Designing Effective Inter-Pixel Information Flow for Natural Image Matting

Yağız Aksoy, Tunç Özcan Aydın, and Marc Pollefeys

Abstract—We present a novel, purely affinity-based natural image matting algorithm. Our method relies on carefully defined pixel-to-pixel connections that enable effective use of information available in the image. We control the information flow from the known-opacity regions into the unknown region, as well as within the unknown region itself, by utilizing multiple definitions of pixel affinities. Among other forms of information flow, we introduce color-mixture flow, which builds upon local linear embedding and effectively encapsulates the relation between different pixel opacities. Our resulting novel linear system formulation can be solved in closed-form and is robust against several fundamental challenges of natural matting such as holes and remote intricate structures. Our evaluation using the alpha matting benchmark suggests a significant performance improvement over the current methods. While our method is primarily designed as a standalone matting tool, we show that it can also be used for regularizing mattes obtained by sampling-based methods. We extend our formulation to layer color estimation and show that the use of multiple channels of flow increases the layer color quality. We also demonstrate our performance in green-screen keying and further analyze the characteristics of the affinities used in our method.

Index Terms—Natural image matting, transparency, compositing, green screen keying

1 INTRODUCTION

Extracting the opacity information of foreground objects from an image is known as natural image matting. Natural image matting has received great interest from the research community through the last decade and can nowadays be considered as one of the classical research problems in visual computing. Mathematically, image matting requires expressing pixel colors in the transition regions from foreground to background as a convex combination of their underlying foreground and background colors. The weight, or the opacity, of the foreground color is referred to as the alpha value of that pixel. Since neither the foreground and background colors nor the opacities are known, estimating the opacity values is a highly ill-posed problem. To alleviate the difficulty of this problem, typically a trimap is provided in addition to the original image. The trimap is a rough segmentation of the input image into foreground, background, and regions with unknown opacity.

Affinity-based methods [1], [2], [3] constitute one of the prominent natural matting approaches in literature. These methods make use of pixel similarities to propagate the alpha values from the known-alpha regions to the unknown region. They provide a clear mathematical formulation, can be solved in closed-form, are easy to implement, and typically produce spatially consistent mattes. However, current methods fail to effectively handle alpha gradients spanning large areas and spatially disconnected regions (i.e. holes) even in simple cases as demonstrated in Figure 2. This is because a straightforward formulation using the pixel-to-pixel affinity definitions can not effectively represent the complex structures that are commonly seen in real-life objects.

In order to alleviate these shortcomings, we rely on a careful, case-by-case design of how alpha values should propagate inside the image. We refer to this propagation as information flow. The key idea of our paper is a novel strategy for controlling information flow both from the known trimap regions to the unknown region, as well as within the unknown region itself. We formulate this strategy through the use of a variety of affinity definitions including the color-mixture flow, which is based on local linear embedding and tailored for image matting. Step-by-step improvements on the matte quality as we gradually add new building blocks of our information flow strategy are illustrated in Figure 1. Our final linear system can be solved in closed-form and results in a significant quality improvement over the state-of-the-art. We demonstrate the matting quality improvement quantitatively, as well as through a visual inspection of challenging image regions. We also show that our energy function can be reformulated as a post-processing step for regularizing the spatially inconsistent mattes estimated by sampling-based natural matting algorithms.

A preliminary version of this paper has been published elsewhere [4]. In this extended version, we additionally (i) propose a novel foreground color estimation formulation where we introduce a new form of local information flow, (ii) demonstrate that our method achieves state-of-the-art quality in green-screen keying, (iii) provide an in-depth spectral analysis of individual forms of information flow, and (iv) present a discussion on how our method relates to sampling-based matting methods, as well as new quantitative and qualitative results.
The numerous natural matting alpha values of different pixels should relate to each other.

similarity and spatial proximity for determining how the methods [7], [8], [9] as proposed by Gastal and Oliveira [10]. Also adopted in a post-processing step in sampling-based propagate alpha values accordingly. The matting affinity is pixel to determine the amount of information flow and matting affinity [1], operate on a local patch around the known opacity. Local affinity definitions, prominently the proximity and propagate the alpha values from regions with pixel similarity metrics that rely on color similarity or spatial learning-based matting methods.

concepts presented in the paper will inspire next-generation dataset with labels. We hope that our formulation and the all current learning-based methods in the alpha matting images with ground-truth mattes. Our method outperforms train such a network, Xu et al. [14] generated a dataset of 50k knowledge that can be extracted from the image. In order to that generates high-quality mattes with the help of semantic deep neural network architecture has been proposed [14] other matting methods for a better matte [13]. Recently, a trimap in constrained settings [12] or to combine results of the alpha in a semi-supervised setting [11], to estimate a ground and background. While they perform well especially around remote and challenging structures, they require affinity-based regularization to produce spatially consistent mattes. Also, our experience with publicly available matting code suggests that implementing sampling-based methods can be challenging at times.

Machine learning has been used to aid in estimating the alpha in a semi-supervised setting [11], to estimate a trimap in constrained settings [12] or to combine results of other matting methods for a better matte [13]. Recently, a deep neural network architecture has been proposed [14] that generates high-quality mattes with the help of semantic knowledge that can be extracted from the image. In order to train such a network, Xu et al. [14] generated a dataset of 50k images with ground-truth mattes. Our method outperforms all current learning-based methods in the alpha matting benchmark [15] despite not taking advantage of a large dataset with labels. We hope that our formulation and the concepts presented in the paper will inspire next-generation learning-based matting methods.

Affinity-based matting methods mainly make use of pixel similarity metrics that rely on color similarity or spatial proximity and propagate the alpha values from regions with known opacity. Local affinity definitions, prominently the matting affinity [1], operate on a local patch around the pixel to determine the amount of information flow and propagate alpha values accordingly. The matting affinity is also adopted in a post-processing step in sampling-based methods [7], [8], [9] as proposed by Gastal and Oliveira [10].

Methods utilizing nonlocal affinities similarly use color similarity and spatial proximity for determining how the alpha values of different pixels should relate to each other. KNN matting [2] determines several neighbors for every unknown pixel and enforces them to have similar alpha values relative to their distance in a feature space. The manifold-preserving edit propagation algorithm [3] also determines a set of neighbors for every pixel but represents each pixel as a linear combination of its neighbors in their feature space.

Chen et al. [16] proposed a hybrid approach that uses the sampling-based robust matting [17] as a starting point and refines its outcome through a graph-based technique where they combine a nonlocal affinity [3] and the matting affinity. Cho et al. [13] combined the results of closed-form matting [1] and KNN matting [2], as well as the sampling-based method comprehensive sampling [9], by feeding them into a convolutional neural network.

In this work, we propose color-mixture flow and discuss its advantages over the affinity definition utilized by Chen et al. [3]. We also define three other forms of information flow, which we use to carefully distribute the alpha information inside the unknown region. Our approach differs from Chen et al. [16] in that our overall information flow strategy goes beyond combining various pixel affinities, as we discuss further in Section 3 while requiring much less memory to solve the final system. Instead of using the results of other affinity-based methods directly as done by Cho et al. [13], we formulate an elegant formulation that has a closed-form solution. To summarize, we present a novel, purely affinity-based matting algorithm that generates high-quality alpha mattes without making use of sampling or a learning step.

Layer Color Estimation For a given alpha matte, the corresponding foreground colors should also be estimated before compositing. Although the alpha matte is assumed to be given for the foreground color estimation, the problem is still underconstrained as there are 6 unknowns and 3 equations. Levin et al. [1] use the gradient of the alpha matte as a spatial smoothness measure and formulate the layer color estimation as a linear problem. Using only a smoothness measure limits their performance especially in remote regions of the foreground. Chen et al. [2] use the color-similarity measure they employ for matte estimation also for layer color estimation. Typically, using only a color-similarity metric results in incorrectly flat-colored regions and suppressed highlight colors in the foreground. In this work, we introduce a second spatial smoothness measure for the layer colors. We use in total 4 forms of information flow together for the layer estimation and show that our
linear system improves the layer color quality especially in remote parts of the matte.

**Green-Screen Keying** A more constrained version of the natural image matting problem is referred as green-screen keying, where the background colors are homogeneous in a controlled setting. While this problem can be seen as a simpler version of natural matting, as green-screen keying is heavily utilized in professional production, the expected quality of the results is immense. In the movie post-production industry, multiple commercial software such as Keylight or Primatte are used by professional graphical artists to get high-quality keying results. These software typically use chroma-based or luma-based algorithms and provide many parameters that help the artist tweak the results. In their early work, Smith and Blinn [19] formulate the use of the compositing equation for a fixed background color. Recently, an unmixing-based green-screen keying method has been proposed [18] that uses a global color model of the scene and a per-pixel nonlinear energy function to extract the background color in high precision. In their paper, they compare their method to state-of-the-art natural matting methods and show that their method fails to give acceptable results in green-screen settings. In this paper, we show that our matting and color estimation methods outperform the natural matting methods and generate comparable results to that of specialized keying methods or commercial software without any parameter tweaking.

3 Method

Trimaps are typically given as user input in natural matting, and they consist of three regions: fully opaque (foreground), fully transparent (background) and of unknown opacity. \( F \), \( B \) and \( U \) will respectively denote these regions, and \( K \) will represent the union of \( F \) and \( B \). Affinity-based methods operate by propagating opacity information from \( K \) into \( U \) using a variety of affinity definitions. We define this flow of information in multiple ways so that each pixel in \( U \) receives information effectively from different regions in the image.

The opacity transitions in a matte occur as a result of the original colors in the image getting mixed with each other due to transparency or intricate parts of an object. We make use of this fact by representing each pixel in \( U \) as a mixture of similarly-colored pixels and defining a form of information flow that we call color-mixture flow (Section 3.1). We also add connections from every pixel in \( U \) to both \( F \) and \( B \) to facilitate direct information flow from known-opacity regions to even the most remote opacity-transition regions in the image (Section 3.2). In order to distribute the information from the color-mixture and \( K \)-to-\( U \) flows, we define intra-\( U \) flow of information, where pixels with similar colors inside \( U \) share information on their opacity with each other (Section 3.3). Finally, we add local information flow, a pixel affecting the opacity of its immediate spatial neighbors, which ensures spatially coherent end results (Section 3.4). We formulate the individual forms of information flow as energy functions and aggregate them in a global optimization formulation (Section 3.5).

3.1 Color-mixture information flow

Due to transparent objects as well as fine structures and sharp edges of an object that cannot be fully captured due to the finite-resolution of the imaging sensors, certain pixels of an image inevitably contain a mixture of corresponding foreground and background colors. By investigating these color mixtures, we can derive an important clue on how to propagate alpha values between pixels. The amount of the original foreground color in a particular mixture determines the opacity of the pixel. Following this fact, if we represent the color of a pixel as a weighted combination of the colors of several others, those weights should correspond to the opacity relation between the pixels.

In order to make use of this relation, for every pixel in \( U \), we find \( K_{CM} = 20 \) similar pixels in a feature space by an approximate \( K \) nearest neighbors search in the whole image. We define the feature vector for this search as \([r, g, b, \tilde{x}, \tilde{y}]^T\), where \( \tilde{x} \) and \( \tilde{y} \) are the image coordinates normalized by image width and height, and the rest are the RGB values of the pixel. This set of neighbors, selected as similar-colored pixels that are also close-by, is denoted by \( N_{CM} \).

We then find the weights of the combination \( w_{p,q}^{CM} \) that will determine the amount of information flow between the pixels \( p \) and \( q \in N_{CM} \). The weight of each neighbor is defined such that the weighted combination of their colors yields the color of the original pixel:

\[
\arg\min_{w_{p,q}^{CM}} \left\| c_p - \sum_{q \in N_{CM}^p} w_{p,q}^{CM} c_q \right\|_2^2, \quad (1)
\]
where $c_p$ represents the 3x1 vector of RGB values. We minimize this energy using the method by Roweis and Saul [5]. Note that since we are only using RGB values, the neighborhood correlation matrix computed during the minimization has a high chance of being singular as there could easily be two neighbors with identical colors. So, we condition the neighborhood correlation matrix by adding $10^{-3}I_{KCM \times KCM}$ to it before inversion, where $I_{KCM \times KCM}$ is the identity matrix.

Note that while we use the method by Roweis and Saul [5] to minimize the energy in (1), we do not fully adopt their local linear embedding (LLE) method. LLE finds a set of neighbors in a feature space and uses all the variables in the feature space to compute the weights in order to reduce the dimensionality of input data. Manifold-preserving edit propagation [3] and LNSP matting [16] algorithms make use of the LLE weights directly in their formulation for image matting. However, since we are only interested in the weighted combination of colors and not the spatial coordinates, we exclude the spatial coordinates in the energy minimization step. This increases the validity of the estimated weights, effects of which can be observed even in the simplest cases such as in Figure 2 where manifold-preserving weight propagation and CMF-only results only differ in the weight computation step.

The energy term for the color-mixture flow is defined as:

$$E_{CM} = \sum_{p \in U} \left( \alpha_p - \sum_{q \in N^F_p \cup N^B_p} w^C_{p,q} \alpha_q \right)^2. \tag{2}$$

### 3.2 $K$-to-$U$ information flow

The color-mixture flow already provides useful information on how the mixed-color pixels are formed. However, many pixels in $U$ receive information present in the trimap indirectly through their neighbors, all of which can possibly be in $U$. This indirect information flow might not be enough especially for remote regions that are far away from $K$.

In order to facilitate the flow of information from both $F$ and $B$ directly into every region in $U$, we add connections from every pixel in $U$ to several pixels in $K$. For each pixel in $U$, we find $K_{U} = 7$ similar pixels in both $F$ and $B$ separately to form the sets of pixels $N^F_p$ and $N^B_p$ with $K$ nearest neighbors search using the feature space $[r, g, b, 10 \times \bar{r}, 10 \times \bar{g}]^T$ to favor close-by pixels. We use the pixels in $N^F_p$ and $N^B_p$ together to represent the pixel color $c_p$ by minimizing the energy in (1). Using the resulting weights $w^F_{p,q}$ and $w^B_{p,q}$, we define an energy function to represent the $K$-to-$U$ flow:

$$E_{KU} = \sum_{p \in U} \left( \alpha_p - \sum_{q \in N^F_p} w^F_{p,q} \alpha_q - \sum_{q \in N^B_p} w^B_{p,q} \alpha_q \right)^2 \tag{3}$$

Note that $\alpha_q = 1$ for $q \in F$ and $\alpha_q = 0$ for $q \in B$. This fact allows us to define two combined weights, one connecting a pixel to $F$ and another to $B$, as:

$$w^F_p = \sum_{q \in N^F_p} w^F_{p,q} \quad \text{and} \quad w^B_p = \sum_{q \in N^B_p} w^B_{p,q} \tag{4}$$

such that $w^F_p + w^B_p = 1$, and rewrite (3) as:

$$E_{KU} = \sum_{p \in U} (\alpha_p - w^F_p)^2. \tag{5}$$

The energy minimization in (1) gives us similar weights for all $q$ when $c_q$ are similar to each other. As a result, if $N^F_p$ and $N^B_p$ have pixels with similar colors, the estimated weights $w^F_p$ and $w^B_p$ become unreliable. We account for this fact by augmenting the energy function in (5) with confidence values.

We can determine the colors contributing to the mixture estimated by (1) using the weights $w^F_{p,q}$ and $w^B_{p,q}$:

$$c^F_p = \frac{\sum_{q \in N^F_p} w^F_{p,q} c_q}{w^F_p}, \quad c^B_p = \frac{\sum_{q \in N^B_p} w^B_{p,q} c_q}{w^B_p}. \tag{6}$$

and define a confidence metric according to how similar the estimated foreground color $c^F_p$ and background color $c^B_p$ are:

$$\eta_p = \|c^F_p - c^B_p\|^2 / 3. \tag{7}$$

The division by 3 is to get the confidence values between $[0, 1]$. We update the new energy term to reflect our confidence in the estimation:

$$\tilde{E}_{KU} = \sum_{p \in U} \eta_p (\alpha_p - w^F_p)^2. \tag{8}$$

Fig. 3. Direct information flow from both $F$ and $B$ to even the most remote regions in $U$ increases our performance around holes significantly (top inset). Using confidences further increases the performance, especially around regions where FG and BG colors are similar (bottom inset).

Fig. 4. $K$-to-$U$ flow does not perform well when the foreground object is highly-transparent. See text for discussion.

Input Ground-truth Without $K$-to-$U$ flow Without confidences ($\eta_p$) Our method

Input No $K$-to-$U$ flow With $K$-to-$U$ flow
This update to the energy term increases the matting quality in regions with similar foreground and background colors, as seen in Figure 5.

It should be noted that the $K$-to-$U$ flow is not reliable when the foreground is highly transparent, as seen in Figure 4. This is mainly due to the low representational power of $\mathcal{N}_p^F$ and $\mathcal{N}_p^B$ for $c_p$ around large highly-transparent regions as the nearest neighbors search does not give us well-fitting pixels for $w_{p,q}^{F,T}$ estimation. We construct our final linear system accordingly in Section 3.5.

### 3.2.1 Pre-processing the trimap

Prior to determining $\mathcal{N}_p^F$ and $\mathcal{N}_p^B$, we pre-process the input trimap in order to facilitate finding more reliable neighbors, which in turn increases the effectiveness of the $K$-to-$U$ flow. Trims usually have regions marked as $U$ despite being fully opaque or transparent, as drawing a very detailed trimap is both cumbersome and prone to errors.

Several methods [7], [8] refine the trimap as a pre-processing step by expanding $\mathcal{F}$ and $\mathcal{B}$ starting from their boundaries with $U$ as proposed by Shahrian et al. [9]. Incorporating this technique improves our results as shown in Figure 4(d). We also apply this extended $\mathcal{F}$ and $\mathcal{B}$ regions after the matte estimation as a post-processing. Since this trimap trimming method propagates known regions only to nearby pixels, in addition to this edge-based trimming, we also make use of a patch-based trimming step.

To this end, we extend the transparent and opaque regions by relying on patch statistics. We fit a 3D RGB normal distribution $\mathcal{N}_p$ to the $3 \times 3$ window around each pixel $p$. In order to determine the most similar distribution in $\mathcal{F}$ for a pixel $p \in \mathcal{U}$, we first find the 20 distributions with closest mean vectors. We define the foreground match score $b_p^F = \min_{q \in \mathcal{F}} B(\mathcal{N}_p, \mathcal{N}_q)$, where $B(\cdot, \cdot)$ represents the Bhattacharyya distance between two distributions. We then select a region for pixel $p$ according to the following rule:

$$p \in \begin{cases} \tilde{\mathcal{F}} & \text{if } b_p^F < t_c \text{ and } b_p^B > t_f \\ \tilde{\mathcal{B}} & \text{if } b_p^B < t_c \text{ and } b_p^F > t_f \\ \tilde{\mathcal{U}} & \text{otherwise} \end{cases}$$

Simply put, an unknown pixel is marked as $\tilde{\mathcal{F}}$, i.e. in foreground after trimming, if it has a strong match in $\mathcal{F}$ and no match in $\mathcal{B}$, which is determined by constants $t_c = 0.25$ and $t_f = 0.9$. By inserting known-alpha pixels in regions far away from $\mathcal{U}$-$K$ boundaries, we further increase the matting performance in challenging remote regions (Figure 5(e)).

### 3.3 Intra-$U$ information flow

Each individual pixel in $\mathcal{U}$ receives information through the color-mixture and $K$-to-$U$ flows. In addition to these, we would like to distribute the information inside $\mathcal{U}$ effectively. We achieve this by encouraging pixels with similar colors inside $\mathcal{U}$ to have similar opacity.

For each pixel in $\mathcal{U}$, we find $K_{\mathcal{U}} = 5$ nearest neighbors only inside $\mathcal{U}$ to determine $\mathcal{N}_p^{\mathcal{U}}$ using the feature vector defined as $v = [r, g, b, \bar{e}/20, \bar{y}/20]^T$. Notice that we scale the coordinate members of the feature vector we used in Section 3.1 to decrease their effect on the nearest neighbor selection. This lets $\mathcal{N}_p^{\mathcal{U}}$ have pixels inside $\mathcal{U}$ that are far away, so that the information moves more freely inside the unknown region. We use the neighborhood $\mathcal{N}_p^{\mathcal{U}} = \mathcal{N}_p^{\mathcal{U}} \cup \{q \mid p \in \mathcal{N}_q^{\mathcal{U}}\}$ to make sure that information flows both ways between $p$ to $q \in \mathcal{N}_p^{\mathcal{U}}$. We then determine the amount of information flow using the $L^1$ distance between feature vectors:

$$w_{p,q}^{\mathcal{U}} = \max \left(1 - \|v_p - v_q\|_1, 0\right) \quad \forall q \in \mathcal{N}_p^{\mathcal{U}}.$$  

The energy term for intra-$\mathcal{U}$ flow then can be defined as:

$$E_{\mathcal{U}} = \sum_{p \in \mathcal{U}} \sum_{q \in \mathcal{N}_p^{\mathcal{U}}} w_{p,q}^{\mathcal{U}} (\alpha_p - \alpha_q)^2.$$  

The information sharing between the unknown pixels increases the matte quality around intricate structures as demonstrated in Figure 5(e).

KNN matting [2] uses a similar affinity definition to make similar-color pixels have similar opacities. However, relying only on this form of information flow for the whole image creates some typical artifacts in the matte. Depending on the feature vector definition and the image colors, the matte may erroneously underrepresent the smooth transitions (KNN - HSV case in Figure 2) when the neighbors of the pixels in $\mathcal{U}$ happen to be mostly in only $\mathcal{F}$ or $\mathcal{B}$, or create flat alpha regions instead of subtle gradients (KNN - RGB case in Figure 2). Restricting information flow to be solely based on color similarity fails to represent the complex alpha transitions or wide regions with an alpha gradient.

### 3.4 Local information flow

Spatial connectivity is one of the main cues for information flow. We connect each pixel in $\mathcal{U}$ to its 8 immediate neighbors denoted by $\mathcal{N}_p^L$ to ensure spatially smooth mattes. The amount of local information flow should also adapt to strong edges in the image.

To determine the amount of local flow, we rely on the matting affinity definition proposed by Levin et al. [1]. The matting affinity utilizes the local patch statistics to determine the weights $w_{p,q}^L$ for $q \in \mathcal{N}_p^L$. We define our related energy term as follows:

$$E_L = \sum_{p \in \mathcal{U}} \sum_{q \in \mathcal{N}_p^L} w_{p,q}^L (\alpha_p - \alpha_q)^2.$$  

Despite representing local information flow well, matting affinity by itself fails to represent large transition regions (Figure 2 top), or isolated regions that have weak or no spatial connection to $\mathcal{F}$ or $\mathcal{B}$ (Figure 2 bottom).
3.5 Linear system and energy minimization

Our final energy function is a combination of the four energy definitions representing each form of information flow:

\[ E_1 = E_{CM} + \sigma_{KL} E_{KL} + \sigma_{UL} E_{UL} + \sigma_L E_L + \lambda E_T, \tag{13} \]

where \( \sigma_{KL} = 0.05, \sigma_{UL} = 0.01, \sigma_L = 1 \) and \( \lambda = 100 \) are algorithmic constants determining the strength of corresponding information flows, and

\[ E_T = \sum_{p \in F} (\alpha_p - 1)^2 + \sum_{p \in B} (\alpha_p - 0)^2 \]

is the energy term to keep the known opacity values constant. For an image with \( N \) pixels, by defining \( N \times N \) sparse matrices \( W_{CM}, W_{UL} \) and \( W_L \) that have non-zero elements for the pixel pairs with corresponding information flows and the vector \( w^F \) that has elements \( w^F_p \) for \( p \in U \), for \( p \in F \) and 0 for \( p \in B \), we can rewrite (13) in matrix form as:

\[ E_1 = \alpha^T L_{IFM} \alpha + (\alpha - w^F)^T \sigma_{KL} H (\alpha - w^F) + (\alpha - \alpha_K)^T \lambda T (\alpha - \alpha_K), \tag{14} \]

where \( T \) is an \( N \times N \) diagonal matrix with diagonal entry \( (p, p) \) if \( p \in K \) and 0 otherwise, \( H \) is a sparse matrix with diagonal entries \( \eta_p \) as defined in (7), \( \alpha_K \) is a row vector of the alpha values to be estimated, and \( L_{IFM} \) is defined as:

\[ L_{IFM} = (D_{CM} - W_{CM})^T (D_{CM} - W_{CM}) + \sigma_{UL} (D_{UL} - W_{UL}) + \sigma_L (D_L - W_L), \tag{15} \]

where the diagonal matrix \( D_{(j,i)} = \sum_j W(i,j) \).

The energy in (14) can be minimized by solving

\[ (L_{IFM} + \lambda T + \sigma_{KL} H) \alpha = (\lambda T + \sigma_{KL} H) w^F. \tag{16} \]

We define a second energy function that excludes the \( K \)-to-\( U \) information flow:

\[ E_2 = E_{CM} + \sigma_{UL} E_{UL} + \sigma_L E_L + \lambda E_T, \tag{17} \]

which can be written in matrix form as:

\[ E_2 = \alpha^T L_{IFM} \alpha + (\alpha - \alpha_K)^T \lambda T (\alpha - \alpha_K), \tag{18} \]

and can be minimized by solving:

\[ (L_{IFM} + \lambda T) \alpha = \lambda T \alpha_K. \tag{19} \]

We solve the linear systems of equations in (16) and (19) using the preconditioned conjugate gradients method [20].

As mentioned before, the \( K \)-to-\( U \) information flow is not effective for highly transparent objects. To determine whether to include the \( K \)-to-\( U \) information flow and solve for \( E_1 \), or to exclude it and solve for \( E_2 \) for a given image, we use a simple histogram-based classifier to determine if we expect a highly transparent result.

If the matte is highly transparent, the pixels in \( U \) are expected to mostly have colors that are a mixture of \( F \) and \( B \) colors. On the other hand, if the true alpha values are mostly 0 or 1 except for soft transitions, the histogram of \( U \) will likely be a linear combination of the histograms of \( F \) and \( B \) as \( U \) will mostly include very similar colors to that of \( K \). Following this observation, we attempt to express the histogram of the pixels in \( U \), \( \hat{D}_U \), as a linear combination of \( D_F \) and \( D_B \). The histograms are computed from the 20 pixel-wide region around \( U \) in \( F \) and \( B \), respectively. We define the error \( e \), the metric of how well the linear combination represents the true histogram, as:

\[ e = \min_{a,b} \| a D_F + b D_B - D_U \|^2. \tag{20} \]

Higher \( e \) values indicate a highly-transparent matte, in which case we prefer \( E_2 \) over \( E_1 \).

4 Matte regularization for sampling-based matting methods

Sampling-based natural matting methods usually select samples for each pixel in \( U \) either independently or by paying little attention to spatial coherency. In order to obtain a spatially coherent matte, the common practice is to combine their initial guesses for alpha values with a smoothness measure. Multiple methods [7], [8], [9], [10] adopt the post-processing method proposed by Gastal and Oliveira [10] which combines the matting affinity [1] with the sampling-based alpha values and corresponding confidences. This post-processing technique leads to improved mattes, but since it involves only local smoothness, the results can still be suboptimal as seen in Figure 6(b).

Our approach with multiple forms of information flow can also be used for post-processing in a way similar to that of Gastal and Oliveira [10]. Given the initial alpha values \( \hat{\alpha}_p \) and confidences \( \tilde{\eta}_p \) found by a sampling-based method, we define the matte regularization energy:

\[ E_R = E_2 + \sigma_R \sum_{p \in U} \tilde{\eta}_p (\alpha_p - \hat{\alpha}_p)^2, \tag{21} \]

where \( \sigma_R = 0.05 \) determines how much loyalty should be given to the initial values. This energy can be written in matrix form and solved as a linear system in the same way we did in Section 3.5.

Figure 6 shows that this non-local regularization of mattes is more effective especially around challenging foreground structures such as long leaves or holes as seen in the insets. In Section 6.2 we will numerically explore the improvement we achieve by replacing the matte regularization step with ours in several sampling-based methods.
5 FOREGROUND COLOR ESTIMATION

In addition to the alpha matte, we need the unmixed foreground colors \([18]\) that got into the color mixture in transition pixels for seamlessly compositing the foreground onto a novel background. Similar to Levin et al. \([1]\) and Chen et al. \([2]\), we estimate the foreground colors for a given matte, after the matte estimation.

We propagate the layer colors from opaque and transparent regions in a similar way as we propagate known alpha values in Section 3. We make use of the color-mixture and the intra-\(\hat{U}\) information flows by extending the search space and affinity computation to include the given alpha values together with spatial coordinates and pixel colors. We also use the spatial smoothness measure proposed by Levin et al. \([1]\) in addition to a second spatial smoothness measure we introduce in this paper. Figure 7 shows how our color estimation result improves as we add more forms of information flow.

5.1 Information flow definitions

In the layer color estimation problem, the input is assumed to be the original image together with an alpha matte. This requires us to redefine the three regions using the matte instead of a trimap:

\[
p \in \begin{cases} \hat{F} & \text{if } \hat{\alpha}_p = 1 \\ \hat{B} & \text{if } \hat{\alpha}_p = 0 \\ \hat{U} & \text{otherwise.} \end{cases}
\]  

(22)

\(\hat{\alpha}_p\) denote the alpha values that are given as input. The foreground and background colors to be estimated will be denoted by \(f\) and \(b\). For a pixel \(p\), the compositing equation we would like to satisfy can be written as:

\[
c_p = \hat{\alpha}_p f_p + (1 - \hat{\alpha}_p) b_p
\]  

(23)

We will formulate the energy functions for a single color channel and solve for red, green and blue channels independently. The scalars \(f\) and \(b\) will denote the values for a single color channel.

5.1.1 Local information flows

Levin et al. \([1]\) proposed the use of the gradient of the alpha channel as the amount of local information flow for the problem of layer color estimation. They solely rely on this form of information flow for propagating the colors. This local information flow basically enforces neighboring pixels to have similar colors if there is an alpha transition. This flow, which we refer to as \(\alpha\)-transition flow, can be represented by the following energy:

\[
E_{\nabla \hat{\alpha}} = \sum_{p} \sum_{q \in N_p^\alpha} \|\nabla \hat{\alpha}_{(p,q)}\| \left( (f_p - f_q)^2 + (b_p - b_q)^2 \right),
\]  

(24)

where \(\nabla \hat{\alpha}\) represents the alpha gradient. We compute the gradients in the image plane using the 3-tap separable filters of Farid and Simoncelli \([21]\). Note that the neighborhood is defined as the local \(3 \times 3\) neighborhood similar to the local information flow in Section 3.4.

The transition flow helps around small regions with alpha gradient but does not propagate information in flat-alpha regions, such as pure foreground or background regions or regions with flat opacity. We propose a new smoothness measure to address this issue, which we call no-transition flow. The no-transition flow enforces spatial smoothness in regions with small color and alpha gradients:

\[
E_{\nabla c,\nabla \hat{\alpha}} = \sum_{p,q \in N_p^\alpha} w_{p,q} \|\nabla \hat{\alpha}_{(p,q)}\| \left( (f_p - f_q)^2 + (b_p - b_q)^2 \right)
\]  

(25)

where \(w_{p,q}\) is the local information flow weights between pixels, \(\|\nabla \hat{\alpha}_{(p,q)}\|\) is the \(L_2\) norm of the vector formed by gradients of the individual color channels. This term increases the performance around slow alpha transitions and flat-alpha regions, as well as around sharp color edges in the image.

No-transition flow already improves the performance quite noticeably as seen in Figure 7(b). However, using only local information flows perform poorly in remote areas such as the end of long hair filaments (Figure 10(a)) or isolated areas (Figure 7 bottom inset). In order to increase the performance in these type of challenging areas, we make use of two types of non-local information flows.

5.1.2 Color-mixture information flow

The basic principle of color mixture as introduced in Section 3.4 also applies to the relationship between layer colors of pixels in the same neighborhood — if we represent the color and alpha of a pixel as a weighted combination of the colors and alpha of several others, those weights should also represent the layer color relation between the pixels. Since we have \(\hat{\alpha}\)’s as additional information in the layer color estimation scenario, we extend the formulation of color-mixture flow to better fit the layer color estimation problem. Similar to its use in alpha estimation, it provides a well-connected graph and allows dense share of information. The performance improvement by the introduction of the color-mixture energy can be seen in Figure 7(c).

In the layer color estimation scenario, we optimize for both foreground and background colors in the same formulation. It should be emphasized that, as it is apparent from \(23\), the foreground and background colors are undefined for regions with \(\hat{\alpha} = 0\) and \(\hat{\alpha} = 1\), respectively. This requires us to avoid using color-mixture flow into \(\hat{U}\) from \(\hat{B}\) for \(f\) and from \(\hat{F}\) for \(b\). We address this by defining two different neighborhoods and computing individual color-mixture flows for \(f\) and \(b\).
For \( f \), we define the neighborhood \( N_{i,f}^q \) by finding \( K_{CM} \) nearest neighbors in \( (U \cup F) \) using the feature vector \( [r, g, b, \alpha, \bar{x}, \bar{y}]^T \). We then compute the weights \( w_{p,q}^{CM} \) that will determine the amount of information flow, as we did it Section 5.4.

\[
\arg \min_{w_{p,q}^{CM}} \left\| \left[ \frac{c_p}{\alpha_p} \right] - \sum_{q \in N_p^{CM}} w_{p,q}^{CM} \left[ \frac{c_q}{\alpha_q} \right] \right\|^2. \tag{26}
\]

Notice that the search space and the weight computation includes \( \alpha \) in addition to the color and location of pixels.

We compute the background conjugates of the neighborhood and weights, \( \tilde{N}_{i,f}^q \) and \( \tilde{w}_{p,q}^{CM} \), in the same way, and define our color-mixture energy for layer color estimation:

\[
E_{\tilde{U}CM} = \sum_{p \in \tilde{U}} \left( \sum_{q \in \tilde{N}_p^{CM}} \tilde{w}_{p,q}^{CM} \left( (f_p - f_q)^2 + (b_p - b_q)^2 \right) \right). \tag{27}
\]

5.1.3 Intra-\( \tilde{U} \) information flow

Intra-\( \tilde{U} \) information flow, as detailed in Section 3.5, distributes the information between similar-colored pixels inside the unknown region without giving spatial proximity too much emphasis. Its behaviour is also very useful in the case of color estimation, as it makes the foreground colors more coherent throughout the image. For example, as shown in Figure 7, bottom inset shows that the addition of intra-\( \tilde{U} \) flow helps in getting a more realistic color to the isolated plastic region between the two black lines.

We make modifications to intra-\( \tilde{U} \) flow similar to the modifications we made to color-mixture flow, in order to make use of the available information coming from \( \tilde{\alpha} \)'s.

We find \( K_{\tilde{U}} \) nearest neighbors only inside \( \tilde{U} \) to determine \( N_{\tilde{U}}^{CM} \) using the feature vector defined as \( v^e = [r, g, b, \tilde{\alpha}, \tilde{x}, \tilde{y} / 20, \tilde{y} / 20]^T \). We then determine the amount of information flow between two non-local neighbors as:

\[
w_{\tilde{p}, q}^{CM} = \max \left( 1 - \left\| v^e_{\tilde{p}} - v^e_q \right\|_1, 0 \right) \quad \forall q \in N_{\tilde{p}}^{CM}. \tag{28}
\]

With the weights determined, we can define the energy function representing the intra-\( \tilde{U} \) flow:

\[
E_{\tilde{U}CM} = \sum_{p \in \tilde{U}} \sum_{q \in N_{\tilde{p}}^{CM}} \tilde{w}_{\tilde{p}, q}^{CM} \left( (f_{\tilde{p}} - f_q)^2 + (b_{\tilde{p}} - b_q)^2 \right). \tag{29}
\]

5.2 Linear system and energy minimization

The final energy function for layer color estimation is the combination of the four types of information flow defined in Sections 5.1.1 to 5.1.3:

\[
E_c = \sigma_L E_{\tilde{U}CM} + \sigma_L E_{\tilde{U}U} + E_{CM} + \sigma_L U + \lambda E_{COMP}. \tag{30}
\]

\( E_c \) is defined and minimized independently for each color channel.

Following the same strategy as we did in Section 5.3, we rewrite the energy function \( E_c \) in the matrix form, this time as a \( 2N \times 2N \) linear system, and solve it for foreground and background colors for 3 times, once for each color channel, using the preconditioned conjugate gradients method [20].

6 RESULTS AND DISCUSSION

We evaluate the proposed methods for matting, matte regularization, layer color estimation and green-screen keying with comparisons to the state-of-the-art of each application.

6.1 Matte estimation

We quantitatively evaluate the proposed algorithm using the public alpha matting benchmark [15]. At the time of submission, our method ranks in the first place according to the sum-of-absolute-differences (SAD) and mean-squared error (MSE) metrics. The results can be seen in Table 1. Our unoptimized research code written in Matlab requires on average 50 seconds to process a benchmark image.

We also compare our results qualitatively with the closely related methods in Figure 8. We use the results that are available on the matting benchmark for all except

### Table 1

| Method | Average Rank | Troll | Doll | Donkey | Elephant | Plant | Plastic Bag | Net |
|--------|--------------|-------|------|--------|----------|-------|-------------|-----|
| Ours   | 2.7          | 2.3   | 2.6  | 3.8    | 3.8      | 2.0   | 1.4         | 1.4 |
| DCNN   | 2.9          | 2.3   | 2.8  | 3.9    | 3.9      | 2.1   | 1.2         | 1.2 |
| DCNN+  | 4.0          | 4.3   | 4.3  | 4.3    | 4.3      | 2.1   | 1.1         | 1.1 |
| CSC    | 3.1          | 2.2   | 2.2  | 2.2    | 2.2      | 2.2   | 1.1         | 1.1 |
| LNSP   | 11.7         | 8.3   | 11.3 | 13.3   | 13.3     | 9.4   | 7.1         | 7.1 |

* Some columns do not have a bold number when the best-scoring algorithm for that particular image-trimap pair is not among the top-ranking methods included here.

** The ranks presented here only take the already-published methods at the time of the submission into account, hence could differ from the online version of the benchmark.
Fig. 8. Several examples from the alpha matting benchmark [15] are shown (a) with trimaps overlayed onto the images (b). The mattes are computed by closed-form matting [1] (c), KNN matting [2] (d), manifold-preserving edit propagation [3] (e), LNSP matting [16] (f), DCNN matting [13] (g) and the proposed method (h). See text for discussion.

Table 2

| Param | Def. | Val. | Perf. | Val. | Perf. | Val. | Perf. |
|-------|------|------|-------|------|-------|------|-------|
| $K_{CM}$ | 20 | 10 & 1.07% | 15 & 0.44% | 0.46% | 30 & -0.62% |
| $K_{KU}$ | 7 & -0.83% | 4 & -0.41% | 10 & 0.12% | 13 & 0.22% |
| $K_{LU}$ | 5 & -0.15% | 3 & -0.1% | 7 & 0.08% | 9 & 0.11% |
| $σ_{KU}$ | 0.05 & 0.01 & -6.44% | 0.025 & -2.1% | 0.075 & 0.66% | 0.09 & 0.87% |
| $σ_{LU}$ | 0.01 & 0.001 & -0.7% | 0.005 & -0.1% | 0.02 & -0.47% | 0.05 & -3.12% |

Table 3

| Method | Sum of Absolute Differences | Mean Squared Error | Overall | S | L | Overall | S | L |
|--------|-----------------------------|--------------------|---------|---|---|---------|---|---|
| KL-D [8] | 24.4 % | 22.4 % | 26.5 % | 28.5 % | 25.9 % | 31.0 % |
| SM [10] | 6.0 % | 3.7 % | 8.4 % | 13.6 % | 8.5 % | 18.8 % |
| CS [9] | 4.9 % | 10.0 % | -0.1 % | 18.7 % | 25.3 % | 11.8 % |

manifold-preserving matting [3] which we implemented ourselves. Figure (c,d,e) show that using only one form of information flow is not effective in a number of scenarios such as wide unknown regions or holes in the foreground object. The strategy DCNN matting [13] follows is using the results of closed-form and KNN matting directly rather than formulating a combined energy using their affinity definitions. When both methods fail, the resulting combination also suffers from the errors as it is apparent in the pineapple and troll examples. The neural network they propose also seems to produce mattes that appear slightly blurred. LNSP matting [16], on the other hand, has issues around regions with holes (pineapple example) or when the foreground and background colors are similar (donkey and troll examples). It can also oversmooth some regions if the true foreground colors are missing in the trimap (plastic bag example). Our method performs well in these challenging scenarios mostly because, as detailed in Section 3, we carefully de-

fine intra-unknown region and unknown-to-known region connections which results in a more robust linear system.

6.2 Matte regularization

We also compare the proposed post-processing method detailed in Section 4 with the state-of-the-art method by Gastal and Oliveira [10] on the training dataset provided by Rheemann et al. [15]. We computed the non-smooth alpha values and confidences using the publicly available source code for comprehensive sampling [9], KL-divergence sampling [8] and shared matting [10]. Table 3 shows the percentage improvement we achieve over Gastal and Oliveira [10] for each...
6.3 Layer color estimation

We evaluate our layer color estimation against state-of-the-art methods by Levin et al. [1] and Chen et al. [2], which will be referred as closed-form colors and KNN colors respectively, qualitatively using two types of alpha values. In the first experiment, we use the ground-truth alphas available for the training dataset by Rhemann et al. [15] as an input to the color estimation methods to compare them in ideal conditions. We also use the alphas estimated by the proposed matte estimation method as well as two state-of-the-art methods in learning-based and sampling-based matting in order to test the color estimation algorithms in a more realistic application scenario.

It is hard to evaluate the color estimation methods nu-
merically in the case of imperfect input alpha values, as the ground-truth colors can only be defined for ground-truth alphas. Moreover, the performance of the colors is further hidden in the details and the most intricate structures. For these reasons, we provide many close-ups in Figures 10-11 and invite the reader to examine them in the digital version.

We observe several characteristic errors when we examine the closed-form and KNN colors. As closed-form color estimation uses only a local flow, we see loss of color in remote regions as seen in Figure 10(1-a) middle inset, (2-a) and (3-a) left inset. KNN colors, on the other hand, may fail to correctly find highlights (Figure 10(3-c) right inset, Figure 11(f,i)), find correct colors for remote isolated regions (Figure 10(1-b) bottom inset, (2-b)), successfully unmix the background color (Figure 10(1-b) top inset, (2-b), Figure 11(f)) or create regions with flat colors (Figure 11(c)).

The proposed color estimation algorithm is able to extract layer colors without these problems thanks to our multi-information flow formulation, which results in a better distribution of information in the transition regions.

6.4 Green-screen keying

Green-screen keying is a more constrained version of the natural image matting problem in which the background is mostly of single color. Despite the more constrained setup, it is challenging to get clean foregrounds for compositing. Aksoy et al. [18] show that common natural matting algorithms fail to get satisfactory results despite their performance on the matting benchmark.

We compare the performance of our method to that of the interactive green-screen keying method by Aksoy et al. [18] (GSK) and unmixing-based soft color segmentation [22] (SCS) as well as KNN matting [2] and comprehensive sampling [9] in Figure 12. GSK requires local color models, a subset of entries in their color model, and SCS requires a binary map to clean the noise in the background. The matting methods including ours require trimaps and we show results for two trimaps used for comparisons in [18]. We computed the foreground colors for our method and comprehensive sampling using our color estimation method, and KNN colors for KNN matting. We observed that the choice of color estimation method does not change the typical artifacts we see in KNN matting and comprehensive sampling. GSK and SCS compute foreground colors together with the alpha values.

Top example in Figure 12 shows that KNN matting overestimates alpha values in critical areas and this results in a green halo around the foreground. In contrast, we see a reddish hue in the hair and around the glasses for comprehensive sampling. This is due to the underestimation of alpha values in those areas. The bottom example shows that both competing matting methods fail to get rid of the color spill, i.e., indirect illumination from the background. The proposed method successfully extracts the foreground matte and colors in both challenging cases and gives comparable results to the state-of-the-art in green-screen keying. It can also be seen that the effect of different trimaps is minimal in both cases. A successful matting approach requires less input than GSK (the local color models are conceptually similar to a multi-channel trimap and requires more time to generate than a trimap) and is robust against color spill unlike SCS, which makes our method a viable option for green-screen keying.

7 Spectral analysis

The spectral clusters formed by Laplacians of affinity matrices can be effectively used to reveal characteristics of affinity matrices that define the connections between pixels. For instance, Levin et al. [1] analyze the matting affinity by looking at eigenvectors corresponding to the smallest eigenvalues of the matting Laplacian. Spectral matting [23] uses the eigenvectors together with a sparsity prior to create a set of soft segments, or alpha components, that represent compact clusters of eigenvectors and add up to one for each pixel.
The alpha components provide a more distilled and clear visualization of the structure of the affinity matrix. In this section, we will use the matting components computed using different subsets of information flows we make use of in our alpha estimation algorithm to reveal the contribution of different flows at a higher level.

We compute the alpha components shown in Figure 13 using the public source code by Levin et al. [23]. We exclude the \( K \)-to-\( U \) flow, which is only defined for the unknown regions as it requires explicitly defined known regions. The resulting Laplacian matrix does not give meaningful spectral clustering because of the pixels with missing connections. We overcome this issue for intra-\( U \) flow by defining it for the entire image instead of only the unknown region.

In our matting formulation, we use the color-mixture flow to create the main source of information flow between close-by similarly-colored pixels. This approach creates densely connected graphs as both spatial and color distances are well accounted for in the neighborhood selection. We observed that spectral matting may fail to create as many components as requested (10 in our experiments) in some images, as many regions are heavily interconnected. Using the weighted average of neighboring colors for the flow creates soft transitions between regions.

The intra-\( U \) flow connects pixels that have similar colors, with very little emphasis on the spatial distance. This creates a color-based segmentation of the pixels, but as we compute the weights based on the feature distances, it is not typically able to create soft transitions between regions. Rather, it creates components with alpha values at zero or one, or flat alpha regions with alpha values near 0.5.

The local information flow, used as the only form of flow in the original spectral matting, creates locally connected components with soft transitions, as expected.

We observed a harmonious combination of positive aspects of these affinity matrices as they are put together to create our graph structure. This provides a nice confirmation of our findings in the evaluation of our algorithm. The examples we present in the remainder of this section demonstrate our comments on the affinity matrix characteristics.

The top example in Figure 13 shows an input image with the matting components that include the green and the pink hair. Color-mixture affinities give components that demonstrate the color similarity and soft transitions, but they typically bleed out of the confined regions of specific colors due to the densely connected nature of the graph formed by corresponding neighborhoods. We clearly see the emphasis on color similarity for intra-\( U \) flow. While the color clusters are apparent, one can easily observe that unrelated pixels get mixed into the clusters especially around transition regions between other colors. We see a significant improvement already when these two flows are combined. When the local information flow is added, which gives spatially confined clusters of many colors when used individually, we see smooth clusters of homogeneous colors. The intricate transitions that were missed in the lack of the local flow are successfully captured when all three flows are included in the Laplacian definition.

The spatial connectivity versus color similarity characteristics are even more clearly observable in the bottom example of Figure 13. We see that bright and dark brown of the fur is clearly separated by intra-\( U \) flow in this example. In contrast, color-mixture and local flows separate the fur into three spatial clusters and the sweater into two separate clusters despite the uniform color. The combination, however, is able to successfully separate the dark and bright brown of the fur with smooth transitions.

The full Laplacian matrix we propose in this work blends the nonlocality of colors and spatial smoothness naturally. This is the key characteristic of the proposed matting method. When combined with \( K \)-to-\( U \) flow which addresses remote regions and holes inside the foreground, the proposed algorithm is able to achieve high performance in a variety of images as analyzed in Section 6.
8 Sampling-based methods and $\mathcal{K}$-to-$U$ flow

The $\mathcal{K}$-to-$U$ flow introduced in Section 3.2 connects every pixel in the unknown region directly to several pixels in both foreground and background. While the amount of flow from each neighbor is individually defined by the computed color-mixture weights, we simplify the formulation and increase the sparsity of our linear system using some algebraic manipulations. These manipulations, in the end, give us the weights $w_p^F$ that go into the final energy formulation.

These weights, which show the connection of the unknown pixel to the foreground, are essentially an early estimation of the matte. This estimation is done by individually selecting a set of neighbors for each pixel and computing an alpha based on the neighbor colors. While our approach is fundamentally defining affinities, it has parallels with sampling-based approaches in natural matting [7], [8], [9], [24], which also select samples from foreground and background and estimates alpha values based on sample colors. We computationally define values for $w_p^F$ that depend on the similarity of colors of neighbors from the foreground and background. Sampling-based approaches also define confidence values for their initial estimation, typically defined by the compositing error, $||e - (\alpha f - (1 - \alpha)b)||^2$.

Conceptually, there are several fundamental differences between our computation of $\mathcal{K}$-to-$U$ flow and common strategy followed by sampling-based methods. The major difference is how the samples are collected. Sampling-based methods first determine a set of samples collected from known-alpha regions and do a selection for unknown pixels from this determined set using a set of heuristics. We, on the other hand, select neighbors for each unknown pixel individually via a k nearest neighbors search in the whole known region. Using the samples, state-of-the-art methods typically use the compositing equation to estimate the alpha value from only one sample pair (a notable exception is CSC matting [7]), while we use 14 samples in total to estimate the alpha by solving the overconstrained system using the method by Roweis and Saul [5]. These differences also change the computation time. $\mathcal{K}$-to-$U$ flow can be computed in several seconds, while sampling-based algorithms typically take several minutes per image due to sampling and sample pair selection steps.

In order to compare the performance of $\mathcal{K}$-to-$U$ flow as a sampling-based method in a neutral setting, in this experiment, we post-process $w_p^F$ and our confidence values using the common regularization step [10] utilized by top-performing sampling-based methods in the benchmark. The quantitative results can be seen in Table 4.

![Fig. 14. Our method fails gracefully in the case of sparse trimaps.](image)

As discussed in Section 3.2, $\mathcal{K}$-to-$U$ flow fails in the case of a highly-transparent matte (net and plastic bag examples). This is due to the failure to find representative neighbors using the k nearest neighbor search. Sampling-based methods are more successful in these cases due to their use of compositing error in the sample selection. However, in the other examples, $\mathcal{K}$-to-$U$ flow appears as the top-performing method among the sampling-based methods in 12 of 18 image-trimap pairs and gives comparable errors in the rest.

The performance of our affinity-inspired approach against the state-of-the-art [7], [8], [9], [24] gives us some pointers for a next-generation sampling-based matting method. While one can argue that the sampling algorithms have reached enough sophistication, selection of a single pair of samples for each unknown pixel seems to be a limiting factor. Methods that address the successful and efficient selection of many samples for each unknown pixel will be more likely to surpass state-of-the-art performance. Furthermore, determining the alpha values using more robust weight estimation formulations such as [1] instead of the more simple compositing equation [23] will likely improve the result quality.

9 Limitations

As discussed in corresponding sections, the $\mathcal{K}$-to-$U$ flow does not perform well in the case of highly-transparent mattes. We solve this issue via a simple classifier to detect highly-transparent mattes before alpha estimation. However, this does not solve the issue for foreground images that partially have transparent regions. For such cases, a local classifier or a locally changing set of parameters could be the solution.

The proposed matte estimation algorithm assumes dense trimaps as input. In the case of sparse trimaps, generally referred as scribble input, our method may fail to achieve its original performance, as seen in Figure 14. This performance drop is mainly due to the $\mathcal{K}$-to-$U$ flow, which fails to find good neighbors in limited foreground and background regions, and intra-$U$ flow which propagates alpha information based solely on color to spatially far away pixels inside the unknown region.

### Table 4

| Method       | Troll S | Doll S | Donkey S | Elephant S | Plant S | Pineapple S | Plastic bag S | Net S |
|--------------|--------|--------|----------|------------|---------|-------------|---------------|-------|
| CSC          | 13.6   | 15.6   | 14.5     | 6.2        | 7.5     | 8.1         | 4.6           | 4.8   |
| Sparse coding | 12.6   | 20.5   | 14.8     | 5.7        | 7.3     | 6.4         | 4.5           | 5.3   |
| KL-Div       | 11.6   | 17.5   | 14.7     | 5.6        | 8.5     | 8.0         | 4.9           | 5.3   |
| $\mathcal{K}$-to-$U$ flow | 12.0   | 13.1   | 14.6     | 7.5        | 9.1     | 8.9         | 3.9           | 4.3   |
| Comp. S.     | 11.2   | 18.5   | 14.8     | 6.5        | 9.5     | 8.9         | 4.5           | 4.9   |

SAD scores of top sampling-based methods on the matting benchmark against the $\mathcal{K}$-to-$U$ flow as a sampling based method, regularized by [10]. Blue shows the best performance among the methods listed here for each image-trimap pair. Red marks the failure cases for the $\mathcal{K}$-to-$U$ flow.
10 Conclusion

In this paper, we proposed a purely affinity-based natural image matting method. We introduced color-mixture flow, a specifically tailored form of LLE weights for natural image matting. By carefully designing flow of information from the known region to the unknown region, as well as distributing the information inside the unknown region, we addressed several challenges that are common in natural matting. We showed that the linear system we formulate outperforms the state-of-the-art in the alpha matting benchmark. The characteristic contributions of each form of information flow were discussed through spectral analysis. We extended our formulation to matte regularization and demonstrate their performance improvements over the state-of-the-art. We demonstrated that the proposed matting and color estimation methods achieve state-of-the-art performance in green-screen keying. We also commented on several shortcomings of the state-of-the-art sampling-based methods by comparing them to our known-to-unknown information flow.

References

[1] 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.
[2] Q. Chen, D. Li, and C.-K. Tang, “KNN matting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 9, pp. 2175–2188, 2013.
[3] X. Chen, D. Zou, Q. Zhao, and P. Tan, “Manifold preserving edit propagation,” ACM Trans. Graph., vol. 31, no. 6, pp. 132:1–132:7, 2012.
[4] Y. Aksoy, T. O. Aydın, and M. Pollefeys, “Designing effective inter-pixel information flow for natural image matting,” in Proc. CVPR, 2017.
[5] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” Science, vol. 290, no. 5500, pp. 2323–2326, 2000.
[6] Q. Zhu, L. Shao, X. Li, and L. Wang, “Targeting accurate object extraction from an image: A comprehensive study of natural image matting,” IEEE Trans. Neural Netw. Learn. Syst, vol. 26, no. 5, pp. 185–207, 2015.
[7] X. Feng, X. Liang, and Z. Zhang, “A cluster sampling method for image matting via sparse coding,” in Proc. ECCV, 2016.
[8] L. Karacan, A. Erdem, and E. Erdem, “Image matting with KL-divergence based sparse sampling,” in Proc. ICCV, 2013.
[9] E. Shahrian, D. Rajan, B. Price, and S. Cohen, “Improving image matting using comprehensive sampling sets,” in Proc. CVPR, 2013.
[10] E. S. L. Gastal and M. M. Oliveira, “Shared sampling for real-time alpha matting,” Comput. Graph. Forum, vol. 29, no. 2, pp. 575–584, 2010.
[11] Y. Zheng and C. Kambhamettu, “Learning based digital matting,” in Proc. ICCV, 2009.
[12] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia, “Deep automatic portrait matting,” in Proc. ECCV, 2016.
[13] D. Cho, Y.-W. Tai, and I. S. Kweon, “Natural image matting using deep convolutional neural networks,” in Proc. ECCV, 2016.
[14] N. Xu, B. Price, S. Cohen, and T. Huang, “Deep image matting,” in Proc. CVPR, 2017.
[15] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, “A perceptually motivated online benchmark for image matting,” in Proc. CVPR, 2009.
[16] X. Chen, D. Zou, S. Zhou, Q. Zhao, and P. Tan, “Image matting with local and nonlocal smooth priors,” in Proc. CVPR, 2013.
[17] J. Wang and M. F. Cohen, “Optimized color sampling for robust matting,” in Proc. CVPR, 2007.
[18] Y. Aksoy, T. O. Aydın, M. Pollefeys, and A. Smolíč, “Interactive high-quality green-screen keying via color unmixing,” ACM Trans. Graph., vol. 35, no. 5, pp. 152:1–152:12, 2016.
[19] A. R. Smith and J. F. Blinn, “Blue screen matting,” ACM Trans. Graph., pp. 259–268, 1996.