Computer Vision and Image Understanding 000 (2017) 1–12

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

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

Three-layer graph framework with the sumD feature for alpha matting

Chao Li[a,b], Ping Wang[a,*], Xiangyu Zhu[a], Huali Pi[a]

[a] School of Mathematics, Tianjin University, Tianjin, PR China
[b] School of Computer Science and Technology, Tianjin University, Tianjin, PR China

ARTICLE INFO

Article history:
Received 5 December 2016
Revised 26 June 2017
Accepted 29 June 2017
Available online xxx

Keywords:
Alpha matting
Graph model
Image feature
Nonlocal principle

ABSTRACT

Alpha matting, the process of extracting opacity mask of the foreground in an image, is an important task in image and video editing. All of the matting methods need exploit the relationships between pixels. The traditional propagation-based methods construct constrains based on nonlocal principle and color line model to reflect the relationships. However, these methods would produce artifacts if the constrains are not reliable. So we improve this problem in three points. Firstly, we design a novel feature called sumD feature to increase the pixel discrimination. This feature is simple and could encourage pixels with similar texture to have similar feature values. Secondly, we design a three-layer graph framework to construct nonlocal constrains. This framework finds constrains in multi-scale range and selects reliable constrains, then unifies nonlocal constrains according to their reliabilities. Thirdly, we develop a new label extension method to add hard constrains. Experimental results confirm that the effectiveness of the three changes, and the proposed method achieves high rank on the benchmark dataset.

1. Introduction

Digital matting technology aims to accurately extract the foreground objects from images or videos. The matting results can be applied in movie and TV industry (Hasinoff et al., 2006), remote sensing, etc. Explicitly, for an image I, image matting aims to find optimal linear combination of foreground F and background B. For any pixel i, let it satisfy the following formula:

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \tag{1}$$

where alpha matte $\alpha \in [0, 1]$ is the combination coefficient and defines the opacity of each pixel. For a colorful image, Eq. (1) shows a highly ill-posed problem, which has three equations with seven unknown values at each pixel. So it requires additional constrains to solve this problem.

Recent matting approaches can be classified into three main categories: propagation-based, sampling-based and machine learning-based approaches (Zhu et al., 2015). Propagation-based approaches use the relativity of pixels in a certain distance to build an effective constraint, then obtain alpha matte by solving an optimization problem (Chen et al., 2013a; Levin et al., 2008). A sampling-based approach picks out the optimal foreground and background sample pairs from known regions for each unknown pixel, then figures out α value directly based on matting equation. Some recent sampling-based approaches combine with propagation-based approach to refine their matting results (Johnson et al., 2016; Karacan et al., 2015). Machine learning-based approaches put image matting into a supervised learning or semi-supervised learning framework, and establish an image matting model to calculate the alpha matte through a learning process (Cho et al., 2016; Xu et al., 2017; Zhang et al., 2012). Recently, Cho et al. (2016) and Xu et al. (2017) use deep learning to achieve outstanding performance. Besides some learning methods, all these methods share a common thing that they build constrains on α values based on color and spatial information of the pixels in the image.

In this section, we will focus on introducing propagation-based approaches, which are most related to our work. A more extensive survey on matting can be found in Zhu et al. (2015).

The most famous local constrain is color line model, which is proposed in closed-form matting (Levin et al., 2008). The color line model suggests that the foreground color values in a small window lie on a single line in the RGB color space, and the same is true for the background color values. So this model represents each foreground color value as a combination of two constant color values in a small window. Then the closed-form matting method constructs a matting Laplacian matrix by the color line model, and obtains alpha matte by solving a quadratic optimization problem. Becasue this method provides a reliable and effective constraint and can obtain a closed-form solution, it has a significant impact on later matting approaches.

Although the color line model provides effective local constraints, it usually fails in a large window. To overcome this shortcoming, nonlocal principle is introduced. Nonlocal principle is

* Corresponding author.
E-mail address: wang_ping@tju.edu.cn (P. Wang).

http://dx.doi.org/10.1016/j.cviu.2017.06.011
1077-3142/

first applied in image denoising (Buades et al., 2005), and its main idea is that the true color value of a pixel is the weighted sum of pixel values with similar features. Lee and Wu (2011) extends this idea to image matting and redescribe it like that each α value is the weighted sum of some nonlocal α values. Because nonlocal principle is not limited in a local region, this method can build effective constrains in a global spatial region. Thereafter, inspired by nonlocal matting method, Chen et al. (2013a) propose KNN matting. This method searches K nearest neighbors of each pixel in the feature space, and then constructs an incidence matrix to reflect connections of α values. This method obtains good results in most cases, while it gets inaccurate results in the region where foreground and background have similar color values. Besides, Chen et al. (2012) propose manifold preserving edit propagation method. They seek to maintain the manifold structure formed by all pixels in the feature space, and use Locally Linear Embedding (LLE) (Saul and Roweis, 2000) to extract the manifold structure. Then α values are mapped to the elements of the manifold structure, so that unknown α values can be solved according to known α values. This method is more robust to color blending region.

The combination of local and nonlocal constrains can make matting method achieve more accurate results. Chen et al. (2013b) propose local and nonlocal smooth priors (LNSP) method, which combines manifold preserving edit propagation method with color line model. This combination overcomes the shortcomings of the above two methods and increases accuracy of matting results. Recently, Aksoy et al. (2017) propose information-flow matting method, which combines color line model and three nonlocal constrains based on the information flow idea. This method has outstanding performance in benchmark database. It is worth noting that many recent sampling methods like KL-divergence based sparse sampling (Karacan et al., 2015) and graph-based sparse matting (Johnson et al., 2016) introduce color line model in their methods. Actually sampling methods collect samples in nonlocal regions, so sampling process can supplement the lack of local constrains. And as a learning method, DCNN matting (Cho et al., 2016) uses the results of closed-form matting and KNN matting as inputs to reconstruct alpha mattes. So this method also utilizes local and nonlocal constrains.

Besides, some image matting methods were extended to video matting field. Choi et al. (2012) extends closed-form matting to video and uses multi-frame nonlocal matting Laplacian to solve matting problem. Li et al. (2013) construct the KNN Laplacian using motion-aware K nearest neighbors.

Although nonlocal constrains can complement local constrains, there are still three basic problems for nonlocal principle.

First, what pixel feature should be used in matting algorithm? Recent nonlocal methods use pixels' color and position information as feature directly and search neighbors in the feature space. But this feature has poor discrimination in the regions where foreground and background have similar color distributions. It is prone to build unreliable constrains in the follow-up process. To increase the discrimination of pixels, we design a novel pixel feature which reflects the distribution characteristic of pixels' neighbors in the feature space. This feature could also encourage pixels to connect to other pixels which have similar texture. Experiments show that this feature helps our method work better in the regions with ambiguous color especially with sparse trimap.

Second, how to build appropriate equations to describe the relationships between the alpha values of the relevant pixels? Recent algorithms use different calculation methods to construct equations respectively. The weights of pixels in the equations play an important role for matting. KNN algorithm (Chen et al., 2013a) calculates the weights of pixels according to the distance between them in the feature space. The method in Chen et al. (2012) uses Local Linear Embedding(LEE) to calculate the weights. But the results of these methods cannot be completely consistent with matting Eq. (1). Aksoy et al. modifies the method in Chen et al. (2012) and propose a new weight calculating method (Aksoy et al., 2017). Here we use the method in Aksoy et al. (2017), and give some mathematica derivation to demonstrate that this method is more consistent with matting Eq. (1).

Third, how to choose the reliable constrains? We need find the reliable constrains in different spacial scales. But even we could construct nonlocal constrains carefully, it is inevitable that the unreliable constrains increase when searching regions are expanded. And if we introduce these unreliable constrains to the final model, it is prone to reduce the accuracy of the matting results. To solve this problem, we design a three-layer graph framework to select the reliable constrains in different spacial scales and unify them according to their reliabilities. Finally, we integrate all constrains into a quadratic optimization, and solve it to achieve the alpha matte.

Moreover, inspired by the research of generalized geodesic distance transform in image editing (Criminisi et al., 2010), we design a label extension method to reduce the difficulty of matting. Experiments show that this method could help our method achieve more accurate results with only about 3 s.

In brief, the main contributions of this paper are in threefold: 1) A novel feature is designed to help discriminate foreground from background. 2) To improve the matting accuracy, we design a three-layer graph framework which could construct nonlocal constrains in different spacial scales and synthesize these constrains according to their reliabilities. Experimental results on benchmark database show that our approach achieves similar or higher matting accuracy comparing to the state-of-the-art methods. 3) To further improve the matting results of complicated images, we propose an effective and quick label expansion method as pre-processing.

The rest of this paper is organized as follows. In Section 2, we introduce the proposed method. Section 3 discusses experimental results, and we conclude the paper in Section 4.

2. Proposed method

In this section, we will describe our algorithm in detail. In our method, we use nonlocal constraints to supplement local constraints. The color line model is a famous and reliable model, so we use it as the local constraints. In order to build effective nonlocal constraints, our work focuses on three points: First, we design a new feature to improve the discrimination of pixels. This feature is based on the sum of distance between pixels and their K nearest neighbors in the feature space, so we call this sumD feature. We use the sumD feature to design three different feature spaces, and search neighbors of each pixel in these feature spaces. Second, to construct nonlocal constrains to be consistent with matting Eq. (1), we suppose that color values of pixels are the combination of their neighbors' colors, and then calculate the linear relationship of them. Third, to choose the reliable constrains, we design a method to measure the reliability of these relationships, and filtrate them to build a three-layer graph model which retains all remaining nonlocal constraints. Finally, we use the color line model as local constraints, the three-layer graph model as nonlocal constraints, to form a quadratic programming. And we solve it to obtain alpha matte. See details below.

2.1. The sumD feature

In order to effectively search the related pixels, we design a new feature to improve the discrimination of pixels. This feature is extracted from distances between pixels and their K nearest neighbors in the feature space. It is indicated as $sumD_i = \sum_{k=1}^{K} d_k$, where

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

$d_k(k = 1, \ldots, K)$ is the distance between pixel i and its neighbors in the feature space. And we define n-layer sumD feature as

$$sumD_i^{(n)} = \sum_{N_k} sumD_{N_k}^{(n-1)}, \tag{2}$$

where N_k is the kth nearest neighbor of pixel i. In this paper, we set $n = 120$ in all experiments.

If we construct a tree, in which the nodes represent the points in the feature space, and the child nodes represent their father nodes' K nearest neighbors. According to the definition, the sumD value represents the sum of distances between the root node and all leaf nodes in the n-layer tree. This feature reflects some distribution characteristics of the pixels neighbors in the feature space, and the n-layer sumD feature can work through connections between pixels and their neighbors. If two pixels belong to one object in an image, and the regions around these pixels have similar texture, then they tend to have similar sumD values. This feature is rotation invariance. And unlike spatial filter, pixels around the margin of an object tend to have similar sumD values with other pixels from the same object.

Here, we use a set of artificial data and natural images to demonstrate the effectiveness of this feature. In Fig. 1, the left column is the original data. In the subgraph of the right column, we map the sumD value to the colormap 'Copper' in octave to get points' color (In these experiments, we set $K = 3$). It is obvious that the points tend to have similar sumD values with their neighbors, especially they have the same distribution in the feature space. And we can see in the third row, if a few points be connected mistakenly, it will not produce big influence in global region. Fig. 2 shows some natural images and their sumD value images. The first row is original images, and the second row is sumDImages. We can see that if objects have the same material or similar texture, their pixels tend to have similar sumD values. And the sumD value is usually higher in the regions with edges or rough material, lower in smooth region.

2.2. Weight calculation

After getting the nearest neighbors of pixels, we need to construct nonlocal constrains by calculating the weights of neighbors. Different weight calculation methods have appeared in KNN matting (Chen et al., 2013a), LNSP matting (Chen et al., 2013b) and information-flow matting (Aksoy et al., 2017). Here we use the method in information-flow matting. Aksoy et al. give some experiments in artificial data to demonstrate the effectiveness of this method. Here we will show that this method is more consistent with matting Eq. (1) in theory.

LNSP matting uses color values and location information of pixels as features, and assumes that the image and the alpha matte have the same manifold structures in this feature space. So this method uses LLE algorithm (Saul and Roweis, 2000) to learn the manifold structure of image in the feature space, then uses the manifold structure as nonlocal constrains of alpha. Specifically, LLE algorithm finds K nearest neighbors in the feature space, then calculates the weights of neighbors by minimizing the reconstruction error. Here, the reconstruction error is expressed as:

$$\varepsilon_{LNSP} = \left\| X(i) - \sum_{k=1}^{K} w_k X(n_k) \right\|_2, \tag{3}$$

where $X(\cdot) = [r, g, b, x, y]^T$ is feature, n_k is the kth neighbor of data point i, w_k is the weight for n_k. Decomposing Eq. (3) we achieve:

$$\begin{aligned} \varepsilon_{LNSP} &= \left\| I(i) - \sum_{k=1}^{K} w_k I(n_k) \right\|_2 + \left| S(i) - \sum_{k=1}^{K} w_k S(n_k) \right\|_2 \\ &= \varepsilon_I + \varepsilon_S, \end{aligned} \tag{4}$$

where $I(\cdot)$ is the color value, $S(\cdot) = [x, y]^T$ is spatial coordinates, ε_I is the reconstruction error of color values, and ε_S is the spacial reconstruction error.

Unlike the above method, Aksoy et al. (2017) calculate weights by minimizing ε_I only. We think this way is more reasonable. Because if we assume

$$\varepsilon = \left\| \alpha(i) - \sum_{k=1}^{K} w_k \alpha(n_k) \right\|_2, \tag{5}$$

then there is

$$\alpha(i) = \sum_{k=1}^{K} w_k \alpha(n_k) + \delta, \tag{6}$$

where $-\varepsilon < \delta < \varepsilon$. In the K nearest neighbors of pixel i, if the first K_1 neighbors' α values are equal to 1, and the remaining neighbors' alpha values are 0, there is

$$\alpha(i) = \sum_{k=1}^{K_1} w_k \alpha(k) + \sum_{k=K_1+1}^{K} w_k \alpha(k) + \delta = \sum_{k=1}^{K_1} w_k + \delta. \tag{7}$$

In this case, α_i is the sum of the alpha values of neighbors belonging to the foreground.

In addition, similar to the ideas of Johnson et al. (2014), 2016), if we extend the matting equation, that we make the color value of pixel i equal to the weighted sum of the color values of foreground and background:

$$I(i) = \sum_k w_k F(k) + \sum_j w_j B(j), \tag{8}$$

where $F(k)$ and $B(j)$ are the color values of the kth foreground pixel and the jth background pixel respectively. So α_i can be regarded as the sum of w_k, then there is

$$\alpha(i) = \sum_k w_k \alpha(k) + \sum_j w_j \alpha(j) = \sum_k w_k. \tag{9}$$

The difference between Eq. (6) and Eq. (9) is δ, and smaller δ can guarantee higher accuracy. So we should minimize (5). Moreover, the spatial coordinates of pixels are not directly related with α. This will affect the approximation of Eq. (5) and the calculation of w_k, and then affect the calculation of α. So minimizing ε_I is better than minimizing ε_{LNSP}.

Besides the above two methods, KNN matting calculates the weights of pixels according to the distance between them in the feature space. This method also can not be consistent with matting equation. So we can conclude that using ε_I as the reconstruction error is more aligned with the requirements of matting problems.

In brief, the weight calculation method in Aksoy et al. (2017) searches pixel's nearest neighbors in the feature space, then calculates weights in the color space. We use this method and calculate weights by minimize ε_I.

2.3. Three-layer graph framework

Based on the multi-scale idea, we construct a multi-layer graph to unify all nonlocal constraints. In this model, each node represents a pixel, each edge represents the weight of pixel's neighbor. As shown in Fig. 3, first, we extract feature from input image and design three feature spaces. Second, we search neighbors in each feature space and get a series of combinations which are composed of pixels and their neighbors. Third, we select reliable combinations and calculate the weights (the edges of the incidence matrix) to get each graph respectively. At last, we calculate the weighed sum of the three graphs to unify all combinations.

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

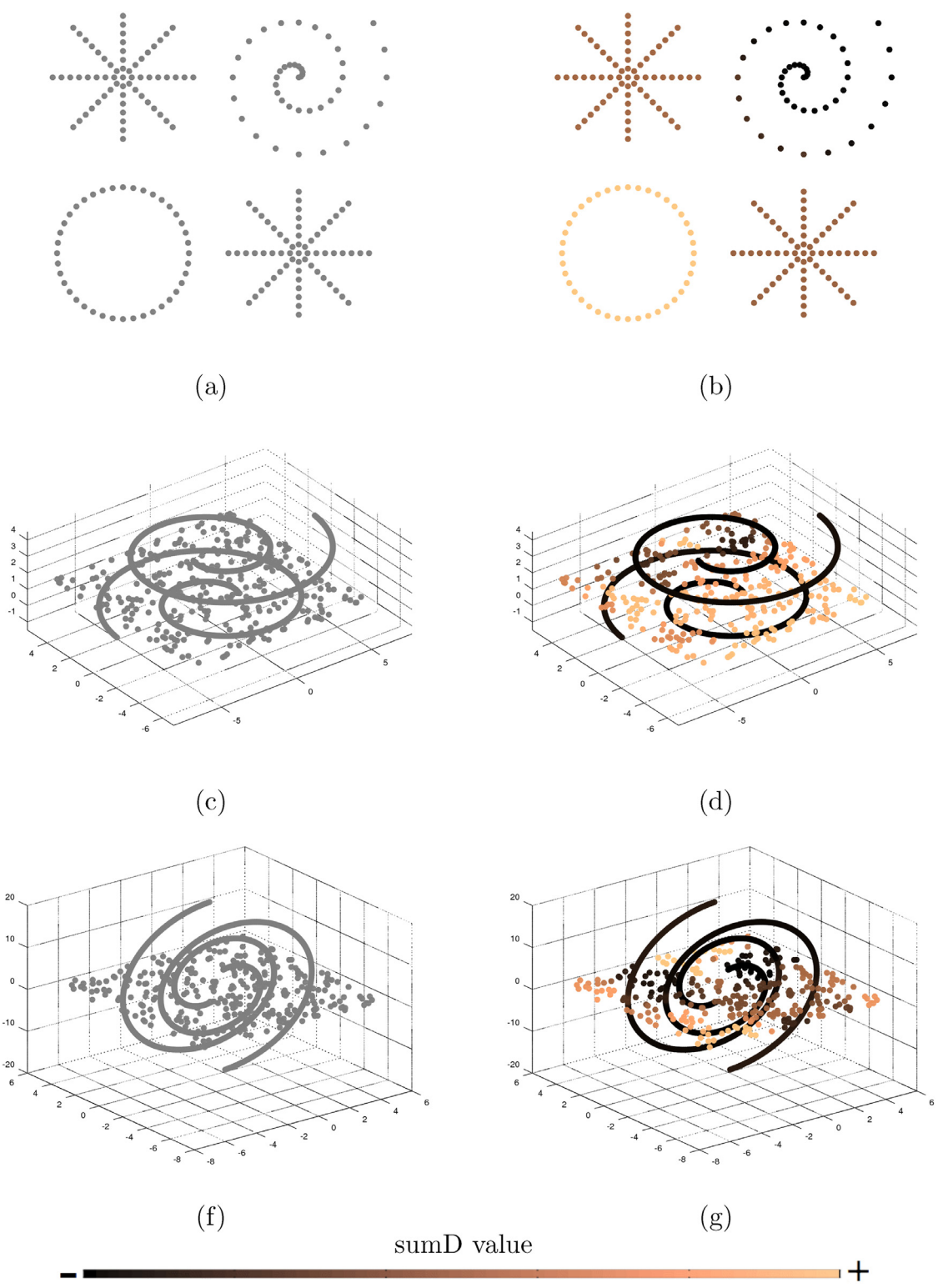

Fig. 1. Performance test with artificial data. (a), (c), (f) is the original data, (b), (d), (g) demonstrate sumD values of points by color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We use the sumD feature to construct the feature spaces. To extract the sumD feature of the first graph, we define the initial feature as follows:

$$X_{initial} = [r \quad g \quad b \quad \mu x \quad \mu y]^T, \tag{10}$$

where r, g and b are the color values of pixels in RGB color space, x and y are the spatial coordinates of pixels, μ is an impact factor. We extract the sumD feature in the initial feature space, then combine $X_{initial}$ and $sumD$ to get a new feature as follows:

$$X_{graph1} = [r \quad g \quad b \quad \mu x \quad \mu y \quad vsumD]^T, \tag{11}$$

where v is the impact factor. We use this feature to construct the first graph. Firstly, we search K_1 nearest neighbors for each pixel in the feature space to obtain N combinations (K_1 is also used

Fig. 2. Performance of the sumD feature with natural images. The first row is original images, the second row is sumD images. The luminance corresponds to the sumD value. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to extract the sumD feature, N is the number of image pixels). Secondly, in order to exclude the outliers, we select $\sigma_1 N$ combinations which have smaller sums of the distances between a pixel and its neighbors, and we set the filter ratio $\sigma_1 = 0.9$. Thirdly, we use these combinations to compute the weights of neighbors so that we get the incidence matrix of the first graph.

Next, we construct the second and the third graph in similar way, and define the features of the second and the third graph as follows:

$$X_{graph2} = [r \quad g \quad b \quad \mu_2 x \quad \mu_2 y \quad v_2 sumD_2]^T, \tag{12}$$

$$X_{graph3} = [r \quad g \quad b \quad \mu_3 x \quad \mu_3 y \quad v_3 sumD_3]^T, \tag{13}$$

where μ_2 and μ_3 are the impact factors of spatial coordinates, and satisfy $\mu \gg \mu_2 > \mu_3$. $sumD_2$ and $sumD_3$ are the sumD features extracted from the feature spaces of graph2 and graph3 respectively, v_2 and v_3 are their impact factors. The numbers of the nearest neighbors are K_2 and K_3, respectively. And the filter ratios of constraints are σ_2 and σ_3 respectively, which satisfy $\sigma_1 > \sigma_2 > \sigma_3$. We choose σ_2 and σ_3 adaptively, the technical details are in Section 3.1. Note that extracting the sumD feature need search pixel's neighbors, so we can exploit the results of neighbors searching process to construct the feature space of next graph. This operation avoid redundant calculations.

As shown in Fig. 3, each graph is based on different features and pixel i is associated with different pixels in the three-layer graph model. Each graph can be represented by an incidence matrix. Note $W_h (h = 1, 2, 3)$ as the incidence matrix of the hth graph, and note $W^h_{iN_k}$ as the weight of the kth neighbor of pixel i in the hth graph. If the combination, which consists of pixel i and its neighbors, is filtered out, there is $\sum_k W^h_{iN_k} = 0$, otherwise $\sum_k W^h_{iN_k} = 1$, and they satisfy $\varepsilon_i = (\alpha_i - \sum_k W^h_{iN_k} \alpha_{N_k})^2$. It can be expressed in a matrix form as follows:

$$\begin{aligned}\sum_i \varepsilon_i^h &= \alpha^T \left(O_h \cdot \left(E - W^h\right)\right)^T \left(O_h \cdot \left(E - W^h\right)\right)\alpha \\ &= \alpha^T G_h \alpha\end{aligned} \tag{14}$$

here, O_h is a diagonal matrix, when the ith combination belongs to the selected combinations, $O_{ii} = 1$, otherwise $O_{ii} = 0$. G_h is a symmetric positive semi-definite matrix. In order to unify all nonlocal constraints in the three-layer graph, we use Eq. (14) to express the reconstruction error of each graph and integrate them according to the influence of constraints in each graph as follows:

$$\varepsilon = \sum_h \gamma_h \sum_i \varepsilon_i^h = \alpha^T (\gamma_1 G_1 + \gamma_2 G_2 + \gamma_3 G_3)\alpha = \alpha^T G_{nonlocal} \alpha, \tag{15}$$

here, γ_1, γ_2, γ_3 are impact factors of the first, second and third graph, respectively. ε is the reconstruction error, $G_{nonlocal}$ is a symmetric positive semi-definite matrix. We use the penalty function method in the optimization process described below. Then we minimize the reconstruction error ε to get the solution that satisfies the nonlocal constraints.

2.4. Closed-form solution

We take foreground and background labels as known conditions and construct local constraints based on the color line model. Meanwhile, we construct nonlocal constraints to further reduce the range of the solution of alpha matte. According to Levin et al. (2008), we minimize $\alpha^T L \alpha$ in the optimization process to make the solution meet local constraints which are based on the color line model. And as mentioned above, we minimize $\varepsilon = \alpha^T G_{nonlocal} \alpha$ to make the solution satisfy the nonlocal constraints. The optimization process can be seen as an extension of Levin et al. (2008). The details are given below.

We first define a subset of pixels M to represent a collection composed of the pixels marked by user. Our optimization can be written in a matrix form as:

$$E(\alpha) = (m - \alpha)^T \Lambda (m - \alpha) + \alpha^T G_{nonlocal} \alpha + \gamma \alpha^T L \alpha, \tag{16}$$

where Λ is a diagonal matrix, and m is a vector. L is a matting Laplacian matrix. Λ_{ii} is 1000 if $i \in M$ and 0 otherwise. m_i is set to

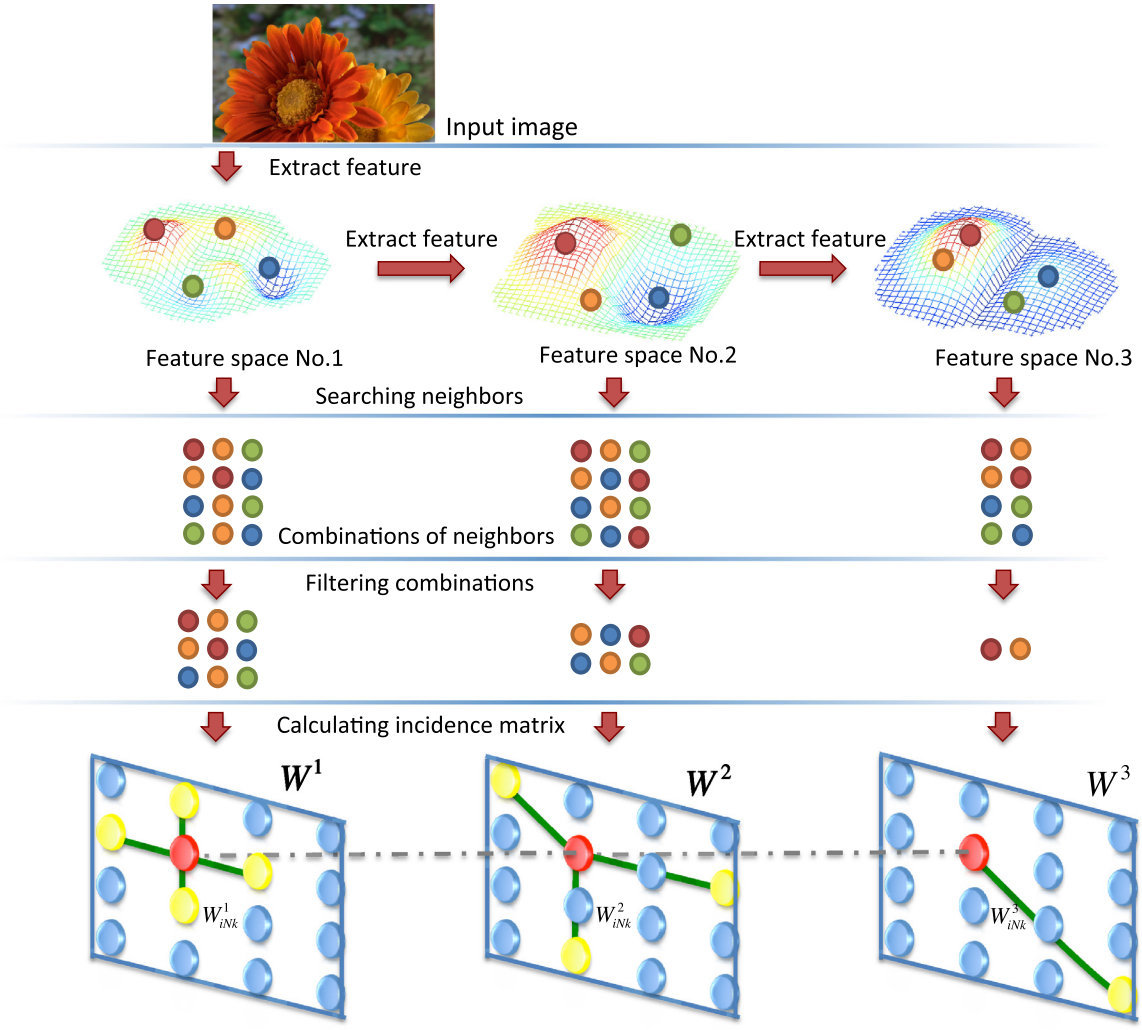

Fig. 3. Illustrative diagram of the three-layer graph framework. This framework consists of four steps: first, extract feature from input image and construct three feature spaces. Second, search neighbors in each feature space and get a series of combinations. Third, select reliable combinations and calculate incidence matrix. Fourth, calculate the weighed sum of the three matrices to unify all combinations.

1 if i belongs to foreground and 0 otherwise. Then

$$\begin{aligned} E(\alpha) &= \alpha^T \Lambda\alpha - 2m^T\Lambda\alpha + \alpha^T G_{nonlocal}\alpha + \gamma\alpha^T L\alpha + m^T\Lambda m \\ &= \frac{1}{2}\alpha^T 2[G_{nonlocal} + \gamma L + \Lambda]\alpha - 2m^T\Lambda\alpha + m^T\Lambda m \\ &= \frac{1}{2}\alpha^T H\alpha - c^T\alpha + m^T\Lambda m. \end{aligned} \tag{17}$$

Note that $H = 2[(G_{nonlocal} + \gamma L + \Lambda]$ is a symmetric positive semi-definite matrix, and we differentiate $E(\alpha)$ w.r.t. α and equate the result to zero:

$$\frac{\partial E}{\partial \alpha} = H\alpha - c = 0. \tag{18}$$

Thus, we can get the optimal solution by solving the linear Eq. (18) in closed-form. Here we leverage the biconjugate gradient stabilized method(BICGSTAB) (Li et al., 2013; Vorst, 1992) to achieve the solution.

2.5. Pre- and post-processing

2.5.1. Label extension

In theory, more valid labels are propitious to achieve accurate alpha matte. Motivated by the recent matting studies (Gastal and Oliveira, 2010; Varnousfaderani and Rajan, 2013), we expand the known regions to the unknown regions by adopting the pre-processing step. Specifically, inspired by the geodesic image and video editing method (Criminisi et al., 2010), we consider an unknown pixel u as a foreground pixel if its geodesic distance from the foreground is smaller than the threshold ξ. Here we define the geodesic distance from the foreground as follows:

$$D(x; F, \nabla) = \min_{x'|x'\in F} d(x, x'), \text{ with} \tag{19}$$

$$d(a, b) = \inf_{\Gamma\in P_{a,b}} \int_0^{\ell(\Gamma)} \sqrt{1 + \tau(x(s))^2 (\nabla I(s)\cdot\Gamma'(s))^2}\, ds, \tag{20}$$

where $I(x)$: $\Psi \to R^3$ is a color image, and $\Psi \subset R^2$ is assumed to be continuous for the time being. $P_{a,b}$ is the set of all possibly differentiable paths in Ψ between points a and b. $\Gamma(s)$: $R \to R^2$ indicates one such path that is parametrized by its arc length $s \in [0, \ell(\Gamma)]$. The geodesic factor τ weighs the contribution of the image gradient versus the spatial distance. Here we define τ as follows:

$$\tau(x) = \min_{x'|x'\in N_x} \frac{1}{J(x')}, \tag{21}$$

where $J(x')$ is the standard deviation of the gray value in 7*7 neighborhood of x'. The integral in (20) can be approximated with

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

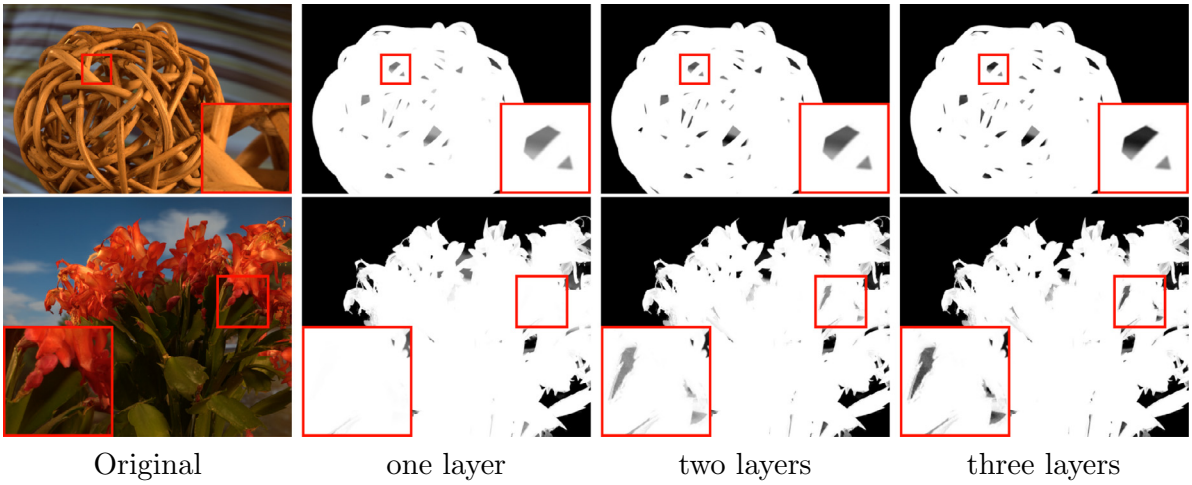

Fig. 4. Analysis on the number of layers. Higher layer can capture pixel connections in larger spacial scale, so the model having more layers tends to get better results in some regions with holes.

Table 1
Summary of notations.

Symbol	Description	Value
K_1	Number of neighbors in graph 1	12
K_2	Number of neighbors in graph 2	7
K_3	Number of neighbors in graph 3	3
σ_1	Filter ratio in graph 1	0.9
$\sigma_i (i = 2, 3)$	Filter ratio in graph $i (i = 2, 3)$	Adaptive with formula (26)
μ	Influence factor of spatial coordinates in graph 1	1/12
μ_2	Influence factor of spatial coordinates in graph 2	1/(12*120)
μ_3	Influence factor of spatial coordinates in graph 3	1/(12*120*12)
$\nu_i (i = 1, 2, 3)$	Influence factor of sumD in graph $i (i = 1, 2, 3)$	Adaptive
γ_1	Influence factor of graph 1	1
γ_2	Influence factor of graph 2	0.3
γ_3	Influence factor of graph 3	0.1
λ	Influence factor of user label	1000
δ	Influence factor of color line model	7
ξ	Threshold in label extension	Adaptive
σ	Standard deviation of PSF	0.5

the following formula:

$$\sum_{k=1}^{n}[|x_k - x_{k-1}|^2 + \tau^2 |I(x_k) - I(x_{k-1})|^2]^{\frac{1}{2}}. \quad (22)$$

Based on this discrete definition of the geodesic distance between two pixels, we use raster-scan method to calculate the geodesic distance of pixel from the foreground (Criminisi et al., 2010).

After obtaining the final map, we set a threshold ξ and choose pixels which satisfy $D(x) < \xi$ as expanded foreground flags. We can use a similar method to get the background flags. In the process of label extension, we make threshold $\xi = \min(d, 20)$, where d is equal to half of the shortest manhattan distance between foreground labels and background labels.

2.5.2. Enhancement of image details

According to the discussion in Rhemann et al. (2010), 2008), blended pixels, whose alpha values are between 0 and 1, may be caused by defocus blur, motion blur, image discretization or light reflected from some translucent medium. In these cases, except that the foreground is translucent object, it is reasonable to suppose that most of the mixed pixels are produced by the point spread function of the camera. When there are too many mixed pixels, the algorithm would be more likely to mistakenly put irrelevant pixels to one combination in the neighbors' searching process.

Based on the above considerations, we perform a preprocessing to enhance the image details. We assume that images are impacted by the point spread function of the camera, and the point spread function is a Gaussian function, as shown in Eq. (23):

$$I = PSF \otimes I^*. \quad (23)$$

In practice, we can calculate I^* as follows:

$$I^* = 2I - G_{\sigma_0} \otimes I + \varepsilon, \quad (24)$$

here G_{σ_0} is a Gaussian kernel function and its parameter $\sigma_0 = 0.5$. Eq. (24) is strictly educed in the Appendix.

According to Eq. (23), we know that the alpha matte satisfies the following formula:

$$\alpha = PSF \otimes \alpha^*. \quad (25)$$

So, after achieving the alpha matte, we use the same point spread function to process it to obtain the alpha values of mixed pixels caused by the point spread function.

3. Experiments

In this section, we determine the parameters in our algorithm and evaluate the proposed approach on a benchmark database (Rhemann et al., 2009a, 2009b). The benchmark database contains 35 natural images. These images show many characteristics of the real-world images, such as the highly textured backgrounds, different degrees of translucency or transparency, different depths

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

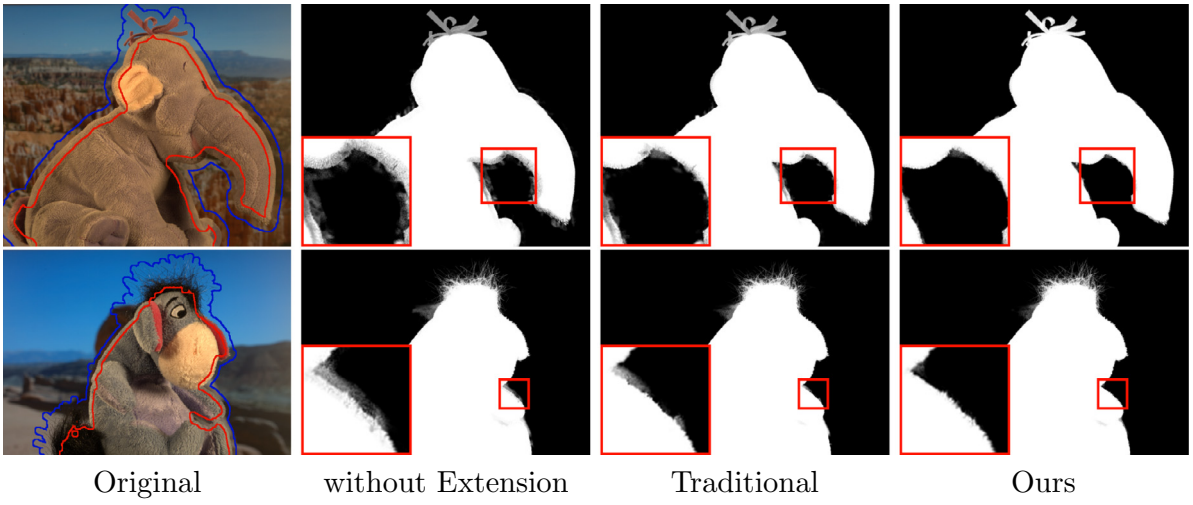

Fig. 5. Qualitative comparison of label extension methods on *elephant* and *donkey* with our matting method. The red line and blue line are label border of foreground and background. The matting result with our label extension method is better than others. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

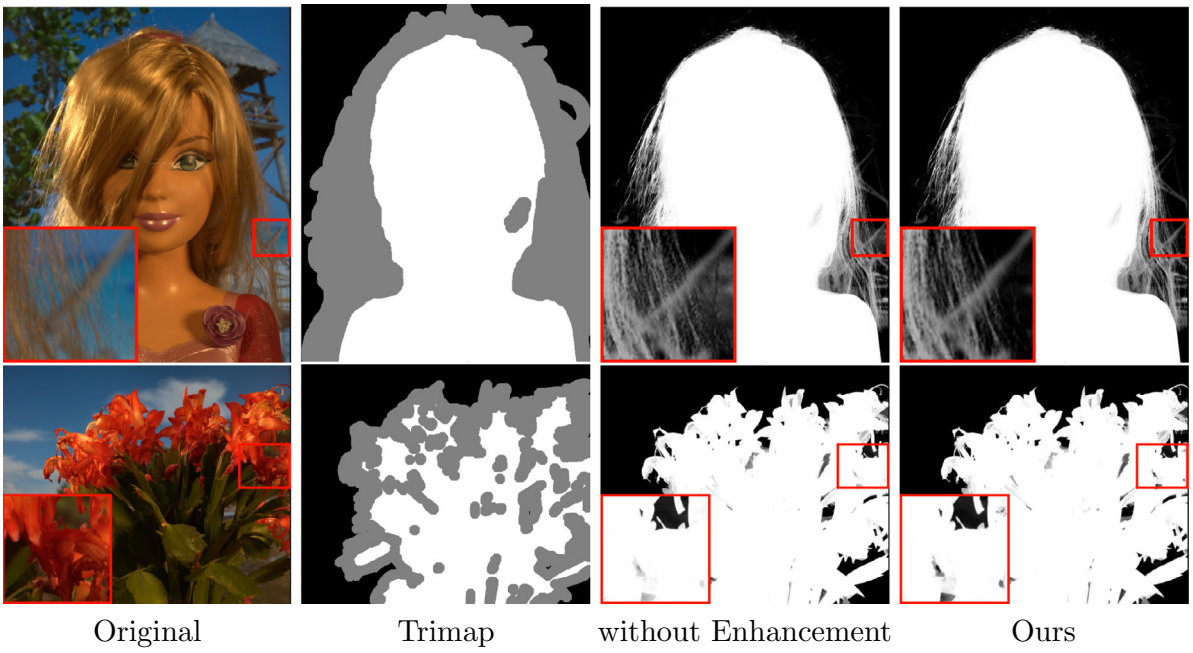

Fig. 6. Effectiveness of the enhancement method. The method with the enhancement method performs better in the zoomed regions which have fine structure with ambiguous color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of fields, and color ambiguity. In this database, 8 images constitute the test set, where the ground truth alpha mattes are hidden from the public to avoid excessive parameter tuning. The remaining 27 images constitute the training dataset where the ground truth alpha mattes are available. We compare our approach with several recent representative methods based on the qualitative and quantitative evaluations.

3.1. Parameters

Choose appropriate σ_2 and σ_3. When building the second graph, we only choose a part of combinations to build nonlocal constraints in order to ensure the reliability of constraints. We denote D as the sum of the squared distance between a pixel and its neighbors in the combination. And we assume that the reliability of a combination has negative correlation to D in an image. If an image has more combinations whose D are smaller, we want to choose more combinations in this image. So we define the following credibility cost formula:

$$y(x) = \frac{\int_0^x \rho(t) D_t dt}{0.7 \cdot D_{0.7}}, \tag{26}$$

here, $x \in [0, 0.7]$ and $y \in [0, 1]$, $\rho(t)$ is the probability density function of D, $D_{0.7}$ is 70th percentile of D. In order to improve the stability of the calculation results, we only select 70% combinations whose D are smaller than others for calculation. Then, we set the threshold of y as 0.15, and calculate σ_2 by formula $y(0.7\sigma) = 0.15$. Similarly, when calculating σ_3 of the third graph, we set the threshold of y as 0.1.

The number of layers. To demonstrate the effectiveness of the number of layers, we use one-layer graph model, two-layer graph model and three-layer graph model with constant param-

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

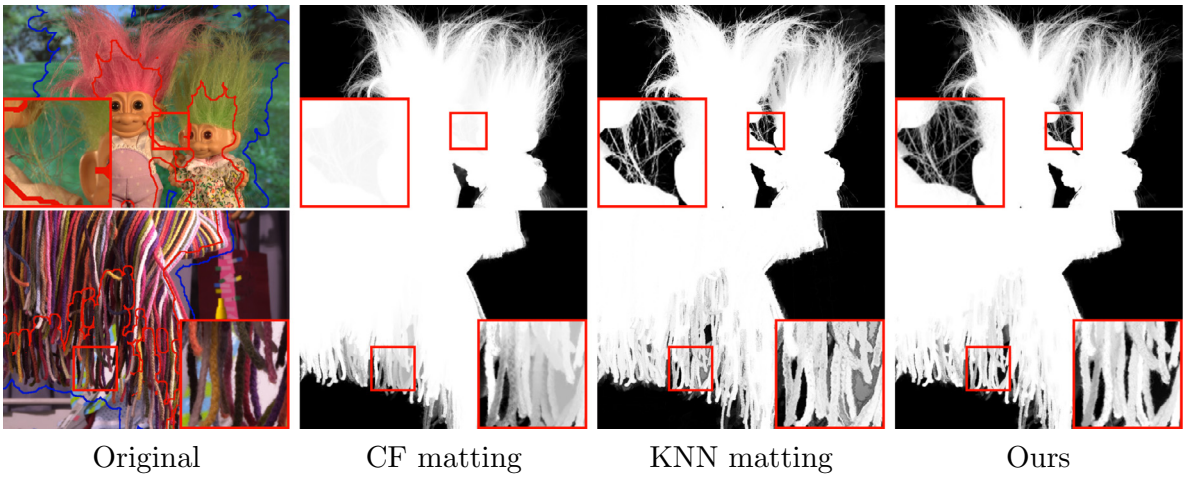

Fig. 7. Comparison without pre- and post-processing. The red line and blue line are label border of foreground and background. The result of our method is more smooth than that of KNN matting and our performance is better than that of CF matting in the regions with holes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 8. Matting results with sparse strokes. The red line and blue line are foreground and background labels respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

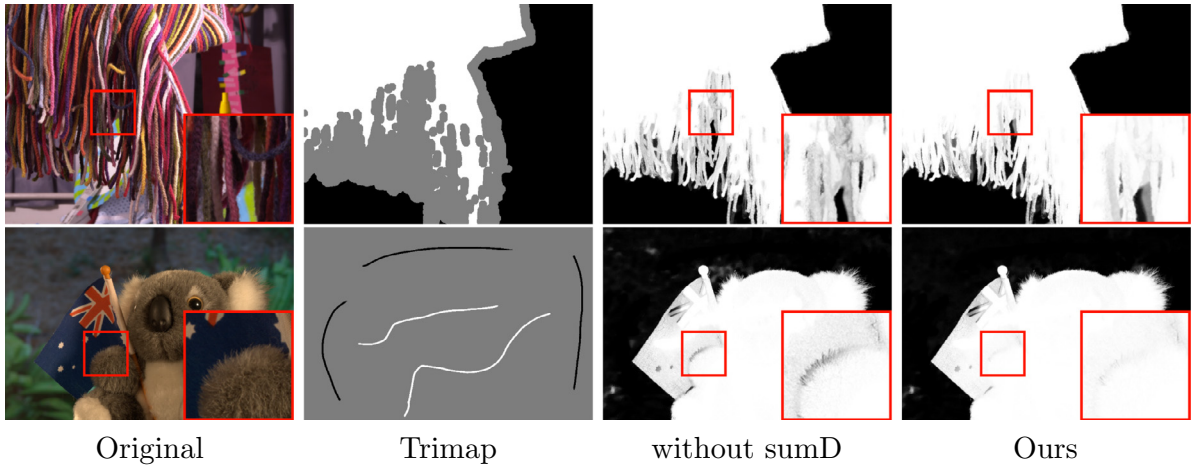

Fig. 9. Effectiveness of the sumD feature. The method with the sumD feature performs better in the regions with ambiguous color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

eters to calculate alpha mattes on the training set, respectively. Fig. 4 shows the effect of graph model with different layers. The model with fewer layers tends to fail in regions with hole, but the model with more layers is more robust in such regions. This is because higher layer graph can capture pixel connections in larger spacial scale.

Other parameters. Considering the reasonableness of the experiments, we give the parameter values in our algorithm. Then we use the controlling variables method to compare the effectiveness of the three-layer graph model. Table 1 gives the parameter values in our algorithm, and they are consistent in the following experiments.

As mentioned above, multi-layer graph includes multi-scale information of image, we set $\mu \gg \mu_2 > \mu_3$. So higher layer graph has larger scale information. And because the confidence of combinations decreases with the increase of spatial distance, we choose combinations carefully in large spatial scale. Based on this principle, we set $K_1 > K_2 > K_3$, $\sigma_1 > \sigma_2 > \sigma_3$ and $\gamma_1 > \gamma_2 > \gamma_3$. Besides, it's reasonable to assume that the confidence of labels is the highest, and the confidence of color line model is higher

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

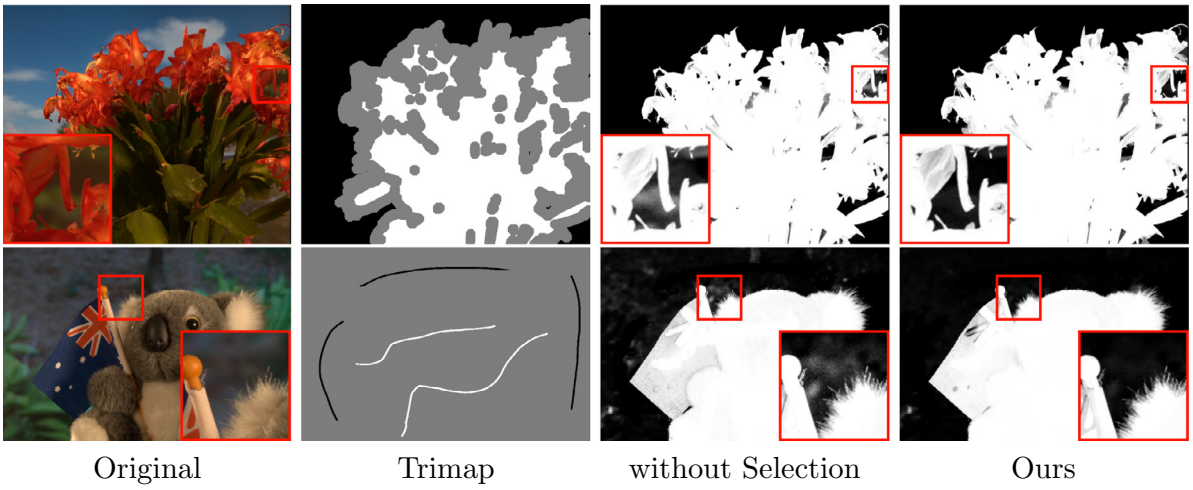

Fig. 10. Effectiveness of the combinations selection. The method with the selection process performs better overall, especially to the tasks with strokes.

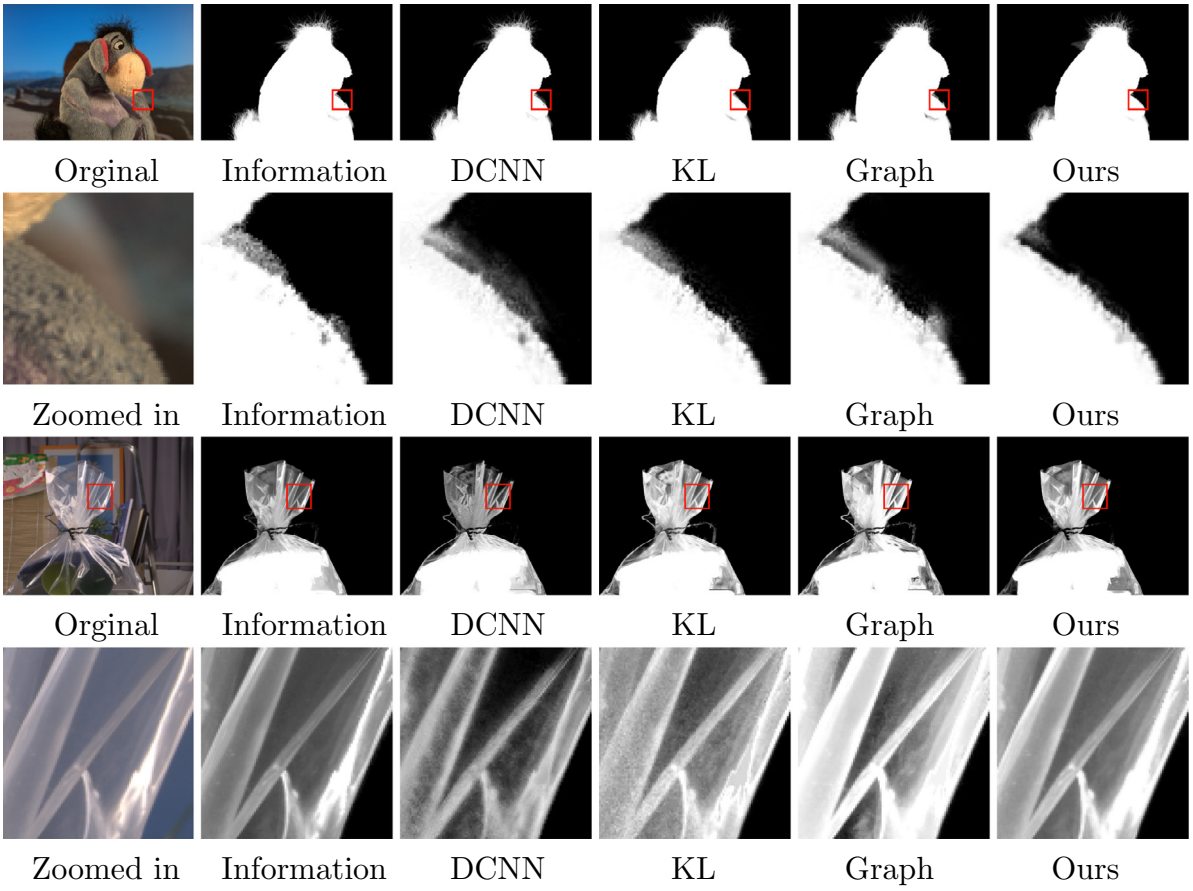

Fig. 11. Qualitative comparison of matting methods on nature images from Rhemann et al. (2009b). Top: Images with medium transparent. In zoomed area, the donkey's leg and the background have similar color, the matting results of this region may have artifacts. Bottom: Images with high transparency. In zoomed area, the opacity of foreground is related to reflected light. More details can be seen on Rhemann et al. (2009b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

than the three-layer graph model, so we set $\lambda \gg \delta > 1$. Other parameters are given by experience.

3.2. Effectiveness evaluation

Effectiveness of our label extension method. Some images have many mixed pixels, and foreground and background have similar colors. Matting method is difficult to achieve accurate alpha matte of these images. So many researchers use label extension method as a pre-processing method (Gastal and Oliveira, 2010; Varnousfaderani and Rajan, 2013). In this paper, we propose a novel label extension method. In order to demonstrate the effectiveness of this method, we consider three situations before executing our algorithm: using our method to extend labels, using method in Gastal and Oliveira (2010) to extend labels, and no label extension. Then we use our matting method to achieve the

Table 2
Rank of different matting methods with Sum of Absolute Differences (SAD), Mean Square Error (MSE) and Gradient Error (GE) at Rhemann et al. (2009b).

Method	SAD			
	Avg. small rank	Avg. large rank	Avg. user rank	Overall rank
Information-flow matting	3.9	2.8	2.9	3.2
Deep matting	4.1	2.5	3.3	3.3
DCNN matting	6.1	2.8	4.8	4.5
Proposed method	5.8	6.1	14.3	8.7
ATPM matting	14.5	12.9	7	11.5
CSC matting	15.8	8.4	12.6	12.3
LNSP matting	9.3	12.6	16.8	12.9
Graph-based sparse matting	13.9	14	12.3	13.4
Patch-based matting	8.6	15.1	16.6	13.5
KL-divergence based sparse sampling	12.9	13.3	15.4	13.8
Method	MSE			
	Avg. small rank	Avg. large rank	Avg. user rank	Overall rank
Information-flow matting	6.4	3.5	4.1	4.7
DCNN matting	5.8	3	5.8	4.8
Deep matting	3.9	4.5	7	5.1
Proposed method	6.5	6.6	14.1	9.1
LNSP matting	8.5	10.6	14.1	11.1
Patch-based matting	8.8	12.8	15.6	12.4
KL-divergence based sparse sampling	12.9	12.6	15.3	13.6
CCM	16.9	14.4	10.3	13.8
ATPM matting	17.8	15.4	9.1	14.1
Graph-based sparse matting	14.6	14.9	13.5	14.3
Method	GE			
	Avg. small rank	Avg. large rank	Avg. user rank	Overall rank
DCNN matting	10.6	7.6	6.5	8.3
Deep matting	6.8	6.6	14.8	9.4
Information-flow matting	11.9	7.8	9.8	9.8
Proposed method	9.1	9	16.9	11.7
Graph-based sparse matting	9.5	10.4	15.5	11.8
KL-divergence based sparse sampling	10.3	11	14.8	12
Patch-based matting	9	11.5	15.6	12
LNSP matting	9.9	12.3	15.3	12.5
Comprehensive sampling	13.6	13.4	13.4	13.5
ATPM matting	15.5	18.5	11.3	15.1

alpha matte and compare the results of these three situations. As shown in Fig. 5, the result with our label extension method is significantly better than other results. Because our label extension method is based on geodesic distance, the results tend to be more smooth than that in Gastal and Oliveira (2010). Moreover, our method is sensitive to significant color changes and could adjust the threshold according to trimap adaptively, so it is not easy to make mistakes and can achieve more sufficient extension results. Our label extension method processes an image in database with only about 3 s in Octave and C++ environments.

Effectiveness of the enhancement method. To analyze the effectiveness of the enhancement method in Section 2.5.2, we compare the results with and without this method in Fig. 6. The improvements often occur in the regions which have fine structure with ambiguous color. In the zoomed region of the first row, some pixels in doll's hair are blended with the background, it is easy to use pixels of background as neighbors when searching neighbors. We can see that the result without enhancement method has some artifact in the zoomed region, but this problem is improved after introducing the enhancement method. Similarly, the regions of the second row have some small holes with ambiguous color, we can see that the result with the enhancement method performs better with these holes. The image in the second row has some regions similar to the zoomed region, and the similar phenomenon occurs in these regions. So we think this phenomenon is not by chance. Here we note that the improvement is not significant, so the enhancement method could be removed for brevity. But we use this method because it is simple and can be finished in little time.

Comparison without pre- and post-processing. To demonstrate the effectiveness of the proposed framework, we compare our method with closed-form matting (Levin et al., 2008) and KNN matting (Chen et al., 2013a) without pre- and post-processing. These two methods are closely related with our method, because we use the color line model in closed-form matting as local constrains and use nonlocal principle inspired by KNN matting. Fig. 7 shows the comparison of the results. We can see CF matting may fail in the regions with holes if the trimap is insufficient, such as the zoomed region in the first row. KNN matting could work in this region, but its result is not smooth than ours. In the zoomed region of the second row, KNN result has halo, while our results have better performance in this region. We think the advantage of our framework is that we combine local and non-local constrains and introduce the sumD feature.

Matting with sparse strokes. To evaluate the practicability and robustness of the proposed method, we also test our method with sparse strokes. Fig. 8 demonstrates some matting results in the training set, which shows our method can work with little labels.

Effectiveness of the sumD feature. Fig. 9 shows some results of methods with and without the sumD feature. If we use only the original feature, our method is prone to achieve artifacts in the regions with ambiguous color. This weakness is more obvious with sparse trimap. We can see that our method with the sumD feature performs better in these regions, and the results with the sumD feature are more smooth. This is because the sumD feature reflects the neighbors' distribution in the feature space and encourages pixels with similar texture to be connected in the graph.

Effectiveness of the combinations selection. To demonstrate the effectiveness of the selection process, we compare the results with and without the selection process in Fig. 10. In the first row, although the result without selection performs better with some holes, there are many artifacts in the zoomed region. This is because if we remove the selection process, some remote regions could be connected together, but the unreliable nonlocal constrains are also introduced. The advantages of the combinations selection become more obvious when matting with sparse strokes. In the second row, the result with selection performs significantly better. So we can conclude that the combinations selection is important to the robustness of the proposed framework.

3.3. Evaluation on benchmark database

Qualitative evaluation on test images. Fig. 11 gives the results of our method and that of the state-of-the-art algorithms in some test images. In image *donkey*, the donkey and the background have similar colors, which easily leads to artifacts. We can see some artifacts in that region, while our method and DCNN matting (Cho et al., 2016) can separate them better. In matting results of *plasticbag*, the alpha values in the result of Graph-based sparse matting (Johnson et al., 2016) is higher than others. The results of DCNN matting (Cho et al., 2016) and KL matting (Karacan et al., 2015) have poorer smoothness. Compared to these two methods, the results of propagation-based methods have stronger correlations among adjacent pixels. In the zoomed area, there is a region which has stronger reflected light, so there are higher alpha values. But the results of DCNN have lower alpha values in this region, while our method and Information flow matting (Aksoy et al., 2017) have better performance. More details can be seen on Rhemann et al. (2009b).

Please cite this article as: C. Li et al., Three-layer graph framework with the sumD feature for alpha matting, Computer Vision and Image Understanding (2017), http://dx.doi.org/10.1016/j.cviu.2017.06.011

Rank on website. We compare our method with state-of-the-art algorithms on website (Rhemann et al., 2009b). As shown in Table 2, our algorithm ranks fourth in SAD,MSE and GE. Note that Deep matting and DCNN matting are deep learning methods, they need training before using them. Our method only needs the natural image as input and could achieve accurate matting results. So comparing to the methods without training process, our method's accuracy ranks the second.

4. Conclusion

We proposed a matting method based on the sumD feature and the three-layer graph framework which builds constrains in different spacial scales and selects more reliable ones. The framework improves sufficiency and reliability of nonlocal constraints. Besides, we introduced a new label expansion algorithm as a pre-processing method. This method helps the proposed framework achieve higher accuracy. Different from the existing methods, our label extension method is based on the geodesic distance transform. Experiments confirm that the effectiveness of the proposed feature, framework and label extension method. And the evaluation on benchmark database demonstrates that our method ranks in front compared with the state-of-the-art methods and ranks the second compared with the methods without training process. In the future, to further improve the nonlocal constraints will be a research direction, and to introduce the sumD feature to other computer vision tasks is also an interesting direction.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 61379014) and the Natural Science Foundation of Tianjin (No. 16JCYBJC15900)

Appendix

Given $I = PSF \otimes I^*$, here PSF is a Gaussian function. Denote G_{σ_0} as a gaussian function and its parameter is σ_0. Then there is a number ε satisfying $I^* = 2I - G_{\sigma_0} \otimes I + \varepsilon$, and when $\sigma_0 \to 0$, $\varepsilon \to 0$.

Proof. Define a function

$$G(I^*, \sigma) = G_{\sigma_0} \otimes I^*, \tag{27}$$

Function $G(I^*, \sigma_0)$ is continuous and derivable for σ_0. If σ_0 is sufficiently small, then there is a number ε satisfying

$$G(I^*, \sigma_0) - G(I^*, 0) = G(I^*, 2\sigma_0) - G(I^*, \sigma_0) + \varepsilon, \tag{28}$$

and when $\sigma_0 \to 0$, $\varepsilon \to 0$. So if $G(I^*, \sigma_0) = I$ and σ_0 is sufficiently small, we can achieve the formulas as follows:

$$\begin{aligned} I^* &= G(I^*, 0) \\ &= G(I^*, \sigma_0) - [G(I^*, 2\sigma_0) - G(I^*, \sigma_0)] + \varepsilon \\ &= I - (G_{\sigma_0} \otimes I - I) + \varepsilon \\ &= 2I - G_{\sigma_0} \otimes I + \varepsilon, \end{aligned} \tag{29}$$

and when $\sigma_0 \to 0$, $\varepsilon \to 0$. □

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cviu.2017.06.011

References

Aksoy, Y., Aydin, T.O., Pollefeys, M., 2017. Designing effective inter-pixel information flow for natural image matting. In: Computer Vision and Pattern Recognition, 2017. CVPR 2017. IEEE Conference on.
Buades, A., Coll, B., Morel, J.-M., 2005. A non-local algorithm for image denoising. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2. IEEE, pp. 60–65.
Chen, Q., Li, D., Tang, C.-K., 2013a. Knn matting. IEEE Trans. Pattern Anal. Mach. Intell. 35 (9), 2175–2188.
Chen, X., Zou, D., Zhao, Q., Tan, P., 2012. Manifold preserving edit propagation. ACM Trans. Graph. 31 (6), 132.
Chen, X., Zou, D., Zhiying Zhou, S., Zhao, Q., Tan, P., 2013b. Image matting with local and nonlocal smooth priors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1902–1907.
Cho, D., Tai, Y.-W., Kweon, I., 2016. Natural image matting using deep convolutional neural networks. In: European Conference on Computer Vision. Springer, pp. 626–643.
Choi, I., Lee, M., Tai, Y.W., 2012. Video matting using multi-frame nonlocal matting laplacian. In: European Conference on Computer Vision, pp. 540–553.
Criminisi, A., Sharp, T., Rother, C., Pérez, P., 2010. Geodesic image and video editing. ACM Trans. Graph. 29 (5), 134.
Gastal, E.S., Oliveira, M.M., 2010. Shared sampling for real-time alpha matting. In: Computer Graphics Forum, vol. 29. Wiley Online Library, pp. 575–584.
Hasinoff, S.W., Kang, S.B., Szeliski, R., 2006. Boundary matting for view synthesis. Comput. Vision Image Understanding 103 (1), 22–32.
Johnson, J., Rajan, D., Cholakkal, H., 2014. Sparse codes as alpha matte. In: BMVC, vol. 1, p. 5.
Johnson, J., Varnousfaderani, E.S., Cholakkal, H., Rajan, D., 2016. Sparse coding for alpha matting. IEEE Trans. Image Process. 25 (7), 3032–3043.
Karacan, L., Erdem, A., Erdem, E., 2015. Image matting with kl-divergence based sparse sampling. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 424–432.
Lee, P., Wu, Y., 2011. Nonlocal matting. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, pp. 2193–2200.
Levin, A., Lischinski, D., Weiss, Y., 2008. A closed-form solution to natural image matting. IEEE Trans. Pattern Anal. Mach. Intell. 30 (2), 228–242.
Li, D., Chen, Q., Tang, C.K., 2013. Motion-aware knn laplacian for video matting. In: IEEE International Conference on Computer Vision, pp. 3599–3606.
Rhemann, C., Rother, C., Kohli, P., Gelautz, M., 2010. A spatially varying psf-based prior for alpha matting. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, pp. 2149–2156.
Rhemann, C., Rother, C., Rav-Acha, A., Sharp, T., 2008. High resolution matting via interactive trimap segmentation. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, pp. 1–8.
Rhemann, C., Rother, C., Wang, J., Gelautz, M., Kohli, P., Rott, P., 2009a. A perceptually motivated online benchmark for image matting. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, pp. 1826–1833.
Rhemann, C., Rother, C., Wang, J., Gelautz, M., Kohli, P., Rott, P., 2009b. http://www.alphamatting.com.
Saul, L. K., Roweis, S. T., 2000. An introduction to locally linear embedding. unpublished. Available at: http://www.cs.toronto.edu/~roweis/lle/publications.html.
Varnousfaderani, E.S., Rajan, D., 2013. Weighted color and texture sample selection for image matting. IEEE Trans. Image Process. 22 (11), 4260–4270.
Vorst, H.A.V.D., 1992. Bi-cgstab: a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. Siam J. Sci. Stat. Comput. 13 (2), 631–644.
Xu, N., Price, B., Cohen, S., Huang, T., 2017. Deep image matting. In: Computer Vision and Pattern Recognition, 2017. CVPR 2017. IEEE Conference on. IEEE arXiv:1703.03872.
Zhang, Z., Zhu, Q., Xie, Y., 2012. Learning based alpha matting using support vector regression. In: 2012 19th IEEE International Conference on Image Processing. IEEE, pp. 2109–2112.
Zhu, Q., Shao, L., Li, X., Wang, L., 2015. Targeting accurate object extraction from an image: a comprehensive study of natural image matting. IEEE Trans. Neural Netw. Learn. Syst. 26 (2), 185–207.