Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Learning Natural Colors for Image Recoloring

H.-Z. Huang[1], S.-H. Zhang[1], R. R. Martin[2], S.-M. Hu[1]

[1]Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, China
[2]School of Computer Science and Informatics, Cardiff University, UK

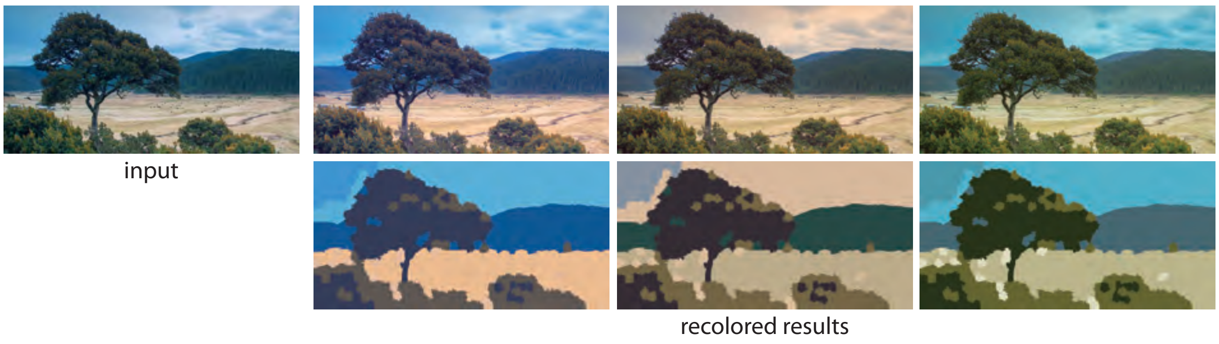

Figure 1: *Top: An input image and our three best recoloring results. Bottom: Segmentation and suggested colors for regions.*

Abstract

We present a data-driven method for automatically recoloring a photo to enhance its appearance or change a viewer's emotional response to it. A compact representation called a RegionNet summarizes color and geometric features of image regions, and geometric relationships between them. Correlations between color property distributions and geometric features of regions are learned from a database of well-colored photos. A probabilistic factor graph model is used to summarize distributions of color properties and generate an overall probability distribution for color suggestions. Given a new input image, we can generate multiple recolored results which unlike previous automatic results, are both natural and artistic, and compatible with their spatial arrangements.

Categories and Subject Descriptors (according to ACM CCS): I.3.m [Computer Graphics]: Miscellaneous—Computational photography; I.3.m [Computer Graphics]: Miscellaneous—Image Processing;

Keywords: Recoloring; Image Filter; Image Enhancement; Image Segmentation; Edit Propagation.

1. Introduction

Recoloring a photo can enhance its appearance, and change a viewer's emotional response to it e.g. dark colors may suggest sadness. Recoloring should also preserve the natural appearance of a photo, especially for e.g. outdoors scenes, both in terms of the colors used, and color composition. For brevity, we will here refer to these two aspects of recoloring as a desire to produce *artistic* and *natural* results.

Many methods have been devised for recoloring. Most map color distributions between input and reference images [FPC*14]. They are limited in use, as they require skillful selection of the reference image, which must be appropriate to the input image. One way to overcome this limitation is to use a set of reference image colorings to learn implicit rules about suitable image color combinations. The idea of utilizing large database has been proven effective on media manipulation, synthesis and understanding [HCX*13]. Recently, Lin et al. gave a data driven method for colorization of patterns [LRFH13], taking into account the spatial correlation of colored regions in the pattern. They used a factor graph to suggest aesthetic colorings: a probabilistic graphical model factorizing a complex overall probability distribution into simpler probability density functions. However, users also expect a natural appearance when recoloring a photo. Without considering color correlations in the natural world, Lin's method can generate unnatural results (e.g. a green sky) when applied to photos.

We show here how to generate photo recolorings that are both artistic and natural. Our approach can be considered to be a data-driven image filter. Inspired by [LRFH13], we use a factor graph to generate recolorings. The novelty in our work lies in producing suggestions with natural colors for real world objects, with suitable spatial arrangements of compatible colors. A *RegionNet*, a novel compact image representation, is used to describe image regions. Nodes store geometric features and mean color of each region; edges store geometric relationships between regions. Training using a database of 3000 images captures the relationships between color property distributions and geometric features of image regions. The color property distributions are combined using a probabilistic factor graph model.

We divide regions into main regions and detail regions, with different recoloring strategies, to ensure robust recoloring results in the face of imprecise image segmentation. We use Markov chain Monte Carlo (MCMC) methods to sample the overall probability density function encoded by the factor graph, generating multiple color suggestions for a given photo. To avoid discontinuities at region boundaries, the color of each pixel is determined by blending suggested colors for nearby regions, using blending weights computed by edit propagation.

Our contributions are threefold: (i) a novel framework for automatically generating multiple appealing yet natural recoloring results for a photo, (ii) the RegionNet, a compact image representation specifically designed for photo recoloring, and (iii) factors for a factor graph model specifically designed to produce artistic yet natural results.

2. Related Work

Our work is inspired by earlier work on color editing, but uses a compact image representation and edit propagation.

Color editing. A widely used method for color editing is to map the color distribution of a reference image to the input image [RAGS01, TJT05, PKD07]. The color transfer method in [WDK*13] creates natural recolorings by taking geometric context into account. By considering temporally consistency, color transfer can also be applied to video [BSPP13]. Color transfer based recoloring methods need careful user selection of a compatible reference. Similar ideas have been used to colorize grayscale images [WAM02, LLW04, CHS08]. To ensure natural colorization results, [CZG*11] considers reference images with similar content found by use of text labels and object contours. [GCR*12] requires the user to supply a semantically similar reference and focuses on matching superpixels. More can be found on color mapping in [FPC*14].

In a data-driven approach to recoloring, [WYW*10] captures color histograms of different textures in a database to steer colors in an input image towards a specific color theme to ensure natural recoloring results, while [WJC13] connects colors with words that describe emotions, allowing users to recolor an image by typing a single word. Both methods require user input, and can give poor results if the theme or word selected is incompatible with the input image. Our method can *automatically* provide multiple appealing recolored results without user input, avoiding poor user choices.

Lin [LRFH13] uses a factor graph model to colorize patterns, utilizing the distributions of different color properties (e.g. intensity and saturation) learned from a database. While it can suggest multiple good alternatives for an input pattern, it generates unrealistic, exaggerated recolored results when applied to natural images, e.g. a green sky. Our model employs a similar factor graph model, but takes into account that results should be both natural and artistic. While simple patterns can be segmented readily, it is more difficult to segment natural images. We finesse this problem by classifying image regions as either main or detail, and only learn for main regions to improve learning reliability.

Compact image representation. The bag-of-visual-words model [FFP05] represents an image in terms of abstract, discriminating features. However, it does not represent geometric features of image regions and geometric relationships between them. Other work [KWR07] summarizes an image using an arrangement of patches but does not adequately capture geometric relationships between regions. In [HZW*13], a *PatchNet* structure is used to capture geometric relationships between regions for image synthesis. However, they do not extract geometric features for each individual region. Furthermore, constructing regions based on patches is time consuming and creates many small meaningless pieces, which can be simplified by using superpixels. Thus, we have devised a new compact image representation with simpler construction, the RegionNet, which summarizes color and geometric features for image recoloring.

Edit propagation. Intuitive stroke-based methods can help users propagate editing operations from one image area to another with similar appearance: see [LLW04, LFUS06, XLJ*09, LCG10]. In particular, [LJH10] formulates edit propagation as function interpolation in a high-dimensional space. This can be done rapidly, as needed for interactive applications. We adapt this work to efficiently compute weights to determine output pixel colors from multiple regions, to avoid recoloring discontinuities at region boundaries. A similar *soft segmentation* method is used in [WYW*10]. However, while we compute the probability that a pixel belongs to a certain region with an intuitive normalization method, they compute probabilities heuristically.

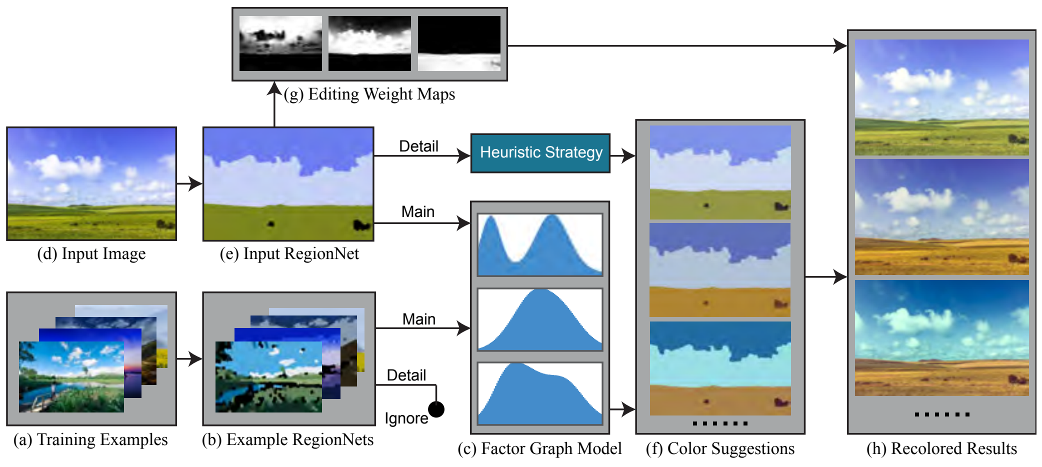

Figure 2: *Pipeline. (a) A set of well-colored images produces (b) one RegionNet for each example. Main regions are used for training; detail regions are ignored. (c) Color property distributions learned from the database are combined into a factor graph model. (d) An input image gives (e) a RegionNet with main and detail regions. (f) Multiple color suggestions are generated for its main regions by MCMC sampling of the factor graph; detail regions are recolored separately. Smooth recoloring output (h) is computed using editing weight maps (g) generated by edit propagation. Results are ranked for user presentation.*

3. Overview

To achieve appealing color effects, professional photographers carefully adjust the colors of regions one by one. We assume that implicit rules guide the photographer's selection of suitable colors, and that these can be learned from a database with sufficient examples. As well as taking into account the artistic quality of the results, photographers must also avoid unnatural recoloring. Here, we assume that such rules may be expressed in terms of geometric features of regions, e.g. area, position, and texture. Texture helps provide an understanding of the semantic content of a region, e.g. whether it is grass or rock. Other geometric features also contribute to the meaning of a region, and furthermore, geometric relationships between regions also determine appropriate recoloring. Once these implicit rules have been obtained from the database, they can be applied to an input image to suggest various recoloring schemes to generate multiple suggested recolorings, allowing the user to choose the output they prefer. Figure 2 shows our pipeline.

Offline learning is based on a database of 3000 well-colored photos. We build a RegionNet for each. To capture artistic character from the database, we train logistic regression models which can predict distributions of color properties for regions based on geometric features. To capture natural character, we construct hue distributions for textures in the database. In the online stage, after building a RegionNet for the input image, we predict distributions of color properties based on the geometric features in the RegionNet, and use a factor graph to combine all distributions to generate an overall distribution for color suggestions. We can use MCMC to sample it to get as many recolorings as wanted.

To get well-colored photos, we downloaded 10000 photos from the Internet and asked students majoring in photography to manually select 3000 which were artistic yet natural. The photos fell into categories including scenery, close-ups and city view. The input image to be recolored should belong to at least one of these categories to achieve satisfactory recoloring results, as content arrangement and color character vary between categories. Obviously, the system could be trained with any other desired categories.

Our RegionNet is a graph. Its nodes correspond to image regions, storing mean color, a geometric feature vector and a texture descriptor for each. Its edges summarize geometric relationships between regions. Each region is classified as either *main* or *detail* according to size. Main regions have large areas and visually dominate the image; color processing is based around them.

To capture artistic character from the image database, we train a multinomial logistic regression model for each unary color property (such as the lightness of a region) to predict a distribution of color property values based on the geometric features of a single main region. We also train a similar model for each pairwise color property (such as the difference of saturation between two regions) based on the geometric features of a pair of adjacent main regions. To capture natural character from the database, we cluster texture descriptors for main regions and calculate a hue histogram for each class. However, for detail regions, similar content may appear just a few times in the database, so no useful color property distributions can be learned: we ignore them during offline learning.

In the online stage, given an input image, we build a Re-

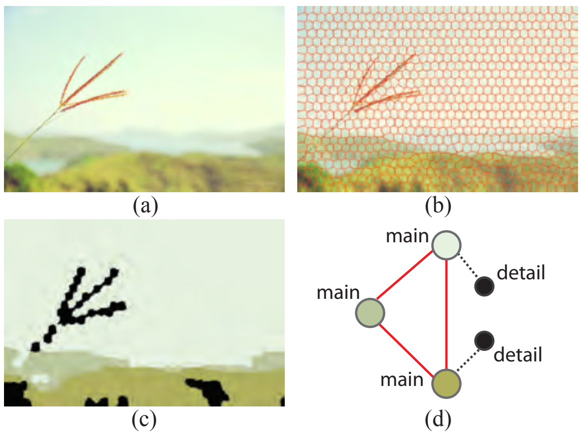

Figure 3: *Constructing a RegionNet. (a) An input image is (b) divided into superpixels, which are grouped by color to give (c) regions, classified as either main (colored) or detail (black). (d) A RegionNet stores region colors, geometric features of regions and their geometric relationships.*

gionNet for it. To ensure artistic results, new color property distributions (both unary and pairwise) for main regions are predicted using corresponding multinomial logistic regression models using the geometric features of the input image. To ensure natural results, when given a texture descriptor corresponding to a main region to be recolored, we use a k nearest neighbor approach to determine its class, and find the corresponding hue histogram. We refer to the mapping between regions and their proposed colors as a color suggestion for an image. In order to combine all color property distributions to generate an overall probabilistic distribution for color suggestions, a probabilistic graph model called factor graph is used. Using an MCMC approach, we sample multiple color suggestions to match the predicted overall distribution for color suggestions. After colors for main regions are determined, colors for detail regions are simply suggested by a heuristic rule to be compatible with colors of main regions. Edit propagation is then used to propagate color suggestions from region level to pixel level. Further user interaction can be provided, e.g. to hold one region unchanged while suggesting colors for other regions.

4. RegionNets

We now discuss the RegionNet structure. Each node represents a region. For each main region, it stores its mean color, a geometric feature vector and a texture descriptor. For each detail region, it only stores the mean color. Edges store geometric relationships between adjacent regions. Figure 3 illustrates our RegionNet construction process. Firstly, the image is segmented into regions with similar color, and then classified as either main or detail regions according to area. Next, we extract geometric features and texture descriptors for main regions, and geometric relationships between regions. Finally, the graph is built.

4.1. Region Extraction

We now consider how to extract regions with similar color. Segmentation of natural images is challenging, and state-of-the-art algorithms , e.g. [AMFM11], can be time consuming. However, careful segmentation is not necessary when learning color property distributions: the spatial arrangement of colors using coarse regions suffices to describe the visual color properties of natural images. Variations in segmentation caused by changes to thresholds or constants also accordingly have little effect. For rapid segmentation, we first divide the input image into *superpixels* [ASS*12], which typically belong to a single object, and have a similar size. They allow us to exploit local similarity of appearance and avoid the need to compare individual pixels when determining regions. Using 800 superpixels gives a good balance between time and precision of regions. This number was determined by trial and error, and does not depend on image size: it is the *relative* sizes of regions and geometric features which are important. To combine superpixels with similar color into regions, we first compute the mean color of each superpixel, and its mean gradient magnitude using a Sobel filter. Using a superpixel occupancy map Q that marks superpixels not yet covered, we then iteratively generate initial regions by combining similar superpixels:

1. Choose the unoccupied superpixel in Q with minimal mean gradient magnitude as a representative superpixel S_r for a new initial region R_n.
2. Find all superpixels in the image with similar appearance to S_r and add them to the current region R_n. A pair of superpixels S_r and S_s are similar if their color distance $d_c(S_r, S_s) < \delta_c$ and gradient distance $d_g(S_r, S_s) < \delta_g$. d_c is the L_2 norm distance in Lab space and d_g is the absolute difference between mean gradient magnitudes. δ_c and δ_g are constants balancing color consistency within each region and the number of regions. They are set to 0.07 and 0.05, determined by trial and error; the method is insensitive to changes in these values since coarse regions suffice to represent the spatial arrangement of colors.
3. Mark superpixels in R_n as occupied in Q.

Note that a region may be disconnected. Initially, a superpixel may belong to more than one region: this helps overcome the problem that the first region a superpixel allocated to may not be the best one (as we use a threshold-based method). We resolve ambiguities by first merging each pair of initial regions which overlap by more than 20% of the area of either. Any remaining shared superpixels are assigned to the region whose representative superpixel has smallest color distance in Lab space to it. Our implementation takes about 5 s to segment a 960×640 image; a pixel-based approach as used by PatchNets takes about 50 s for a similar task [HZW*13].

4.2. RegionNet Construction

We now construct a RegionNet from the extracted regions. Main regions are those with more than 30 superpixels. We compute their mean color, and several geometric features: **Normalized Area**: region area / total image area. **Normalized Centroid**: the region's centroid assuming the image's dimensions are $[0,1]^2$. **Squared Relative Perimeter**: perimeter^2 / area. **Texture Descriptor**: the mean and standard deviation of pixelwise Gabor filter responses (at 4 scales and 6 orientations) within the region [MM96]. This 48-dimensional texture descriptor for every region is resolution independent. The geometric features, apart from the texture descriptor, are extracted into a geometric feature vector, which is later used to learn *artistic* character from the database. The texture descriptors are used to learn *natural* character from the database.

For each detail region, we extract the mean color, which provides a suitable color for each such region.

These features and descriptors for each region are now assembled into a RegionNet. Each node for a main region stores the geometric feature vector, the texture descriptor and the mean color; for a detail region, it stores the mean color. For every pair of main regions (R_i, R_j), we compute an adjacency strength, the total number of pixels from either R_i or R_j that are within a two-pixel distance from the other, normalized by the sum of all adjacency strengths. We add an edge to the RegionNet connecting the corresponding nodes, storing the adjacency strength, if it is not zero.

5. Learning Colors from the Database

In this section, we explain how we learn to predict artistic yet natural color suggestions for main regions from the information in the database.

5.1. Learning to be Artistic

Predicting color *properties* instead of colors generalizes better to colors absent from the training dataset; the relative importances of color properties can be adjusted via weights. For each main region of an input image, we predict *unary color properties*: **Hue**, **Saturation**, **Lightness**. For each pair of adjacent main regions, we predict *pairwise color properties*: **Perceptual Distance** (L_2 norm distance between colors in Lab space), and **Hue Distance**, **Saturation Distance**, and **Lightness Distance**, as absolute channel differences in HSV space (hue treated as a circular quantity).

We wish to predict a distribution of color property values for each color property π separately. We use the k-means algorithm to discretize the property space into $k = 10$ bins. We then train a multinomial logistic regression model on (π, f) pairs from the database. For unary color properties, f is the geometric feature vector of a main region. For pairwise color properties, f comprises the concatenated geometric feature

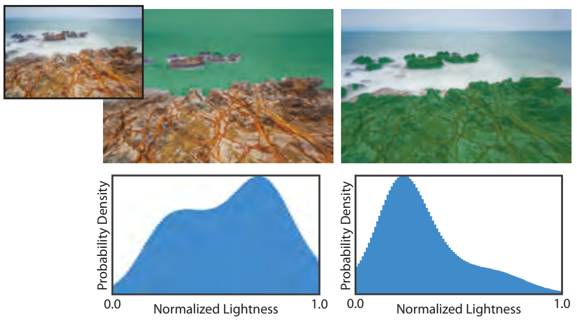

Figure 4: *Lightness distributions for different regions. Top: Input image and two main regions (in green). Bottom: Probability density distributions of lightness for these regions*

vectors of the pair of adjacent main regions, ordered such that the vector with the smaller L_2 norm comes first. (The geometric feature vector does not include the texture descriptor: see Section 4.2). Given this model, for any new input f, we can predict the probability that its color property falls into each bin: we can predict a color property histogram for f. Using a Gaussian mixture model, we can estimate the underlying probability density distribution $p(\pi(c)|f)$ for each histogram, following [WYW*10]. Each color property $\pi(c)$ is a function of a color suggestion c, which represents a suggested mapping between colors and regions. Figure 4 illustrates distributions of lightness for two different regions in an image. For a unary color property, $p(\pi(c)|f)$ indicates the suitability of a certain color suggestion for a main region. For a pairwise color property, $p(\pi(c)|f)$ indicates whether two adjacent main regions are colored compatibly.

5.2. Learning to be Natural

Although semantic information is already encoded in the geometric feature vector, e.g. regions near the top are more likely to be sky, this is insufficient to ensure that color suggestions will result in natural colors. However, *textures* are also a strong cue to the type of an object. We thus learn color property distributions for texture classes to help suggest natural colors. Putting texture information in the same vector as the other geometric features would result in a feature space with too many dimensions for effective learning, so we treat the texture information separately. We follow [WYW*10] to capture texture-color relationships. A k-means algorithm is used to cluster texture descriptors of all main regions in the database into k classes (empirically, $k = 200$ following [WYW*10]). We again ignore detail regions as before. We then make a hue histogram for each texture class, and a Gaussian mixture model is used to estimate probability density functions for these histograms; these indicate how natural a given hue is for some texture class. Given the texture descriptor of an input image region, a k-nearest neighbours approach is used to find the texture class it belongs to. The corresponding hue distribution can evaluate the naturalness of a color for that input region.

5.3. The Factor Graph Model

We use a factor graph model to compute the overall probability distribution for color suggestions for an input image. This probabilistic graphical model factorizes the complex overall probability distribution for color suggestions into factors, which are simpler probability density functions for color properties. For each main region node N_i in the RegionNet, its unary factor in the factor graph is:

$$F_{\text{unary}}(c_i) = F_1 F_2, \qquad \text{where} \tag{1}$$

$$F_1 = \exp\big(\sum_k w_k \ln p_k(\pi(c_i)|f_i) + w_t \ln p_t(\pi(c_i)|t_i)\big), \tag{2}$$

$$F_2 = \mathcal{N}(|c_i - c_i'|, \sigma_n). \tag{3}$$

Here c_i is the proposed color, f_i is the geometric feature vector, t_i is the texture descriptor, p_k are probability densities for color properties, p_t is the hue distribution for t_i's texture class, w_k weight different unary color properties, w_t weights the texture, $|c_i - c_i'|$ is the distance from the proposed color c_i to the original mean color c_i' in Lab space, and $\mathcal{N}(|c_i - c_i'|, \sigma_n)$ is a Gaussian of width σ_n. A value of 0.03 was used in our experiments, but this can be increased if users prefer greater color change.

Similarly, each pair of adjacent main nodes N_i and N_j in the RegionNet has a pairwise factor in the factor graph:

$$F_{\text{pairwise}}(c_i, c_j) = \exp\big(\sum_l w_l B_{ij} \ln p_l(\pi(c_i, c_j)|f_{ij})\big), \tag{4}$$

where B_{ij} is the adjacency strength of the two regions, f_{ij} is their concatenated geometric feature vector, p_l are probability density functions for different pairwise color properties, and w_l weights different pairwise color properties.

Finally, to produce an overall harmony between all colors, we add a global factor for color compatibility:

$$F_{\text{global}}(c) = \exp\big(w_{\text{cp}} \ln(\text{cp}(c_1, \cdots, c_5)/5)\big), \tag{5}$$

where c denotes a color suggestion for all main regions, $\text{cp}(c_1 \cdots c_5)$ is the color compatibility score in $[0,5]$ as given in [OAH11], and $c_1, \cdots, c_5$ are the colors of the five largest main regions.

The overall distribution encoded by the factor graph is:

$$p(c|\mathcal{I}:w) = \frac{1}{Z(\mathcal{I}:w)} \prod_m F_m(c|\mathcal{I}:w), \tag{6}$$

where $\mathcal{I}$ is the input image, the F_m are various unary, pairwise and global factors, and $Z(\mathcal{I}:w)$ is the partition function that normalizes the distribution. (Using MCMC to sample results avoids the need to explicitly normalize the probability density function). The weights for different terms are determined using maximum likelihood parameter estimation as in [LRFH13]. Given a database $\mathcal{D}$, gradient ascent is used to maximize the log-likelihood of the weights:

$$l(w:\mathcal{D}) = \ln \prod_{(\mathcal{I},c)\in\mathcal{D}} \Big(\frac{1}{Z(\mathcal{I}:w)} \prod_m F_m(c|\mathcal{I}:w)\Big). \tag{7}$$

Values are given in the supplementary materials.

6. Recoloring Results Generation

This learned factor graph model can now be used to generate multiple recoloring results for an input image.

6.1. Main Regions

We generate color suggestions for main regions using MCMC sampling of the modelled overall distribution, as in [LRFH13]. Color suggestions with higher probability density, i.e. which are more artistic and natural, are more likely to appear. We use the Metropolis-Hastings algorithm [MRR*04, Has70], a variant of MCMC, with parallel tempering [Gey91] using 5 chains at temperatures (1.0, 0.5, 0.2, 0.05, 0.01). ‘Hotter’ chains jump across the state space, while ‘colder’ chains perform local hill-climbing. We sample the RGB cube in $[0,255]^3$, to ensure colors fall within the display gamut. The maximum marginal relevance criterion [CG98] is used to re-rank the 5000 sampling results from all 5 chains to improve diversity; the top 10 suggestions are used as a basis for generating output images.

6.2. Detail Regions

Having determined colors for the main regions, we use a heuristic to suggest colors for detail regions. The probability that a region has color c is:

$$p(c) = \mathcal{N}(|c - c'|, \sigma_n)\text{cp}(c_1, \cdots, c_4, c)/5, \tag{8}$$

where cp is the color compatibility score from Section 5.3, $\mathcal{N}$ is the same Gaussian controlling the amount of color change from Section 5.3, c' is the original mean color of the detail region, c is the suggested color for the detail region, and $c_1, \cdots, c_4$ are the colors of the four largest main regions adjacent to the detail region. If there are fewer than 4 such main regions, the color of the largest main region c_1 is repeatedly used instead. The Metropolis-Hastings algorithm is used to sample 20 color suggestions; we use the one with the highest probability density for each detail region. Parallel tempering or a greater number of samples are not needed, as we only need an *acceptable* choice for detail regions.

6.3. Smooth Recoloring

The resulting color suggestions are simply an *average* color for each region. Although every pixel belongs to a single region, we compute its output color using all nearby regions—without doing so, color discontinuities would result at region boundaries. We assign weights for these regions to each

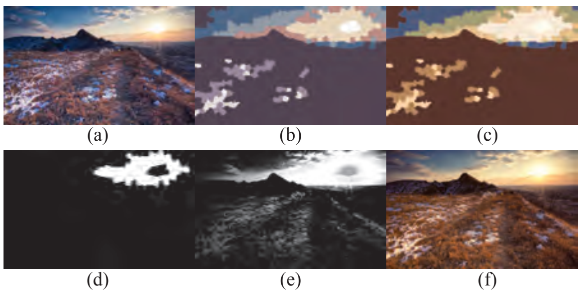

Figure 5: *(a) An input image. (b) Original mean colors for regions. (c) Suggested colors for regions. (d) A region mask. (e) Editing weight map generated by edit propagation from the region mask. (f) Smoothly recolored result.*

pixel, and determine the color accordingly; see Figure 5. Taking each region R_i, we morphologically erode it by size r, half the average radius of a superpixel, to give a mask inside the region. This mask is used as an initial 'stroke' to control an edit propagation technique [LJH10] which generates an edit weight map w_{ij} in $[0,1]$ for every pixel p_j: the larger the weight, the greater the influence region R_i has upon the pixel when determining its color. We combine these weight maps to give a blending weight B_{ij} for pixel p_i and region R_j:

$$B_{ij} = w_{ij} / \sum_{j=1}^{N} w_{ij}, \tag{9}$$

where N is the number of regions. For each channel in Lab space, the final color for pixel p_i is a linear combination of the color changes suggested by the different regions:

$$c_i = \sum_{j}^{N} B_{ij}(c'_i + C_j - C'_j), \tag{10}$$

where N is the number of regions, C_j is the suggested average color for R_j, C'_j is the original average color for region R_j, and c'_i is the original color for pixel p_i.

7. Results

7.1. Performance

Our algorithm has been implemented in C++ and tested on images from sources such as Flickr. Our experiments used a PC with an Intel Core i7-3770 3.4GHz processor and 8GB RAM. An input image of size 960×640 might typically contain 5 main regions. It takes about 5 s to build a RegionNet, and 7 s to compute blending weights. 1 s is needed to sample and rank 5000 recoloring suggestions for main regions to give 10 best results. Computing colors for detail regions takes under 1 s. The total time to recolor a new input image is thus about 14 s. Most time is spent constructing a RegionNet and computing blending weights. Both steps can be parallelized. Note that these steps do not have to be repeated to add different user constraints or use a model learned from another database.

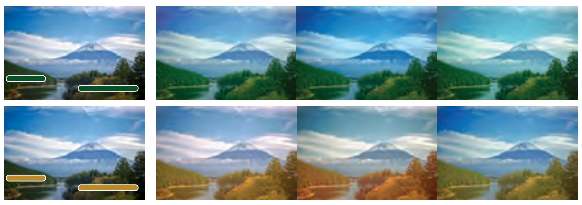

Figure 8: *Top: Hard color constraints fix the color of the trees; the background is adjusted to suit. Bottom: Soft color constraints fix the color of the trees to lie in a certain range.*

Figure 1 shows the three top ranked results with their segmentation and suggested colors for regions. Appealing colors result for the sky, clouds and woods which vary from picture to picture but always stay natural. Figure 6 shows the six top ranked results for another image. The color of the sky varies more widely, whereas the grass always tends to remain green or yellow. (Also see the supplementary materials). Multiple artistic yet natural recoloring results are achieved for each input image; users can choose from them according to their personal preferences. Figure 7 shows recoloring results which do and do not consider naturalness. The latter are exaggerated and unrealistic, although their color schemes are balanced.

7.2. Recoloring with Color Constraints

Hard constraints. Sometimes, a user may want specific colors for certain regions. This can be achieved by fixing colors for certain regions and sampling compatible colors for the remaining regions during the MCMC process. An example in the first row of Figure 8 fixes the color of the trees to dark green, and only the color of the sky is changed.

Soft constraints. Alternatively, a user might have a general preference for certain regions, and may want to see results having a limited range of colors for such regions. We can handle such soft color constraints by adding an additional unary factor for these target regions R_i in the factor graph:

$$F_{\text{soft}}(c_i|\mathcal{P}) = \mathcal{N}(|c_i - c_t|, \sigma_{\text{soft}}), \tag{11}$$

where c_t is the user's desired color and σ_{soft} controls the extent of the variability of the new color. See the second row of Figure 8, where the trees are generally yellow but vary from photo to photo. The sky changes color to follow.

7.3. Database with a Given Style

As our model captures the artistic character of a database, we can use alternative databases to achieve a desired style. Figure 9 shows use of two different databases, to produce results with 'dark' and 'fresh' appearances.

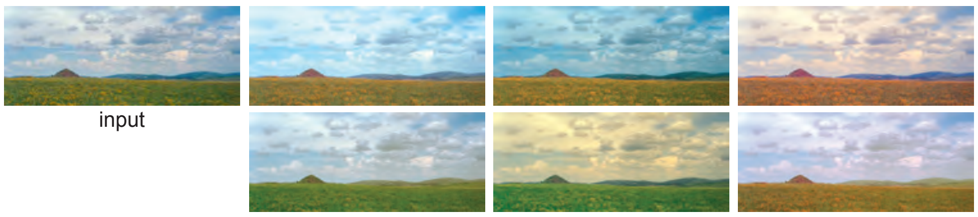

Figure 6: *An input image and the six top ranked recolored results.*

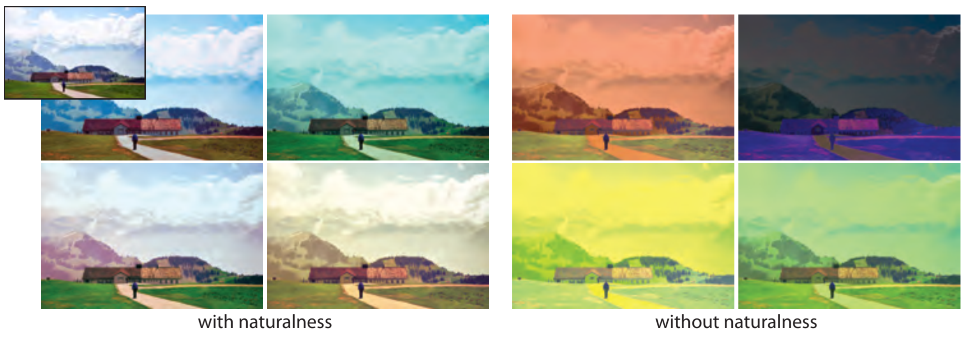

Figure 7: *Left: An input image and recoloring results using terms to achieve natural color. Right: Recoloring results without these terms. Although color schemes are balanced, they are not always plausible.*

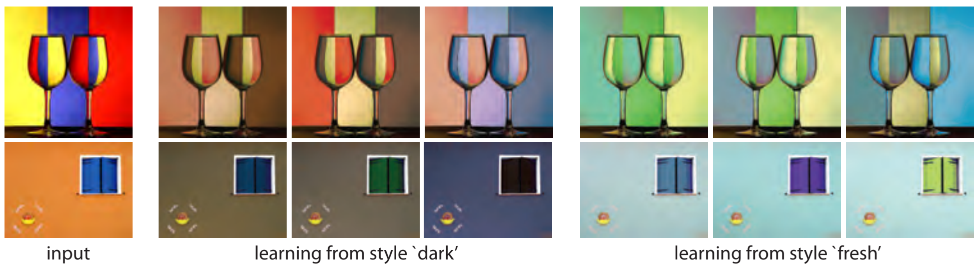

Figure 9: *Left: Input Images. Middle: Results created using 'dark' database. Right: Results crested using 'fresh' database.*

7.4. Comparison to Related Work

Color Theme Enhancement. To recolor an input image using the method in [WYW*10], users must choose a color theme. The output is a single recolored result: see Figure 10(b). Our method can also perform color theme enhancement if given a theme as an extra input. Taking an input image and a theme, we can use our model to compute a score for each possible assignment of the color theme to the image regions. The result with the highest score achieves a satisfactory color theme enhancement effect: see Figure 10(c).

Sometimes a poorly chosen color theme may lead to a disappointing result, as shown in Figure 10(d). A good color theme for one image may be a bad choice for another, due to differences in content. It can be difficult for users to choose a color according to input. Our method can generate recolored results without the need to choose a theme: see Figure 10(e).

Color Transfer. Color transfer methods (e.g. [PKD07, WDK*13]) recolor images based on a user-chosen reference image—see Figures 11(a),(b),(c). Results can be unsatisfactory if the source and reference images are incompatible: Figures 11(b), (c) are over-saturated. Although [WDK*13] tries to transfer color between regions with the same semantic meaning, the user may not always provide a good reference. In contrast, our approach can generate artistic and natural results without the need for a reference image. Results generated by our approach are shown in Figures 11(d). We can also control recoloring by adding hard or soft color constraints, which is easier than using a reference image.

7.5. User Study

We carried out a user study to compare our recoloring approach to other automatic methods. We invited 15 males

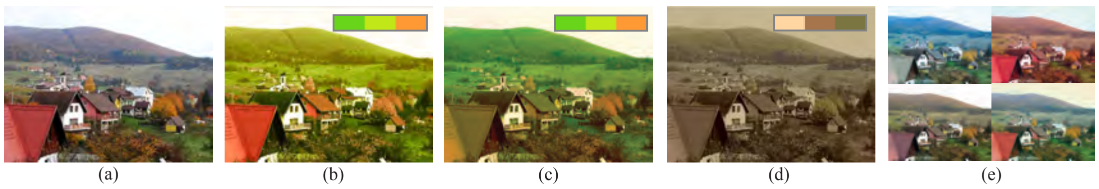

Figure 10: *(a) An input image. (b), (d) Enhancement results using the method of [WYW* 10], based on a chosen theme. (c) A color theme enhancement result generated by our method when given an extra color theme input. (e) We can also generate multiple satisfactory recolored results without choosing a certain color theme.*

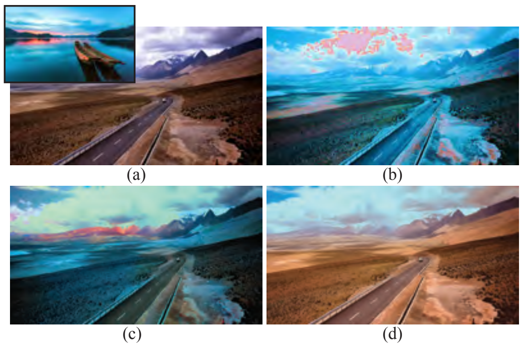

Figure 11: *(a) A source image and a reference image. (b) Color transfer result by [PKD07]. (c) Color transfer result by [WDK* 13]. (d) The first recolored result generated by our method.*

and 9 females, aged 22 to 30, to participate in our study. We randomly selected 30 input photos and recolored them using 4 approaches: manual adjustment by artists, random recoloring, recoloring without considering naturalness (using [LRFH13]) and our method. For the first two approaches, two arbitrary results were used; for the latter, the two top ranked results were taken. Participants were presented with these 8 recolored results for each input photo and asked to score them in a range 1–5 according to color. The input image and all 8 recoloring results were shown in a random order on the same screen at the same time. The average scores and standard deviations for the 4 methods were $(4.4, 1.4, 1.4, 3.8)$ and $(0.31, 0.27, 0.29, 0.33)$. Recolored results produced by our method achieved significantly higher scores. Results which do not consider naturalness score no better than randomly recolored results. Although we do not achieve the scores of manually adjusted results, most participants suggested that it is mainly because artists also enhance sharpness, local contrast and other aspects which contribute to the overall quality of a photo. Figure 12 shows that our result produces comparable colors to artists' work, but that local contrast of the grass and flower affect the impression.

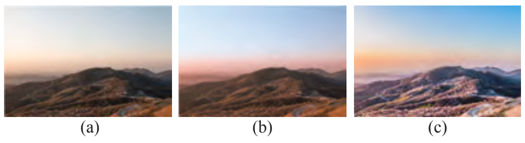

Figure 12: *(a) An input image. (b) One of our top 10 results. (c) Artist's work.*

8. Limitations and Future Work

Our system has several limitations which merit further work. Firstly, our framework uses just one plausible set of color properties and geometric features. Other color distances, color spaces or geometric features are worth investigating, e.g. CIEDE2000 distance, LCH space or contour shapes. Additional use of location as well as texture might provide improved object-color associations. Secondly, our results do not yet achieve the same quality as manually adjusted photos—other factors than color (e.g. sharpness) are also important to image appearance. Thirdly, textureless regions do not give clues to natural color, and we can only rely on the original color to determine the output color. User assistance is necessary for such regions. More sophisticated techniques like in [FCNL13] based on semantic tagging could help. Fourthly, our model is based on average region colors, ignoring color differences within each region. Pixels in the same region undergo similar color changes, which is not always appropriate. Extending the ideas from images to video is a further obvious direction.

9. Conclusions

We have presented a data-driven method for recoloring images, giving artistic and natural results by sampling color suggestions with high probabilistic density in a distribution encoded by a factor graph This summarizes multiple color property distributions learned from geometric features of image regions stored in a compact representation called a RegionNet. Our results and user studies show the effectiveness and flexibility of our method.

Acknowledgements

We thank the reviewers for helpful comments. This work was supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Numbers 61120106007, 61373069), the National High Technology Research and Development Program of China (Project Number 2012AA011802), a Research Grant from the Beijing Higher Institution Engineering Research Center, the Tsinghua University Initiative Scientific Research Program, and a UK EPSRC Travel Grant.

References

[AMFM11] ARBELAEZ P., MAIRE M., FOWLKES C., MALIK J.: Contour detection and hierarchical image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence 33*, 5 (2011), 898–916. 4

[ASS*12] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SUSSTRUNK S.: Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2274–2282. 4

[BSPP13] BONNEEL N., SUNKAVALLI K., PARIS S., PFISTER H.: Example-based video color grading. *ACM Trans. on Graphics 32*, 4 (2013), 39. 2

[CG98] CARBONELL J., GOLDSTEIN J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of the 21st Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval* (1998), pp. 335–336. 6

[CHS08] CHARPIAT G., HOFMANN M., SCHÖLKOPF B.: Automatic image colorization via multimodal predictions. In *Proc. ECCV 2008* (2008), pp. 126–139. 2

[CZG*11] CHIA A. Y.-S., ZHUO S., GUPTA R. K., TAI Y.-W., CHO S.-Y., TAN P., LIN S.: Semantic colorization with internet images. *ACM Trans. on Graphics 30*, 6 (2011), 156. 2

[FCNL13] FARABET C., COUPRIE C., NAJMAN L., LECUN Y.: Learning hierarchical features for scene labeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence 35*, 8 (2013), 1915–1929. 9

[FFP05] FEI-FEI L., PERONA P.: A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (2005), vol. 2, pp. 524–531. 2

[FPC*14] FARIDUL H. S., POULI T., CHAMARET C., STAUDER J., TRÉMEAU A., REINHARD E., ET AL.: A survey of color mapping and its applications. In *Eurographics 2014-State of the Art Reports* (2014), pp. 43–67. 1, 2

[GCR*12] GUPTA R. K., CHIA A. Y.-S., RAJAN D., NG E. S., ZHIYONG H.: Image colorization using similar images. In *Proc. of the 20th ACM international conference on Multimedia* (2012), pp. 369–378. 2

[Gey91] GEYER C. J.: Markov chain monte carlo maximum likelihood. In *Proc. of the 23rd Symposium on the Inteface: Computing Scicence and Statstics* (1991), pp. 156–163. 6

[Has70] HASTINGS W. K.: Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*, 1 (1970), 97–109. 6

[HCX*13] HU S.-M., CHEN T., XU K., CHENG M.-M., MARTIN R.: Internet visual media processing: a survey with graphics and vision applications. *The Visual Computer 29*, 5 (2013), 393–405. 1

[HZW*13] HU S.-M., ZHANG F.-L., WANG M., MARTIN R. R., WANG J.: Patchnet: a patch-based image representation for interactive library-driven image editing. *ACM Trans. on Graphics 32*, 6 (2013), 196. 2, 4

[KWR07] KANNAN A., WINN J., ROTHER C.: Clustering appearance and shape by learning jigsaws. In *Advances in Neural Information Processing Systems* (2007). 2

[LCG10] LIANG C.-K., CHEN W.-C., GELFAND N.: Touchtone: Interactive local image adjustment using point-and-swipe. *Computer Graphics Forum 29*, 2 (2010), 253–261. 2

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Trans. on Graphics 25*, 3 (2006), 646–653. 2

[LJH10] LI Y., JU T., HU S.-M.: Instant propagation of sparse edits on images and videos. *Computer Graphics Forum 29*, 7 (2010), 2049–2054. 2, 7

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Trans. on Graphics 23*, 3 (2004), 689–694. 2

[LRFH13] LIN S., RITCHIE D., FISHER M., HANRAHAN P.: Probabilistic color-by-numbers: suggesting pattern colorizations using factor graphs. *ACM Trans. on Graphics 32*, 4 (2013), 37. 1, 2, 6, 9

[MM96] MANJUNATH B. S., MA W.-Y.: Texture features for browsing and retrieval of image data. *IEEE Trans. on Pattern Analysis and Machine Intelligence 18*, 8 (1996), 837–842. 5

[MRR*04] METROPOLIS N., ROSENBLUTH A. W., ROSENBLUTH M. N., TELLER A. H., TELLER E.: Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*, 6 (2004), 1087–1092. 6

[OAH11] O'DONOVAN P., AGARWALA A., HERTZMANN A.: Color compatibility from large datasets. *ACM Trans. on Graphics 30*, 4 (2011), 63. 6

[PKD07] PITIÉ F., KOKARAM A. C., DAHYOT R.: Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1 (2007), 123–137. 2, 8, 9

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 5 (2001), 34–41. 2

[TJT05] TAI Y.-W., JIA J., TANG C.-K.: Local color transfer via probabilistic segmentation by expectation-maximization. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (2005), vol. 1, pp. 747–754. 2

[WAM02] WELSH T., ASHIKHMIN M., MUELLER K.: Transferring color to greyscale images. *ACM Trans. on Graphics 21*, 3 (2002), 277–280. 2

[WDK*13] WU F., DONG W., KONG Y., MEI X., PAUL J.-C., ZHANG X.: Content-based colour transfer. *Computer Graphics Forum 32*, 1 (2013), 190–203. 2, 8, 9

[WJC13] WANG X., JIA J., CAI L.: Affective image adjustment with a single word. *The Visual Computer 29*, 11 (2013), 1121–1133. 2

[WYW*10] WANG B., YU Y., WONG T.-T., CHEN C., XU Y.-Q.: Data-driven image color theme enhancement. *ACM Trans. on Graphics 29*, 6 (2010), 146. 2, 5, 8, 9

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using kd tree. *ACM Trans. on Graphics 28*, 5 (2009), 118. 2