



Fast Colorization Using Edge and Gradient Constrains

Yao Li, Ma Lizhuang, Wu Di
Digital Media, Shanghai Jiao Tong University

What is colorization?



The term is generically used now to describe the process of adding color to monochrome still images and movies



Why do we need colorization techniques?

- Easier
- Faster
- Cheaper



Current techniques

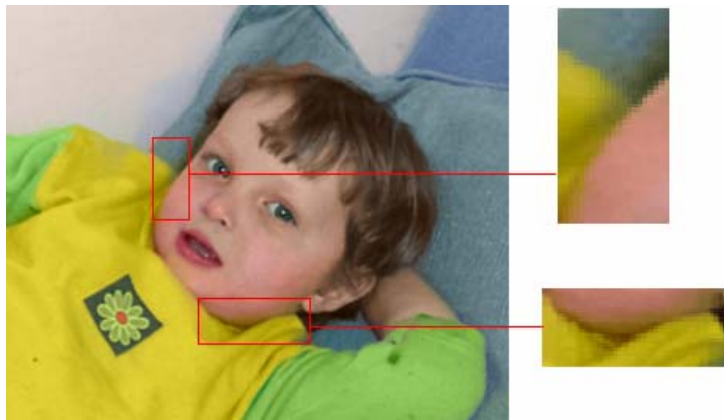
■ Automated colorization

- ▶ Need source images
- ▶ Transferring Color to Grayscale Image
- ▶ Colorization by example

■ Semi-automated colorization

- ▶ Naive segmentation
- ▶ Optimization

Color Using Optimization(Levin04)

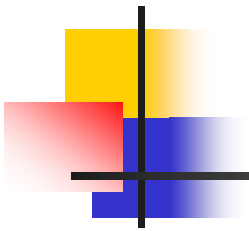


there is color confusion near the edge between two images even after careful color choosing and scribbling



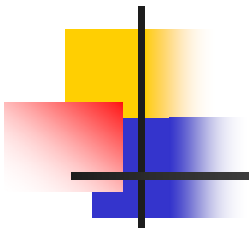
Our algorithm

- A **User-guided Colorization** Inspired by Levin's Optimization method
- 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.

- 
-
- 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:

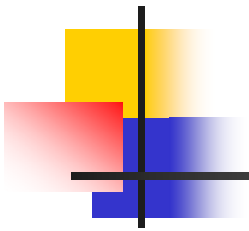
$$dist(s, t) = \min_{P_{s,t}} \sum J(p, q)$$

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

- 
-
- 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.



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

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

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

Where I_E is the boundary, 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 , $i \in (0, 8)$, $j \in (-2, 2)$.

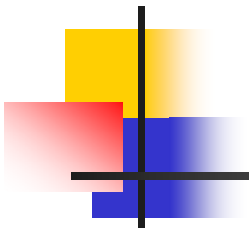


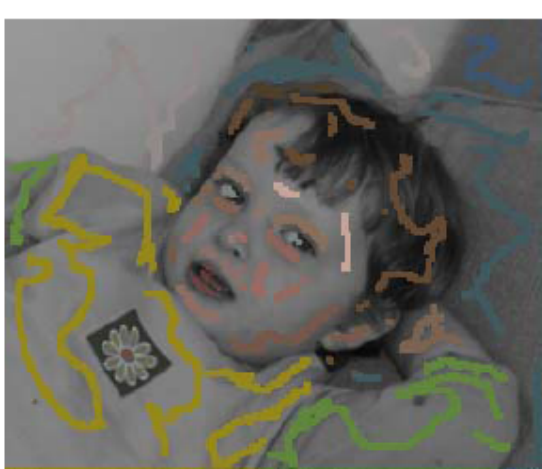
Color blending

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

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

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

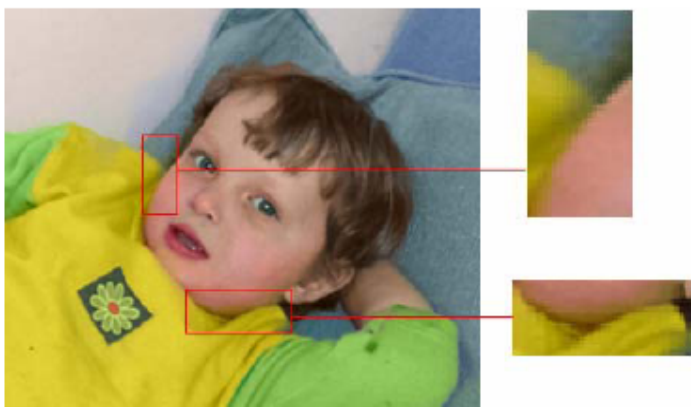
- 
-
- The chrominance with lower intrinsic distance effect the point's color more. We found that we can get the satisfactory results using three closest chrominance to blend the color. So we only use the chrominance with the three closest distances to blend the point. This can fast the blending speed.



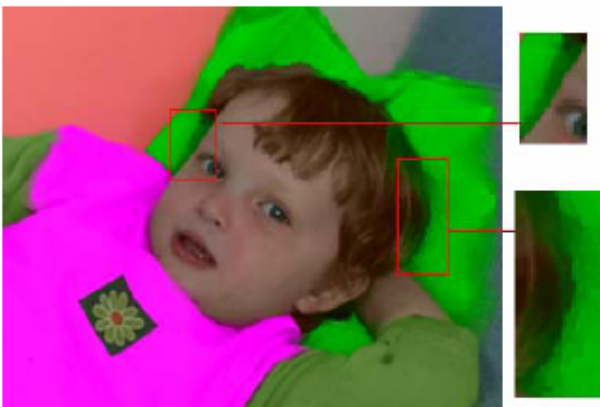
(a)



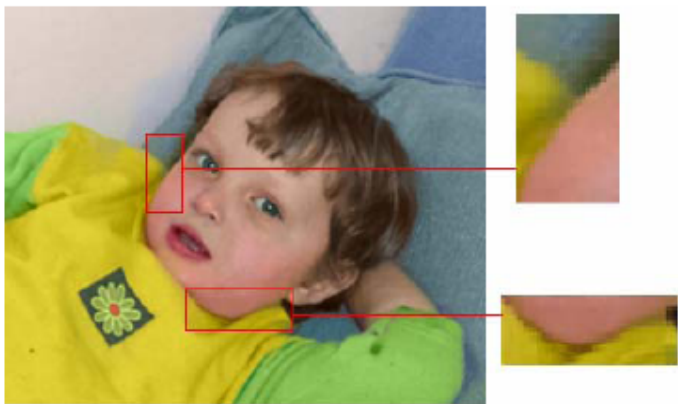
(b)



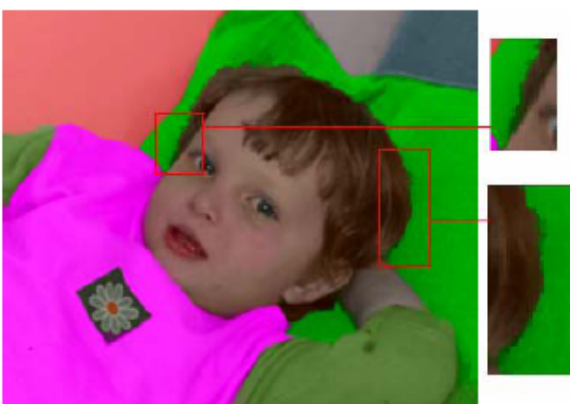
(c)



(d)



(e)



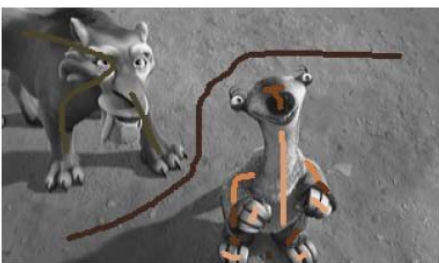
(f)

- (a) is the gray-scale image with color scribbles from “Colorization using optimization” (SIGGRAPH2004), (b) is the gray-scale image with color scribbles which’s 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.



Video

- Levin uses optical flow tracking to propagate colors across time. We avoid this explicit computation. 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.



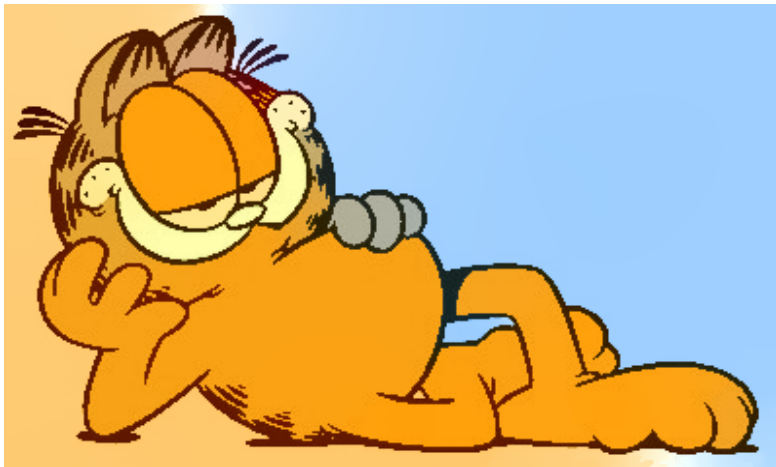


Recolorization

- 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 color blending equation

Recolorization Results





Because we take edges into accounts, our proposed algorithm needs smaller scribbles, especially for the images with strong edges, such as cartoons. The left is levin's result. The right is ours.



More still examples





Future Work

- Weighting function
- The use of the colorization approach for image **compression**
- need to understand what is the **simplest** set of color scribbles to reconstruct the color without any visual artifacts.
- it is important to understand the **best** position to place the scribbles