Local texture-based color transfer and colorization

Benoit Arbelot, Romain Vergne, Thomas Hurtut, Joëlle Thollot

Local Texture-Based Color Transfer and Colorization

B. Arbelot[a], R. Vergne[a], T. Hurtut[b], J. Thollot[a]

[a] *Univ. Grenoble Alpes, CNRS, Inria*
[b] *Polytechnique Montréal*

Abstract

This paper targets two related color manipulation problems: *Color transfer* for modifying an image's colors and *colorization* for adding colors to a grayscale image. Automatic methods for these two applications propose to modify the input image using a reference that contains the desired colors. Previous approaches usually do not target both applications and suffer from two main limitations: possible misleading associations between input and reference regions and poor spatial coherence around image structures. In this paper, we propose a unified framework that uses the textural content of the images to guide the color transfer and colorization. Our method introduces an edge-aware texture descriptor based on region covariance, allowing for local color transformations. We show that our approach is able to produce results comparable or better than state-of-the-art methods in both applications.

Keywords:
Texture Analysis, Color Transfer, Colorization, Stroke-based Edition

1. Introduction

In this paper, we propose a method to automatically apply local color transfer and colorization between images. Manually colorizing a grayscale image, or tuning colors to obtain a desired ambiance is challenging, tedious and requires advanced skills. Exemplar-based methods offer an intuitive alternative by automatically changing colors of an **input** image according to a **reference** image (the exemplar) containing the desired colors. The main challenge of these methods is to accurately match content between the input and reference image.

The first color transfer algorithms were based on global approaches reshaping the input image color histogram to match the histogram of the reference image. While these approaches can be simple and successful with carefully chosen image pairs, they often mismatch regions in the input and reference images, and are not suited for the colorization problem when the input image does not have a color histogram to begin with.

Alternatively, local approaches (soft-)segment an image into several subregions that can be processed independently. Colors are then added or transferred between similar regions. Those regions can be either manually provided, or automatically computed based on image descriptors.

Our approach is automatic and relies on regions defined as areas of similar textural content. This choice was driven by the fact that textures can be found everywhere in nature, and thus in a lot of photographs. Moreover, perceptual studies showed that the early stages of human vision are composed of several filters to analyze textures and color variations in our visual field [1, 2]. This suggests that textures are important when observing images and should be a pertinent basis for local color transformations. Furthermore, textures can be efficiently described by a summary of first and second order statistics, and present an attractive middle ground between low-level descriptors (luminance, chromaticity) that cannot efficiently describe textured regions, and high-level descriptors (object and region semantic) that are complex, error-prone and slow to compute.

While our approach automatically matches every region of the input and reference images, as presented in [3], we extend it here to allow the user to define this matching through simple strokes. This is done using the edge-aware texture descriptor introduced in this paper, and gives the user the ability to use different reference images to quickly edit and fine-tune the result of our automatic approach locally. When strokes are given, our texture description is used to automatically segment homogeneously textured regions from the strokes, and restrict the color manipulation to those regions.

To apply color transfer between textured regions, our descriptors are computed on a large scale to be able to characterize large textures, but they must also preserve image structures. Existing methods for texture and structure decomposition are not well suited for our application: edge-aware image descriptors (such as bilateral filtering) have trouble analyzing highly contrasted textures and may introduce discontinuities in the color transfer. The alternative consists in detecting variations of the descriptors themselves (such as region covariance), but in that case, image edges are smoothed, leading to halos in the transfer.

Our solution to estimate texture properties is based on a texture analysis, followed by an edge-aware processing to compute edge-aware texture based descriptors. Our main contribution is to compute accurate textural information while preserving image structure. We use it in a generic framework for local color transfer and colorization between images based on textural properties.

2. Related Work

In this section, we review previous work on color transfer and colorization, before discussing several approaches to extract and analyze textures for image manipulation.

Color Transfer. An extensive review of color transfer methods can be found in [4]. Color transfer consists in changing the colors of an input image to match those of a reference image. It was first introduced in [5] as a simple histogram reshaping, where the mean and variance of each channel are transferred separately, using the decorrelated $L\alpha\beta$ color space. This rather straightforward method can be surprisingly effective with well chosen input images. A rotation component was added in the matching process by Xiao and Ma [6], allowing the transfer to be done in a correlated color space (such as RGB). Instead of processing each channel independently, Pitié *et al.* [7] proposed to tightly match the 3-dimensional histograms using iterative 1-dimensional matchings. While the matching offered by this approach is very good, it is almost "too good" for the color transfer application as it tends to produce artifacts by forcing the input to have exactly the same number of pixels of each color as the reference. Finally, a more recent approach based on multiscale histogram reshaping was proposed in [8] where the user can control how tightly the histograms should be matched. Overall, these global methods are simple, but histogram matchings do not ensure colors to be transferred between similar regions. When such automatic methods fail, manual segmentations can be provided to locally transfer between selected regions [9, 10, 11].

In order to automatically apply a local color transfer, Tai *et al.* [12] used mixtures of Gaussians to segment the input images and transfer colors between regions of similar luminance. A method to color grade videos based on color transfer between sequences was proposed in [13]. Their color transformation segments the images using the luminance and transfer chrominance between shadows, mid-tones and highlight regions. In a similar vein, Hristova *et al.* [14] partition the images into Gaussian distributed clusters considering their main features between light and colors. Color-based segmentation was also used in [15] to extract color palettes and transfer between them using optimal transportation. While more accurate than global transfers, these approaches are still only based on first order information to segment the image and do not take higher order information to match regions between images. Consequently, regions with different textural properties but similar luminance cannot be distinguished.

Other approaches similar to Image Analogies [16] have been applied to color transfer [17, 18]. However they differ from our approach as they use an additional input to compute the transformation.

Colorization. Colorization deals with the problem of adding colors to a grayscale image. One of the first approaches to tackle this issue relies on user input scribbles being extended via optimization across regions of similar luminance [19]. This optimization is used with automatically generated scribbles in a lot of example-based colorization methods [20, 21, 22]. Because they rely on a luminance-based optimization in their final step, these methods tend to have trouble with highly contrasted textures where the optimization does not propagate colors properly. More recently, Jin *et al.* [23] proposed a randomized algorithm to better match color distributions between user segmented regions.

Since last year, deep learning algorithms such as convolutional neural networks were also successfully used for automatic image colorization [24, 25, 26]. However those approaches require extensive datasets to train the algorithms and the learned image statistics are complex and hard to interpret.

Closer to our approach, other methods rely on higher-order information to transfer the chrominance between pixels containing similar statistics [27, 28, 29, 30]. However, they often produce halos due to the window used in the statistics computation. These methods also rely on an energy minimization which typically makes them slow and hard to use on large images.

Texture Analysis. Many different descriptors have been used to manipulate images according to their textural content. Previous automatic colorization methods used SURF, Gabor features, or the histogram of oriented gradients as base tools for texture analysis [28, 21, 22]. These descriptors are known to be discriminative, but also computationally and memory intensive due to their high number of features. Similarly, the shape-based texture descriptors introduced in [31, 32], although offering multiple invariants, are too complex for an image manipulation application where we expect to compute results in a reasonable time for relatively large images. The recent approaches proposed in [33, 34] precisely separate texture from structure using a relative total variation, but their descriptors are not accurate enough to discriminate textures among themselves. Finally, Karacan *et al.* [35] proposed to use region covariance as a texture descriptor for image smoothing. Our method also relies on a variant of this descriptor, as it is compact and efficient in describing textural properties. One main drawback is that most of these descriptors tend to be unreliable around image edges and texture transitions, especially when estimated on large neighborhoods. For that reason, we also briefly describe edge-aware filtering methods that could be used to solve this issue.

Edge-aware filters are crucial to preserve image structures when smoothing, denoising, enhancing details, or extracting textural information from images. A well known approach regarding that goal is the bilateral filter [36], which efficiently smoothes images while mostly preserving luminance edges. However, it tends to locally introduce halos and gradient reversal artifacts which can modify textural properties. The guided filter [37] offers a different approach by using a linear transform of a guidance image to filter an image but may also produce halos around edges. The anisotropic diffusion [38] or the unnormalized bilateral filter [39] are more appropriate for our descriptors, since they avoid both halos and gradient reversal when large scale diffusions are needed.

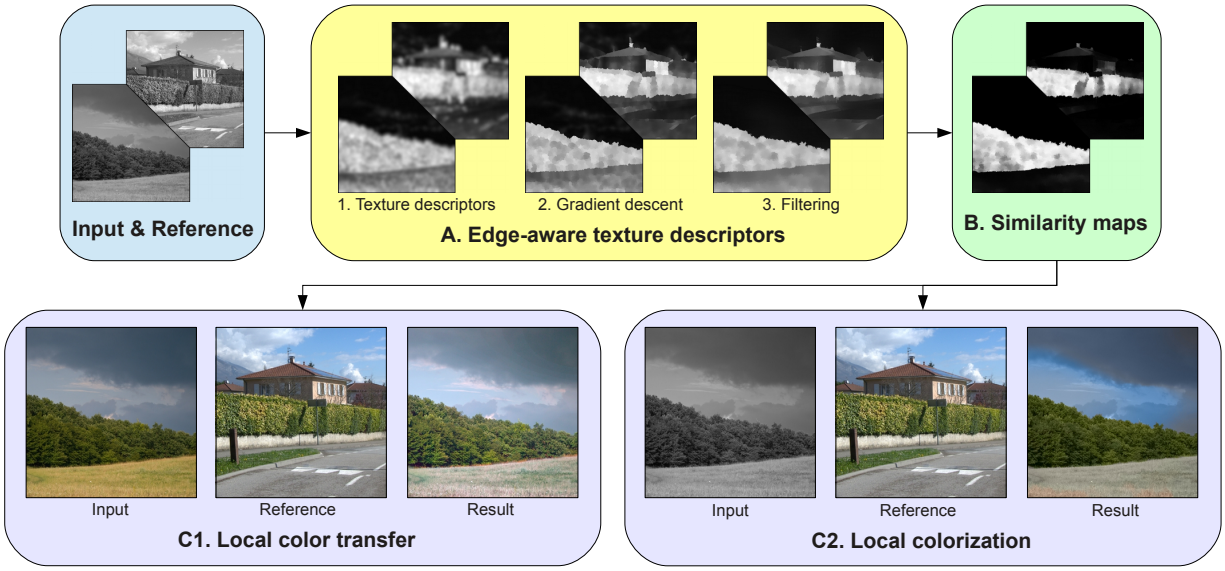


Figure 1: **Pipeline overview.** Edge-aware descriptors are first computed to accurately describe the textural content of the input and reference images (A). They are then used to compute per-pixel distances and allows similar regions to be associated, as shown for the vegetation in (B). We finally use these distance maps for both color transfer (C1) and colorization (C2), where attributed colors depends on pixel similarities.

3. Overview

Our approach for automatically editing image colors based on textural content is summarized in Figure 1. First, descriptors are computed for the input and reference images in three steps (A): covariance matrices of several local image features are computed over a coarse scale to roughly characterize the textural content of each region (A.1). A multi-scale gradient descent then locally displaces descriptors in order to recover texture edges lost during the coarse scale analysis (A.2). Finally, an edge-aware filter is applied to obtain descriptors that accurately discriminate homogeneous textural regions while preserving detailed texture transitions (A.3).

Our descriptors allow the computation of similarities between pixels. As such, they also enable soft segmentations of the input and reference images, where smooth and sharp structures are preserved. This is illustrated in Figure 1 (B), where the vegetation is automatically isolated in both the input and reference images. Finally, similarity maps locally control the transfer of colors between images (C1) or colorize regions according to similar textural content (C2). The remainder of the paper is organized as follows: Descriptors are described in Section 4 and local color manipulation algorithms are detailed in Section 5. Results and comparisons are then presented in Section 6 before concluding in Section 7.

4. Edge-aware Texture Descriptors

4.1. Local Texture Descriptors

We want to analyze the textural information surrounding each pixel in both the input and the reference images. To that end, we chose to use *region covariance* [40, 35] as it is an efficient and compact way of describing image regions. Region covariance captures the underlying texture by computing a small set of second order statistics on specific image features such as the luminance or the gradient. Let us consider a pixel $\mathbf{p}$, described by a d-dimensional feature vector $\mathbf{z}(\mathbf{p})$. The region covariance is defined as the following $d \times d$ covariance matrix:

$$\mathbf{C}_r(\mathbf{p}) = \frac{1}{W} \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} (\mathbf{z}(\mathbf{q}) - \boldsymbol{\mu}_r)(\mathbf{z}(\mathbf{q}) - \boldsymbol{\mu}_r)^T w_r(\mathbf{p}, \mathbf{q}),$$

where $N_r^{\mathbf{p}}$ is a square neighborhood centered on $\mathbf{p}$ of size $(2r+1) \times (2r+1)$ and $\boldsymbol{\mu}_r$ is a vector containing the mean of each feature inside this region. Unlike [40], we add a Gaussian weighting function with standard deviation $r/3$ that ensures descriptors to be smoothly defined from pixel to pixel: $w_r(\mathbf{p}, \mathbf{q}) = exp(\frac{-9\|\mathbf{q}-\mathbf{p}\|^2}{2r^2})$. Note that this weight function should also be used to compute the mean features $\boldsymbol{\mu}_r$. W is the normalization factor: $W = \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} w_r(\mathbf{p}, \mathbf{q})$. In practice, r should be set to the lowest value that still allows to capture most textural properties accurately. For 512×512 natural images, we typically use $r \in [20, 30]$ and rely on a 6-dimensional feature vector based on luminance derivatives to capture coarse scale textural content:

$$\mathbf{z}(\mathbf{p}) = \left[L(\mathbf{p}) \quad \frac{\partial L(\mathbf{p})}{\partial x} \quad \frac{\partial L(\mathbf{p})}{\partial y} \quad \frac{\partial^2 L(\mathbf{p})}{\partial x^2} \quad \frac{\partial^2 L(\mathbf{p})}{\partial y^2} \quad \frac{\partial^2 L(\mathbf{p})}{\partial x \partial y} \right]^T,$$

where $L(\mathbf{p})$ denotes the luminance of pixel $\mathbf{p}$. In practice, each feature is first centered and normalized (i.e. we substract its mean and divide by its standard deviation) to equally contribute to the analysis. Note that other features, such as color

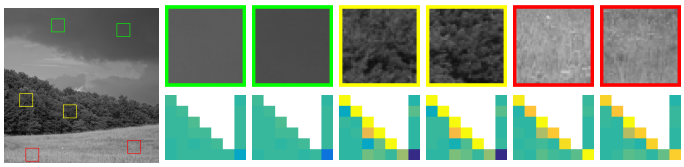
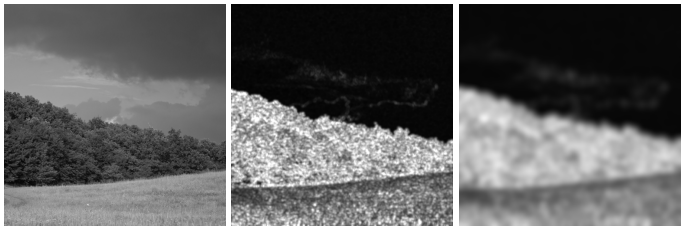


Figure 2: **Texture descriptors.** Patches taken from several regions of the image in Figure 1 (top) and their respective descriptors computed for the central pixel of the window (bottom). Yellow and blue values correspond to positive and negative values respectively. Patches from similar regions have similar descriptors.



(a) Input (b) $\boldsymbol{\mathcal{S}}_6$ (c) $\boldsymbol{\mathcal{S}}_{21}$

Figure 3: **Descriptors scales.** Small scales lead to noisy descriptors (b). Large scales lead to more homogeneous descriptors and smooth sharp texture transitions. For visualization clarity, only the first element of $\boldsymbol{\mathcal{S}}_r$ is shown (i.e. the first value of $\mathbf{L}_r^1$) but the rest of the set presents the same behavior.

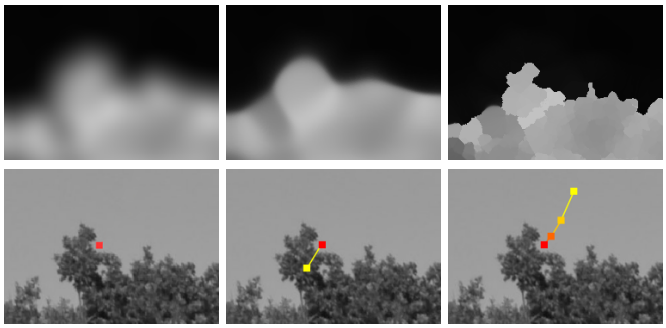
derivatives, could also be used for different applications. For our color manipulations, we found that luminance carried most of the relevant texture information, especially in natural images.

As explained in [41, 35], region covariances only describe second-order statistics, which can be a limitation when describing textural content as it cannot separate two distributions which only vary with their mean. Moreover, computing distances between covariance matrices is expensive because they do not lie in a Euclidean space. We thus follow the solution proposed by Karacan *et al.* [35] who use the Cholesky decomposition to transform covariance matrices into vectors that can be easily compared and enriched with first-order statistics. Our descriptor is then represented by:

$$\boldsymbol{\mathcal{S}}_r = \begin{pmatrix} \mathbf{L}_r^1 & \cdots & \mathbf{L}_r^d & \boldsymbol{\mu}_r \end{pmatrix}, \tag{1}$$

where $\mathbf{L}_r^i$ is the i^{th} column of the lower triangular matrix $\mathbf{L}_r$ obtained with the Cholesky decomposition $\mathbf{C}_r = \mathbf{L}_r \mathbf{L}_r^T$ at scale r and $\boldsymbol{\mu}_r$ are the first-order mean features in the corresponding region.

Visualizations of our descriptors are shown in Figure 2 where we can see that their values are similar when computed on the same types of regions. On the other hand, these values are dissimilar between different regions, making our descriptor able to discriminate different textural regions. Figure 3 shows how descriptors are affected by the scale r. Small scales (b) preserve edges but tend to produce noisy descriptors. Conversely, larger scales successfully describe uniform regions but fail to accurately preserve sharp texture transitions that often occur inside images. This is shown in (c), where the sharp transition between trees and sky is blurred when computing the descriptor with a large neighborhood. This phenomenon is due to the fact



(a) Descriptor $\boldsymbol{\mathcal{S}}_{21}$ (b) Standard descent (c) Multiscale descent

Figure 4: **Gradient descent illustration.** (a) A zoom in the sky/trees transition of the image shown in Figure 3. (b) A gradient descent guided by the variance of the coarse scale descriptor tends to sharpen edges (top), but may mistakenly assign descriptors to the wrong side of the edges: The red sky pixel (bottom) is considered as part of the trees here. (c) A gradient descent gradually performed at multiple scales (from fine to coarse) better preserves complex texture transitions. The red pixel is now successfully assigned to the sky.

that on these particular pixels both tree and sky features are mixed to compute the descriptor, which then tend to represent this transition as a third texture. However, this is problematic for our color manipulation applications, where such descriptors will produce halos around edges. Note that we cannot integrate luminance edges in the weight function w_r (as in the bilateral filter for instance). Indeed, this would prevent highly contrasted textures to be accurately captured since such textures would be fragmented into multiple pieces. For our purpose, we need both constraints to be satisfied: homogeneous descriptors inside regions and sharp texture edges preserved.

4.2. Multiscale Gradient Descent

To prevent texture transitions from being blurred, we propose to use a multiscale gradient descent algorithm to give these regions valid descriptors. Intuitively this multiscale gradient descent locally propagates relevant descriptor values (occurring inside homogeneous textural regions) to replace irrelevant ones (occurring around region borders). In order to do so, we use the variance of the descriptors to guide a gradient descent as this variance is low on homogeneous regions and high around texture edges. This gradient descent will then replace descriptors with high variance by those contained in uniform regions. Formally, the variance of a pixel $\mathbf{p}$ is computed as follows:

$$V_r(\mathbf{p}) = \| \frac{1}{W} \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} (\boldsymbol{\mathcal{S}}_r(\mathbf{q}) - \mathbf{v}_r)(\boldsymbol{\mathcal{S}}_r(\mathbf{q}) - \mathbf{v}_r)^T w_r(\mathbf{p}, \mathbf{q}) \|, \tag{2}$$

where $\boldsymbol{\mathcal{S}}_r(\mathbf{p})$ is the descriptor at pixel $\mathbf{p}$ and $\mathbf{v}_r$ is the weighted average of the descriptors over the neighborhood $N_r^{\mathbf{p}}$.

The gradient descent replaces the descriptors on either side of the variance (*e.g.* texture edges) by descriptors with lower variance, consequently sharpening descriptor edges. Figure 5 (top) shows the pseudo-code of the gradient descent, where the returned map contains the coordinates of the descriptor that should be used for each pixel. The result is shown in the top

row of Figure 4, where initial descriptors (a) are replaced by descriptors from homogeneous regions by following the gradient of the variance (b). The result obviously depends on the scale at which descriptors are computed. On large scales, complex texture transitions are smoothed out and consequently, some descriptors might be incorrectly attributed to different regions. This is illustrated in the bottom row of Figure 4, where the red pixel located in the sky (a) is mistakenly associated with the descriptor of a tree (b) after the gradient descent pass. Our solution to preserve complex texture changes with large scale descriptors is to use a multiscale gradient descent, where the scale of both descriptor and variance are gradually increased to guide the gradient descent of the initial (large scale) descriptor.

Figure 5 (bottom) shows the pseudo-code of the proposed multi-scale gradient descent process. The idea is to iteratively apply gradient descents, from fine to coarse scales, in order to propagate desciptors from homogeneous regions while preserving complex texture edges. At small scales, the descent accurately preserves edges, but quickly falls into local minima. Increasing scales slowly select pixels further and further away from the detailed edges, ensuring that the descriptors are consistent. In practice, the number of iterations used for a given scale is set to the size of the neighborhood (small and large scales may respectively lead to small and large propagations). Note that, even if small scale descriptors are needed to compute the variance, the resulting new coordinates only modify the coarse scale descriptor. The result is shown in Figure 4 (c). The obtained descriptor (top) better preserves complex texture transitions. The red pixel (bottom) now successfully takes descriptor values of a homogeneous region inside the sky.

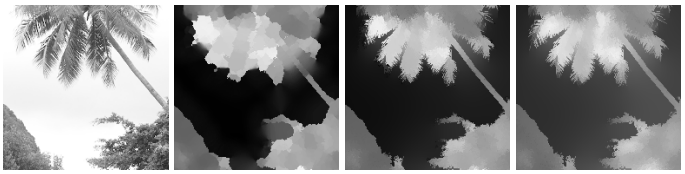
Multiscale gradient descent

Initialize M with pixel coordinates
Compute $\boldsymbol{\mathcal{S}}_{r_{max}}$ using Equation. (1)
for $r = 1$ to r_{max} **do**
    Compute V_r using Equation. (2)
    $M \leftarrow$ Gradient descent(M, V_r, r)
end for
for all pixels **p do**
    $\boldsymbol{\mathcal{S}}_{r_{max}}(\mathbf{p}) \leftarrow \boldsymbol{\mathcal{S}}_{r_{max}}(M(\mathbf{p}))$
end for

Gradient descent

Input: coordinate map M, variance map V, number of steps n
for all pixels **p do**
    for $i = 1$ to n **do**
        $M(\mathbf{p}) \leftarrow M(\mathbf{p}) + \nabla V(M(\mathbf{p}))$
    end for
end for
Return M

Figure 5: Multiscale gradient descent algorithm.



Input (a) (b) (c)

Figure 6: **Unnormalized bilateral filter.** (a) The descriptor obtained from the image after gradient descent. (b) The unnormalized bilateral filter accurately propagates descriptors and follows luminance edges. (c) Without the multiscale gradient descent, halos are propagated inside regions and descriptors are altered. In these examples, we used 2000 iterations with $\sigma_s = 2$ and $\sigma_l = 0.05$.

4.3. Unnormalized Bilateral Filtering

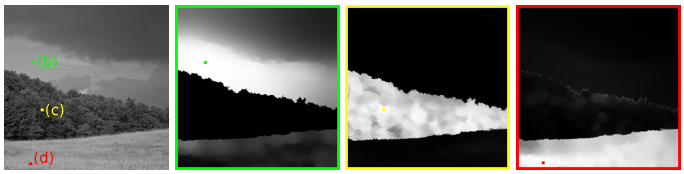
Gradient descent ensures the precise capture of textural properties around each pixel, even near texture edges. Yet, descriptors might still contain some variations that do not appear in the original image. These might happen around U-shaped texture transitions (as in the left part of Figure 4 (c)) or when a region cannot be properly defined by its textural content (such as a fine edge on a uniform background). This has to be prevented since any variations in the descriptors might lead to color changes during transfer or colorization. In a last step, we thus smooth the descriptor using an edge-aware filter to perfectly fit to the image structure. To that end, we adapt the unnormalized bilateral filter [39], such that it iteratively smoothes the descriptor according to luminance variations. This filter is simple, efficient, and introduces very little halos if any. However, any other edge-aware filter could have been used [36, 37, 38]. Formally, we use the unnormalized bilateral filter as follows:

$$\boldsymbol{\mathcal{S}}^{ubf}(\mathbf{p}) = \boldsymbol{\mathcal{S}}(\mathbf{p}) + \sum_{\mathbf{q} \in \mathcal{N}\mathbf{p}} \frac{G_{\sigma_s}(\mathbf{q}-\mathbf{p}) G_{\sigma_l}(L(\mathbf{q})-L(\mathbf{p}))(\boldsymbol{\mathcal{S}}(\mathbf{q})-\boldsymbol{\mathcal{S}}(\mathbf{p}))}{\sqrt{2\pi\sigma_s^2}}, \tag{3}$$

where $G_\sigma(\mathbf{x}) = exp(\frac{-\|\mathbf{x}\|^2}{2\sigma^2})$ is a standard Gaussian kernel. σ_s and σ_l respectively control the influence of spatial distances and luminance variations. In practice, we iteratively apply Equation (3) with rather small values of σ_s and σ_l (typically 2 and 0.05) in order to accurately diffuse descriptors on large neighborhoods. Figure 6 shows the effect of the filter on a problematic region, where the descriptors do not precisely follow edges around the palm tree (a). The unnormalized bilateral filter accurately brings back the leaf edges, as shown in (b). The last image (c) shows the effect of the filter when applied on the original descriptor (i.e. without gradient descent). In that case, halos are propagated inside regions and create unreliable descriptors.

4.4. Effect of Each Step on Color Transfer Result

An example showcasing the effect of each of the previous steps on a color transfer result is given in Figure 7. Initial descriptors (a) tend to produce strong color halos around texture transitions. The multiscale gradient descent (b) produces much sharper transitions, but may also introduce discontinuities



(a) (b) (c) (d)

Figure 8: **Similarity maps.** (a) Input image luminance. The green, yellow and red pixels are compared with all pixels using Equation (4) to obtain the corresponding similarity maps (b), (c) and (d). The similarity measure allows the three regions to be accurately discriminated. Similarities were computed with $\sigma_d = 1$ in these examples.

around U-shaped regions. When applied alone, the unnormalized bilateral filtering (c) smudges halos instead of suppressing them. Combining multiscale gradient descent and unnormalized bilateral filtering (d) creates a clean result where even strong initial halos are efficiently removed.

5. Local Color Manipulation

Now that we have obtained reliable descriptors, we propose to use them for color manipulations by defining transfer functions that only rely on similar pixels between the input and reference images.

5.1. Pixel Similarity

We define a similarity measure based on the $L2$ Euclidean distance between two descriptors:

$$\mathcal{D}_{\sigma_d}(\mathbf{p},\mathbf{q}) = \exp\left(\frac{-\|\boldsymbol{\mathcal{S}}(\mathbf{p})-\boldsymbol{\mathcal{S}}(\mathbf{q})\|^2}{2\sigma_d^2}\right), \tag{4}$$

where $\boldsymbol{\mathcal{S}}(\mathbf{p})$ and $\boldsymbol{\mathcal{S}}(\mathbf{q})$ are the descriptors at locations $\mathbf{p}$ and $\mathbf{q}$ and σ_d is the standard deviation that controls how close descriptors should be to contribute to the similarity measure. Note that other metrics could have been used as detailed in [41, 35], but we did not find any significant differences for our purpose. An example of similarity measure is shown in Figure 8, where pixels (b), (c) and (d) are compared with all the other pixels of the input image (a). We can observe that trees, sky and grass regions are accurately selected and distinguished in the results.

5.2. Color Transfer

The main idea for transferring colors between images is to rely on local histogram matchings between input and reference images, where both sets of color points are defined by their texture similarities. The matching process is based on a translation and scaling of the distribution in a decorrelated color space, as originally proposed by Reinhard *et al.* [5]. Input and reference images are therefore first transformed into the uncorrelated and perceptually uniform CIE-Lab color space before being processed. The following transfer function is then applied on each channel $c \in \{L,a,b\}$ separately:

$$\mathcal{T}_{\sigma_d}(\mathbf{p}) = \frac{std^{ref}(\mathbf{p})}{std^{in}(\mathbf{p})}\left(c^{in}(\mathbf{p})-\mu^{in}(\mathbf{p})\right)+\mu^{ref}(\mathbf{p}), \tag{5}$$

where superscripts "in" and "ref" denote the input and reference images. "μ" and "std" are the weighted mean and standard deviations respectively, computed as follows, according to the similarities of the pixel $\mathbf{p}$ of the input image:

$$\mu^{img}(\mathbf{p}) = \frac{1}{W}\sum_{\mathbf{q}} c^{img}(\mathbf{q})\mathcal{D}_{\sigma_d}(\mathbf{p}^{in},\mathbf{q}^{img})$$

$$std^{img}(\mathbf{p}) = \sqrt{\frac{1}{W}\sum_{\mathbf{q}}\left(c^{img}(\mathbf{q})-\mu^{img}(\mathbf{p})\right)^2\mathcal{D}_{\sigma_d}(\mathbf{p}^{in},\mathbf{q}^{img})},$$

where $img \in \{in, ref\}$, $\mathbf{q}$ iterates over img and W is the normalization factor:
$W = \sum_{\mathbf{q}}\mathcal{D}_{\sigma_d}(\mathbf{p}^{in},\mathbf{q}^{img})$. A color transfer example is shown in Figure 9 (top) where we can observe the effect of the σ_d parameter. When σ_d is small, colors are transferred only between highly similar regions, such as the sea or the clouds of the input and reference images here. Wider and wider regions are considered when increasing σ_d, leading to results closer to the global matching of [5].

5.3. Colorization

Histogram matching techniques cannot be used directly for colorizing images that do not contain chrominance channels. In this case, we assign the mean chrominance of the reference image to each input pixel, weighted by our similarity measure:

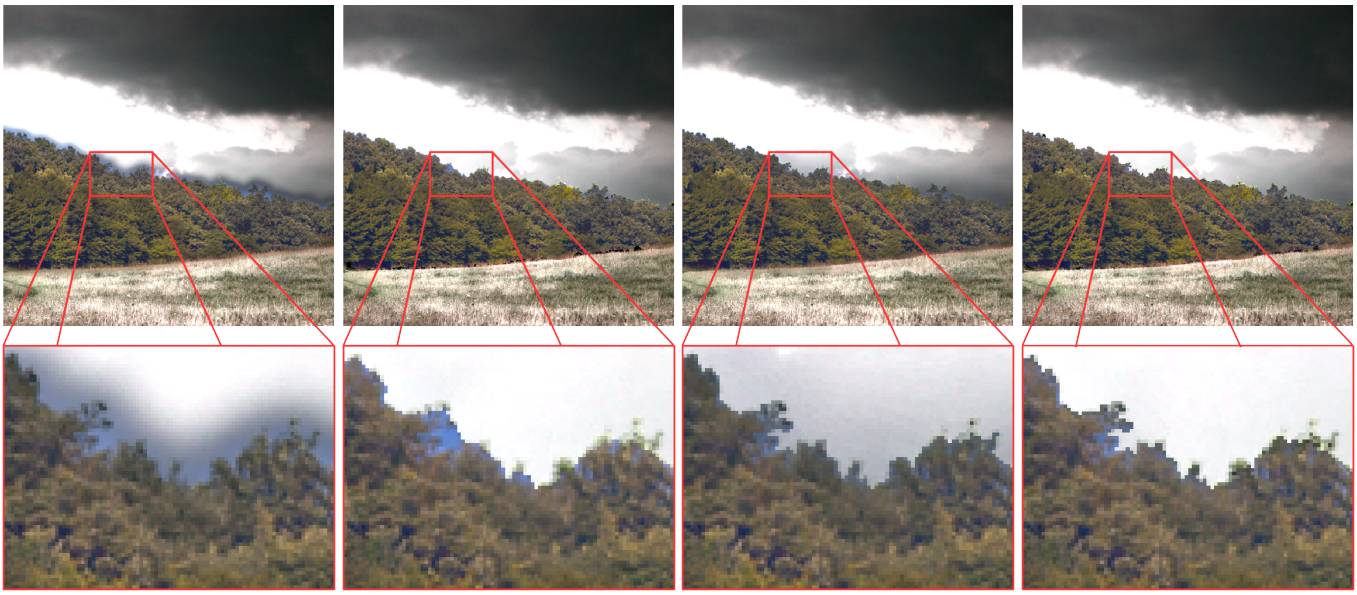
$$C_{\sigma_d}(\mathbf{p}) = \frac{\sum_{\mathbf{q}} c^{ref}(\mathbf{q})\mathcal{D}_{\sigma_d}(\mathbf{p}^{in},\mathbf{q}^{ref})}{\sum_{\mathbf{q}}\mathcal{D}_{\sigma_d}(\mathbf{p}^{in},\mathbf{q}^{ref})}. \tag{6}$$

Note that this transfer function is applied on chrominance channels only, although the luminance could also be modified depending on the purpose. A colorization example is shown in Figure 9 (bottom). Large values of σ_d tend to average colors on large regions and consequently create pale and monochrome results. Therefore σ_d should be kept small enough for colorization purpose, in order to only average colors over regions of highly similar descriptors.

5.4. Implementation & Performances

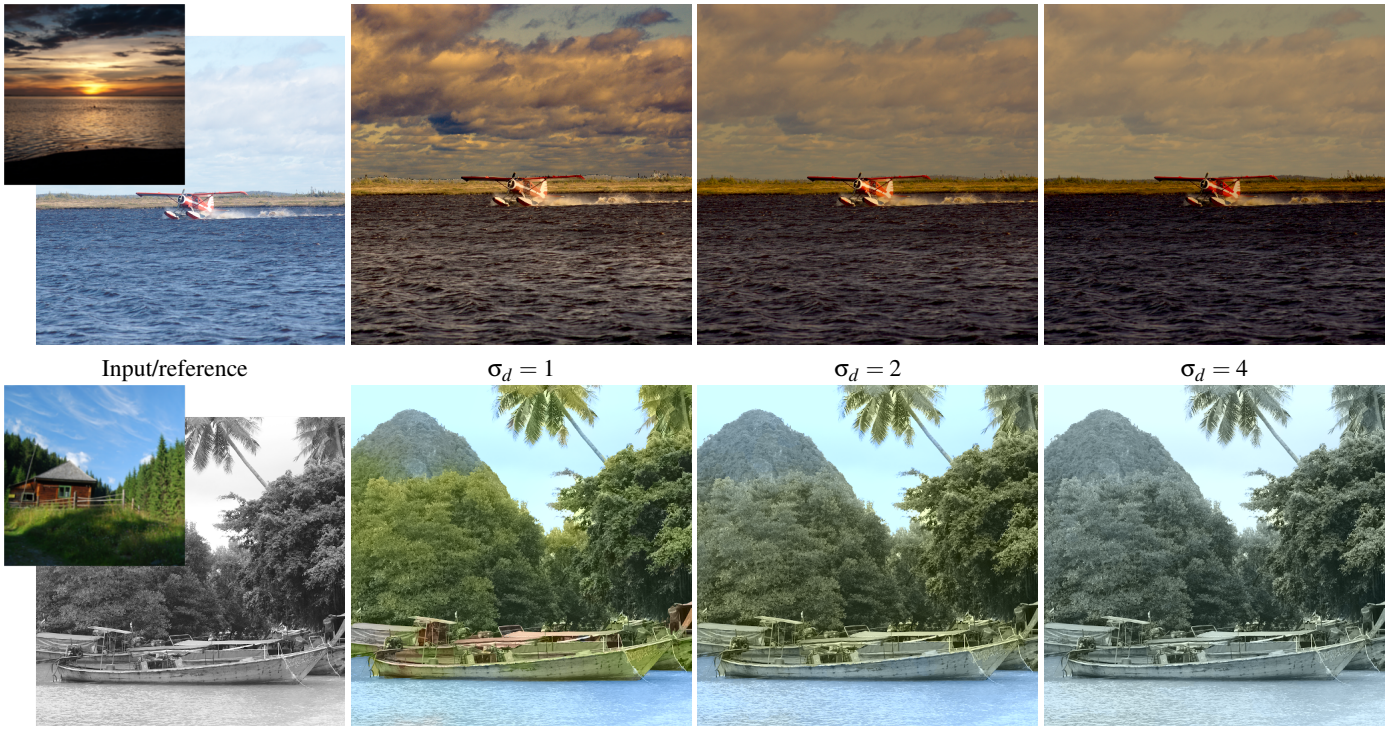
We fully implemented our color manipulation functions on the GPU using Cuda. All the results presented in this paper were obtained with a NVIDIA Quadro 6000 graphics card. In practice, we first precompute the descriptors $\boldsymbol{\mathcal{S}}$ for both the input and reference images before applying a transfer or a colorization. However, Equations (5) and 6 require to iterate over all the pixels of the input image, and compute the similarities with the whole reference for each of them in order to obtain the weighted mean and standard deviations. A naïve implementation of these equation leads to extensive computation times.

To achieve reasonable speed, we propose to quantify similarities using a user-defined distance τ that controls how close two descriptors should be to be considered as equal. Considering a particular input pixel $\mathbf{p}$, all the other pixels $\mathbf{p}_i$ such as $\mathcal{D}_{\sigma_d}(\mathbf{p},\mathbf{p}_i) < \tau$ are processed using the same similarity function. That way, increasing τ decreases the total number of iterations needed to obtain the result. The effect of this optimization



(a) Initial result (b) w/ MGD (c) w/ UBF (d) w/ MGD+UBF

Figure 7: **Processing effect on the transfer result.** In this example, initial descriptors are blurry and create strong color halos above the trees in the transfer result (a). The Multiscale Gradient Descent (MGD) prevents the apparition of halos but some incorrect edges remain in U-shaped transitions between the sky and the trees (b). The Unnormalized Bilateral Filtering (UBF) accurately preserves the structure but smudges halos instead of suppressing them when used alone (c). The combination of both MGD and UBF leads to a cleaner result as shown in (d).



Input/reference $\sigma_d = 1$ $\sigma_d = 2$ $\sigma_d = 4$

Input/reference $\sigma_d = 0.2$ $\sigma_d = 1$ $\sigma_d = 2$

Figure 9: **Impact of σ_d on transfer functions.** Top: color transfer example. When increasing σ_d, more and more pixels are considered as similar, resulting in a transfer close to a basic global histogram matching. Bottom: colorization example. As colors are obtained from the weighted average of similar pixels in the reference image, increasing σ_d tends to produce a monochrome result.

$\tau = 0$, 20min | $\tau = 0.01$, 21s | $\tau = 0.1$, 6s | $\tau = 0.5$, 1s

Figure 10: **Optimization impact.** Color transfer results for increasing τ values for 512×512 images. The lower τ, the higher the speed-up and the probability of quantization artifacts. In this example, $\tau = 0.1$ allows for a fast transfer which can be used for efficient results exploration with minimal visual artifacts.

can be seen in Figure 10, where important speed-up is achieved without visual impacts. High values of τ tend to produce quantization artifacts, but may be used to interactively explore the result space.

To summarize, the user can tune the following parameters to achieve the desired results:

- r_{max} controls the size of the window on which descriptors are computed and thus defines the scale at which textures are estimated. Typically, we found that $r_{max} = 21$ works well for natural images of resolution 512×512.
- σ_s and σ_l respectively control the influence of spatial distances and luminance variations when smoothing the descriptor with the unnormalized bilateral filter. All the results in the paper were done with $\sigma_s = 2$ and $\sigma_l = 0.05$. The number of iterations used for this filter depends on the complexity of texture edges. We typically used 500 iterations for our results.
- σ_d controls how strongly the weight between two pixels is influenced by their distances in the descriptors space. In practice, we respectively used $\sigma_d = 1$ and $\sigma_d = 0.2$ for most color transfer and colorization results.
- τ controls the quantization step. In our results, we used $\tau = 0.01$ as it provides a good speed-up while keeping a good visual quality in almost every case.

The timings of our algorithm for the image in Figure 10 using those parameters are described in the following table:

Algorithm	Image Size	Desc.	Transfer	Total
Colorization	512×512	19s	47s	66s
($\sigma_d = 0.2$)	1024×1024	78s	490s	568s
Color Transfer	512×512	19s	21s	40s
($\sigma_d = 1$)	1024×1024	78s	132s	210s

where "Desc." stands for the descriptors computation and "Transfer" stands for the color transfer or colorization step. Typically the colorization takes longer because of the lower σ_d used which create less similarity between pixels (see Equation (4)), leading to more computation during the transfer step.

Note that our code was designed to be strongly flexible, while maintaining decent timings as much as possible. We believe further optimization could still provide significant speed-up. For example a bilateral grid could be used to compute the filtering step much faster during the descriptors computation. The transfer step could also be further optimized by considering subsampled versions of the reference and input images which would greatly reduce the transfer time for large images. However this unoptimized code still allows for computation times similar to other color transfer or colorization methods relying on image descriptors.

6. Results

Results and comparisons presented in the paper and in the supplemental materials were all made with the default parameters given in the previous section.

6.1. Color Transfer Results

Figure 12 (top) shows the results of our color transfer against other state-of-the-art methods. The results of [5] were computed with our own implementation of their method. The results of [7, 8] were computed using the available code on the authors webpage, we used a full match (100%) for [8]. The results of [15] were provided by the authors. The results of [14] were taken from the authors webpage and drove our choice of images.

These results show that global approaches [5, 7] tend to produce saturated colors due to the stretching of the input color histogram. Furthermore, global histogram matchings match regions of similar colors and luminance, failing in transferring colors between similar textured regions if they have highly different luminance or colors. This is showcased in the bottom row where the orange color of the reference buildings is transferred to the input sky. The progressive approach of [8] also fails to accurately preserve the colors of the reference in their results. Local approaches based on color information [15, 14] lead to better results, but also fail in matching regions of similar textural content because they define similar regions by their luminance and color distributions.

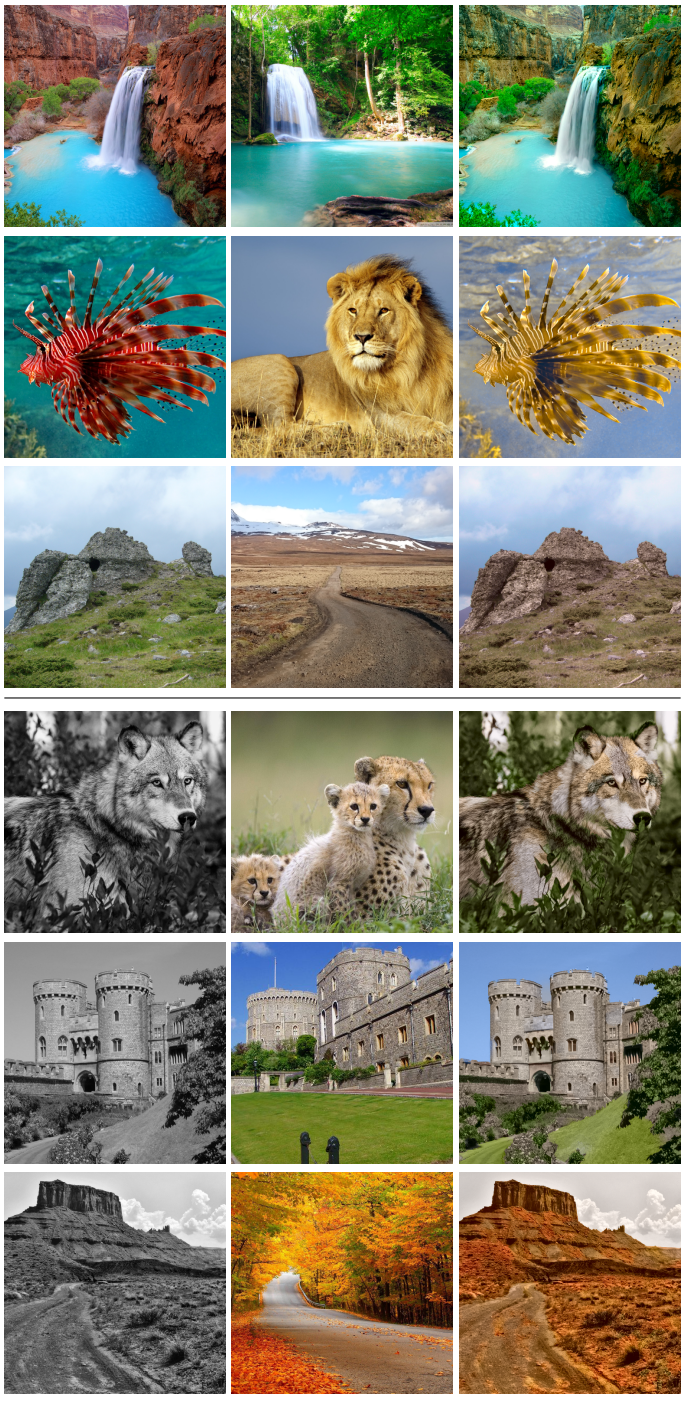


Figure 11: **Transfer and colorization results.** Different colors are clearly associated with different regions based on their textural content.

Our approach successfully matches those regions, as shown in the third row, where the flower field of the reference is matched to the grass of the input (making it yellow); or in the fourth row where the buildings of the reference are matched to those of the input (making them orange). Figure 11 shows three more examples where the matching between different regions is clearly effective thanks to our descriptors.

6.2. Colorization Results

Figure 12 (bottom) compares the results of our colorization against other state-of-the-art methods. The results of [27, 28, 21] were taken from [21]. The results of [29, 42] were computed using the code provided by the authors, using the default parameters suggested in their code.

Those results show that the method of [27] based on luminance matching fails when the input images are too complex: different regions with similar luminance get the same colors, such as the building and clouds in the first example. The method of [28] uses SURF descriptors and Gabor filters which are strongly discriminative, leading to efficient colorization when the input and reference images have identical or very similar content. However, they have to crop image borders and colors often smudge in their results. The method from [21] produces better results by combining superpixel segmentation and similarly robust descriptors (i.e. image intensity, standard deviation features, Gabor filters and SURF features). While achieving better results than previous methods, they still fail to distinguish between intricate regions such as the clouds and the sky in the first and fourth row, or the river and the land in the fourth row. Finally, the method from [29] uses descriptors based on standard-deviation, discrete Fourier transform and cumulative histograms of image patches. It is very prone to halos due to the window used in the descriptors computation and was improved in [42] where a new luminance-chrominance model was used to better propagate colors. While this model is very good to avoid artifacts, the final colors are not always faithful to the reference image colors, as seen in the first and third rows.

As seen in the last column, our approach accurately matches corresponding textures and produces colorful results: sky, cloud, vegetation, mountain and building colors of the references are successfully transferred into the input images. Figure 11 shows three more results demonstrating a clear separation between regions of the input image and correct color associations from the reference image.

To evaluate the coherency of our descriptors, we also tried to colorize a desaturated image using the original color image as reference. These results are shown in Figure 13, where we can observe that color differences between the reference and the output images depends on σ_d: the lower σ_d, the higher the fidelity. This is due to the fact that input and reference images have exactly the same descriptors in that case. In the limit case, when $\sigma_d \to 0$, only one pixel will be taken into account when comparing descriptors (cf. Equation (6)) and the result will be equal to the reference. The pixel-wise difference between the result and the reference image was computed as the sum of the absolute RGB differences. Note that, when input and reference images differ, σ_d should be given a higher value to avoid color artifacts.

6.3. Combining Colorization and Transfer

Since our framework is the same for colorization and color transfer, we can easily apply a combination of both to a grayscale input by adding chrominance via colorization, while modifying the luminance by transferring only the luminance from the

Figure 12: **Comparison with previous methods.** Top and bottom respectively compare color transfer and colorization results with previous state-of-the-art methods. See the text for more details.

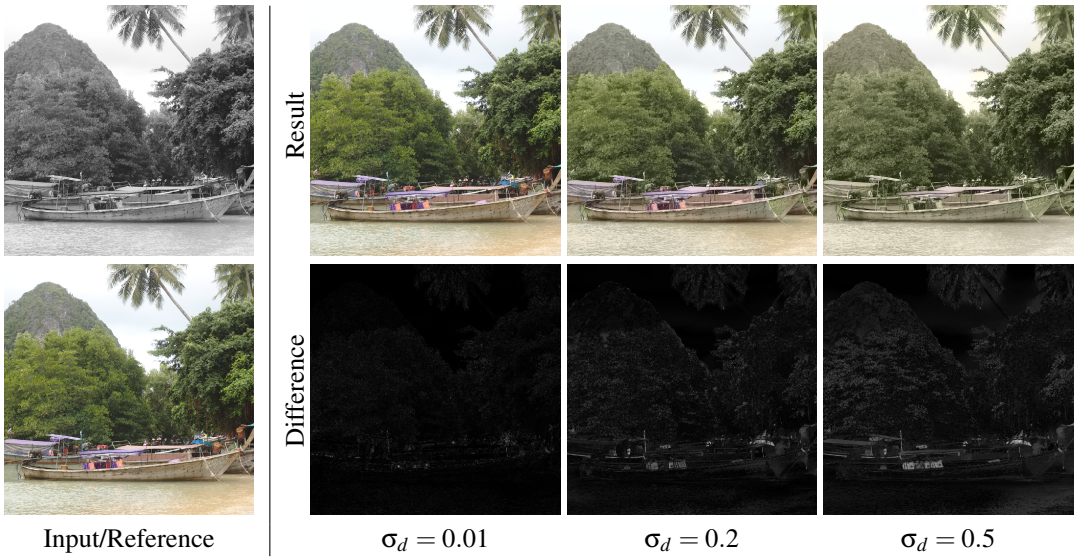


Figure 13: **Self colorization.** The reference color image was desaturated to create the input image. In the difference images, the brighter a pixel, the higher the color difference between the result and the reference. The result colors fidelity depends on σ_d: as $\sigma_d \to 0$, the result approaches the reference.

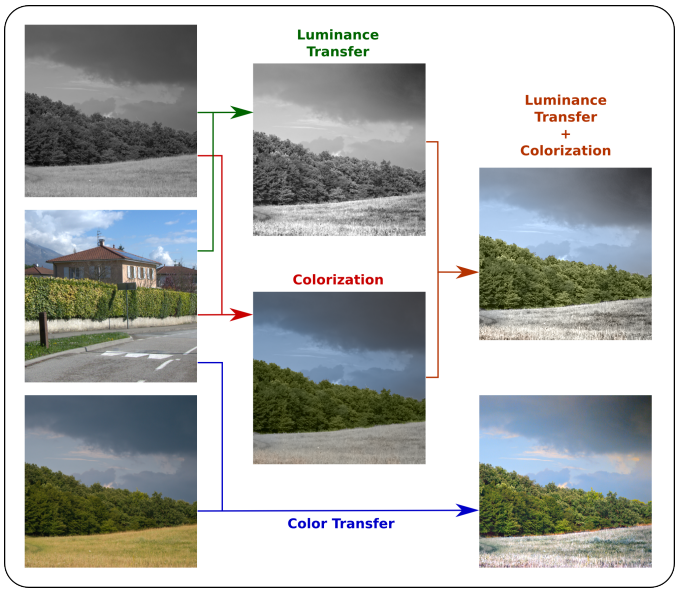


Figure 14: **Combining colorization and luminance transfer.** Our framework allows for an easy combination of colorization and luminance transfer. This combination provides a good style transfer between the input and reference images. While less colorful than a color transfer result, this result only requires a grayscale input. In those results, $\sigma_d = 0.5$.

reference image. The results of this approach can be seen in Figure 14. They show that this combination can produce a result closer to the style of the reference image, while still using only the input luminance. Comparing this to the result of the color transfer (which also transfers luminance), we see that color transfer remains more colorful because the chrominance information of the input image is also used, however it requires a color image as input which is more restrictive.

6.4. Stroke-based local transfer and colorization

The results presented in the previous sections were obtained fully automatically, however our method can easily be extended to support user-provided strokes. These strokes are used to quickly define user desired regions in the input and reference images, through the use of our descriptors. We can then apply a local transfer or colorization between the two user-selected regions of the input and reference images. This allows the user to edit a single part of an image, or to fine-tune the result of the automatic approach.

To extract the homogeneous texture region corresponding to some user strokes in an image, we compute a region-based image segmentation. More specifically, we compute the similarity maps of this image for each pixel of the strokes, then average them to obtain the similarity map $\mathcal{D}^S_{\sigma_d}$ corresponding to the strokes. From this similarity map, we compute a connected binary mask to segment the desired texture region. We chose to make this binary mask connected to each stroke in order to make the selection process more intuitive and avoid a fragmented selection from a single stroke. However several different regions can still be simultaneously selected by using several strokes. The mask computation is detailed in Figure 15.

This mask is initialized with the strokes, then iteratively extended to include pixels whose similarity is close to the similarity of their neighbor in the mask. In practice, if a pixel $\mathbf{p}$ is outside of the mask and one of its neighbors $\mathbf{q}$ is inside the mask, $\mathbf{p}$ is added to the mask if $\left|\mathcal{D}^S_{\sigma_d}(\mathbf{p}) - \mathcal{D}^S_{\sigma_d}(\mathbf{q})\right| \leq \lambda_m$ where λ_m is a threshold controling how similar two neighbors should be to belong to the mask. The higher λ_m, the larger the mask. We used $\lambda_m = 0.05$ for most examples and lowered it to 0.01 when trying to distinguish between similarly textured regions. An example of the strokes similarity maps and the resulting connected mask is shown in Figure 16. We can see that the mask accurately selects the region underlying the stroke (the sky and the hedge). In the second example, making the mask connected

Connected binary mask computation

Input: image I with stroke(s) S
Output: binary connected mask M
Compute the stroke similarity map $\mathcal{D}^S_{\sigma_d}$
Add the stroke pixels to M
while M is expanding **do**
    for every pixel $p \notin M$ **do**
        if a neighbor q of p is in M **and**
        $\left|\mathcal{D}^S_{\sigma_d}(\mathbf{p}) - \mathcal{D}^S_{\sigma_d}(\mathbf{q})\right| \leq \lambda_m$ **then**
            Add p to M
        end if
    end for
end while

Figure 15: Binary connected mask algorithm.

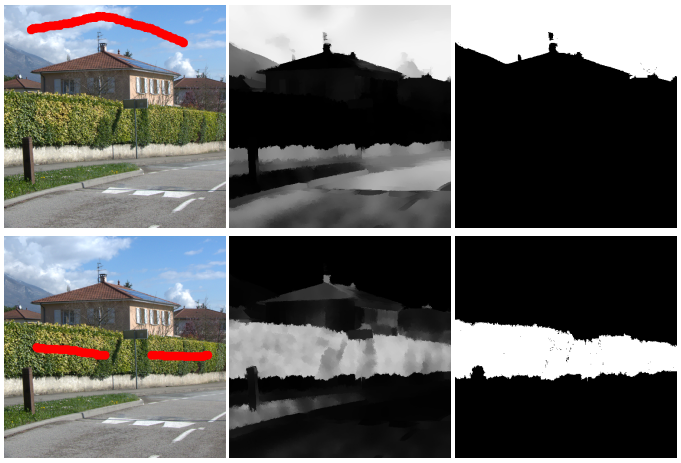


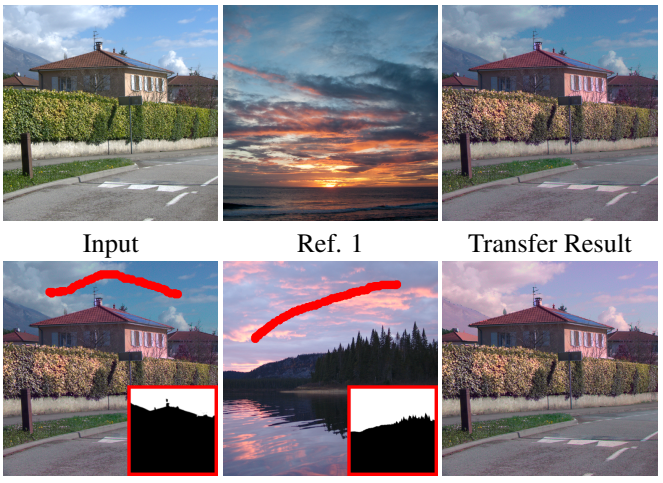
Image strokes Stroke similarity map Binary mask

Figure 16: **Strokes and the resulting mask.** The binary mask defines a connected homogeneous texture region according to the stroke similarity map. In these examples, $\lambda_m = 0.05$.

avoids selecting the grass with the hedge stroke even though their textures are slightly similar.

In Figure 17, we use strokes to fine-tune a color transfer result. The automatic transfer result with the first reference image creates a nice red tint on the house and hedge, however the sky remains blue because of the blue part of the sky in the reference. To increase the sunset feeling, we use a stroke to transfer colors from another reference into the sky of the automatic result, resulting in a red tinted sky. Note that an automatic transfer from the sunset shot of the second reference only would have globally produced a much darker image.

In Figure 18, we see an example where the automatic method fails because of the absence of sky or bright area in the reference image. With additionnal user strokes, we manage to get a pleasing result using only relevant regions from the reference image. Similarly in Figure 19, the input image is homogeneously colorized because the reference trees contain similar textural information and their colors are then blended together.



Input Ref. 1 Transfer Result

Result w/ stroke Ref 2. w/ stroke Final Result

Figure 17: **Local color transfer tuning with strokes.** The automatic color transfer provides a result with a blue sky similar to the first reference blue sky. We used a second reference to fine-tune the sky with a pair of strokes. The final result colors are closer to the red colors of a sunset.

Using strokes, we are able to colorize only the trees of the input image of Figure 14 with different tree colors from the reference image. Other images can then be used to colorize the remaining input regions as showcased in Figure 20 where we used a second reference image to separately colorize the sky and grass regions of the input image. Note that this approach differs from the color strokes used in [19, 42, 24] as our strokes are used to segment texture regions based on their luminance through our descriptors. The colors of the segmented reference region are then transferred to the segmented input region according to the descriptor similarities inside those regions.

Since the stroke similarity map is computed using each pixel of the stroke independently, the shape of the stroke does not matter. When the textured region to segment is homogeneous, as in Figures 16, 17 and 19, the strokes can be very simple. However, when the region is more heterogeneous, as in the input of Figure 18, detailed strokes can be necessary. In this image, the grass and rock textures contain many different parts (bushes and rocks in the grass, shadows and holes in the rock) that cannot be easily segmented with a simple stroke.

7. Discussion and Future Works

In this paper, we presented a generic framework for both color transfer and colorization. Our edge-aware descriptor accurately captures similar textural content in images while being robust to texture transitions. It allows local color transfer and colorization between similar regions of an input and reference images. Furthermore, the user can select, with strokes, regions to match between the input and the reference images. This allows to fine-tune parts of the automatic approach results and to locally edit an image colors. Our method suffers from two main limitations, as described below.

(1) Considering colorization, the input and reference images should be similar enough to produce coherent results. If a par-

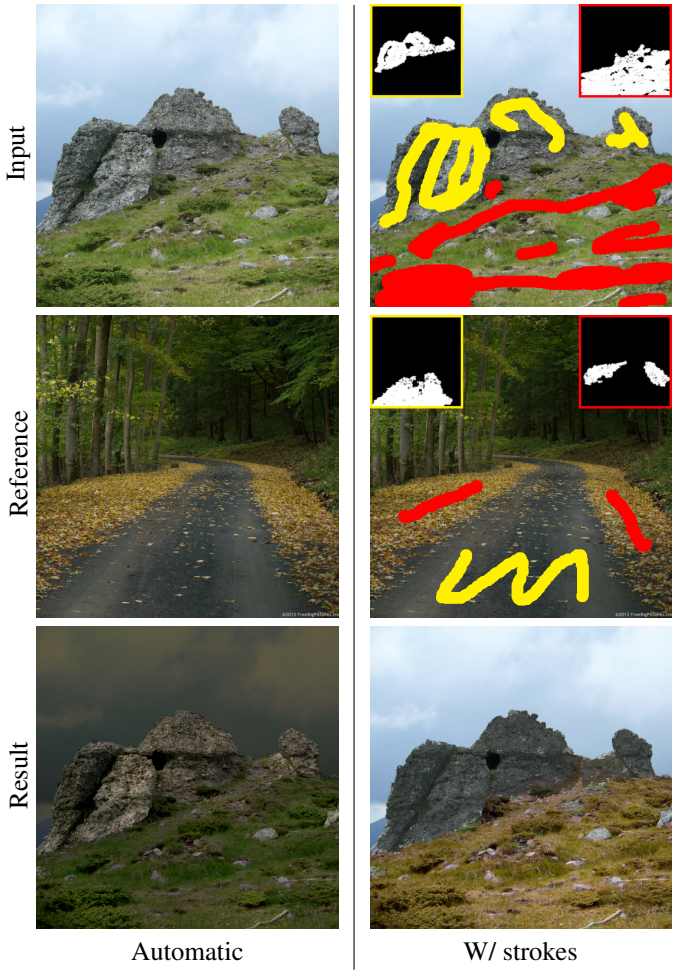


Figure 18: **Improving color transfer with strokes.** The automatic color transfer of the input image is very dark because of absence of bright area (like a sky) in the reference image (left column). Using strokes, only selected regions of the reference are used (the road and the leaves). In this example, $\lambda_m = 0.01$.

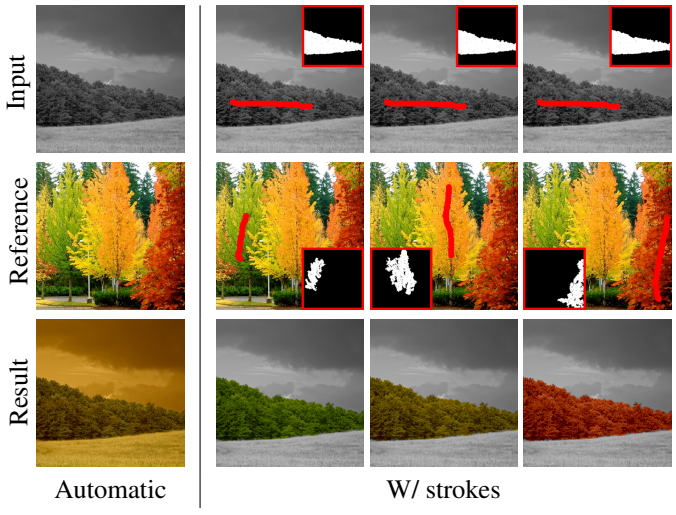


Figure 19: **Improving colorization with strokes.** The automatic colorization of the input image fails because of the similarity in the tree textures in the reference image (left column). Using strokes, the colors of different trees of the reference image are segmented (middle row) and used to colorize the trees of the input image (bottom row). In this example, $\lambda_m = 0.01$.

Figure 20: **Colorization with several targets and strokes.** The color of the trees is taken from the first reference with the red strokes, the colors of the sky and the grass are taken from the second reference with the blue and yellow strokes.

Figure 21: **Failure case.** Semantic information such as man-made objects or faces may locally modify the descriptors and produce incoherent colorizations. For example here, the motorbike wheels are colored in blue because they are detected as a texture resembling the one of the input girl hat.

ticular region in the input image does not have any correspondence in the reference one, the similarity function (based on a Gaussian distance) tends to give the same weight to all pixels, resulting in a monochrome colorization. Note that this is equivalent to increasing σ_d for this particular region, as seen in Figure 9 (bottom-right). This problem also occurs for color transfer but is much less visible since the mean and variance are only used to modify the histogram. To prevent this, one possibility would be to automatically detect mismatched regions and ask the user to disambiguate the transfer by providing more specific reference images.

(2) The proposed descriptors efficiently capture texture regions and their transitions, but are not able to detect higher-level semantic information such as faces, man made objects or background and foreground. Our descriptors might be altered by such objects, thus affecting the quality of the transfers. Again, this is most visible in colorization results, as shown in Figure 21. The yellow color obtained in the top left part of the image is due to the electric wires that are associated to the warning sign of the reference. The wheels of the motorbike contain fine structures associated to the girl's hat, resulting in a bluish color. One way to mitigate these issues would be to rely on more complex, but slower, descriptors combining both semantic and texture information.

Despite these limitations, we believe that our descriptor constitutes a good basis that could contribute to other applications such as tone mapping, edge-aware image decomposition, and color content modification of videos.

Acknowledgements

We thank the anonymous reviewers for their relevant comments. We also thank Oriel Frigo for helping generating the comparison results in Figure 12, and Olivier Le Meur for his helpful comments. Finally, for providing access to their code, we thank Fabien Pierre, Raj Kumar Gupta and all authors whose code was accessible online. The input image used in Figures 1, 2, 3, 4, 8, 10, 14 and the house reference image in Figure 9 are courtesy of `freebigpictures.com`. Color transfer comparisons were made using the images from [14]. Colorization comparison images were taken from [21]. This work is partially supported by the French National Research Agency, MapStyle project [ANR-12-CORD-0025].

References

[1] Yellott Jr JI, et al. Implications of triple correlation uniqueness for texture statistics and the julesz conjecture. JOSA A 1993;10(5):777–93.

[2] Balas BJ. Texture synthesis and perception: using computational models to study texture representations in the human visual system. Vision Research 2006;46(3):299–309.

[3] Arbelot B, Vergne R, Hurtut T, Thollot J. Automatic texture guided color transfer and colorization. In: Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering. Expressive '16; Eurographics Association; 2016, p. 21–32.

[4] Faridul HS, Pouli T, Chamaret C, Stauder J, Reinhard E, Kuzovkin D, et al. Colour mapping: A review of recent methods, extensions and applications. Computer Graphics Forum 2015;.

[5] Reinhard E, Ashikhmin M, Gooch B, Shirley P. Color transfer between images. IEEE Comput Graph Appl 2001;21(5):34–41.

[6] Xiao X, Ma L. Color transfer in correlated color space. In: Proc. ACM Int. Conf. on Virtual Reality Continuum and its Applications (VRCIA). ISBN 1-59593-324-7; 2006, p. 305–9.

[7] Pitié F, Kokaram AC, Dahyot R. Automated colour grading using colour distribution transfer. Comput Vis Image Underst 2007;107(1-2):123–37.

[8] Pouli T, Reinhard E. Progressive color transfer for images of arbitrary dynamic range. Computers & Graphics 2011;35(1):67–80.

[9] Dong Y, Xu D. Interactive local color transfer based on coupled map lattices. In: Proc. Computer-Aided Design and Computer Graphics. 2009, p. 146–9.

[10] An X, Pellacini F. User-controllable color transfer. Computer Graphics Forum 2010;.

[11] Liu S, Sun H, Zhang X. Selective color transferring via ellipsoid color mixture map. Journal of Visual Communication and Image Representation 2012;23(1):173 –81.

[12] Tai YW, Jia J, Tang CK. Local color transfer via probabilistic segmentation by expectation-maximization. In: Proc. IEEE CVPR; vol. 1. 2005, p. 747 – 754 vol. 1.

[13] Bonneel N, Sunkavalli K, Paris S, Pfister H. Example-based video color grading. ACM Trans Graph 2013;32(4):39:1–39:12.

[14] Hristova H, Le Meur O, Cozot R, Bouatouch K. Style-aware robust color transfer. In: Proc. Computational Aesthetics. 2015, p. 67–77.

[15] Frigo O, Sabater N, Demoulin V, Hellier P. Optimal transportation for example-guided color transfer. In: Computer Vision – ACCV 2014; vol. 9005 of *Lecture Notes in Computer Science*. Springer International Publishing. ISBN 978-3-319-16810-4; 2015, p. 655–70.

[16] Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH. Image analogies. In: Proc. of the 28th annual conference on Computer graphics and interactive techniques. SIGGRAPH '01. ISBN 1-58113-374-X; 2001, p. 327–40.

[17] Shih Y, Paris S, Durand F, Freeman WT. Data-driven hallucination of different times of day from a single outdoor photo. ACM Trans Graph 2013;32(6):200:1–200:11.

[18] Okura F, Vanhoey K, Bousseau A, Efros A, Drettakis G. Unifying color and texture transfer for predictive appearance manipulation. Computer Graphics Forum (Proc of the Eurographics Symposium on Rendering) 2015;34(4).

[19] Levin A, Lischinski D, Weiss Y. Colorization using optimization. ACM Trans Graph 2004;23(3):689–94.

[20] Irony R, Cohen-Or D, Lischinski D. Colorization by example. In: Proc. EGSR. ISBN 3-905673-23-1; 2005, p. 201–10.

[21] Gupta RK, Chia AYS, Rajan D, Ng ES, Zhiyong H. Image colorization using similar images. In: Proc. ACM Int. Conference on Multimedia. ISBN 978-1-4503-1089-5; 2012, p. 369–78.

[22] Kuzovkin D, Chamaret C, Pouli T. Descriptor-based image colorization and regularization. In: Proc. Computational Color Imaging Workshop. ISBN 978-3-319-15978-2; 2015, p. 59–68.

[23] Jin SY, Choi HJ, Tai YW. A randomized algorithm for natural object colorization. Computer Graphics Forum 2014;33(2):205–14.

[24] Endo Y, Iizuka S, Kanamori Y, Mitani J. DeepProp: Extracting Deep Features from a Single Image for Edit Propagation. Computer Graphics Forum 2016;.

[25] Zhang R, Isola P, Efros AA. Colorful image colorization. ECCV 2016;.

[26] Iizuka S, Simo-Serra E, Ishikawa H. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics (Proc of SIGGRAPH 2016) 2016;35(4):110:1–110:11.

[27] Welsh T, Ashikhmin M, Mueller K. Transferring color to greyscale images. ACM Trans Graph 2002;21(3):277–80.

[28] Charpiat G, Hofmann M, Schölkopf B. Automatic image colorization via multimodal predictions. In: Proc. ECCV. ISBN 978-3-540-88689-1; 2008, p. 126–39.

[29] Bugeau A, Ta VT. Patch-based image colorization. In: Pattern Recognition (ICPR), 2012 21st International Conference on. 2012, p. 3058–61.

[30] Bugeau A, Ta VT, Papadakis N. Variational exemplar-based image colorization. IEEE Trans Image Processing 2014;23(1):298–307.

[31] Xia GS, Delon J, Gousseau Y. Shape-based invariant texture indexing. International Journal of Computer Vision 2010;88(3):382–403.

[32] Xu Y, Huang S, Ji H, Fermüller C. Scale-space texture description on sift-like textons. Computer Vision and Image Understanding 2012;116(9):999 – 1013.

[33] Xu L, Yan Q, Xia Y, Jia J. Structure extraction from texture via relative total variation. ACM Trans Graph 2012;31(6):139:1–139:10.

[34] Cho H, Lee H, Kang H, Lee S. Bilateral texture filtering. ACM Trans Graph 2014;33(4).

[35] Karacan L, Erdem E, Erdem A. Structure-preserving image smoothing via region covariances. ACM Trans Graph 2013;32(6):176:1–176:11.

[36] Tomasi C, Manduchi R. Bilateral filtering for gray and color images. In: Proc. ICCV. ISBN 81-7319-221-9; 1998, p. 839–.

[37] He K, Sun J, Tang X. Guided image filtering. IEEE Trans Pattern Analysis and Machine Intelligence 2013;35(6):1397–409.

[38] Perona P, Malik J. Scale-space and edge detection using anisotropic diffusion. IEEE Trans Pattern Analysis and Machine Intelligence 1990;12(7):629–39.

[39] Aubry M, Paris S, Hasinoff SW, Kautz J, Durand F. Fast local laplacian filters: Theory and applications. ACM Trans Graph 2014;33(5):167:1–167:14.

[40] Tuzel O, Porikli F, Meer P. Region covariance: A fast descriptor for detection and classification. In: Proc. ECCV. 2006, p. 589–600.

[41] Hong X, Chang H, Shan S, Chen X, Gao W. Sigma set: A small second order statistical region descriptor. In: Proc. IEEE CVPR. 2009, p. 1802–9.

[42] Pierre F, Aujol JF, Bugeau A, Papadakis N, Ta VT. Luminance-chrominance model for image colorization. SIAM Journal on Imaging Sciences 2015;8(1):536–63.