.push(*c*,0); //set the point in the scribbles to be the chrominance of the scribbles; and the distance to be *0*;

Link.push(*P*(*p*));

End for

While *Link* is not empty

For all *P*, Get the pair (c_p,$dist_p$)with smallest distance

Pop *P* from *Link*;

For all $q\in N(p)$

Get *P(q)*;

Calculate dist*(p,q)*;

$dist_q = dist_p + dist(p,q)$;

P(q).M.push(pair(c_p,c_q));

If(*P(q).M.pair_no*>=3

P(q).M.sort;

P(q).M.pop_back;//only remain the three smallest distance

End if

Link.push*(P(q))*;

End for

For $t \in \Omega$

Caculate $P(t)$.blend equation(1)

End for

Video

Levin[2] uses optical flow tracking to propagate colors across time. We avoid this explicit computation. We opt for a simpler formulation. Following the color constancy constraint often assumed in optical flow, and if the gradient fields and motion vectors of all the movie channels are the same, then of course we can consider:

$$\frac{\partial Y}{\partial t} = \frac{\partial U}{\partial t} = \frac{\partial V}{\partial t},$$

where t is the time coordinate in the movie.

Recolorization

Recolorization is to replace the colors of an image by new colors. The original color in the image give us more cue than the gray images. We assume that the color changes indicate the new color changes. That's to say, in the recolorization process, the original chrominance give more information than intensity. So we can replace the Y by U or V in equation (1).

4. RESULTS AND ANALYZE

Our blending algorithm's time and space complexity is $O(|\Omega| \cdot |c|)$, where c is the chrominance of Ω_c. Because $|c|$ is small and we only use the three chrominance with lowest distance to blend. So the time and space complexity are reduced to $O(|\Omega|)$.

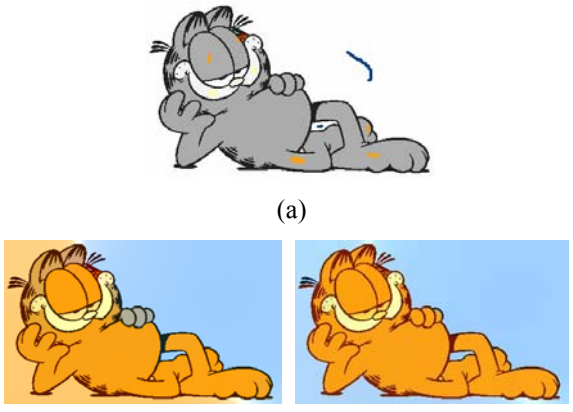
The colorization system is built on a Pentium IV 1.9Ghz CPU and 512M RAM. Most images can be colorized reasonably well in under a second. We get the scribbled pictured from Levin's website and using our algorithm on them. As figure 2 shows, we get almost the same satisfied results as Levin's. Levin takes more than ten seconds, while our algorithm only takes less than one second (Table 1). Figure 1.c and d is our results. From the insets, we can see there are less confusion around the face and hair. We give more results on nature scenes in figure 5.

Because we take edges into accounts, our proposed algorithm needs smaller scribbles than [2], especially for the images with strong edges, such as cartoons. In figure 3, our algorithm colorizes the gray-scale Garfield good. Figure 4 shows more results on natural scene. Our algorithm also gives good results. Garfield's right and left area are colorized correctly because of their connection.

Figure 2 The first column is the gray images marked with color scribbles from [2]. The second column is our colorization results.

	Pixels	Time in [2] (second)	Ours (second)
Cats	319*267	15.209	0.712
Girl	318*238	10.128	0.421
Building	399*299	15.267	0.917

Table 1 The comparison of the processing time between ours and Levin's under the same platform.



(a) (b) (c)

Figure 3 Colorization of Garfield. (a) The gray-scale image with color scribbles, (b) The colorization using algorithm in[2] (c) Our colorization result.

Figure 5 shows how our algorithm can be applied to recolorization. There is no need to mask the object as [2]. What the user need is to scribble the color on the areas where he want change the color. Our algorithm can recolor the object according to their original color change with the edge and gradient constrains.

Figure 6 shows 8 frames from the movie "Ice Age 2". We change their color to gray to test our algorithm. We only scribble color on one frame. The remaining frames can be colorized around 2 seconds. And the effect is good.

Figure 4: More still examples. Right column: the input images with scribbled colors. Left column: colored images.

5. SUMMARY

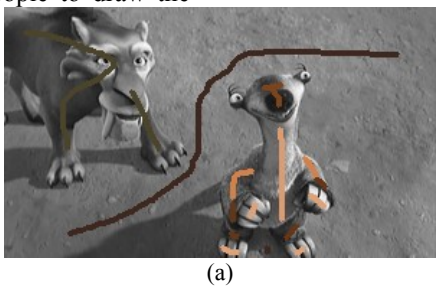
A simple, effective and fast colorization framework was introduced in this paper. While keeping the quality at least as good as state-of-art algorithm, our propose method colorize images within a second. Besides the improvement on speed, our algorithm reduces the color confusion near the boundary by bringing in edge and gradient cost. Our algorithm also can be applied on movie efficiently.

There are many important topics to explore in the future work. First, it is hard to people to draw the scribble in a right region to make the algorithm more efficient. So we need to find a way to understand what kind of information is needed in the chroma channels for error controlled colorization. This will be helpful in editing images, directing the user to the crucial regions to provide the strokes. Second, we can use the colorization onto compression. Only a few color samples can colorize the entire gray-scale image. This can be explored to a new color image compress method. Third, this colorization algorithm can be used on wireless image transmission. When some data lost in transformation process, using the relation between the chrominance and intensity, the lost data can be repaired.

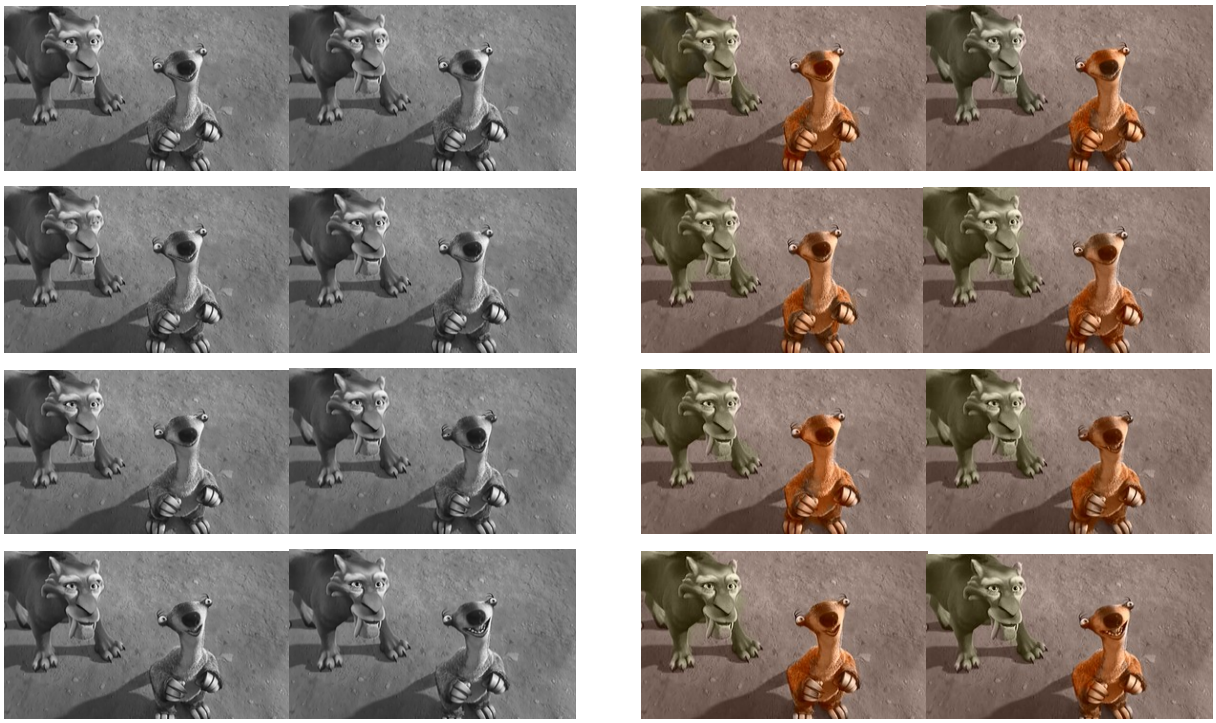
Figure 5 Recolorization results. The first row is the original images, the second row is the images with color scribbles, and the last row is the result.

6. ACKNOWLEDGMENTS

This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No.2006CB303105 and the National Natural Science Foundation of China under Grant Nos. 60373070.



(a)



(b) (c)

Figure 6 Colorizing the movie by our colorization algorithm. (a) The first frame with the color scribbles. (b) The 8 gray-scale frames. (c) Colorization results.

7. REFERENCES

[1] G. Burns, Colorization. Museum of Broadcast Communication:Encyclopedia of Television. Http://www.museum.tv/archives/etv/index.html

[2] Levin A., Lischinski D., and Weiss Y. 2004. Colorization using optimization. ACM Transactions on Graphics, 3, 689–694.

[3] Markle. W. 1984. The development and application of colorization. SMPTE Journal, 632–635.

[4] Welsh T., Ashikhmin M., and Mueller K. 2002. Transferring color to grayscale images. ACM Transaction on Graphics 21,3(July),277-280

[5] Hwerzmann A., Jacobs C. E., Oliver N., Curless B., Salesin D. H.: Image analogies. In Proc. SIGGRAPH, 2001 (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 327–340.

[6] Irony, R., Cohen-or, D., and Lischinski, D. 2005. Colorization by example. In Proc. of Eurographics Symposium on Rendering 2005, 201–210.

[7] Sykora, D., Burianek, J., and Jirizara. 2004. Unsupervised colorization of black-and-white cartoons. In Proc. of Third International Symposium on Non Photorealistic Animation and Rendering, 121–128.

[8] Sapiro G.: Inpainting the colors. IMA Preprint Series, Institute for Mathematics and its Applications University of Minnesota, 1979 (2004).

[9] Yatziv L., Sapiro G.: Fast image and video colorization using chrominance blending. IMA Preprint Series, Institute for Mathematics and its Applications University of Minnesota, 2010 (2004).

[10] Yingge Qu, Tien-Tsin Wong, Phen-Ann Heng, Manga Colorization, In Proc. SIGGRAPH 2006, Computer Graphics Proceedings, Annual Conference Series pp: 1214 - 1220.

[11] E.W.Dijkstra,"A note on Two problems in Connexion with Graphs" numerische Mathematik, vol1.1, pp.269-270, 1959

[12] N.J.Nilsson, Principles of Artificial Intelligence. Palo Alto, CA: Tioga, 1980.