Fast Closed-form Matting Using Hierarchical Data Structure

Chunxia Xiao, Meng Liu, Donglin Xiao, Zhao Dong, and Kwan-Liu Ma, *Fellow, IEEE*

***Abstract*—Image/video matting is one of the key operations in many image/video editing applications. Although previous methods can generate high-quality matting results, their high computational cost in processing high-resolution image and video data often limits their usability. In this paper, we present a unified acceleration method for closed-form image and video matting using a hierarchical data structure, which achieves an excellent compromise between quality and speed. We first apply Gaussian KD-tree to adaptively cluster the input high-dimensional image and video feature space into a low-dimensional feature space. Then, we solve the affinity-weighted Laplacian alpha matting in the reduced feature space. The final matting results are derived using detail-aware alpha interpolation. Our algorithm can be fully parallelized by exploiting advanced graphics hardware, which can further accelerate the matting computation. Our method accelerates existing methods by at least an order of magnitude with good quality, and also greatly reduces the memory consumption. This acceleration strategy is also extended to support other affinity-based matting approaches, which makes it a more general accelerating framework for a variety of matting methods. Finally, we apply the presented method to accelerate image and video dehazing, and image shadow detection and removal.**

***Index Terms*—Gaussian KD-tree, matting, dehazing, acceleration, GPU**

I. Introduction

MATTING aims at accurately extracting foreground objects from input image or video, which is an important technique in image and video processing and film production applications. Although matting has been extensively studied (refer to [1], [2] for a survey), its accuracy and efficiency still need to be improved, so as to develop intelligent, user-friendly, computationally efficient matting tools. Recently, several affinity-based matting methods [3], [4], [5], [6], [7], [8], [9] succeed in generating more accurate results than early approaches. However, these methods are generally expensive to compute, as they need to solve a large linear system. Moreover, the high memory consumption of these methods also limit their performance especially when dealing with gigapixel images or video sequences.

Closed-form matting (CFM) [3] is one representative affinity-based matting approach and its matting quality ranks top among the methods in the online image matting benchmark list [10]. The CFM method derives a cost function from local smoothness assumption on foreground and background colors. By eliminating the foreground color (F) and background color (B), this cost function can be deduced to a quadratic cost function only containing alpha value. The globally optimal alpha matte can then be computed by solving a sparse linear system of equations. Although the CFM method [3] works well for image matting, its high time and memory consumption for solving a large linear system make it intractable to process high-resolution images or videos.

However, for the natural image and video matting, most pixels are definite foreground or background, and there are many clusters of pixels that are highly similar in feature space. Thus, it is wasteful to perform the CFM optimization for every pixel. Motivated by it, we accelerate the CFM method [3] using a hierarchical data structure. Firstly, the input pixels are grouped into clusters based on feature similarity metric using the affinity-based Gaussian KD-tree [11]. Hence, a much smaller number of Gaussian weighted feature samples of the original data set are obtained, which forms the reduced feature space approximating the original data set. Then we construct an affinity-weighted Laplacian matrix with the feature samples to obtain the matting alpha values in the reduced feature space. Finally, we generate the matting results using a new detail-aware alpha interpolation method. With such a hierarchical strategy, we can construct the Laplacian matrix using a small number of samples and solve it in low-resolution space, which breaks the time and memory consumption bottleneck in the original CFM method. By applying the Gaussian KD-tree on video input and constructing affinity-weighted Laplacian matrix based on 3D local neighboring samples, our method can be easily extended to efficiently handle video matting. Hence, our method is a unified acceleration strategy for both image and video matting.

Our method can be further applied as a general acceleration approach for various other affinity-based matting methods [3], [4], [5], [6], [12], [9] with slight modifications. We apply the presented strategy to spectral matting [5] and robust matting [9], which demonstrates that although the underlying energy functions for these methods are quite different from each other, they can share our method as a

Chunxia Xiao, Meng Liu and Donglin Xiao are with the Computer School, Wuhan University, Wuhan, Hubei, China, 430072. E-mail: cxxiao@whu.edu.cn, mengliu.whu@gmail.com, aydone.xiao@gmail.com.

Zhao Dong is with the program of computer graphics, Cornell University, NY, USA, 14853. E-mail:zd@graphics.cornell.edu

Kwan-Liu Ma is with the department of computer science, University of California-Davis, CA, USA, 95616-8562. E-mail: ma@cs.ucdavis.edu

Corresponding to Chunxia Xiao: cxxiao@whu.edu.cn

general matting acceleration framework.

We notice that the haze imaging equation has a similar form as the matting equation [13], where a transmission map is exactly an alpha map. As an application, we apply the accelerated CFM as a fast extrapolation tool to refine the computed transmission using prior-dark channel prior [13] for fast image and video dehazing. As shadow detection and removal is important in image/video editing and segmentation [14], [15], we further apply our method to accelerate and refine this application.

The Gaussian KD-tree strategy for CFM acceleration has the following advantages: Firstly, high image/video matting quality as the CFM method can be obtained, while the computation and memory costs are significantly reduced. Secondly, CFM can be further accelerated by one order of magnitude via the evolving graphics hardware, since the three main steps of the acceleration procedure, including Gaussian KD-tree construction, affinity-based Laplacian matrix construction, and alpha interpolation, can be implemented in parallel. Thirdly, it can be further applied as a general acceleration approach for various other affinity-based matting methods with slight modifications.

In summary, our major contributions are:

- A unified acceleration method for the closed-form image/video matting using hierarchical data structure, which achieves both good quality and high performance.
- Extending the acceleration method to support other affinity-based matting methods, which makes it a more general matting acceleration framework.
- Accelerating the image/video dehazing and image shadow removal by applying the presented accelerated CFM as a fast interpolation tool.

II. Related Works

A complete review of all existing image and video matting methods is beyond the scope of this article and we refer the reader to [1] for a detailed introduction. Here, we only discuss the most relevant works.

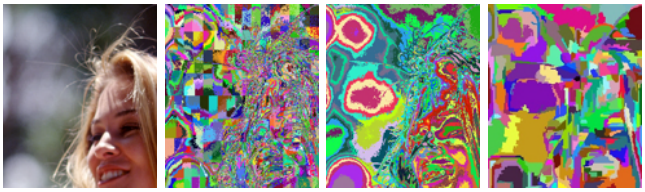
Image and Video Matting: Existing image matting methods can be roughly classified into three categories: affinity-based, sampling-based, and approaches combining sampling with affinity. Affinity-based matting methods compute the alpha values for unknown region based on the alpha interpolation from the known regions determined by user inputs. It defines the affinities between neighboring pixels by modeling the matte gradient for each pixel in its local neighbor window. The interpolation can therefore be achieved by solving an affinity matrix [16], [4], [3], [17], [18]. However, the complex computations incurred by affinity-based matting always becomes the bottleneck when dealing with high-resolution images. The affinity-based methods compute alpha value without explicitly accounting for the foreground and background colors. In contrast, the sampling-based methods [19], [20], [21], [22] assume that the true foreground and background colors of a pixel in unknown region can be explicitly estimated by analyzing nearby pixels in known region.

Both sampling-based and affinity-based approaches have pros and cons. The sampling-based methods work better when dealing with distinct foreground and background color distributions along with carefully specified trimaps, but tend to generate large errors when their underlying assumptions are violated. Comparatively, the affinity-based approaches are relatively insensitive to different user inputs and always generate smooth mattes, although their results might not be accurate enough for long and thin (i.e. furry) foreground structures. Moreover, they are usually time-consuming. Matting approaches combining sampling with affinity have been developed [8], [7], [9], which achieve a good trade-off between accuracy and robustness. These methods normally defines an energy function taking the form of a "Gibbs" energy to be minimized. Although these methods produce satisfactory mattes, their performance is still low for handling high-resolution image.

Video matting is inherently a more challenging task and the difficulties mainly arise from the following aspects: (1) large data size, (2) difficulty in maintaining temporal coherence, and (3) tedious user interactions. Many video matting methods have been presented [12], [23]–[26], and most of them extends an image matting method to handle 3D video data with time as the third dimension. For example, Bayesian video matting [23], [24] follows the technical routine of Bayesian image matting [20], and adopts a bidirectional optical flow algorithm to interpolate the trimaps in key frames. Please refer to [1] for more discussions about video matting methods.

CFM acceleration: The optimization problem of closed-form matting (CFM) [3] is to minimize a quadratic cost function subject to linear constraints, and the solution can be obtained by solving a large set of sparse linear equations. Solving linear system with this structure is a well-studied problem with various solutions, such as preconditioning [27] and multigrid [28] methods. Levin et al. [3] applied a simple multi-resolution solver to accelerate matte extraction. Although multi-resolution solver can perform CFM in a couple of seconds for a low-resolution image, this method will degrade the matte quality. He et al. [18] presented a large kernel matting Laplacian matrices to speed up the CFM. Although this method can receive pleasing matting results, matting efficiency is still too low for interactive applications. To apply GPU acceleration strategy for CFM, Huang et al. [29] proposed a multi-scale windowing scheme, which first computes solutions in local windows one at a time, and then combines the local solutions to achieve the global results. Although this method is efficient for accelerating CFM, our acceleration method based on hierarchical data structure is much more general compared with it, which can be applied to accelerate various affinity-based matting methods [4], [5], [6], [12], [9], not just CFM.

KD-tree Acceleration: As an efficient hierarchical data structure for accelerating data clustering, KD-tree has been widely applied in computer graphics and vision applications. Utilizing the massive parallelism of modern GPU, it is possible to construct a KD-tree on GPU in real-



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

Fig. 1. Comparison of different clustering techniques: (a) Input image, and clustering results using Gaussian KD-tree (b), k-means (c) and mean-shift (d).

time. Horn et al. [30] and Zhou et al. [31] presented GPU-based KD-tree construction algorithms achieving real-time performance. KD-Tree also has been used in image and video processing [32]. Adam et al. [11] developed Gaussian KD-tree to accelerate a class of non-linear filters, including bilateral image filtering, bilateral mesh filtering, and image denoising with nonlocal means. This method [11] achieved both magnificent performance improvement and high-quality results. Our method adopts Gaussian KD-tree as the hierarchical data structure to accelerate the image and video matting. Note, although sharing the similar motivation with [11], we developed different acceleration strategies in our method for fast matting applications, which include (1) constructing a new affinity-weighted Laplacian matrix for the feature samples to obtain the matting alpha values in the reduced feature space, and (2) generating the matting results using a new detail-aware alpha interpolation.

Many clustering methods have been proposed, such as the popular k-means clustering (KMC) [33], and mean-shift clustering (MSC) [34]. Although KMC works fast for data clustering, for the image/video matting task, k-means clustering may not generate accurate and fine sampling/clustering result along the image edges, which is critical for extracting accurate matte along the boundaries of the foreground object, e.g. the fine hair in Fig.1. MSC works by defining a Gaussian kernel density estimate for underlying data, and then clustering the similar points together through a fixed-point iterative process until converging to the same mode. MSC works well for small-scale data clustering and segmentation, while its performance is slow for clustering large data set. Further, MSC cannot produce accurate and fine sampling along the image edges, either. In comparison, the Gaussian KD-tree clustering (GKDC) generates accurate and fine enough results along the image edges. As shown in Fig.1, the result using GKDC produces finer and more accurate sampling along the image edge compared with both KMC and MSC. The GKDC also greatly reduces both time and memory consumption, and hence easily deals with large data set. Moreover, using the KD-tree structure, it is more convenient to control the hierarchical clustering scheme, which helps accelerating the matting process using hierarchical data structure.

III. Closed-form Matting

The closed-form matting (CFM) approach [3] explicitly derives a cost function using local smoothness assumption on foreground and background colors, referred to as F and B, respectively. During the derivation, it is possible to analytically eliminate F and B to finally obtain a quadratic cost function of matte α , which can be further solved as a sparse linear system. The underlying assumption made in CFM is that each F and B is a linear mixture of two colors over a small window w_j (typically 3×3 or 5×5) around each pixel, which is referred to as the color line model. Under this assumption, the alpha value in a small window can be expressed as

$$\alpha_i = \sum_c a^c I_i^c + b, \forall i \in w \tag{1}$$

where c refers to color channels, and a^c and b are constants in local window. The matting cost function is then defined as

$$J(\alpha, a, b) = \sum_{j \in I} \left(\sum_{i \in w} (\alpha_i - a_j^c I_i^c - b_j)^2 + \varepsilon a_j^c \right) \tag{2}$$

a^c and b can be further eliminated from the cost function, yielding a quadratic cost with α alone:

$$J(\alpha) = \alpha^T L \alpha \tag{3}$$

where L is an $n \times n$ Laplacian matrix (n pixels in the image). The optimal alpha value is then computed as

$$\alpha = \text{argmin}(\alpha^T L \alpha) \tag{4}$$

which is essentially a problem of minimizing a quadratic error, and can be solved by a linear system solver. Although CFM achieves very good image matting quality, it works on the per-pixel basis and hence introduces a large linear system for high-dimension image and video, the computational cost and memory consumption of which are prohibitive.

IV. Fast Closed-form Matting

In this section, we describe how to accelerate the CFM [3] method using the Gaussian KD-tree [11]. Our method takes an image I (or video sequence) with n pixels and the interactive user strokes as input. It consists of the following major steps: (a) Building the affinity-based KD-tree for input image/video while maintaining the local smooth assumption. (b) Propagating the input user strokes to each tree leaf node. (c) Constructing and solving the affinity-based Laplacian matrix in low-dimensional space. (d) Restoring the matte result for input image using up-sampling. In the following subsections, we introduce the algorithmic details for each step.

A. Gaussian KD-tree partitioning

Affinity space definition: We adopt the affinity measure from [35], [32] to define the affinity space of the image/video data. For each pixel i, we represent it using a 5-dimensional feature vector $f_i = (p_i, v_i)$ where p_i and v_i are the 2D position and the color (appearance) of i, respectively. For video data, the feature vector can be expanded to include the frame index (i.e. time) t_i of i, which becomes a 6-dimensional vector $f_i = (p_i, v_i, t_i)$. Further weighting

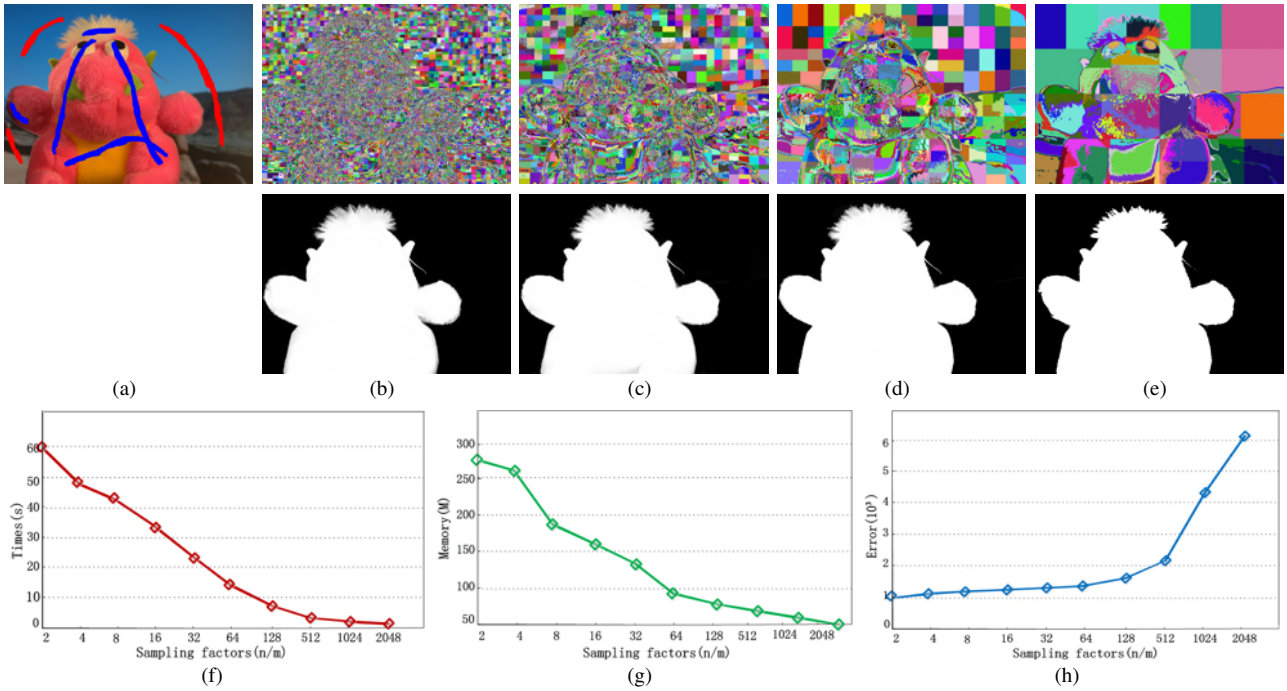


Fig. 2. Image matting using different sampling factor. (a) Original image with size 800×600, (b) n/m =13, (c) n/m=70, (d) n/m =300, (e) n/m =2800. Here, n is the number of image pixels and m is the number of sampling points in the reduced feature space. For different sampling factors, we give the (f) time consumption, (g) memory consumption, and (h) the error from the ground truth results.

f_i by the parameters σ_p, σ_v and σ_t, the similarity measure (or affinity) between two image pixels can be defined as $z_{ij} = exp(-|f_i - f_j|)^2$, where $f_i = (p_i/\sigma_p, v_i/\sigma_v)$ and for video data, $f_i = (p_i/\sigma_p, v_i/\sigma_v, t_i/\sigma_t)$. Here, we set σ_v as the standard deviation of the color in image or video, and set σ_p and σ_t to be image resolution and video length, respectively.

Gaussian KD-tree partitioning: Since nearby pixels in the affinity space represent similar appearance and hold similar matte values, the extracted matte values in the affinity space form a smooth function. For natural image and video, the close-by pixels share similar color and should also have similar matte values. Clearly, it is wasteful to perform the CFM optimization for every pixel. Therefore we can apply hierarchical strategy to adaptively partition the input data set in the feature space based on affinity measure. The regions where the point feature vectors having large variances can be subdivided finely, while the regions where the feature vectors are similar can be subdivided coarsely. Representing each generated cluster with one sample point, an approximate feature space of the original data set with much less samples is built.

The partition in the feature space is pursued in a top-down fashion. Starting from a rectangular root node containing all the sample points, we recursively split a node into two child nodes midway along an axis that is alternated at successive tree levels. Inspired by Gaussian KD-tree [11], each inner node of the tree T represents a d-dimensional rectangular cell which stores six variables: the dimension ID along which the tree cuts, the cut value T_{cut} on that dimension, the bounds of the cell in that dimension T_{min} and T_{max}, and the pointers to its children T_{left} and T_{right}. Each leaf node stores two variables: its ID and a d-dimensional sample point representing all the points in current node.

For each leaf node, to meet the local smoothness assumption in CFM, we analyze the color variance within it. If the variance is above a user-defined threshold δ_{cv}, the cell is further subdivided. Using such a strategy, in the cell of each leaf node the local smoothness assumption is satisfied. Terminating the splitting procedure is based on two factors: the threshold δ_{cv} and the diagonal length η of the cell. If the color variance is less than δ_{cv} and η is less than another user-defined threshold δ_l, we regard this node as a leaf node, and create an associated sample point at the center of the cell. The final KD-tree stores m d-dimensional samples $\{y_j\}_{j=1}^m$ of the original data set with n samples ($m \ll n$), which constructs the reduced feature space $\Re$ of the original data set.

In Fig.2, we show the clustering results with different sampling factors. The image is adaptively clustered with respect to the feature distribution. Hence, at the edge regions, more samples are placed, while at the inner regions there are less samples. The sampling factor can be changed by adjusting the stopping criteria through setting different values for the parameters δ_{cv} and δ_l. Fig.3 shows that better clustering results can be received using color variance.

B. Reduced Laplacian matrix construction

The CFM assumes that in a small window w_j of a color image, each of F and B is a linear mixture of two colors. For the m samples $\{y_j\}_{j=1}^m$ of the original data set, as

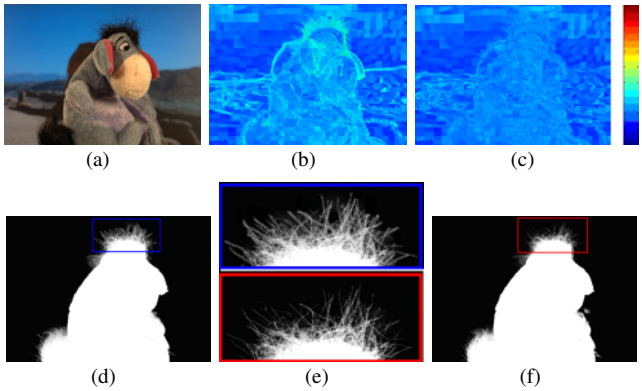


Fig. 3. Clustering using color variance. (a) Input image, (b) (c) the variance visualization before and after leaf nodes adjustment, (d) (f) matting result before and after leaf nodes adjustment, (e) is zoom in results.

we do not have regular lattice for each sample, we replace the local window by the nearest neighborhood θ_j around each sample y_j with neighborhood size θ. For each sample, we apply feature affinity-based KD-tree query to find its nearest neighborhood θ_j in the feature vector space $\Re$. The query takes a query sample y_j, a standard deviation δ around this sample, and a number $|\theta_j|$, which is the number of the samples in θ, as input, and returns a list of $|\theta_j|$ nearest points y_is with the corresponding weights w_is. $w_i = exp(-|f_i - f_j|^2/2\delta^2))$, where f_i and f_j are feature vectors of sample y_i and sample y_j. In our experiments, we set δ in the same way as σ in z_{ij}.

As the samples $\{y_j\}_{j=1}^m$ are sampled non-uniformly from the original data, the density distribution of samples in each neighborhood θ_j may vary. For the samples at the object boundaries, the samples in θ_j are densely sampled, while in the inner regions, the samples in θ_j are coarsely sampled. To make the color linear model work well in the neighborhood, we introduce the affinity weight factor $z_{ij} = exp(-\|f_i - f_j\|^2)$ into Eq.(2). Hence, we can get the following energy formula:

$$J(\alpha, a, b) = \sum_{j \in I} \left(\sum_{i \in \theta_j} z_{ji} (\alpha_i - a_j^c I_i^c - b_j)^2 + \varepsilon a_j^2 \right) \quad (5)$$

Define $J(\alpha, a, b) = min_{a,b} J(\alpha, a, b)$, and the following equation can be obtained:

$$J(\alpha) = \alpha^T L \alpha \quad (6)$$

where L is a $m \times m$ matrix, and each matrix element L_{ij} is defined as:

$$\sum_{k|(i,j)\in\theta_k} z_{ij} \left(\delta_{ij} - \frac{1}{|\theta_k|} \left((I_i - \mu_k)(\Sigma_k + \frac{z_{ij}\varepsilon}{|\theta_k|} I_3)^{-1} (I_j - \mu_k) \right) \right) \quad (7)$$

The matrix L is so-called affinity weighted matting Laplacian, where Σ_k is a 3×3 covariance matrix, μ_k is a 3×1 vector of the mean color in the neighborhood θ_k, I_3 is the 3×3 identity matrix, and $|\theta_k|$ is the number of points in this window. In the matrix L, besides imposing the local smoothness assumption in local neighborhood of the reduced feature space, we further impose that in the neighborhood, the local smoothness assumption also accounts for feature variation, which is captured by the affinities z_{ij}. Thus, the matrix L is more robust for non-uninformly sampled feature space.

From Eq.(7) we observe that the following three terms $\frac{1}{|\theta_k|}$, μ_k, $(\Sigma_k + \frac{z_{ij}\varepsilon}{|\theta_k|} I_3)^{-1}$ are independent of index i, j for computing L_{ij}. Hence, we first precompute and store these three terms for each θ_k, then construct the Laplacian matrix L, making the construction of Laplacian matrix L more efficient.

This affinity-weighted matting Laplacian can be easily extended to deal with video data. If applying the Laplacian matting naively for each frame, it may generate temporal incoherent result. To maintain temporal coherence, we treat the whole video sequence as a 3D volume. We sample m feature points $\{y_j\}_{j=1}^m$ from the original video data set. Then in each small 3D window typically 15 nearest points are sampled around each sample, similar to the image case. Assuming that within each window, each of F and B is a linear mixture of two colors, thus, the color linear model also holds in 3D window. We then build the matting cost function for the video sequence and obtain affinity-weighted Laplacian matrix similar to Eq.7, where Σ_k and μ_k are defined as the covariance matrix and mean vector of the colors in a 3D window θ_k.

C. Reduced linear system construction

Without any additional constraint, directly minimizing the energy function Eq.6 will lead to a trivial solution. To properly extract semantically meaningful foreground objects, almost all matting approaches, including the CFM [3], start by having the user to segment the input image into three regions: definite foreground R_f, definite background R_b and unknown R_u. Such inputs from user are referred to as a trimap. Instead of requiring a carefully specified trimap, our method allows the user to just specify a few foreground and background scribbles as inputs to pursue matting. These scribbles define a very coarse trimap by marking the majority of the pixels as unknowns. For video matting, we apply volume-based approaches [26] to compute video trimap.

Having the specified trimap on the input data, to perform CFM in the reduced feature space using the tripmap as the constraints, we must map the trimap into the reduced feature space. The trimap is mapped using the following steps: (1) for each scribbled pixel p with feature vector f, we search its nearest sample L_f in the reduced feature space $\Re$, (2) if pixel p is a foreground or background pixel, then L_f is set to be foreground or background sample. As the reduced feature space contains much less points than original data, the mapped foreground and background samples in the reduced space may be too coarse to provide enough training examples for effective matting. To resolve this issue, for each scribbled pixel p, we find several nearest samples in the feature space $\Re$ as the corresponding set of samples.

After constructing the reduced feature space $\Re$ and project the trimap into it, the matting problem is thus

reduced to estimate F, B and α for pixels within the unknown region in the reduced feature space, based on the mapped foreground and background samples. We provide user-supplied constraints on Eq.6 to extract the alpha matte. We set $\alpha = 1$ for user specified foreground points, and $\alpha = 0$ for user specified background points. We minimize the following energy function to extract the alpha matte, matching the user's constraints in the reduced feature space:

$$J^*(\alpha, a, b) = J(\alpha, a, b) + \lambda_1 \sum_{\theta_k} \sum_{(i,j)\in\theta_k} z_{ij}(\alpha_i - \alpha_j)^2 + \lambda_2(\alpha^T - b_S^T)D_S(\alpha - b_s) \tag{8}$$

This energy function consists of three terms. The first term $J(\alpha, a, b)$ is a cost function. The second term enforces that the matte values α are similar for regions with similar appearances f, which is captured by the affinity z_{ij}. The second term also helps to avoid the incorrect mattes diffusion, especially in the edges regions, since the affinity z_{ij} accounts for both the position and color of the sampled points. The third one is responsible for satisfying the user-specified constraints, where D_s is a diagonal matrix whose diagonal elements are 1 for constrained pixels and 0 for all other pixels, and b_s is the vector containing the specified alpha values for the constrained pixels, and $b_s = 1$ for foreground pixels and $b_s = 0$ for background pixels. The relative contributions of these three terms are controlled by λ_1 and λ_2. Based on our experiments, $\lambda_1 = 0.5$ and $\lambda_2 = 3$ are optimal in usual cases.

By eliminating a and b from the first term, to extract an alpha matte matching the user's constraints, we solve for

$$\alpha = \operatorname{argmin} \alpha^T L \alpha + \lambda_1 \sum_{\theta_k} \sum_{(i,j)\in\theta_k} z_{ij}(\alpha_i - \alpha_j)^2 + \lambda_2(\alpha^T - b_S^T)D_S(\alpha - b_S) \tag{9}$$

Since the above cost function is quadratic in alpha α, the global minimum can be found by differentiating it and setting the derivatives to be zero. This amounts to solving the following sparse linear system:

$$(L + 2\lambda_1 D - 2\lambda_1 Z + \lambda_2 D_S)\alpha = \lambda_2 b_S \tag{10}$$

where L is the affinity-weighted Laplacian matrix (Eq.7), Z is the affinity matrix whose elements are z_{ij}, and D is a diagonal matrix with $D_{ij} = \sum_j z_{ij}$. As we construct the energy function in the reduced feature space, the matrix L, Z, D and D_s are much smaller than the matrix built upon original data set. Hence, the computational complexity is greatly reduced by solving a much smaller linear system, which can be solved by various linear solvers, such as conjugate gradient descent method [28].

D. Alpha Interpolation

The popular interpolation methods, such as Gaussian or Bicubic interpolation, typically suffer from blurring the sharp alpha edges. To achieve better quality, we develop a new multi-scale detail-aware alpha interpolation method. Let I be the input image, using an edge preserving filter [36], [37], we compute M progressively coarser version $I_1, ..., I_M$ of I, which suppress the noises while preserving the strong features in all the subsequent decomposition iterations. We then compute M ($M = 3$ in our experiments) subsequent image detail levels:$\{D_j\}_{j=1}^{M}$, where $D_j = I_j - I_{j-1}$. The final matting result for original image I is restored by upsampling the alpha solution α^* computed in the reduced feature space using the detail-guided alpha interpolation.

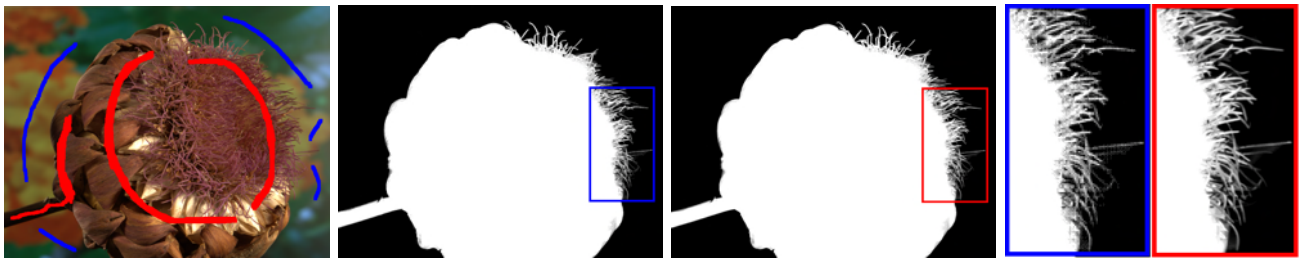
The method works as follows: (1) For each pixel p in I with feature vector f, we search a set of its nearest samples $N^*(p)$ in feature space, and compute the affinity weight $w_i = e^{-|f-f_i|^2/2\sigma}$ between each sample p_i^* and p. Then, the weighted details sum of p is computed as: $\tilde{D} = \sum_{j=1}^{M} \phi_j D_j / \sum_{j=1}^{M} D_j$, where the weight $\phi_j = g_\sigma * (j \cdot e^{-\frac{1}{n}\sum |\nabla I_j|})$, n is the number of the image pixels, and the Gaussian convolution g_σ is used to locally smooth the weight. This weighted details sum $\tilde{D}$ prefers the coarser detail levels by assigning them larger weights, since the noises are smoothed out and also the prominent edges are better preserved in the coarser levels. Hence, the alpha edge information is recovered by suppressing the noises in the final matte value. We then define the weight $\omega_i = e^{-|\tilde{D}-\tilde{D}_i|^2/2\sigma}$, where $\tilde{D}_i$ is the enhanced detail of a pixel in I which corresponds to p_i^* in the reduced feature space. (2) The final alpha value for pixel p is computed as a weighted average of the alpha values α_is of all the nearest samples in $N^*(p)$: $\alpha_p = \sum_{i\in N^*(p)} \alpha_i w_i \omega_i / \sum_i w_i \omega_i$. The interpolation artifacts such as alpha edges blurring, alpha noises, etc, can be efficiently alleviated.

Fig.4 shows the results using two different interpolations, one is the Gaussian interpolation, and the other is our detail-aware alpha interpolation. Clearly, our method produces smoother and more accurate results. Not only the interpolation blurring artifact is attenuated, but also the fine structures and features are better enhanced.

E. Incremental refinement

Current matting approaches can achieve good results when the foreground and background are smooth and well-separable in the color space. However, if the foreground and/or background contain (s) highly-textured regions with complex color patterns, existing approaches tend to generate noisy results with noticeable artifacts. This is a difficult open problem [1] for matting. Levin et al. [3] applied incremental refinement technique by adding additional scribbles in the regions containing mixed pixels to address this issue. Our method further improves this incremental refinement scheme [3] using our hierarchical data structure. When there are some regions where the matting results are unsatisfactory, the user can add new background or foreground scribbles in these regions. Instead of re-computing the CFM for every unknown pixel with an unreliable initial value, we use the previously computed alpha matte as the initial value to solve the Eq.10, which is much faster and generate more accurate results in a progressive style.

Fig.5 shows the incremental matting refinement process.



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

Fig. 4. (a) Input image with coarse trimap, matting results using (b) Gaussian and (c) detail-aware interpolation, (d) the close-ups in results.

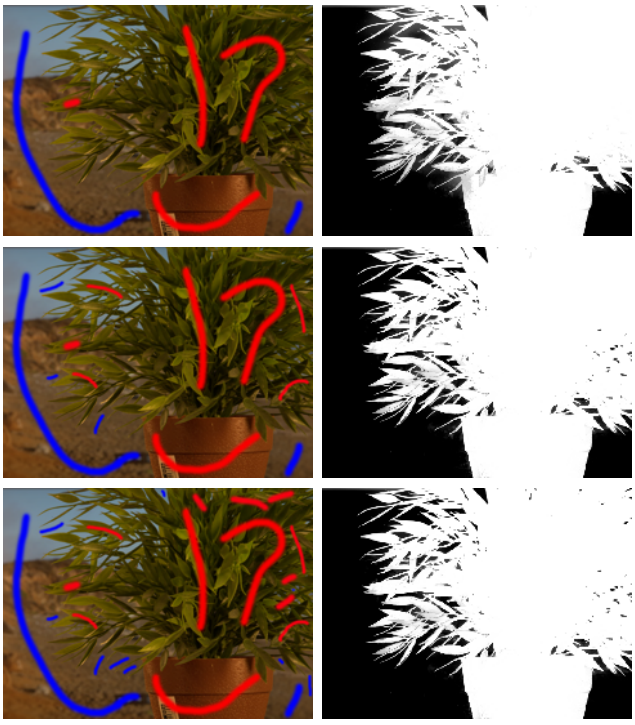


Fig. 5. Incremental refinement results including the incremental trimaps (left) and the incremental matting results (right).

First, the user scribbles a small subset of pixels in the fuzzy regions or regions with incorrect matting values needed to be re-estimated, and sets constrained value for these pixels. Then, we map the constrained values of these pixels to the samples in the reduced feature space, and update the values of D_s and b_s in Eq.10. We use the alpha value computed in previous computation as the initial value to resolve the Eq.10, with the new constrained values D_s and b_s, we achieve better matting results.

Our incremental method is fast and accurate due to the following reasons:

- KD-tree partition and Laplacian matrix reconstruction only need to be performed once and can been reused in following incremental refinement.
- The Eq.10 is solved in a reduced feature space.
- We use the previously computed α value as the initial value for solving Eq.10, which makes the solution converge much faster, and produce more accurate result.

Fig.5 presents an incremental refinement example. For an image of size 800×595, the initial solution of Eq.10 takes 5.864 seconds (s), and the number of iterations is 96. The first and the second incremental refinement take $1.742s$ and $1.103s$, respectively, and the corresponding number of iterations is 64 and 43. During the iterative solving procedure of Eq.10, we stop when the error of consequent two solutions is below 1×10^{-15}. As shown in the Fig.5, the matting results are improved significantly after refinement.

F. Complexity analysis

Using the hierarchical data structure, the time complexity of matting is greatly reduced. As observed in Fig. 2, large areas of the image are grouped into clusters, indicating that much fewer clusters are used than the number of actual pixels. The number of clusters depends on several factors including the number of user scribbled pixels, the distribution of the pixels in the affinity space, and the depth of the KD-tree. In general, more clusters are generated if the number of user-edited pixels increases or the pixels appearance is more non-uniformly distributed in the affinity space. If both the image and the user-strokes are fixed, the cluster number grows with the depth of the KD-tree, which is effectively governed by the size of the smallest leaf cell and the size of the root cell that represents the bounding box of all pixels.

Using the KD-tree, we construct a reduced feature space for n input d-dimensional data points with m feature vectors $\{x_i\}_{i=1}^{n} \rightarrow \{y_j\}_{j=1}^{m}$, and $m \ll n$. Assuming the depth of KD-tree is $O(\log(m))$, the complexity of tree construction is $O(nd\log(m))$. Searching nearest neighborhood with s points into the tree for each of the n input points takes $O(ns(\log(m)+d))$ time. To solve the reduced linear system (Eq.10), using the conjugate gradient, the time complexity is $O(m^{1.5})$. Hence, if performing τ conjugate gradient iterations, computing the linear system in the reduced space with m feature vectors takes $O(\tau m^{1.5})$ time. In the last stage, we up-sample the clustering results to the original data sets takes $O(ns(\log(m)+d))$ time (for simplicity, we assume each of the n pixels find s nearest points in the reduced feature space for interpolation), and the detail-aware alpha interpolation takes $O(ns^2)$ time.

With above discussions, the total complexity is $O(n((s+d)\log(m)+sd+s^2)+\tau m^{1.5})$. Recall that $m \ll n$, and s is a sampling constant, which is usually set as: $6 \leq s \leq 16$, this

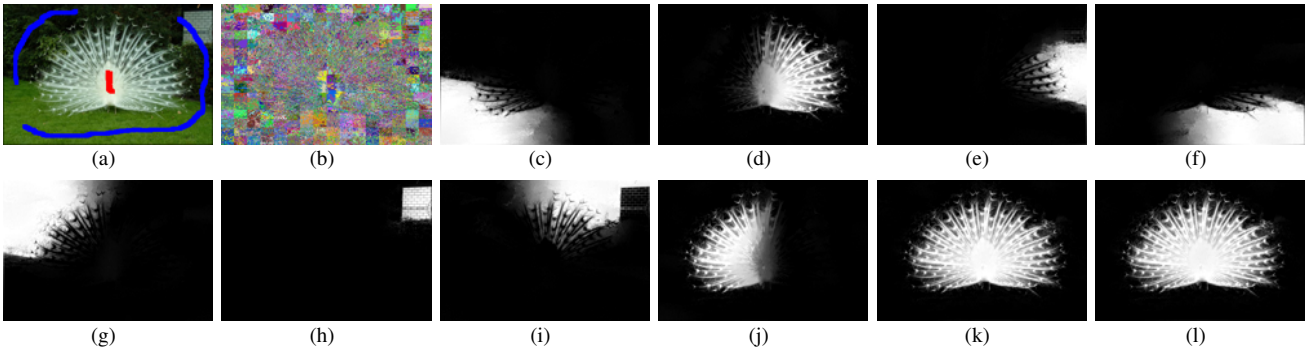


Fig. 6. Accelerating Spectral matting [5]. (a) Input image with coarse trimap, (b) Gaussian Kd-tree partition, (c)-(j) eight matting components, (k) the final matte, (l) the result of [5].

results in a total complexity $O(n(s+d)\log(m)+\tau m^{1.5})$. Compared with solving linear system for the original n pixels with complexity $O(\tau n^{1.5})$, our method is significantly faster. Further, the memory consumption is greatly reduced due to construct a much smaller matrix. The much less time and memory consumption enable our method to efficiently process high-resolution image and video data.

V. GPU ACCELERATION

All the algorithmic steps of our method, including partitioning affinity-based KD-tree, preprocessing matrix element, constructing Laplacian matrix, solving the linear system and interpolating matte values, can be fully parallelized on GPU. Applying the existing techniques from [31], [35], the affinity-based KD-tree can be efficiently built on GPU for the sampling points with high-dimensional feature vector. Once the tree is built, the steps of preprocessing matrix element and interpolating matte values are inherently parallel, since the high-dimensional nearest neighbor search can be fully implemented in parallel using GPU [11]. Hence, these two steps are efficient for handling large data set with high-dimensional feature vectors.

When preprocessing the matrix element, for each sample, we need to compute three terms: $\frac{1}{|\theta_k|}$, μ_k, and $(\Sigma_k+\frac{z_{ij}\varepsilon}{|\theta_k|}I_3)^{-1}$. As every term can be computed independently for each sample, these steps can be processed in parallel. Using GPU-based nearest neighborhood queries, computing these terms can be further accelerated. When constructing the Laplacian matrix L, we treat the construction of each row of elements as an independent thread block and perform searching in parallel (i.e. the task of computing each row of elements can be processed using one thread block, and using all threads in the thread block to complete the search processing). We apply the conjugate gradient method to solve the linear system (Eq.10), which is accelerated by the GPU-based CUBLAS library [38] from NVIDIA®. Interpolating matte values is achieved by computing the weighted average of the nearest alpha values in the reduced feature space for each pixel from original image. Performing the nearest neighborhood queries and matte interpolation for each pixel in parallel on GPU, this step can also achieve real-time performance.

Our method is implemented in CUDA, and realizes a typical speedup of 20x over the single-threaded CPU implementation, which allows our method to be a real-time image matting framework for moderate-sized images, and applicable to perform video matting with moderate size in an acceptable speed. For example, our video matting method costs about 8 seconds for a video with size $632\times 268\times 76$.

VI. ACCELERATING AFFINITY-BASED MATTING METHODS

Using the same acceleration framework, our hierarchical strategy can be extended to many other affinity-based matting approaches [4], [5], [6], [12], [8], [9]. Similarly, we first adaptively sample the data set, and then define the energy functions on the samples. By using the presented hierarchical strategy, the size of data set is greatly reduced, and the computational complexity of optimizing the energy function is reduced accordingly. In this section, to show the generality of our method, we extend the present acceleration framework to spectral matting [5] and robust matting [9]. Although the underlying energy functions of other affinity-based methods [4], [6], [8], [12] are quite different, these approaches can share the same optimization framework with CFM, and hence can be accelerated in the similar way, with only minor modifications.

Fast spectral matting: Spectral matting [5] is a matting approach that tries to pull out a foreground matte in an automatic fashion. In this approach, the input image is modeled as a convex combination of K image layers as $I_z=\sum_{k=1}^{K}a_z^kF_z^k$, where F_z^k is the kth matting component of the image. The most important conclusion from this approach is that the smallest eigenvectors of the matting Laplacian matrix L [3] span the individual matting components of the image. Thus, recovering the matting components of the image is equivalent to finding a linear transformation of the eigenvectors.

The spectral matting approach is inspired by spectral image hard segmentation. However, the matting Laplacian L is constructed in the same way as CFM, thus it is very slow to compute the smallest eigenvectors L of for a high-resolution image. Computing an optimal linear transformation of the eigenvectors to find the final

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

Fig. 7. Accelerating robust matting method [9]. (a) Input image with coarse trimap, (b) the result of [9], (c) our result, (d) the ground truth matting result.

matting components of the image hence becomes time-consuming. Using our matting acceleration framework, by constructing the Gaussian KD-tree for the input image, and computing and transforming the smallest eigenvectors in a low-dimensional feature space, the matting computation becomes much faster with much less memory consumption. As illustrated in Fig.6, with the same user interactions, we receive nearly the same matting quality compared with the standard spectral matting. The standard method takes 185.21 seconds to complete the matting extraction, while our accelerated version only costs 8.45 seconds.

Fast robust matting: The robust matting method [9] combines the advantages of both sampling-based and affinity-based matting, and is a well-balanced approach that can generate high quality mattes while maintaining reasonable robustness against different user inputs. Specifying a trimap for the underlying image, this method minimizes the following energy function:

$$E = \sum_{z \in w} [\hat{f}_z(\alpha_z - \hat{\alpha}_z)^2 + (1 - \hat{f}_z)(\alpha_z - \delta(\hat{\alpha_z} > 0.5))^2] + \lambda \cdot J(\alpha, a, b) \quad (11)$$

where $\hat{\alpha_z}$ and $\hat{f}_z$ are, respectively, the estimated alpha and confidence values in the color sampling step using trimap, and $J(\alpha, a, b)$ is the neighborhood energy defined already in CFM. The energy function can be minimized as a quadratic function of α, which can be solved by solving a linear system $LA = B$. The matrix L is a sparse, positive-definite Laplacian matrix.

Robust matting [9] can also be accelerated using our framework. After specifying a coarse trimap for the image, we subsample the image data, and generate a reduced feature space. Then we estimate the $\hat{\alpha}_z$ and $\hat{f}_z$ in the reduced feature space, and solve a much simpler linear system. Using the detail-aware interpolation, we receive the final mattes. As illustrated in Fig.7, with the same coarse trimap, the accelerated robust matting achieves very similar matte quality as the standard robust matting [9]. For an image of size 800×618, it takes the standard method 35 seconds to extract the matte, while our method only spends 5 seconds. Further, the memory cost is reduced from $0.085G$ to $0.012G$.

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

Fig. 8. Image dehazing acceleration: (a) Input haze image, and image dehazing results of [39] (b), [13] (c), and our method (d).

VII. Applications

Our method can be further applied to accelerate the image and video dehazing, and image shadow detection and removal.

Image and video dehazing: In computer vision and computer graphics, the formation of a haze image is usually defined as follows: $I(x) = J(x)t(x) + A(1 - t(x))$ [13], where I is the observed intensity, J is the scene radiance, A is the global atmospheric light, and t is the medium transmission. The goal of haze removal is to recover J, A and t from I. He et al. [13] proposed the dark channel prior (DCP), a kind of statistics of the haze-free outdoor images, to directly estimate the thickness $t(x)$ of the haze and recover a high quality haze-free image. The DCP method estimates the transmission map $t(x)$ from an input haze image using the patch (with size 15×15). Its dehazing quality is roughly good but contains some blocky effects since the transmission is not always constant within a patch. Hence, the DCP method refines the transmission map using a similar Laplacian matrix based interpolation as CFM [3], which is also time-consuming for high-resolution image.

We apply the presented method to accelerate the refinement of the transmission map. We first compute the rough transmission map $t(x)$, and cluster both the input image $I(x)$ and $t(x)$ using the Gaussian KD-tree to produce the reduced feature space $\widetilde{I}(x)$ and map $\widetilde{t}(x)$. We then compute the Laplacian matrix L and haze thickness $\hat{t}(x)$ in $\widetilde{I}(x)$. Finally, by using affinity-based interpolation, we interpolate

Fig. 9. Video dehazing acceleration. Top row, the input video, bottom row, dehazed video. From left to right, the 3th, 45th and 80th frame.

$\hat{t}(x)$ to get the final smooth transmission map.

We further employ our method for fast video transmission map interpolation to accelerate video dehazing. To maintain temporal coherence, we estimate transmission map $\widetilde{T}(x,t)$ from an input haze video using a spatio-temporal patch (e.g. $15 \times 15 \times 12$). Similarly, the generated spatio-temporal transmission map $\widetilde{T}(x,t)$ also contains some blocky effects since the transmission is usually nonuniform within a patch. Similar to the image case, we compute the transmission map in the reduced feature space by solving a much simpler sparse linear system. By performing affinity-based interpolation, the temporally coherent transmission map for the input video is generated. The final dehazing video can be recovered from the refined transmission map.

Fig.8 shows the dehazing results using our method, which only exhibit minor visual differences with [13]. The computation time of the standard DCP method [13] is $18.905s$ for this image with size 441×450, while our method takes $2.223s$. We also compare our method with Tarel et al. [39], which is a fast dehazing algorithm using median filter. It take [39] $1.623s$ to finish the dehazing process. The speed of our method is comparable with Tarel et al. [39], while our quality is better.

Fig.9 shows the video dehazing results using our method. For input video with size $512 \times 280 \times 82$, it is difficult to directly apply the CFM for interpolating transmission map due to the large memory requirement. Using our acceleration strategy, it only takes about $0.25G$ memory and $8.5s$ on GPU to complete the dehazing process. The video dehazing results are satisfactory, and the temporally incoherent artifacts are avoided effectively.

Shadow detection and removal: Image shadow detection and removal are important research topics in computer vision. The shadow in an image can described with the following formula: $I_t = (t_i cos\varphi_i L_d + L_a)R_i$, where I_i is the intensity value of ith pixel, L_d is the direction illumination, L_a is the ambient illumination, R_i is the surface reflectance of ith pixel, φ_i is the angle between direction illumination and the surface norm, and t_i $(0 < t_i < 1)$ is a value determining how much direct illumination gets to current pixel, which is referred to as shadow coefficient.

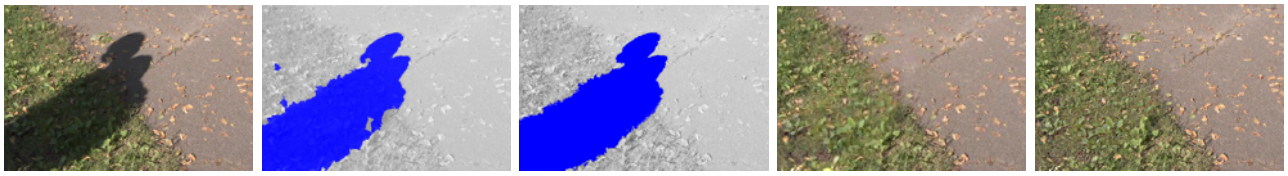
The CFM [3] can be used to improve shadow detection and removal [40], [41]. With the detected coarse shadows, we treat shadowed pixels as foreground and lit pixels as background. Similar as the transmission map interpolation used in image dehazing, the CFM can be applied for interpolation to obtain more smooth shadow coefficients t_i, which can used for further accurate shadow detection. Based on the shadow coefficients, we calculate the ratio between direct light and environment light [40], and then perform shadow removal by relighting each pixel. Obviously, we can also apply the accelerated closed-form interpolation method to efficiently estimate the shadow coefficients t_i. As shown in Fig. 10, in comparison with [40] on both shadow detection and removal, our results achieve very competitive quality with much less time and memory consumption. The time for interpolation is reduced from about 8 seconds to 0.56s for the shadowed regions in a image with size of 523×348.

VIII. Results and discussions

Performance data of all the results is reported on a PC with Intel® Pentium Dual-Core CPU E5200@2.50 GHz and 2 GB memory, and a NVIDIA® GeForce GTX 285 (1 GB) graphics card. Our method is implemented using CUDA 4.1. In this section, we present various experimental results, and the comparisons with the state-of-art methods [22], [18], [3], [9], for the matting quality, time complexity and memory requirement. For a fair comparison, all these four methods are implemented in the same environment as ours. The limitations of our method are also discussed.

Fig.2 gives the matting results generated applying our fast CFM method with different sampling factors. As shown in Fig.2, for an image with size 800×600, even with very high sampling factor, such as n/m = 300, the matting result is still pleasing. However, when using too high sampling factor such as n/m = 2800, the matte values in the complex regions, like hair, can not be effectively extracted. In Fig.2, for each sampling factor, we also give the corresponding time and memory consumption, and also the errors from the ground truth. Obviously, with gradually higher sampling factor n/m, the matting speed is faster and the memory consumption is less, while the error is increasing. We apply the perceptually motivated gradient-based MSE error function [10] to measure the errors from the ground truth.

Fig.3 gives the clustering result comparison between using and without using color variance. We map the color variance value into the color range between blue (minimum) and red (maximum). The pixels in each leaf node are represented using the color variance of the cell. As shown in Fig.3 (c), we get leaf nodes with lower color variance compared with adjustment, which better satisfy the local smoothness assumption. As shown in Fig.3 (d), (e) and (f),



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

Fig. 10. (a) Input image, (b) shadow detection result of [40] (result from manuscript [40]), (c) our shadow detection result, (d) shadow removal result of [40] (results from manuscript [40]), (e) our shadow removal result.

TABLE I
TIME-COMPLEXITY AND MEMORY REQUIREMENT COMPARISON WITH METHODS [22], [18], [3] AND [9].

Image	Size	Time-complexity (seconds)							Memory(G)				
		[3]	[9]	[18]	[22]	n/m	Our(CPU)	Our(GPU)	[3]	[9]	[18]	[22]	Our (CPU) (GPU)
Donkey(trimap)	800×569	116.46	13.72	9.016	16.26	22	4.71	0.42	0.31	0.07	0.12	0.071	0.054 (0.015) (0.039)
Pineapple(trimap)	800×602	138.42	15.98	11.89	24.65	25	4.82	0.45	0.32	0.05	0.13	0.058	0.055 (0.013) (0.042)
Pineapple(stroke)	800×602	140.68	28.39	14.71	86.71	20	5.03	0.48	0.346	0.085	0.12	0.058	0.061 (0.014) (0.047)
Peacock(stroke)	600×400	61.91	16.64	8.234	46.39	15	3.85	0.33	0.129	0.034	0.08	0.068	0.021 (0.009) (0.012)

the accelerated matting computed in the reduced feature space incorporating color variance produces more accurate result, and the fine structures are better extracted.

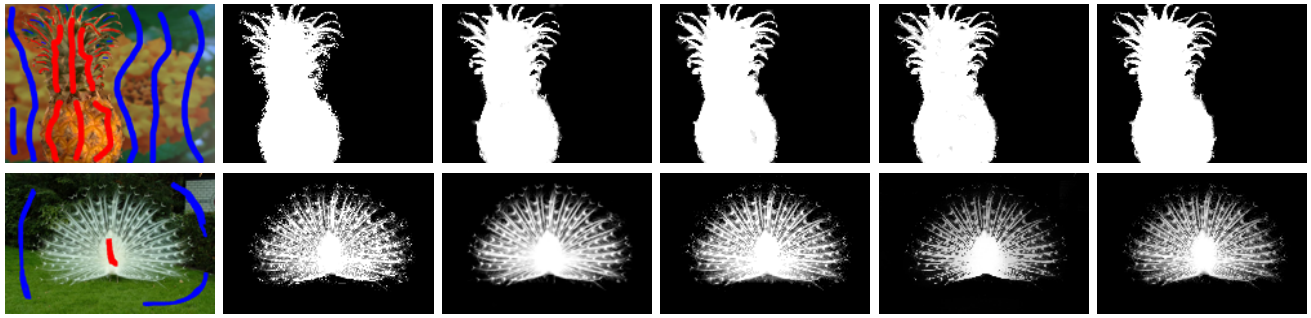
In Fig.11, Fig.12, Table I and Table II, our method is compared with the state-of-the-art methods: CFM [3], Robust matting [9], Shared matting [22] and Large kernel matting [18]. We perform CFM [3], Shared matting [22] and Large kernel matting [18] in C++, and perform Robust matting [9] using the code presented by the authors. For a fair comparison, we set the parameters for all methods using the values reported in the original papers. The comparison is performed for two cases: the input image with an accurate trimap, and with a coarse trimap. As illustrated in Fig.12, for accurate trimap, our results are comparable with [3], [18] and [9], with much faster speed. Although [22] can be performed on GPU and obtain real-time matting results, with CPU implementation this method is still slower than our method on CPU and our quality is also much better. For image with size 800×602, as illustrated in Table I, [3], [18], [9] and [22] and our method cost $138.42s$, $11.89s$, $15.98s$, $24.65s$, $4.82s$ respectively. For the 4.82s used in our method, the Gaussian KD-tree sampling takes 2.21s, affinity-weighted Laplacian matrix construction takes 1.80s, and alpha interpolation take 0.71s. Fig.11 shows the comparisons with coarse trimap. Our matting quality is comparable with [3] and much better than [18], [9] and [22]. The results demonstrate that [18], [9] and [22] require accurate (tight) trimap to produce satisfactory results, while our method and [3] can generate good results with only coarse trimap. Moreover, with coarse trimap, our method shows much more advantage in speed. For the same image, [3], [18], [9] and [22] and our method cost $140.68s$, $14.78s$, $28.39s$, $86.71s$, $5.03s$, respectively. Note the speed of [9] depends on the number of the pixels of the unknown regions, and hence larger unknown regions means higher time complexity. Further, coarse trimap is important for image and video matting, as for complex cases, such as the peacock image in Fig.11 and the video input in Fig.14), it is tedious to specify an accurate trimap.

In Table II, we further pursue an quantitative analysis to compare the matting quality between our method and all these state-of-the-art methods [3], [9], [18], [22]. We tested 27 training images for evaluation, using both accurate trimaps and coarse trimaps (scribbles) as inputs. The training images and the corresponding ground truth mattes are freely accessible at http://www.alphamatting.com. Inspired by Rhemann et al. [10], we compute the accuracy of the resulting alpha matte with respect to five error measures via the comparison with the ground truth, i.e. SAD (sum of absolute differences), MSE (mean squared error), SD (standard deviation), Gra.(gradient error) and Con.(connectivity error). In the experiments, for each training image with corresponding trimap, we give a ranking of all algorithms. The final rank in the table is averaged over all training images. As illustrated in Table II, when using accurate trimaps, we observe that, CFM [3] and Shared matting [22] outperform our method, and our method surpasses Large kernel matting [18] and Robust matting [9]. When dealing with coarse trimaps, we observe that, only CFM [3] outperforms our method in quality, while our method outperforms all the other methods [9], [18], [22]. The quantitative analysis of matting quality confirm that our method works well for coarse trimaps, which is important for processing foreground with complex boundaries in both image and video data. In Fig.13, we compare the results

TABLE II
QUANTITATIVE COMPARISONS WITH [3], [18], [9] AND [22].

Methods	Coarse trimaps (Scribbles)					Accurate trimaps				
	SAD	MSE	SD	Gra.	Con.	SAD	MSE	SD	Gra.	Con.
[3]	**1.3**	**1.5**	**1.4**	**1.5**	**1.8**	1.7	1.6	1.5	1.6	1.8
Ours	2.2	1.9	1.8	2.6	2.8	3.5	3.7	3.7	4.2	3.6
[18]	2.5	2.6	2.8	2.3	3.3	4.1	4.2	4.2	4.4	4.2
[9]	4.4	4.5	4.5	4.2	3.7	4.3	4.0	4.1	3.5	4.1
[22]	4.7	4.8	4.9	5.0	4.4	**1.3**	**1.4**	**1.3**	**1.3**	**1.2**

using multi-resolution solver presented in [3]. Levin et al. [3] presented a simple multi-resolution solver for CFM in their implementation. As shown in Fig.13, using the default level (which is set to 1) in the authors' code, this method receives pleasing result (Fig.13(b)). When the level is set

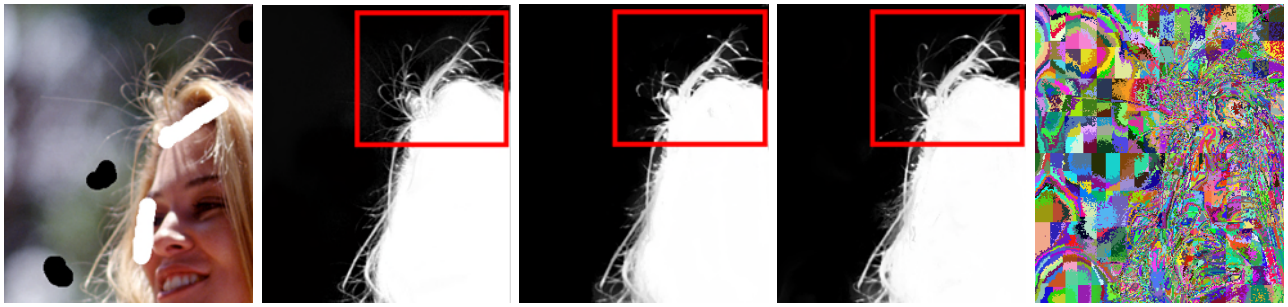


(a) (b) (c) (d) (e) (f)

Fig. 11. Matting results comparison based on coarse trimap specified by a few foreground and background scribbles. (a) Input image, (b) result of [22], (c) result of [3], (d) result of [18], (e) result of [9], (f) our result.

(a) (b) (c) (d) (e) (f)

Fig. 12. Matting results comparison based on accurate trimap.(a) Input image, (b) result of [22], (c) result of [3], (d) result of [18], (e) result of [9], (f) our result.



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

Fig. 13. Limitation: (a) Original image with trimap, (b) result of multi-resolution solver [3] using 1 levels, (c) result of multi-resolution solver [3] using 3 levels, (d) our result using sampling factor n/m=200, (e) the generated clusters in our method with sampling factor n/m=200.

to 3 (Fig.13(c)), the fine features such as the hair are not effectively extracted. Using our method, even when we set sampling factor n/m =200, the result (Fig.13(d)) is still much better than Fig.13(c).

Fig.14 presents the video matting results using our method. It is usually time and memory prohibitive to directly apply CFM to process video matting. In comparison, our fast CFM method can process video matting efficiently. As shown in Fig.14, by sampling the video data into reduced feature space, and performing video matting incorporating both the spatial and temporal information, we produce temporally coherent video matting results. Note that, for input video whose foreground and/or background contain(s) highly-textured regions with complex color patterns, we apply the presented incremental refinement techniques to produce more pleasing results, which incrementally improve the matting quality.

As our method can be performed in parallel, we process video matting on GPU. For the Maria video in Fig.14 with size of $400 \times 300 \times 77$, our method takes $1.105s$ for KD-tree construction, $6.629s$ for matrix solution, and $2.251s$ for matting alpha interpolation, and in total around $10s$ finish matting. In comparison, applying Bayesian video matting [23], [24], which is a dominant video matting method, to finish the matting of the same video costs more than one hour on CPU. Moreover, considering the matte quality, our method produces visually comparable results as [20]. These experiments demonstrate that our system is computationally efficient to extract dynamic matting values from videos with moderate size.

Limitations: Although our hierarchical subsampling method captures the image features and produces repre-

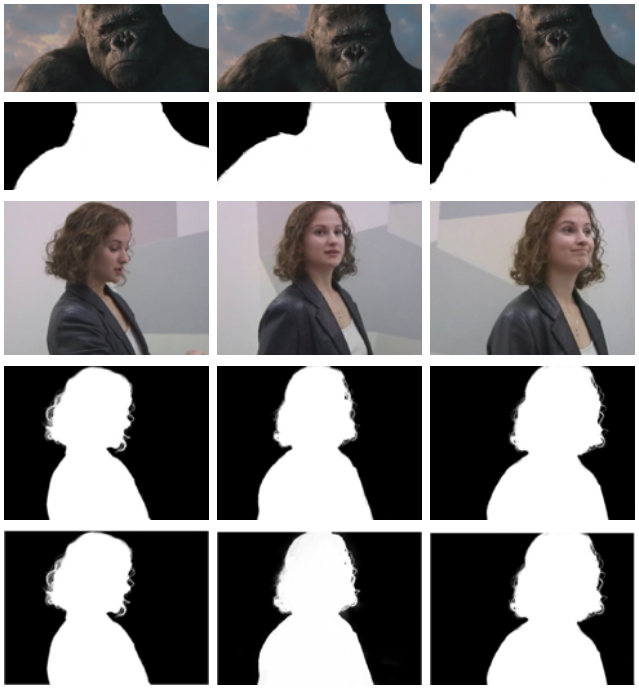


Fig. 14. Video matting acceleration. Top row, the input video with size $632 \times 268 \times 76$; the second row, our matting acceleration results, from left to right is the 2th, 50th and 63th frame; the third row, the input video with size $400 \times 300 \times 77$; the fourth row, video matting results of [23]; bottom row, our matting results. From the left to right is the 12th, 25th and 76th frame.

sentative samples in the feature space, when the samples are extremely sparse, our method still fails to extract the matting value in the complex regions, or from the fine and thin features, such as the hair in Fig.13. To address this issue, one possible solution is to put more user interactions on these regions, and another strategy is to increase the sampling ratio. However, the latter strategy may increase the time complexity and require relatively more memory for the KD-tree construction. We have to leverage between matting quality and the time-memory consumption. Compared with the Bayesian video matting [23] that process video matting frame by frame, our method is inconvenient to process very long video sequence. One potential solution is to divide the input video into several subsequences, while making some overlap between the adjacent parts. Finally the matte value for each part can be extracted using the matte value on the overlap regions as the constrained value.

IX. Conclusion and Future Work

In this paper, we present a unified acceleration method for closed-from image and video matting using hierarchical data structure. By downsampling the large data set adaptively into much smaller feature space, and solve the matting optimization on such a reduced space, we greatly reduce the time and memory consumption, while maintaining the matting quality. Another advantage of our method is that it can be further accelerated by exploiting the evolving graphics hardware. We also extend our acceleration strategies to other affinity-based matting methods and demonstrate that it can be considered as a general acceleration framework for a variety of matting methods.

In the future, we plan to work on automatically determining the optimal sampling factors n/m to achieve better balance between quality and performance for different inputs. In most experiments, to compromise between quality and performance, we set the sampling factor n/m between 50 and 80 and receive satisfactory results. Defining guidelines to automatically choose optimal n/m is an important and interesting future research direction. Extending our method to sample-based matting methods is also an interesting future research topic.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and insightful suggestions. This work was partly supported by the National Basic Research Program of China (No. 2012CB725303), the NSFC (Nos. 61070081 and 41271431), the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1208), and the Luojia Outstanding Young Scholar Program of Wuhan University.

References

[1] J. Wang and M. Cohen, "Image and video matting: a survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 3, no. 2, pp. 97–175, 2007.

[2] D. Singaraju and R. Vidal, "Estimation of alpha mattes for multiple image layers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 7, pp. 1295–1309, 2011.

[3] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 228–242, 2008.

[4] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," in *Proc. of VIIP*, vol. 2005, 2005, pp. 423–429.

[5] A. Levin, A. Rav-Acha, and D. Lischinski, "Spectral matting," in *Proc. of CVPR*. IEEE, 2007, pp. 1–8.

[6] Y. Zheng, C. Kambhamettu, J. Yu, T. Bauer, and K. Steiner, "Fuzzymatte: A computationally efficient scheme for interactive matting," in *Proc. of CVPR*. IEEE, 2008, pp. 1–8.

[7] Y. Guan, W. Chen, X. Liang, Z. Ding, and Q. Peng, "Easy matting-a stroke based approach for continuous image matting," in *Computer Graphics Forum*, vol. 25, no. 3, 2006, pp. 567–576.

[8] J. Wang and M. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *Proc. of ICCV*, vol. 2. IEEE, 2005, pp. 936–943.

[9] J. Wang and M. F. Cohen, "Optimized color sampling for robust matting," in *Proc. of CVPR*. IEEE, 2007, pp. 1–8.

[10] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *Proc. of CVPR*. IEEE, 2009, pp. 1826–1833.

[11] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian kd-trees for fast high-dimensional filtering," in *ACM Trans. Graph.*, vol. 28, no. 3, 2009, p. 21.

[12] X. Bai and G. Sapiro, "Geodesic matting: A framework for fast interactive image and video segmentation and matting," *International journal of computer vision*, vol. 82, no. 2, pp. 113–132, 2009.

[13] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *Proc. of CVPR*. IEEE, 2009, pp. 1956–1963.

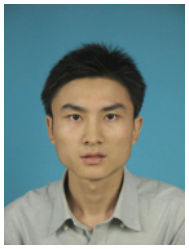
[14] D. Xu, J. Liu, X. Li, Z. Liu, and X. Tang, "Insignificant shadow detection for video segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 1058–1064, 2005.

[15] T. Shih, N. Tang, and J. Hwang, "Exemplar-based video inpainting without ghost shadow artifacts by maintaining temporal continuity," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 347–360, 2009.

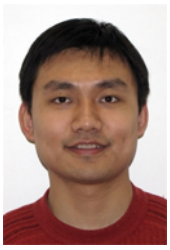
[16] J. Sun, J. Jia, C. Tang, and H. Shum, "Poisson matting," in *ACM Trans. Graph.*, vol. 23, no. 3, 2004, pp. 315–321.
[17] D. Singaraju, C. Rother, and C. Rhemann, "New appearance models for natural image matting," in *Proc. of CVPR*. IEEE, 2009, pp. 659–666.
[18] K. He, J. Sun, and X. Tang, "Fast matting using large kernel matting laplacian matrices," in *Proc. of CVPR*. IEEE, 2010, pp. 2165–2172.
[19] M. Ruzon and C. Tomasi, "Alpha estimation in natural images," in *Proc. of CVPR*, vol. 1. IEEE, 2000, pp. 18–25.
[20] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski, "A bayesian approach to digital matting," in *Proc. of CVPR*, vol. 2. IEEE, 2001, pp. II–264.
[21] C. Rhemann, C. Rother, and M. Gelautz, "Improving color modeling for alpha matting," in *Proc. of BMVC*, 2008.
[22] E. Gastal and M. Oliveira, "Shared sampling for real-time alpha matting," in *Computer Graphics Forum*, vol. 29, no. 2, 2010, pp. 575–584.
[23] Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski, "Video matting of complex scenes," in *ACM Trans. Graph.*, vol. 21, no. 3, 2002, pp. 243–248.
[24] N. Apostoloff and A. Fitzgibbon, "Bayesian video matting using learnt image priors," in *Proc. of CVPR*, vol. 1. IEEE, 2004, pp. I–407.
[25] Y. Li, J. Sun, and H. Shum, "Video object cut and paste," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 595–600, 2005.
[26] J. Wang, P. Bhat, R. Colburn, M. Agrawala, and M. Cohen, "Interactive video cutout," in *ACM Trans. Graph.*, vol. 24, no. 3. ACM, 2005, pp. 585–594.
[27] R. Szeliski, "Locally adapted hierarchical basis preconditioning," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 1135–1143.
[28] Y. Saad and Y. Saad, *Iterative methods for sparse linear systems*. PWS publishing company Boston, 1996, vol. 20.
[29] M. Huang, F. Liu, and E. Wu, "A gpu-based matting laplacian solver for high resolution image matting," *The Visual Computer*, vol. 26, no. 6, pp. 943–950, 2010.
[30] D. Horn, J. Sugerman, M. Houston, and P. Hanrahan, "Interactive kd tree gpu raytracing," in *Proc. of I3D*. ACM, 2007, pp. 167–174.
[31] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time kd-tree construction on graphics hardware," in *ACM Trans. Graph.*, vol. 27, no. 5, 2008, p. 126.
[32] K. Xu, Y. Li, T. Ju, S. Hu, and T. Liu, "Efficient affinity-based edit propagation using kd tree," in *ACM Trans. Graph.*, vol. 28, no. 5, 2009, p. 118.
[33] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297. California, USA, 1967, p. 14.
[34] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.
[35] X. An and F. Pellacini, "Appprop: all-pairs appearance-space edit propagation," in *ACM Trans. Graph.*, vol. 27, no. 3, 2008, p. 40.
[36] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. of ICCV*. IEEE, 1998, pp. 839–846.
[37] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Trans. Graph.*, vol. 26, no. 3, p. 51, 2007.
[38] *CUBLAS Library*, NVIDIA, Feburary 2012.
[39] J.-P. Tarel and N. Hautiere, "Fast visibility restoration from a single color or gray level image," in *ICCV*. IEEE, 2009, pp. 2201–2208.
[40] R. Guo, Q. Dai, and D. Hoiem, "Single-image shadow detection and removal using paired regions," in *Proc. of CVPR*. IEEE, 2011, pp. 2033–2040.
[41] Y. Shor and D. Lischinski, "The shadow meets the mask: Pyramid-based shadow removal," in *Computer Graphics Forum*, vol. 27, no. 2, 2008, pp. 577–586.

Chunxia Xiao received his BSc and MSc degrees from the Mathematics Department of Hunan Normal University in 1999 and 2002, respectively, and received his Ph.D. from the State Key Lab of CAD & CG of Zhejiang University in 2006, China. He is currently a professor at the Computer School, Wuhan University, China. His research interests include digital geometry processing, image and video processing, and computational photography.

Meng Liu received a B.S. degree in 2009 and a M.S. degree in 2011, both from the School of Computer, Wuhan University. His research interests include image and video editing, GPGPU applications.



Donglin Xiao received the BS degree from the School of Computer, Wuhan University in 2011, is currently a master student at School of Computer, Wuhan University, China. His research interests include image and video processing.



Zhao Dong is a postdoc researcher at Program of Computer Graphics of Cornell University. He received his Ph.D. degree in Computer Graphics from Max-Planck-Institut Informatik, Germany in 2011, and M.S. and B.S degrees from Zhejiang University, China, in 2005 and 2001, respectively. His current research interests include physically-based material modelling and rendering, real-time global illumination rendering and volume graphics.

Kwan-Liu Ma is a professor of computer science and the chair of the Graduate Group in Computer Science (GGCS) at the University of California, Davis. He leads the VIDi research group and directs the UC Davis Center for Visualization. Professor Ma received his PhD degree in computer science from the University of Utah in 1993. He was a recipient of the PECASE award in 2000. His research interests include visualization, high-performance computing, computer graphics, and user interface design. Professor Ma was a paper chair of the IEEE Visualization Conference in 2008 and 2009, and an associate editor of IEEE TVCG (2007-2011). He is a founding member of the IEEE Pacific Visualization Symposium and the IEEE Symposium on Large Data Analysis and Visualization. Professor Ma presently serves on the editorial boards of the IEEE CG&A, the Journal of Computational Science and Discoveries, and the Journal of Visualization. Professor Ma is an IEEE Fellow.