Interactive High-Quality Green-Screen Keying via Color Unmixing

YAĞIZ AKSOY
ETH Zürich and Disney Research Zürich
TUNÇ OZAN AYDIN
Disney Research Zürich
MARC POLLEFEYS
ETH Zürich
and
ALJOŠA SMOLIĆ
Disney Research Zürich

Due to the widespread use of compositing in contemporary feature films, green-screen keying has become an essential part of postproduction workflows. To comply with the ever-increasing quality requirements of the industry, specialized compositing artists spend countless hours using multiple commercial software tools, while eventually having to resort to manual painting because of the many shortcomings of these tools. Due to the sheer amount of manual labor involved in the process, new green-screen keying approaches that produce better keying results with less user interaction are welcome additions to the compositing artist's arsenal. We found that—contrary to the common belief in the research community—production-quality green-screen keying is still an unresolved problem with its unique challenges. In this article, we propose a novel green-screen keying method utilizing a new energy minimization-based *color unmixing* algorithm. We present comprehensive comparisons with commercial software packages and relevant methods in literature, which show that the quality of our results is superior to any other currently available green-screen keying solution. It is important to note that, using the proposed method, these high-quality results can be generated using only one-tenth of the manual editing time that a professional compositing artist requires to process the same content having all previous state-of-the-art tools at one's disposal.

Categories and Subject Descriptors: I.3.8 [**Computer Graphics**]: Applications; I.4.6 [**Image Processing and Computer Vision**]: Segmentation—*Pixel classification*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Color*

General Terms: Image and Video Processing

Additional Key Words and Phrases: Green-screen keying, image matting, soft segmentation, interactive segmentation

ACM Reference Format:

Yağız Aksoy, Tunç Ozan Aydin, Marc Pollefeys, and Aljoša Smolić. 2016. Interactive high-quality green-screen keying via color unmixing. ACM Trans. Graph. 35, 5, Article 152 (August 2016), 12 pages.
DOI: http://dx.doi.org/10.1145/2907940

1. INTRODUCTION

As computer-generated imagery became convincingly realistic, compositing synthetic backgrounds and objects into live-action shots became a common practice in feature-film production. The widespread use of composite shots over pure live action is often motivated by the higher degree of artistic control over the final shot, as well as the potential to reduce production costs. Usually, the first step in a digital compositing workflow is the performance capture of the actors and various other live-action elements against a controlled—typically green—background. Then, in postproduction, one needs to obtain RGBA foreground layers corresponding to the live-action elements that ideally carry no trace of the green-screen background. This process is often referred to as *keying*. Finally, one or more foreground layers are combined with the desired computer-generated scene elements to obtain the composite shot.

Keying is a crucial intermediate step in any compositing workflow, as later in the workflow, seamless blending between the synthetic and live-action elements is highly dependent on obtaining high-quality keying results. The keying process usually starts with the compositing artist obtaining preliminary foreground layers by using multiple software tools in concert, some of the most popular ones being The Foundry's *Keylight*, Nuke's Image-Based Keyer *(IBK)* and Red Giant's *Primatte*. Often, this first step already involves significant manual labor in the form of parameter tweaking or drawing roto-masks. Ideally, the preliminary foreground layers would already be sufficiently high quality so that one can move on to consecutive steps in the compositing pipeline. Unfortunately, this

Authors' addresses: Y. Aksoy, T. O. Aydin, and A. Smolic, Disney Research Zurich, Stampfenbachstrasse 48, CH-8006, Zurich, Switzerland; emails: {yaksoy, tunc, smolic}@disneyresearch.com; M. Pollefeys, Dept. of Computer Science, ETH Zurich, Universitatstrasse 6, CH-8092, Zurich, Switzerland; email: marc.pollefeys@inf.ethz.ch.

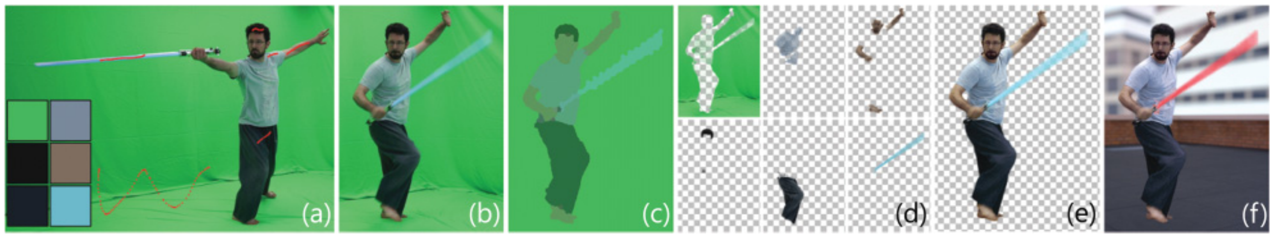

Fig. 1. Major steps of our method. First, parameters of a global color model are obtained from a key frame via a simple scribble interface (a) (Section 4.1). For a different query frame (b), the global color model is refined into local color models (c) (Section 4.2), which are utilized for extracting multiple color layers via color unmixing (d) (Section 3). A subset of layers is then combined to get the final keying result (e). The layers can be used for compositing as well as color editing (f).

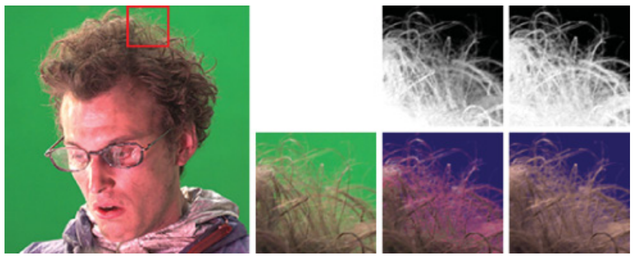

Fig. 2. High-quality alpha maps do not necessarily result in high-quality foreground layers for keying. While both alpha maps capture the intricate details of the actor's hair, the foreground layer computed by comprehensive sampling [Shahrian et al. 2013] (left) has noticeable color artifacts, while the foreground layer computed by our method has the correct colors.

is rarely the case in practice; the imperfections in the foreground layer still have to be corrected by manual painting before moving forward. In professional circles, the combined manual work required for both obtaining preliminary keying results and later their refinement by manual painting is recognized as a significant bottleneck in postproduction.

While the shortcomings of the currently available keying tools are well known in the industry, the focus of relevant academic research is almost exclusively on the related *natural matting* problem. An important distinction between natural matting and keying is in their end goals. While the end result of the keying process is one or more RGBA foreground layers with both correct colors *and* precise alpha maps, natural matting methods very often solely focus on the extraction of alpha maps. In fact, the widely used natural matting benchmark [Rhemann et al. 2009] evaluates performance based only on alpha masks and not foreground layer colors. Figure 2 shows two seemingly high-quality alpha maps with significantly different corresponding foreground layers: while one is almost perfect, the other has significant color artifacts. In fact, our experiments with the state-of-the-art natural matting methods show that their performance in the alpha matting benchmark does not necessarily carry over to green-screen keying challenges.

The feedback that we collected from industry professionals as well as our own experience showed that commercial software tools have difficulties dealing with image regions where the colors of multiple objects mix, either due to motion blur, intricate object boundaries (e.g., hair), or color spill (color cast due to indirect illumination from green-screen). Influenced by this observation, we propose a novel energy function for solving the fundamental problem of *unmixing* a color mixture, that is, computing both the individual underlying colors as well as their mixing ratios (alpha values). We efficiently minimize this energy function by utilizing priors for the underlying colors in the mixture, which are obtained and refined through a two-step user interaction. We demonstrate the application of our color unmixing framework to green-screen keying. In a comprehensive set of quantitative and qualitative evaluations utilizing a paid compositing artist, we show that our method consistently outperforms both the current commercial software tools and the state-of-the-art natural matting methods in the domain of green-screen keying. It is important to note that the superior results of our technique can be obtained, on average, by using only *one-tenth* of the manual interaction time required by a trained artist for processing the same content with the current state-of-the-art tools. Major steps of our pipeline can be seen in Figure 1.

2. RELATED WORK

Green/blue screen keying has received little attention in the research community. In order to solve the underconstrained keying problem, Smith and Blinn [1996] and Grundhöfer et al. [2010] proposed methods that capture the same foreground with two different backgrounds, providing additional equations to the linear system. A radiometric compensation method was proposed by Grundhöfer and Bimber [2008] in order to solve the problem against arbitrary backgrounds as well. However, these methods require specialized setups that limit their practical use. Our method, in comparison, requires a regular video stream shot against only a single background. Thus, its practical use is similar to commercial keying software such as Keylight, IBK, or Primatte.

Commercial keying tools often use chroma-based or luma-based algorithms. In feature-film postproduction, these tools are operated by specialized compositing artists for obtaining a *preliminary* keying result. Preliminary results often require further manual processing because, despite the parameter tweaking and the usage of roto-masks, they often fall short of the quality level demanded in professional productions. Figure 10 shows various examples in which a trained artist simply cannot achieve production-level quality due to the various limitations of currently available tools. Since such keying results are unacceptable in professional production, the preliminary keying results undergo an extremely tedious manual painting process, in which each pixel in the video is cleaned of keying errors by hand.

Natural alpha matting methods are generally classified as sampling- or propagation-based. Local propagation-based methods [Sun et al. 2004; Levin et al. 2008a, 2008b; Singaraju et al. 2009] typically rely on the assumption that there is a smooth transition between foreground and background layers, and solve the matting problem by identifying these transitions. The *matting Laplacian* introduced by Levin et al. [2008a] has been employed or improved by numerous methods [Singaraju et al. 2009; Gastal and Oliveira

2010] and applied to multiple layers [Singaraju and Vidal 2011]. Nonlocal propagation-based methods [Lee and Wu 2011; He et al. 2013; Shi et al. 2013; Chen et al. 2013a] make use of the *nonlocal principle* introduced by Lee and Wu [2011].

Sampling-based methods can be divided into parametric and nonparametric ones. Nonparametric sampling-based methods [He et al. 2011; Shahrian and Rajan 2012; Shahrian et al. 2013; Johnson et al. 2014] propose strategies to effectively select many samples from both foreground and background, and conduct matting by finding foreground–background sample pairs that can represent an observed mixed pixel by their weighted sum. Parametric sampling-based methods [Ruzon and Tomasi 2000; Chuang et al. 2001; Wang and Cohen 2005] estimate a color distribution for each pixel using nearby known pixels and solve the matting problem accordingly. The soft segmentation method proposed by Tai et al. [2007] also utilizes a parametric representation of the colors in an image.

Other methods combine propagation and sample-based approaches [Wang and Cohen 2007; Rhemann et al. 2008; Chen et al. 2013b; Jin et al. 2014] and pose an energy minimization problem using the possible samples as a unary energy component and an alpha-based propagation term as smoothness. Our formulation is also based on energy minimization, while we only use the appearance of a pixel without any propagation.

In literature, the matting Laplacian utilized by Levin et al. [2008a] and Gastal and Oliveira [2010] among others, has been criticized for having overly strong regularization [Lee and Wu 2011] and creating unnecessary ambiguities [Singaraju et al. 2009]. Input in the form of trimaps, which is needed for methods such as Ruzon and Tomasi [2000], Chuang et al. [2001], and Gastal and Oliveira [2010] among others, has been said not to have direct influence on the result [Levin et al. 2008a], making users unable to predict or reiterate the results without actually running the algorithm every time, or to be very time-consuming to be practical [Wang and Cohen 2005]. In contrast, our method is parametric, but it does not rely on propagation or automated sampling. We obtain our model parameters directly through a two-step user interaction scheme. In the first step, the user identifies an arbitrary number of dominant colors in the scene that are used to build a global color model (Section 4.1), which can be locally refined further in a second user interaction step (Section 4.2).

Video matting methods are often extensions of their image matting counterparts. While some methods use propagated or edited trimaps for each frame and apply image matting methods directly [Chuang et al. 2002; Li et al. 2005; Bai et al. 2011; Tang et al. 2012; Fan et al. 2012], others apply consistency constraints or sampling strategies in video volume instead of image plane [Wang et al. 2005; Bai et al. 2009; Choi et al. 2012; Zhong et al. 2012; Li et al. 2013; Shahrian et al. 2014].

Matting for specialized applications involves making specific assumptions or utilizing additional user input. For example, shadow matting allows specific assumptions on appearance; the method proposed by Wu et al. [2007] outperforms natural matting methods in this application. Similarly, motion-blurred objects can be extracted more precisely when user-guided motion vectors are utilized [Lin et al. 2011]. Extracting smooth transparent layers has also been studied [Yeung et al. 2008], which can be applied to shadow removal. The downside of these methods is their rather limited applications, as they do not generalize beyond the specific subproblems on which they focus.

Color unmixing is an interesting problem for a wider range of applications. Using an image formation model, Carroll et al. [2011] propose illumination decomposition for material recoloring. For effective image-based rendering in the presence of reflective surfaces, Sinha et al. [2012] decompose the scene into reflected and transmitted surfaces. Shih et al. [2015] remove reflections from glass windows using an attenuation model for ghosting cues. Despite sharing the common high-level goal of unmixing scene colors, these methods are designed for substantially different use cases than green-screen keying, which is our main focus in this work.

3. COLOR UNMIXING

The central component of our method is an energy minimization framework, in which the color $\boldsymbol{c}$ of a pixel is hypothesized to be a mixture of a number of *underlying colors* $\boldsymbol{u_i}$. The problem solved by our framework is the estimation of the underlying colors and their mixing ratios (α_i), such that the linear combination of the underlying colors weighted by corresponding mixing ratios gives the original pixel color $\boldsymbol{c}$. To that end, we build and utilize a parametric representation of all the colors present in the scene, which we simply call the *color model*. The color model comprises N distributions in RGB space. Both the number and the parameters of these distributions are obtained through user interaction. We assume that the color model for an input scene is already known to us throughout this section; rather, we focus on the formulation and efficient solution of the color-unmixing problem. A detailed discussion on building the color model of an input scene will follow in Section 4.

We start formulating our color unmixing framework by defining three basic constraints that each pixel should satisfy: (i) an *alpha constraint*, which states that the alpha values α_i should sum up to unity; (ii) a *color constraint*, which states that we should obtain the original color $\boldsymbol{c}$ of the pixel when we mix the underlying colors $\boldsymbol{u_i}$ using the corresponding alpha values; and (iii) a *box constraint* that limits the space of possible alpha and color values. Formally, we express these constraints as follows:

$$\sum_i \alpha_i = 1, \quad \sum_i \alpha_i \boldsymbol{u}_i = \boldsymbol{c}, \quad \text{and} \quad \alpha_i, \boldsymbol{u}_i \in [0, 1]. \tag{1}$$

The cost associated with the occurrence of an underlying color $\boldsymbol{u_i}$ in a mixture $\boldsymbol{c}$ is defined by how well it fits to the corresponding distribution $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, where μ and Σ denote the mean vector and the covariance matrix, and $\mathcal{N}$ is the normal distribution. We use the squared Mahalanobis distance as our measure of goodness of fit:

$$\mathcal{D}_i(\boldsymbol{u}) = (\boldsymbol{u} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{u} - \boldsymbol{\mu}_i), \tag{2}$$

and define our energy function $\mathcal{F}$ of selecting a particular mixture of N underlying colors accordingly:

$$\mathcal{F} = \sum_i \alpha_i \mathcal{D}_i(\boldsymbol{u_i}). \tag{3}$$

This energy function favors layer colors that have the best likelihoods according to their corresponding color distributions, especially for the layers with higher alpha values. Minimization of this energy subjected to the color constraint makes sure that the resultant layers successfully represent the color mixture that formed the observed pixel color.

While the energy function $\mathcal{F}$ may seem straightforward, we found that its minimization is nontrivial. Since the energy function $\mathcal{F}$ and the color constraint defined in Equation (1) are nonlinear, we are faced with a nonlinearly constrained nonlinear optimization problem. Specifically, the color constraint in Equation (1) constrains a single alpha value for three underlying color channels at once. This makes our energy function $\mathcal{F}$ prone to get stuck in local minima within the vicinity of the initial point if the constraints are enforced from the start. What we need instead is an algorithm that strictly enforces the constraints only after allowing us to find some reasonable alpha and color values first. To that end, we utilize the method called *the original method of multipliers* [Bertsekas 1982]. We express the deviation from the constraints in Equation (1) as

$$\mathcal{G}_\alpha = \left(\sum_i \alpha_i - 1\right)^2 \quad \text{and} \quad \boldsymbol{\mathcal{G}}_u = \left(\sum_i (\alpha_i \boldsymbol{u}_i) - \boldsymbol{c}\right)^{\bullet 2}, \qquad (4)$$

where $(\cdot)^{\bullet 2}$ denotes the elementwise squaring operation. This leads to the constraint vector $\boldsymbol{\mathcal{G}} = [\boldsymbol{\mathcal{G}}_u^T \ \ \mathcal{G}_\alpha]^T$. The vector containing the variables $\boldsymbol{x}$ that are the arguments of the optimization is

$$\boldsymbol{x} = \left[\alpha_1 \ \ldots \ \alpha_N \ \ \boldsymbol{u}_1^T \ \ldots \ \boldsymbol{u}_N^T\right]^T. \qquad (5)$$

Note that $\boldsymbol{x}$ contains the variables for unmixing a single pixel. The optimization is performed independently for every pixel; for each pixel, we solve for both the alpha values and the underlying colors simultaneously. Further details of our optimization procedure are discussed in Section 3.1.

Once $\boldsymbol{x}$ is computed for all pixels of an input video frame, for each pixel we obtain N underlying colors and corresponding alpha values. If we visualize the i^{th} underlying color for all pixels of the video frame with their alpha values, we obtain the RGBA layer corresponding to the distribution $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$. Green-screen keying can be seen as a special case in which we remove the RGBA layer corresponding to the green-screen background.

In contrast to our color-unmixing method, related works on parametric natural matting use Bayesian formulations with either local [Ruzon and Tomasi 2000; Chuang et al. 2001] or global models [Tai et al. 2007]. Local methods solve for alpha values first, then estimate colors. On the other hand, the method in Tai et al. [2007] iteratively estimates the alpha values and colors for all pixels of an input image, which makes it feasible for only low-resolution images. In contrast, our formulation is easily parallelizable as each pixel is treated independently; thus, our method easily scales to HD resolutions and beyond.

Sampling-based natural matting methods such as comprehensive sampling [Shahrian et al. 2013] take alternate approaches to compute foreground layer colors where they try all the possible background-foreground color pairs to get the best match from a limited set of color samples. Certain priors commonly utilized by these methods, such as matte sparsity [Wang and Cohen 2007; Gastal and Oliveira 2010], are often violated in green-screen keying due to color spill.

On the other hand, commercial chroma-based keying tools simply suppress the background green-screen color everywhere in the frame, which often distorts the colors of the foreground objects, especially if they are similar to the color of the green-screen background. Around intricate object boundaries or motion blur, they extend the foreground region without actually unmixing the colors. As a result, they leave an unnatural halo around difficult regions.

To summarize, the natural matting methods in the literature, as well as commercial keying tools, fail to achieve production-level quality in green-screen keying due to their various shortcomings discussed earlier. The main advantages of our color unmixing formulation are the following:

—Our method does not enforce a matte-sparsity constraint, nor rely on the suppression of the color of the green-screen background.
—Our formulation is highly scalable and parallelizable, as each pixel is processed independently.
—The proposed energy minimization successfully unmixes even mixtures of very similar colors (demonstrated later in Section 5.1) and is agnostic to the scene colors, that is, we do not require a strong chroma or luma component, as in commercial software.
—Similar to KNN Matting [Chen et al. 2013a], our method computes multiple RGBA layers as its output, which enables further interesting applications beyond green-screen keying, such as color editing.

In the next section, we continue with a discussion of the two-step user interaction process and other details of the color model, which we treated as a black box so far.

3.1 Minimization of the Color Unmixing Energy

The color unmixing energy introduced in Equation (3) is optimized using Algorithm 1. The function minimized in Line 1 is composed of the original energy function and the deviations from the constraints. Minimization at this step is done using the nonlinear conjugate gradient method that takes $\boldsymbol{x}_k$ as the initial value. The step size of the nonlinear conjugate gradient at each iteration is determined by a line search in the direction determined via the Polak–Ribière formula. The box constraints are enforced at each iteration of the nonlinear conjugate gradient method by clipping the elements to be in the range [0, 1] and setting the gradients of the elements at the boundaries 0 and 1 to zero if they are positive or negative, respectively. As the parameters $\rho_{(\cdot)}$ and $\boldsymbol{\lambda}_{(\cdot)}$ increase at each iteration of Algorithm 1 (Lines 2 and 3), the energy $\mathcal{F}(\boldsymbol{x})$ is minimized while allowing increasingly smaller deviations from the alpha and color constraints in Line 1. $\boldsymbol{\lambda}_{(\cdot)}$ punishes deviation from individual constraints, while $\rho_{(\cdot)}$ increases the constraint enforcement globally. The input to Algorithm 1, initial values for α_i and $\boldsymbol{u}_i$, are taken as:

$$\alpha_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad \boldsymbol{u}_i = \begin{cases} \boldsymbol{c} & \text{if } i = j \\ \boldsymbol{\mu}_i & \text{otherwise} \end{cases},$$

where $j = \arg\min_i \mathcal{D}_i(\boldsymbol{c})$, that is, only the alpha value corresponding to the most likely distribution in the color model is initialized to be 1. Note that the optimization procedure that we described is independent for each pixel in an image.

ALGORITHM 1: The Original Method of Multipliers

```
Input: $\boldsymbol{x}_0$
Define: $k = 0, \rho_0 = 0.1, \boldsymbol{\lambda}_0 = [0.1\ 0.1\ 0.1\ 0.1]^T, \beta = 10, \gamma = 0.25, \epsilon > 0$
1: $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \left(\mathcal{F}(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{\mathcal{G}}(\boldsymbol{x}) + \frac{1}{2}\rho_k \|\boldsymbol{\mathcal{G}}(\boldsymbol{x})\|^2\right)$
2: $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \rho_k \boldsymbol{\mathcal{G}}(\boldsymbol{x}_{k+1})$
3: $\rho_{k+1} = \begin{cases} \beta\rho_k & \text{if } \|\boldsymbol{\mathcal{G}}(\boldsymbol{x}_{k+1})\| > \gamma \|\boldsymbol{\mathcal{G}}(\boldsymbol{x}_k)\| \\ \rho_k & \text{otherwise} \end{cases}$
4: if $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| > \epsilon$ then
5:     $k \leftarrow k + 1$
6:     go to Step 1
7: else
8:     return $\boldsymbol{x}_{k+1}$
```

4. BUILDING THE COLOR MODEL

The energy function $\mathcal{F}$ that we defined in Equation (3) requires a parametric representation of the colors that formed the color mixture, which we refer as the color model. A set of distributions is obtained in the first step of the user interaction of our method. The resulting *global* color model (Section 4.1) is assumed to be able to represent the whole image. The global color model is locally overcomplete since, very often, each pixel color $\boldsymbol{c}$ is a mixture of only a subset of the scene colors. We call the subset of distributions that participate in the color mixture in a certain region of an image the *active color distributions*. In Section 4.2, we refine the global color model such that each pixel is associated only with its active color distributions. This refinement process is performed automatically by utilizing a Markov Random Field optimization, but we also allow the user to edit the resulting *local* color models in an optional second user interaction step.

In comparison, commercial green-screen keying software packages offer a multitude of interaction modes ranging from background/foreground color selection to rotoscoping interfaces. They also typically offer user control over various parameters that control the amount of chroma suppression, matte blurring, or matte bleed. Although this high level of control allows compositing artists to fine-tune keying results, it also makes the process highly time-consuming. On the other hand, natural matting methods typically require *trimaps* (dense approximate segmentation of the image into foreground, background, and unknown regions), which are in practice extremely tedious to generate, especially for video sequences, and have been criticized for influencing the result only indirectly [Levin et al. 2008a]. Some natural matting methods instead rely on the user drawing a sparse set of scribbles, which often results in a more convenient user interaction.

The goal of the user interaction in our method is to extract the information we need to build the color model as intuitively and efficiently as possible. Consecutively, instead of relying on complex user interactions such as commercial keying tools or requiring prohibitively time-consuming inputs such as a trimap, we utilize a two-step interaction that involves drawing a small number of scribbles (typically 7–8) and an optional pointing-and-clicking step.

4.1 Global Color Model

The user interaction typically starts with the user loading the first frame of an input video using the interface of our method. The goal of the first user interaction step is building the global color model, which is achieved by the user drawing a scribble over each of the *dominant* scene colors. The number of the scribbles N, thus the number of dominant scene colors, is determined by the user depending on the scene. For example, in Figure 8(b) (our result, input) each different color on the person's wig is selected separately as a dominant color, whereas in Figure 8(a) (our result, input), the actor's natural hair color is marked as a single dominant color.

Each scribble identifying a dominant color is used to extract the parameters of a distinct normal distribution. The mean and covariance of each distribution are computed simply from the pixels underneath the corresponding scribbles (note that we do not use any scribble propagation). Importantly, the results of our color-unmixing method are not sensitive to the exact placement, size, or shape of the scribbles (Figure 3). This property is very useful in practice, as high-quality results can be obtained quickly from roughly drawn scribbles. Additionally, once the global color model is created for a single frame, it can typically be used for the remaining frames of the shot assuming that the dominant colors do not change significantly. In fact, the global color models of all video results presented in this work were generated from a single frame (typically the first frame). The motivation behind this first user interaction step is utilizing the inherently good cognitive skills of the users for clustering colors. These cognitive skills are especially helpful in dealing with specific situations, such as the presence of strong color spill. Figure 10(d) (original) shows an example in which the color of the actor's robe is affected by the indirect illumination from the green-screen, except for only very few small regions. In this case, recognizing the color spill and selecting unaffected regions as a dominant color are trivial for a human user while the same tasks

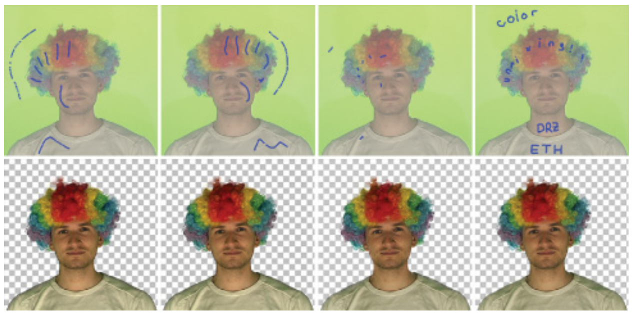

Fig. 3. The keying results generated using four different scribbles demonstrate the robustness of our algorithm against different user inputs.

are extremely difficult for an automatic color-clustering algorithm. In fact, although we experimented with methods for automatically building the global color model (see Section 5.3), we found that, in most practical cases, user interaction would be necessary and, therefore, favored our current interactive approach. The ability to select the dominant colors also gives the user artistic control over the color composition of the resulting RGBA layers, which is especially useful for compositing artists.

4.2 Local Color Model

One shortcoming of the global color model is the assumption that the color of each pixel of the input video is a mixture of N underlying colors from the N distributions that make up the color model. However, in practice, this assumption is almost always incorrect. For example, in the original image in Figure 4, skin tones are present only in a small region near the actor's face and neck. If we solely rely on the global color model, we would have to use the distribution corresponding to the skin tones for unmixing pixels in completely unrelated image regions, such as the far edges of the green-screen background. This may cause the color unmixing to hallucinate nonexistent colors with small alpha values in such regions. Thus, we perform a Markov Random Field (MRF)-based optimization procedure over superpixels to estimate the active subset of color distributions for different regions in an image.

If desired, the result of this optimization procedure can be edited through user interaction via a simple point-and-click interface. Since the automatic color activation is rather computationally costly, and it would be cumbersome to perform the local color model edits repeatedly for every frame, we propagate the local color model of an edited frame to the following frame through simple superpixel matching. For every superpixel in a new frame, we find a corresponding superpixel in the previous frame in a small spatial neighborhood with the closest mean color. The active distributions of a superpixel in the new frame is defined as the active distributions of its match in the previous frame. An example of local color models, a typical user edit, and propagation to consecutive frames are illustrated in Figure 4.

The local color model computation step can loosely be related to the sample selection process employed by sampling-based natural matting methods such as shared sampling [Gastal and Oliveira 2010], in which the goal is also to find the best-fitting distributions for every pixel. However, their brute-force approach is fundamentally different from our MRF optimization process.

Several natural matting methods, such as comprehensive sampling [Shahrian et al. 2013], utilize localized color models. While we select a subset of the global color model as the local color

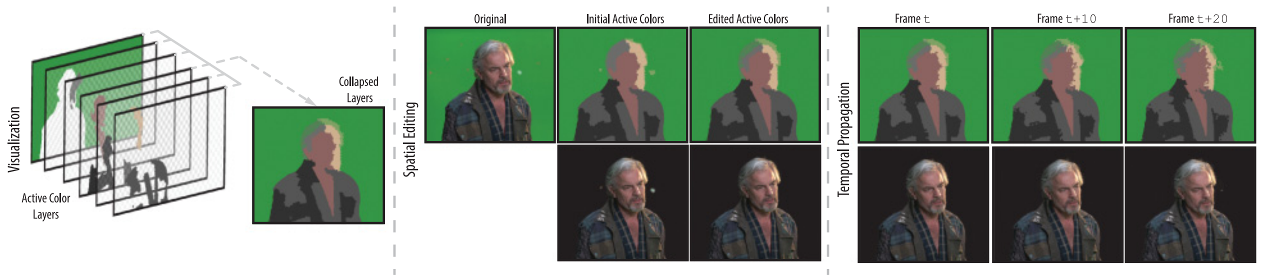

Fig. 4. **Visualization:** The result of local color model estimation can be visualized as a cascade of layers that illustrate the active color distributions by their mean colors. **Editing:** The MRF optimization for the local color models may fail to distinguish between different objects with similar colors (such as the markers and the actor's face), or may give suboptimal results when one of the colors is present only faintly in a region (such as the color spill in the actor's hair from the green screen). Such situations can be alleviated by refining the local color model via a simple point-and-click user interface. **Propagation:** The user interaction can be streamlined by propagating the local color model to consecutive frames.

model, comprehensive sampling estimates a set of normal distributions from the close-by foreground and background regions for a mixed-color pixel. Although this approach provides some robustness against complex backgrounds, it has several shortcomings in the green-screen keying case. Under heavy color spill, estimating distributions locally is typically insufficient since the pure-color regions may occur in a very limited part of the image and cannot be integrated into the local models. It also inherently increases the number of necessary distributions to represent the image, making the direct user-edits inconvenient, if not impossible. The resulting localized layers then require additional temporal coherency steps to be applied to image sequences, since spatially they are expected to change from frame to frame. Hence, we found our definition of local color models as a subset of a global model to be practically well-fitting to our target application of green-screen keying.

4.2.1 *Local Color Model Estimation.* We represent the active distributions of a pixel as a binary vector $\boldsymbol{A}$ of length N, and define the cost of activating a subset of distributions for a pixel as the sum of two terms. The first term is the minimum energy defined in Equation (3) when the subset of distributions are fed to the energy minimization algorithm detailed in Section 3, denoted by $\mathcal{F}_{\boldsymbol{A}}$. The intuition here is that, if the optimization is conducted with distributions that fail to effectively represent an observed color, the minimized energy will still be high. The second term $\mathcal{G}_{\boldsymbol{A}} = \|\boldsymbol{A}\|$, $\|\cdot\|$ representing the Euclidean norm, is added to this cost in order to favor fewer active colors for each pixel. Following these definitions, the unary potentials are defined as

$$\mathcal{U}_{\boldsymbol{A}} = \mathcal{F}_{\boldsymbol{A}} + \delta \mathcal{G}_{\boldsymbol{A}}, \tag{6}$$

where δ is a user-specified weight parameter typically in the range [5 10]. The binary potentials between neighboring pixels are defined as

$$\mathcal{B}_{p,q} = \|\boldsymbol{A}_p - \boldsymbol{A}_q\| e^{-\|\boldsymbol{c}_p - \boldsymbol{c}_q\|}. \tag{7}$$

The energy function that we want to minimize in order to determine active color distributions is

$$\mathcal{E} = \underset{\boldsymbol{A}_{(\cdot)}}{\arg\min} \sum_p \mathcal{U}_{\boldsymbol{A}_p} + \sigma \sum_{(p,q)\in\Omega} \mathcal{B}_{p,q}, \tag{8}$$

where σ is the smoothness parameter, typically selected in the range [0.01 0.05], and Ω is the set of 8-connected pixels.

The problem that we defined in this section is analogous to multilabel segmentation if we treat each possible subset of active color distributions as a label. The minimization of the energy defined in Equation (8) is NP-hard [Boykov et al. 2001]. We approximate the global solution of this energy minimization using the $\alpha - \beta$ *swap* algorithm presented by Boykov et al. [2001], using the publicly available implementation by the authors [Kolmogorov and Zabih 2004; Boykov and Kolmogorov 2004].

Although we presented our energy formulation in this section at the pixel level, computing $\mathcal{F}_{\boldsymbol{A}}$ for every subset and every pixel can be time-consuming, especially if N is high. In order to make the local color model estimation more efficient, we instead construct the random field using SLIC superpixels [Achanta et al. 2012] (typically 10k superpixels for a 1080p frame). This allows a user controllable trade-off between quality and computational efficiency.

5. RESULTS

Our method is suitable for parallel computation, as discussed in Section 3. For a 1080p frame, our current C++/CUDA implementation typically requires 10s for local color estimation (assuming 8 dominant colors), another second to propagate the local color model to the following frame, and approximately 3s for color unmixing. Thus, at this resolution, the total computation time for a still image is 13s, which drops to 4s per frame for image sequences.

In this section, we evaluate our method and present results for various applications. In the absence of a comprehensive ground-truth dataset of green-screen content, in our experiments, we utilize computer-generated ground truth, as well as keying results generated by a paid independent professional compositing artist. In contrast, all user interaction with our method was performed by people with no prior experience in digital keying or compositing.

5.1 Statistical Validation

In this experiment, we test how distinct two colors have to be for our unmixing algorithm to work successfully. To that end, we generated a total of 480 images, each obtained by overlaying 2 or 3 images created by randomly sampling from one of 720 different normal distributions with varying mean vectors and covariance matrices. The images were overlaid via a known alpha matte, which served also as the ground truth. Examples of these test images are shown in Figure 5. For the distinctiveness measure, we use Bhattacharyya distance,[1] which models the amount of overlap between two normal distributions. Figure 5 shows that our method can successfully

[1]Bhattacharyya distance between $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ and $\mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$ is defined as $\frac{1}{8}(\mu_i - \mu_j)^T \Sigma^{-1}(\mu_i - \mu_j) + \frac{1}{2} ln(\frac{det(\Sigma)}{\sqrt{det(\Sigma_i) det(\Sigma_j)}})$; $\Sigma = \frac{\Sigma_i + \Sigma_j}{2}$.

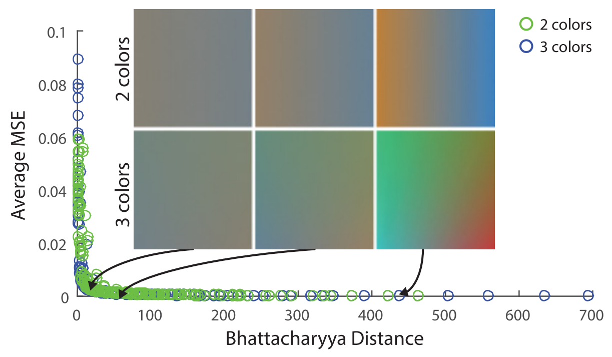

Fig. 5. The average MSE error plotted with respect to the distance between the distributions in the color model. The rightmost images show two cases in which distributions are very distinct. The leftmost images are at the point that our energy function starts to fail at discriminating between colors, effectively illustrating the limits of the proposed color unmixing.

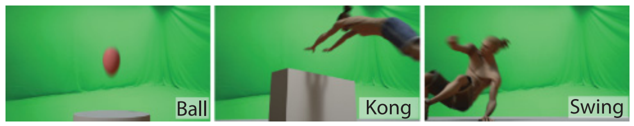

Fig. 6. Three animated image sequences are overlaid onto a challenging green-screen in order to create data with ground truth. *Ball* represents a simple scene with high motion blur, while *Kong* and *Swing* represent live-action scenes with fast motion.

unmix colors up to a point when they become hard to distinguish by a human observer.

5.2 Evaluation on Synthetic Video

Due to the absence of ground-truth data for green-screen keying, we prepared a test set of computer-generated video sequences (Figure 6) rendered with a live-action green-screen in the background. We used this ground-truth data to compare the performance of our method with three leading commercial keying tools (IBK, Keylight, and Primatte). In the first experiment, we compared the *out-of-the-box* performance by providing only minimal user input to all methods, that is, by selecting a reasonable background color for the commercial tools and selecting 5 to 9 dominant colors for our method.

In the second experiment, we asked a paid compositing artist to generate the best possible result separately with each commercial tool. The artist reported spending 105min to 120min with each tool. For comparison, we also processed the same sequences with our method to achieve the best possible keying result, for which we spent 10min mostly refining the local color maps.

Table I shows that our keying results are objectively better than the three commercial tools for all test sequences, both with minimal and optimal level of user input. In some cases, such as the performance of Primatte in the Swing sequence, we observed that further processing by the artist is essential to get a more reasonable result, which means that, for a novice user, it is harder to get a good initial estimate. Note also that the user interaction of our method is an order of magnitude more efficient when one seeks to obtain the best possible result.

Table I. Quantitative Comparison of the Proposed Algorithm with Industrial Keying Tools

		1000 × MSE Color			1000 × MSE Alpha		
		Ball	*Kong*	*Swing*	*Ball*	*Kong*	*Swing*
Minimal	*IBK*	0.0170	0.0553	0.1139	0.5353	0.5954	2.1232
	Keylight	1.6001	0.5247	0.4831	2.3645	1.3389	2.7036
	Primatte	5.8097	1.6830	27.0635	6.8404	2.6980	32.0337
	Ours	**0.0096**	**0.0250**	**0.0489**	**0.1286**	**0.4114**	**1.2722**
Optimal	*IBK*	0.0129	0.0504	0.1658	0.0583	0.1510	0.2291
	Keylight	0.0239	0.0842	0.1301	0.0200	0.4841	0.1583
	Primatte	0.0492	0.2348	0.2587	0.1391	0.5487	0.6166
	Ours	**0.0034**	**0.0189**	**0.0304**	**0.0089**	**0.0421**	**0.0585**

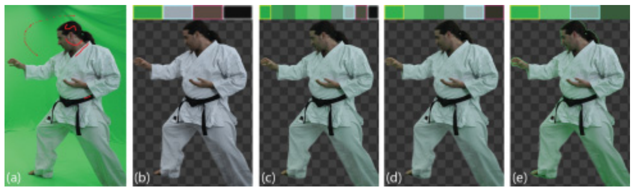

Fig. 7. The results of our algorithm when the color model is inferred from the scribbles (b) and when the color model is estimated by expectation maximization using different numbers of distributions (10 for (c), 6 for (d) and 4 for (e)). The EM algorithm is run using all the pixels in a small region of interest (a). The highlighted colors are the colors estimated by EM that are closest to our original four distributions.

5.3 Color Model Estimation Using EM

As an alternative to scribble-based interaction to infer the global color model, we tried to estimate the distributions using expectation maximization.

The main problem with expectation maximization is that it is unable to separate the areas with color spill (indirect illumination from the green-screen material) from the clean areas. As Figure 7 shows, the distribution corresponding to the white robe of the actor appears greenish regardless of the number of distributions estimated by EM. This is expected since the pure white color appears in very limited regions while the greenish white is dominant due to the strong color spill.

Using our scribble interface, the user can select regions without color spill, and our color unmixing algorithm is able to separate the spill from the robe.

5.4 Green-Screen Keying

5.4.1 *Comparison with Natural Alpha Matting Methods.* We compare our method to four natural matting methods with publicly available implementations. All four methods—KNN matting (KNN) [Chen et al. 2013a], shared matting (SM) [Gastal and Oliveira 2010], weighted color and texture sampling (WCTS) [Shahrian and Rajan 2012] and comprehensive sampling (CS) [Shahrian et al. 2013]—compute not only alpha values but also the corresponding foreground colors. For this comparison, we first prepared a very detailed and narrow trimap and dilated the unknown regions by 6 and 12 pixels to obtain two additional trimaps (following the procedure from the alpha matting benchmark [Rhemann et al. 2009]). For scenes with substantial color spill, we prepared two sets of trimaps, in which one considers the regions with spill as

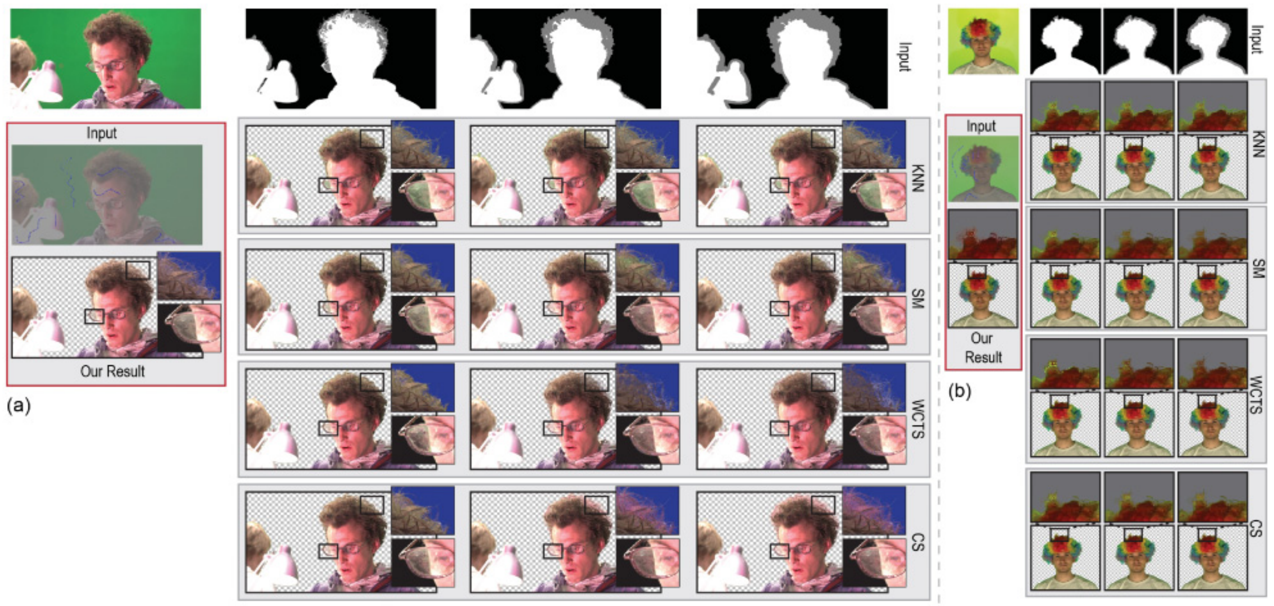

Fig. 8. Results of KNN matting [Chen et al. 2013a] (KNN), shared matting [Gastal and Oliveira 2010] (SM), weighted color and texture sampling [Shahrian and Rajan 2012] (WCTS) and comprehensive sampling [Shahrian et al. 2013] (CS) are presented using different trimaps together with our input scribbles and keying results. Note that our scribbles are drawn on the first frames of the corresponding videos. Plate (a) shows an example with intricate object boundaries as well as translucent regions, and (b) shows another example with many foreground colors that also include a green tone close to the background color.

unknown and the other as foreground. The final trimaps and corresponding results can be seen in Figure 8 and in the supplementary material.

The intricate object boundaries in Figure 8(a) demonstrate a fail case for sample selection strategies of WCTS and CS, as they partly use samples from the actor's face rather than his hair, causing the hair to appear to have a red hue. SM gives the cleanest result in this case among the natural matting methods. Figure 8(b) shows that the presence of the color green on the actor's wig degrades the performance of KNN, WCTS, and CS, while the local color model assumption of SM helps to extract a cleaner foreground. However, SM fails to extract the fine details as our method does, possibly due to the sparsity assumption of SM.

The scenes shown in Figure 8 are selected to highlight several challenges of green-screen keying. The results show that our method performs favorably against the state-of-the-art natural matting methods.

5.4.2 *Comparison with Commercial Keying Software.* As mentioned in Section 2, several methods have been proposed to solve the keying problem by capturing the same foreground against different background colors. Figure 9 shows that our algorithm gives comparable results to such a method [Grundhöfer et al. 2010] using only a single background.

The keying tools that are widely used in production do not rely on any special setups. In this section, we compare our method with some of the leading commercial keying tools: Keylight, Primatte and IBK. To that end, we used green-screen shots from the open-source movie Tears of Steel[2] as well as some content that we shot with a Sony α7s camera.

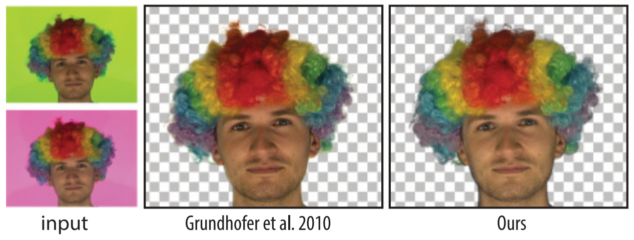

Fig. 9. Our result obtained using only the image with the green background is comparable to the result by Grundhöfer et al. [2010] obtained with both input images.

In order to present a fair comparison, we asked a paid professional compositing artist to generate a separate result with each tool for each test scene. Based on the artist's feedback, that in most real-world scenarios all three tools would be used sequentially to take advantage of their individual strengths, we decided to ask the artist to generate another set of results in which he is allowed to use all three tools. We did not impose any constraints on the artist other than asking him to avoid manually painting pixels.

For the four sequences in our test set, the artist reported a total of 9h to get the results using multiple tools and reported an estimated 12h for fixing any remaining issues. Our results, on the other hand, were generated by us using our tool in less than 1h. Almost the entire time was spent on refining the local color models using the point-and-click interface of our method[3].

The results presented in Figure 10 show that our results compare favorably to the artist's results, even when the artist uses all the tools at his disposal and spends approximately an order of magnitude

[2](CC) Blender Foundation—mango.blender.org.

[3]Refer to the supplemental video for a demonstration of user interaction.

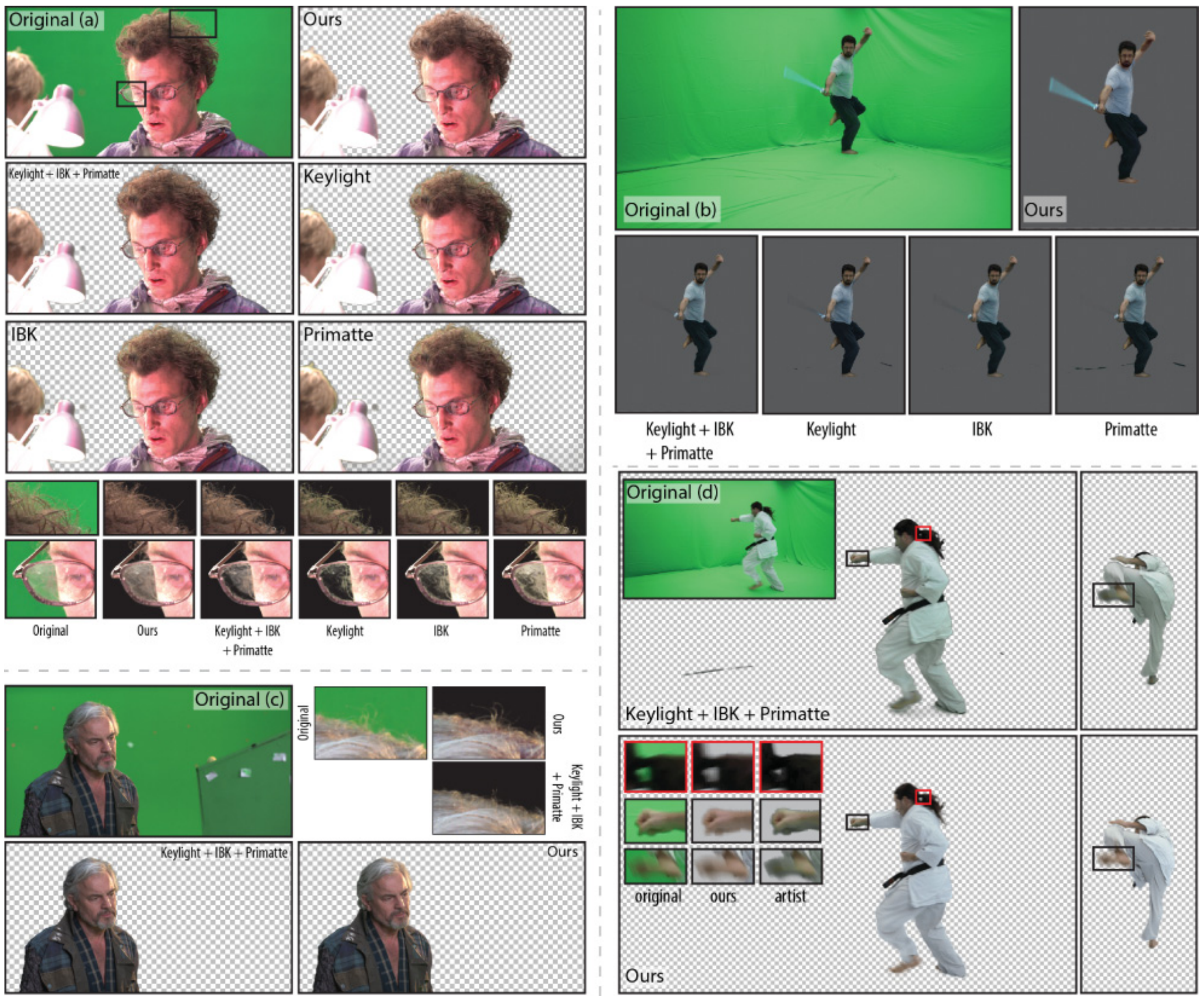

Fig. 10. Commercial keying tools, even when operated by a specialized compositing artist, may not be able to extract the fine details near intricate object boundaries (a), may fail to extract highly blurred objects (b), may distort the foreground color if it is mixed with the background color (c), or may create unnatural artifacts around blurred regions (d), while our algorithm is robust against such scenarios.

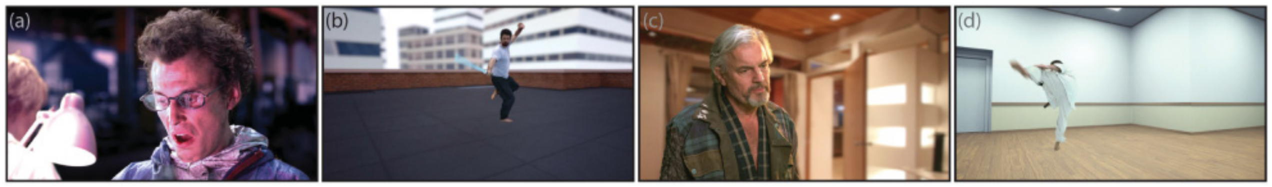

Fig. 11. The main real-world application of our method is digital compositing. The figure shows a number of toy examples that we generated using the foreground layers obtained with our prototype implementation. Background images courtesy of Flickr users *milanboers* (a) and *jeremylevinedesign* (c).

more time on manual editing. Additionally, the complex workflow and heavy local editing employed by the artist may result in temporal coherence artifacts. In contrast, our results for the same sequences do not suffer from such artifacts, as illustrated in Figure 13.

Because of the high amount of spill on the actor in scenes shown in Figures 10(b) and 10(d), actors appear transparent in the extracted foreground layer. Discriminating between transparency occurring from color spill or motion blur in a principled way is not a trivial problem. In order to account for this, we apply a simple post-processing composed of boosting α values of the foreground layers with high spill to 1 except for the edges of the layers. For instance, the layer corresponding to the white robe in Figure 10(d) appears

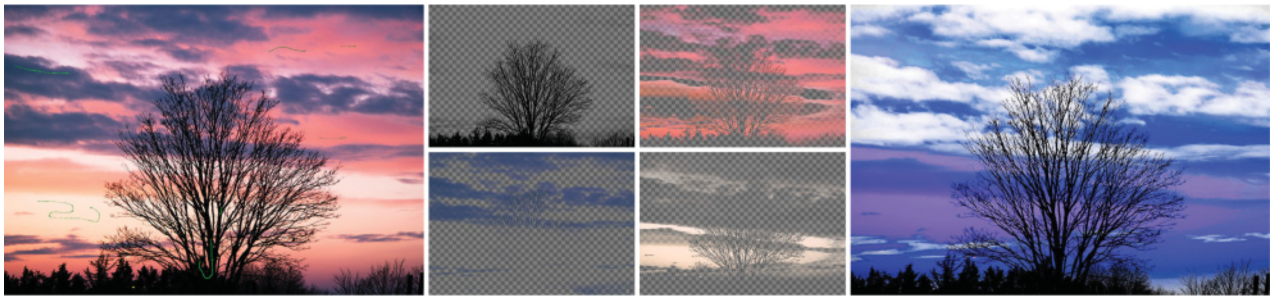

Fig. 12. A possible application scenario for our color unmixing algorithm is interactive color editing. Here, an image with input scribbles and corresponding color layers are shown with an edited image using our layers.

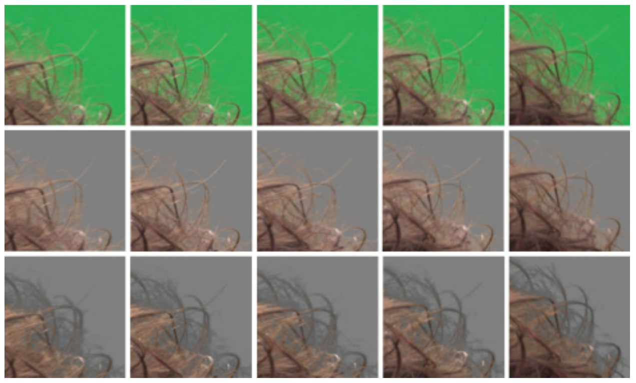

Fig. 13. Close-up around the same single filament in several frames of the video. Note that the filament that sticks out is captured using our method even when it is motion-blurred (middle), but the artist was only able to capture it in some of the frames (bottom).

transparent after color unmixing. The robe layer is postprocessed such that it has unity alpha values in regions that are not on the edges of the robe. The edges are left untouched to account for the smooth transition and the motion blur around the edges. While this postprocessing is not completely foolproof, that is, its performance will degrade if there is strong color spill on layers with high transparency, we found it to be helpful for compositing and left the classification of nonunity alpha values to color spill or transparency as future work. Figure 11 shows examples of compositing results generated using the foreground layers extracted by our method.

5.5 Further Applications

5.5.1 *Non-Green-Screen Keying.* We also tested our method using scenes with non-green-screen backgrounds. Figure 14 shows an example in which our per-pixel color unmixing approach proves to be robust against complex foreground structures. Another example, one that includes reflections from a semitransparent medium, is shown in Figure 15. While the backgrounds in these examples are admittedly simple, the results presented in this section suggest that our method could be useful for an extended set of applications beyond green-screen keying. However, it is worth noting that our method is limited to simple backgrounds and is not suitable for general-purpose natural matting.

5.5.2 *Color Manipulation.* Representing the image with multiple layers rather than just foreground and background opens up new application areas, such as color editing. By giving the artist freedom to edit layers of each dominant color in the scene, interesting results can be achieved easily while not being limited by scenes with motion blur, as demonstrated in Figures 1 and 15.

The layers extracted by our unmixing algorithm can also be used for photo recoloring similar to soft segmentation [Tai et al. 2005, 2007] or palette-based recoloring algorithms such as Chang et al. [2015], as seen in Figure 12.

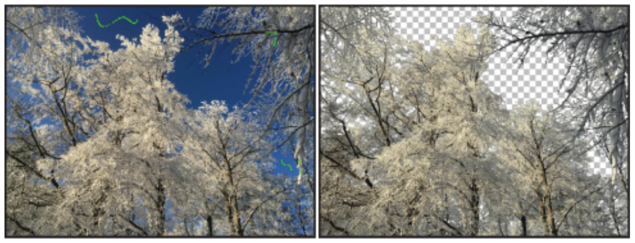

Fig. 14. Despite the very complex scene structure, our algorithm successfully removes the sky in the background, demonstrating an advantage of our per-pixel approach to color unmixing that does not rely on spatial cues.

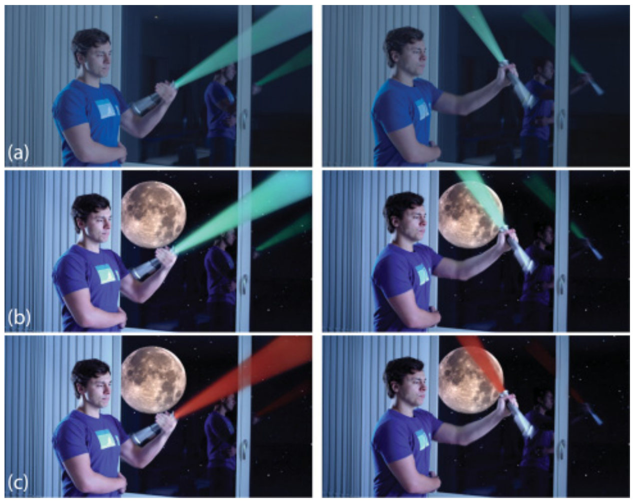

Fig. 15. The layers computed by our algorithm are used to replace the background (b) and change the color of the blurred object (c) while retaining the reflections. Note that the result images are color graded while compositing. Background image courtesy of Flickr user *davejdoe*.

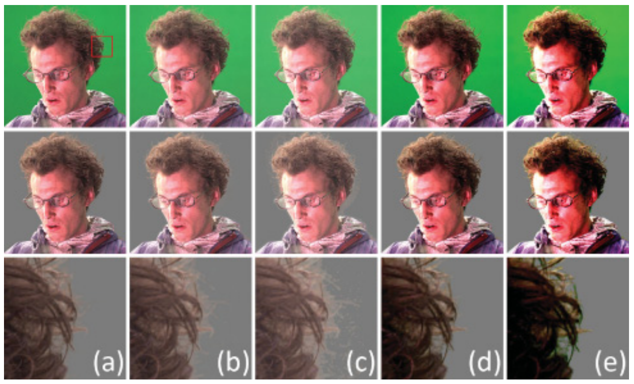

Fig. 16. We changed the illumination or contrast of the input frame and extracted the foreground using the same color model constructed from the original image (a). With slight changes (b, d), our method is able to successfully extract the foreground. With a significant change in brightness (c), we observe a drop in the performance of our method, characterized by the halo around the actor. On the other hand, with very significant contrast change (e), some intricate details are missed and the background color remains in some small regions in the foreground.

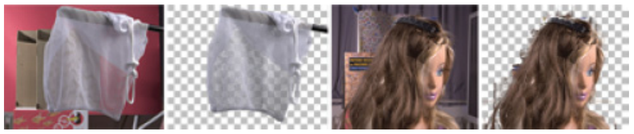

Fig. 17. When our assumption of a small number of scene colors is satisfied, we are able to get a successful foreground layer (left), but the quality drops significantly otherwise. Images courtesy of Rhemann et al. [2009].

6. LIMITATIONS AND DISCUSSION

While, in our experiments, we have not noticed any significant temporal consistency issues, our test scenes had admittedly near-constant illumination. In practice, keying may need to be performed in outdoor scenes (such as driving), in which the illumination can change drastically from one frame to another. Due to the absence of any mechanism to enforce temporal coherence, we expect the performance of our method to decrease in such settings, as demonstrated in Figure 16.

The global color model as a small set of distributions may not be able to effectively represent non-green-screen backgrounds. We tested our method on several images from the alpha matting benchmark [Rhemann et al. 2009]. Figure 17 shows typical natural matting results in which our method works well when our main assumptions are satisfied, but fails when they are violated.

Our scribble interface for extracting the color model requires the unmixed colors to be present in at least one of the frames. For highly transparent media such as thin smoke, the pure color cannot be determined via the proposed interaction; thus, it is not possible for our keying system to extract the layer with only smoke. Devising an algorithm that can infer the colors that only appear mixed with others in a scene is an interesting direction for further research.

The proposed color unmixing algorithm may slightly overestimate the alpha values of some layers in some cases. Since the energy minimization favors underlying colors that are closer to the mean vector of the distributions, the foreground layer might get a small portion of the color mixture since matte sparsity is not enforced in the color unmixing energy minimization by design. This mainly occurs when the underlying color of one of the layers is not well represented by the corresponding distribution. These artificially occurring alpha values being very small, we observed that this behavior does not result in any disturbing artifacts in the keying results.

7. CONCLUSION

In this article, we proposed an interactive technique for green-screen keying, which is a highly relevant problem in the postproduction industry due to the popularity of digital compositing. We presented a novel energy minimization–based color unmixing algorithm that relies on global/local parametric color models and can achieve high-quality keying results even in challenging cases. We show that our algorithm outperforms the state-of-the-art in natural matting in the case of green-screen keying. Our technique also substantially decreases the interaction time required for achieving production-ready keying quality when compared to commercial keying tools.

Future research directions include the investigation of temporal coherency for scenes with dramatic illumination changes, evaluating whether the discriminative power of our algorithm can be improved by using a perceptually uniform color space instead of RGB, and exploring further applications of our color-based soft-segmentation such as local contrast editing.

ACKNOWLEDGMENTS

We would like to thank Alison Marlin for the voiceover of the accompanying video; Alessia Marra and Maurizio Nitti for their support in preparing the compositing examples and animated sequences; Amit Bermano, Alexandre Chapiro, and Torsten Sattler for their performances in our live-action recordings; Michele Dell'Ambrogio for his efforts in extending our code to a Nuke plug-in; Anselm Grundhöfer for our discussions on keying; as well as Changil Kim, Jordi Pont-Tuset, Torsten Sattler, and Nikolce Stefanoski for their feedback on the text.

REFERENCES

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 11, 2274–2282.

Xue Bai, Jue Wang, and David Simons. 2011. Towards temporally-coherent video matting. In *Proceedings of MIRAGE*.

Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. 2009. Video SnapCut: Robust video object cutout using localized classifiers. *ACM Transactions on Graphics* 28, 3, 70:1–70:11.

Dimitri P. Bertsekas. 1982. The method of multipliers for equality constrained problems. In *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, NY, 96–157.

Y. Boykov and V. Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 9, 1124–1137.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 11, 1222–1239.

Robert Carroll, Ravi Ramamoorthi, and Maneesh Agrawala. 2011. Illumination decomposition for material recoloring with consistent interreflections. *ACM Transactions on Graphics* 30, 4, 43:1–43:10.

Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015. Palette-based photo recoloring. *ACM Transactions on Graphics* 34, 4, 139:1–139:11.

Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. 2013a. KNN matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 9, 2175–2188.

Xiaowu Chen, Dongqing Zou, S. Z. Zhou, Qinping Zhao, and Ping Tan. 2013b. Image matting with local and nonlocal smooth priors. In *Proceedings of CVPR*.

Inchang Choi, Minhaeng Lee, and Yu-Wing Tai. 2012. Video matting using multi-frame nonlocal matting Laplacian. In *Proceedings of ECCV*.

Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. 2002. Video matting of complex scenes. *Transactions on Graphics* 243–248.

Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. 2001. A Bayesian approach to digital matting. In *Proceedings of CVPR*.

Jialue Fan, Xiaohui Shen, and Ying Wu. 2012. Scribble tracker: A matting-based approach for robust tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 8, 1633–1644.

Eduardo S. L. Gastal and Manuel M. Oliveira. 2010. Shared sampling for real-time alpha matting. *Computer Graphics Forum* 29, 2, 575–584.

Anselm Grundhöfer and Oliver Bimber. 2008. VirtualStudio2Go: Digital video composition for real environments. *ACM Transactions on Graphics* 27, 5, 151:1–151:8.

Anselm Grundhöfer, Daniel Kurz, Sebastian Thiele, and Oliver Bimber. 2010. Color invariant chroma keying and color spill neutralization for dynamic scenes and cameras. *The Visual Computer* 26, 9, 1167–1176.

Bei He, Guijin Wang, Chenbo Shi, Xuanwu Yin, Bo Liu, and Xinggang Lin. 2013. Iterative transductive learning for alpha matting. In *Proceedings of ICIP*.

Kaiming He, C. Rhemann, C. Rother, Xiaoou Tang, and Jian Sun. 2011. A global sampling method for alpha matting. In *Proceedings of CVPR*.

Meiguang Jin, Byoung-Kwang Kim, and Woo-Jin Song. 2014. Adaptive propagation-based color-sampling for alpha matting. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 7, 1101–1110.

Jubin Johnson, Deepu Rajan, and Hisham Cholakkal. 2014. Sparse codes as alpha matte. In *Proceedings of BMVC*.

V. Kolmogorov and R. Zabih. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2, 147–159.

P. Lee and Ying Wu. 2011. Nonlocal matting. In *Proceedings of CVPR*.

Anat Levin, Dani Lischinski, and Yair Weiss. 2008a. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2, 228–242.

Anat Levin, Alex Rav-Acha, and Dani Lischinski. 2008b. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 10, 1699–1712.

Dingzeyu Li, Qifeng Chen, and Chi-Keung Tang. 2013. Motion-aware KNN Laplacian for video matting. In *Proceedings of ICCV*.

Yin Li, Jian Sun, and Heung-Yeung Shum. 2005. Video object cut and paste. *ACM Transactions on Graphics* 24, 3, 595–600.

Hai Ting Lin, Yu-Wing Tai, and M. S. Brown. 2011. Motion regularization for matting motion blurred objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 11, 2329–2336.

Christoph Rhemann, Carsten Rother, and Margrit Gelautz. 2008. Improving color modeling for alpha matting. In *Proceedings of BMVC*.

Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. 2009. A perceptually motivated online benchmark for image matting. In *Proceedings of CVPR*.

Mark A. Ruzon and Carlo Tomasi. 2000. Alpha estimation in natural images. In *Proceedings of CVPR*.

E. Shahrian, B. Price, S. Cohen, and D. Rajan. 2014. Temporally coherent and spatially accurate video matting. *Computer Graphics Forum* 33, 2, 381–390.

E. Shahrian and D. Rajan. 2012. Weighted color and texture sample selection for image matting. In *Proceedings of CVPR*.

E. Shahrian, D. Rajan, B. Price, and S. Cohen. 2013. Improving image matting using comprehensive sampling sets. In *Proceedings of CVPR*.

Yongfang Shi, O. C. Au, Jiahao Pang, K. Tang, Wenxiu Sun, Hong Zhang, Wenjing Zhu, and Luheng Jia. 2013. Color clustering matting. In *Proceedings of ICME*.

YiChang Shih, Dilip Krishnan, Fredo Durand, and William T. Freeman. 2015. Reflection removal using ghosting cues. In *Proceedings of CVPR*.

D. Singaraju, C. Rother, and C. Rhemann. 2009. New appearance models for natural image matting. In *Proceedings of CVPR*.

D. Singaraju and R. Vidal. 2011. Estimation of alpha mattes for multiple image layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 7, 1295–1309.

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics* 31, 4, 100:1–100:10.

Alvy Ray Smith and James F. Blinn. 1996. Blue screen matting. *ACM Transactions on Graphics* (1996), 259–268.

Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. 2004. Poisson matting. *ACM Transactions on Graphics* 23, 3, 315–321.

Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. 2005. Local color transfer via probabilistic segmentation by expectation-maximization. In *Proceedings of CVPR*.

Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. 2007. Soft color segmentation and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 9, 1520–1537.

Zhen Tang, Zhenjiang Miao, Yanli Wan, and Dianyong Zhang. 2012. Video matting via opacity propagation. *The Visual Computer* 28, 1, 47–61.

Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. 2005. Interactive video cutout. *ACM Transactions on Graphics* 24, 3, 585–594.

Jue Wang and M. F. Cohen. 2005. An iterative optimization approach for unified image segmentation and matting. In *Proceedings of ICCV*.

Jue Wang and M. F. Cohen. 2007. Optimized color sampling for robust matting. In *Proceedings of CVPR*.

Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. 2007. Natural shadow matting. *ACM Transactions on Graphics* 26, 2.

Sai-Kit Yeung, Tai-Pang Wu, and Chi-Keung Tang. 2008. Extracting smooth and transparent layers from a single image. In *Proceedings of CVPR*.

Fan Zhong, Xueying Qin, Qunsheng Peng, and Xiangxu Meng. 2012. Discontinuity-aware video object cutout. *ACM Transactions on Graphics* 31, 6, 175:1–175:10.

Received September 2015; revised February 2016; accepted March 2016