Computers & Graphics 62 (2017) 15–27

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Virtual Special Section on Expressive 2016

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, France
[b] Polytechnique Montréal, Canada

ARTICLE INFO

Article history:
Received 28 October 2016
Received in revised form
25 November 2016
Accepted 3 December 2016
Available online 13 December 2016

Keywords:
Texture analysis
Color transfer
Colorization
Stroke-based edition

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.

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.

* Corresponding author.

http://dx.doi.org/10.1016/j.cag.2016.12.005
0097-8493/

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 [9–11].

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 [20–22]. 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 [24–26]. 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 [27–30]. 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.

3. Overview

Our approach for automatically editing image colors based on textural content is summarized in Fig. 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 Fig. 1 (B), where the vegetation is