Computers & Graphics 38 (2014) 167–173

Contents lists available at ScienceDirect

Computers & Graphics

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

CAD/Graphics 2013

Efficient manifold preserving edit propagation with adaptive neighborhood size

Li-Qian Ma, Kun Xu*

TNList, Department of Computer Science and Technology, Tsinghua University, 3-523 FIT building, Beijing 100084, China

ARTICLE INFO

Article history:
Received 2 August 2013
Received in revised form
1 October 2013
Accepted 19 October 2013
Available online 31 October 2013

Keywords:
Edit propagation
Adaptive neighborhood size
Manifold perserving

ABSTRACT

Recent manifold preserving edit propagation (Chen et al., 2012) [1] provides a robust way for propagating sparse user edits to a whole image or video, which preserves the manifold structure formed by all pixels in feature space during edit propagation. However, it consumes a big amount of time and memory especially for large images/videos, limiting its practical usage. In this paper, we propose an efficient manifold preserving edit propagation method. We accelerate the original method from two aspects. First, instead of using a fixed neighborhood size in building the manifold structure, we adaptively determine neighborhood size for each pixel based on its local complexity in feature space, which largely reduces average neighborhood size. Secondly, following Xu et al. (2009) [2], we adaptively cluster all pixels, and solve the edit propagation problem on clusters instead of pixels. Our experiment shows that, compared to the original method (Chen et al., 2012) [1], our method significantly reduce time and memory costs without reducing visual fidelity.

1. Introduction

The need for image/video editing has become more and more important in our everyday life. In an editing task, artists usually need to adjust the appearance (e.g., color, brightness, tone values, etc.) of the regions of interest. While existing image editing softwares, such as Adobe Photoshop, provide powerful editing tools to achieve this goal, however, it usually requires a careful selection of the region of interest, which is laborious especially for regions with complex textures. Edit propagation avoids explicit region selection, thus provides a more intuitive interaction mode. In an edit propagation scenario, users sparsely draw some strokes on the image/video, where each stroke indicates specific edits, and those user edits are automatically propagated to the whole data satisfying the policy that nearby pixels with similar colors receive similar edits.

Among recent edit propagation methods [1–6], manifold preserving edit propagation [1] is the most robust in handling images containing blended colors of neighboring pixels, which is a frequent case for images with anti-aliasing, semi-transparency or blurring effects. The core of it is a local linear approximation of pixels in feature space, which maintains the linear structure of each pixel with a *fixed number* of neighbors, hence it can preserve the manifold structure formed by all pixels in feature space, leading to more robust results. Despite its advantage, its performance is far from satisfactory. For example, as reported in [1], it takes about 200 s and 350M memory to process an image with 30 million pixels, which largely limit its usage in real applications.

To address it, in this paper, we propose an efficient solution for manifold preserving edit propagation. Our key observation is that different pixels have different local complexities. While pixels in complex regions require a large number of neighbors to describe their local linear structure, a much less number of neighbors are usually enough for pixels in less complex regions. Based on this observation, instead of using a *fixed number* of neighbors, we adaptively determine neighbor size for each pixel accordingly to its *local complexity* in the feature space, which largely reduces average neighborhood size. Our second improvement is combining the hierarchical clustering structure in [2] into manifold preserving edit propagation. Following [2], instead of solving edit propagation on pixels, we solve on clusters. Since the number of clusters is much smaller than the number of pixels, large performance gain is achieved. The experiments demonstrate that our method significantly reduces time and memory cost, while preserving the visual fidelity, largely increases the practicability of manifold preserving edit propagation.

2. Related works

Image/video appearance editing is an increasingly important research topic in computer graphics. Recently, due to its ease of interaction, edit propagation has attracted wide attention. In edit propagation tasks, users specify some sparse edits on images or videos, and those edits are automatically propagated to the whole data. Levin et al. [7] proposed an optimization based approach for

* Corresponding author. Tel.: +86 10 62797001 803; fax: +86 10 62797459.
E-mail address: xukun@tsinghua.edu.cn (K. Xu).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.013

colorizing grayscale images, which propagates user drawn color strokes to nearby regions. Lischinski et al. [8] proposed an interactive method for local tonal adjustment, which propagates tonal edits expressed by strokes to the entire image. Pellacini et al. [3] proposed *AppWand*. It organizes all the pixels into an appearance graph and formulates edit propagation as an optimization problem on the appearance graph, which can be efficiently solved by a sparse linear system. An and Pellacini [4] proposed *AppProp*, which considers all-pairs affinities between pixels. Since it is infeasible to compute and store the full affinity matrix, they further employ a low-rank approximation for the full affinity matrix, which largely reduces memory and time cost.

Many works [2,9,5,10] are later proposed to further accelerate edit propagation, extending edit propagation to videos. Specifically, Xu et al. [2] use a k-d tree in feature space to organize pixels and solved affinity based edit propagation on clusters instead of pixels, achieving speed up of 2–3 orders of magnitude compared to the previous methods [4]. In our work, we also employ this strategy to accelerate manifold preserving edit propagation. Xiao et al. [9] proposed another adaptive clustering method to accelerate edit propagation. Different from [2] which uses a k-d tree in feature space, they employ a quad-tree structure in image space. Li et al. [5] further accelerate edit propagation by formulating it as a function interpolation problem instead of an optimization problem. Bie et al. [10] proposed an efficient edit propagation approach based on an efficient sampling scheme and static clustering.

Various works have also been proposed to improve the quality of edit propagation. Farbman et al. [11] proposed to use diffusion distance, instead of Euclidean distance, to define affinity values between pixels, which better accounts for the global distribution of pixels in feature space. Ma and Xu [12] proposed a method which can preserve antialiased edges during edit propagation. Recently, Chen et al. [1] proposed manifold preserving edit propagation, which organizes all pixels by a manifold structure in the feature space. Specifically, locally linear relationship of each pixel between its neighboring pixels is preserved during edit propagation. This method has demonstrated to be robust in handling images with color blending. However, its performance is far below interactive.

3. Overview

Background: Given an image or video, the goal of edit propagation is to propagate sparse appearance edits, expressed by strokes, to the whole image or video. The process of edit propagation is guided by two principles: first, pixels covered by strokes should satisfy user edits; secondly, similar pixels (i.e. nearby pixels with similar appearance) should receive similar edits. Here we review the basic formulation of manifold preserving edit propagation method [1].

For each pixel i in an image, we define a high dimensional feature vector $\mathbf{f}_i = (\mathbf{c}_i/\sigma_c, \mathbf{p}_i/\sigma_p)$, where $\mathbf{c}_i$ is the appearance vector (e.g. color in Lab space), $\mathbf{p}_i$ is the 2D pixel position (e.g. x and y coordinates), and σ_c, σ_p are parameters to control the relative contribution of the two terms. For videos, an additional frame index term is also combined into the feature vector to incorporate the time dimension, such that $\mathbf{f}_i = (\mathbf{c}_i/\sigma_c, \mathbf{p}_i/\sigma_p, \mathbf{t}_i/\sigma_t)$, where $\mathbf{t}_i$ is the frame index and σ_t is its controlling parameter.

Following the theory of Local Linear Embedding (LLE), local linearity is assumed for each pixel with its neighboring pixels in the high dimensional feature space. In other words, each pixel i can be approximated as a linear combination of its K neighboring pixels. A set of weights that best reflect this local linearity property are computed by minimizing the equation below [13]:

$$\sum_{i=1}^{N} \|\mathbf{f}_i - \sum_{j \in \mathcal{N}_i} w_{ij}\mathbf{f}_j\|^2 \tag{1}$$

where N is the number of pixels, $\mathcal{N}_i$ denotes the neighbor set of pixel i, and the weights satisfy that $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$. The weight matrix W (the $N \times N$ sparse matrix whose elements are w_{ij}) describes the manifold structure of the pixels in feature space and can be solved by a sparse system following [13].

The reconstructed weights can then be used for edit propagation. Specifically, given user edits g_i with strength $u_i \in [0,1]$ (0 indicating no user constraints), The desired propagated edits e_i at all pixels are computed by minimizing the equation below:

$$\lambda \sum_{i=1}^{N} u_i(e_i - g_i)^2 + \sum_{i=1}^{N} \left(e_i - \sum_{j \in \mathcal{N}_i} w_{ij} e_j \right)^2 \tag{2}$$

The two terms respect the two principles we mentioned earlier in this section. Specifically, the first term ensures the user edits are satisfied, and the second term maintains the reconstructed manifold structure, which also ensures that similar pixels receive similar edits. λ is a controlling weight and is set as $\lambda = 1$ in our experiments to make the contributions of the two terms similar.

Summary: The above minimization problem (Eq. (2)) can be solved as a sparse linear system [1]. The time/space complexity depends on the number of unknowns (i.e. the number of pixels N) and the number of non-zero elements in the sparse matrix (i.e. $N \cdot K/2$, where K is the neighborhood size used in Eq. (1) for each pixel). Note that a fixed value of K is used for all pixels. Therefore, decreasing N and K is crucial to improve the performance of manifold preserving edit propagation.

Our Observation: We improve the efficiency of manifold preserving edit propagation from two aspects.

First, we observe that different pixels have different local complexities. While pixels with complex local structure require a large number of neighbors to describe their local linear structure, a much less number of neighbors are usually enough for pixels in smooth regions. Hence, instead of using a fixed number of neighborhood size K, we adaptively determine neighborhood size for each pixel, accordingly to its local complexity in the feature space, which reduces average neighborhood size by 3–5 times. The details will be explained in Section 4.

Second, following [2], we employ a hierarchical clustering structure in solving manifold preserving edit propagation. Pixels are organized into a hierarchical k-d tree structure. Instead of directly solving on pixels, we solve edit propagation on k-d tree corners. Since the number of k-d tree corners is much less than that of pixels, we achieve a large performance gain. The edited parameter of each pixel is then obtained by multi-linear interpolation from k-d tree corners. The details will be discussed in Section 5.

4. Adaptive neighborhood size strategy

In the work of [1], a fixed number of nearest neighbors (typically $K=30$) are used to preserve the manifold structure during edit propagation. However, using a fixed value of K is not always optimal. In our experiment, we observe that, for a pixel, if it locates in a dense region in the feature space (i.e. many pixels are located close-by in the feature space), fewer neighbors are required to preserve its local linear relationship. On the other hand, if it locates in a sparse region (i.e. few feature vectors are located close-by in the feature space), such as edge pixels, much more neighbors are required.

Based on the above observation, we propose to adaptively determine the number of neighbors for each pixel according to its local density in feature space, instead of using a fixed number of neighbors for all pixels. Specifically, we define the local density d_i