A Global Sampling Method for Alpha Matting

Kaiming He[1], Christoph Rhemann[2]*, Carsten Rother[3], Xiaoou Tang[1], Jian Sun[4]
[1]The Chinese University of Hong Kong, [2]Vienna University of Technology,
[3]Microsoft Research Cambridge, [4]Microsoft Research Asia

Abstract

Alpha matting refers to the problem of softly extracting the foreground from an image. Given a trimap (specifying known foreground/background and unknown pixels), a straightforward way to compute the alpha value is to sample some known foreground and background colors for each unknown pixel. Existing sampling-based matting methods often collect samples near the unknown pixels only. They fail if good samples cannot be found nearby.

In this paper, we propose a global sampling method that uses all samples available in the image. Our global sample set avoids missing good samples. A simple but effective cost function is defined to tackle the ambiguity in the sample selection process. To handle the computational complexity introduced by the large number of samples, we pose the sampling task as a correspondence problem. The correspondence search is efficiently achieved by generalizing a randomized algorithm previously designed for patch matching[3]. A variety of experiments show that our global sampling method produces both visually and quantitatively high-quality matting results.

1. Introduction

Alpha matting refers to the problem of softly and accurately extracting the foreground from an image. Specifically, the input image $\mathbf{I}$ is modeled as a linear composite of a foreground image $\mathbf{F}$ and a background image $\mathbf{B}$:

$$\mathbf{I} = \mathbf{F}\alpha + \mathbf{B}(1-\alpha), \tag{1}$$

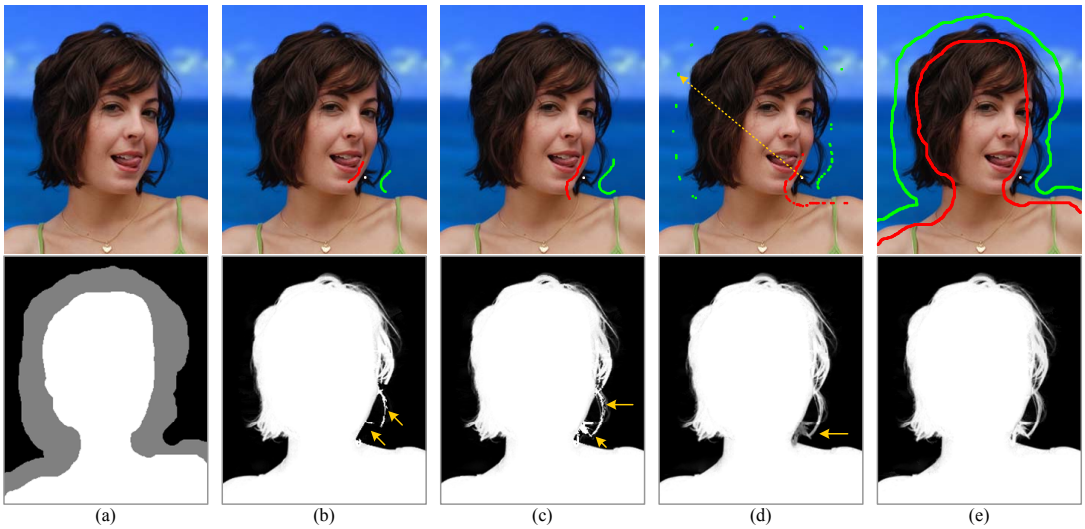
where α, called *alpha matte*, is the opacity of the foreground. Alpha matting plays an important role in many image/video editing tasks, like layer separation, background replacement, and foreground toning. But the matting problem is highly ill-posed, because the number of unknowns ($\mathbf{F}$, $\mathbf{B}$, and α) is much larger than the number of equations. Therefore, a user-specified trimap which indicates the known foreground/background and the unknown pixels is often required.

Existing matting methods can be categorized as propagation-based or sampling-based. Propagation-based methods treat the problem as interpolating the unknown alpha values from the known regions. The interpolation can be done by solving an affinity matrix [17, 8, 11, 16, 9], by optimizing Markov Random Fields [18], or by computing geodesic distance [2]. These methods mainly rely on the image's continuity to estimate the alpha matte, and do not explicitly account for the foreground and background colors. They have shown success in many cases, but may fail when the foreground has long and thin structures or holes. Their performance can be improved when combined with sampling-based methods (*e.g.*, [19, 12, 7]).

Sampling-based methods first estimate the foreground and background colors and then compute the alpha matte. Earlier methods like Ruzon and Tomasi's work [15] and Bayesian Matting [6], fit a parametric model to the color distributions. But they are less valid when the image does not satisfy the model. Recent sampling-based methods [19, 12, 7] are mostly non-parametric: they pick out some color samples from the known regions to estimate the unknown alpha values. These methods perform well in condition that the true foreground and background colors are in the sample set. However, the true foreground/background colors are not always covered, because these methods only collect samples near each unknown pixel, and the number of samples is rather limited.

In order to avoid missing true samples, we propose a global sampling approach that considers all available samples. Conventional wisdom may hold that a large number of samples will increase both ambiguity and computational complexity for selecting good samples. To handle the ambiguity, we define a simple but effective cost function for sample selection. This function simultaneously considers the sample locations and how well the sample colors fit the matting equation (1). Regarding the complexity, we formulate the sample selection task as a correspondence search between the unknown pixels and an "FB search space" (defined in Sec. 3.2). Then we generalize a fast randomized algorithm [3] that was originally developed for patch matching, to our task. Experiments show that our global sampling

* This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT08-019.



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

Figure 1. Sample sets and sampling results of non-parametric methods. (a) Input image and trimap. (b) Robust Matting [19]. (c) Improved Color Matting [12]. (d) Shared Matting [7]. (e) Our method. Upper row: we label an unknown pixel in the hair and indicate its sample sets (red: $\mathbf{F}$; green: $\mathbf{B}$). The foreground sample sets of our competitors (b-d) do not cover any hair-color, so do not include a true foreground sample for the labeled unknown pixel. Lower row: resulting alpha mattes from sampling only (computed without any post-processing like solving the matting Laplacian [11]).

method provides both visually and quantitatively good results and has a time complexity comparable to the previous (local) sampling methods. Our generalized correspondence algorithm also has the potential in other computer vision areas.

2. Related Work

In the following, we review only non-parametric sampling methods, because they are most related to our work. Their basic assumption is that for any unknown pixel, the (approximately) true foreground and background colors can be sampled from the known regions. These methods differ in the way they build the sample set and how they select good samples from this set.

In an early system called *Knockout* [5], an unknown pixel's foreground color is a weighted average of its nearby foreground samples. The weight is determined by the spatial distance between the foreground sample and the unknown pixel. The background is estimated similarly and then refined. The sampling of Knockout depends only on the spatial distance.

Robust Matting [19] is the first non-parametric sampling method that takes into account the color fitness of samples. For any unknown pixel, this method first collects a set of spatially close (in Euclidean distance) foreground/background samples, from which good samples are selected in the next step. Good samples are defined as those whose color fits the matting equation (1) well. This color-based sample selection is more robust than the weighted averaging in Knockout. But since this method collects only a few nearby samples, it can fail when the true samples are not in the sample set, as shown in Fig. 1(b).

Rhemann *et al*. [12] collected samples nearby in geodesic distance, as opposed to the Euclidean distance used in [19]. This can be helpful for more complex object topologies. But similar to [19], the sample set is small and often locally distributed, so the true samples can still be missed (see Fig. 1(c)).

Shared Matting [7] collects samples by shooting rays in different directions: the samples are the nearest foreground and background pixels on every ray (see the dashed line in Fig. 1(d)). Each unknown pixel collects very few samples, but the samples are further shared among neighboring pixels. Thus, the actual sample set of an unknown pixel can be roughly considered as the union set of its own and its neighbors' samples. Fig. 1(d) gives an example of such a union set. Due to the ray-based sampling strategy, we find that the samples can be far away (often background samples), but mostly are nearby (often foreground samples). Thus the true samples can still be missed (Fig. 1(d)).

3. A Randomized Global Sampling Method

All existing non-parametric sampling methods collect nearby (in a certain metric, *e.g*., Euclidean/geodesic distance, or nearest on a ray) samples. Thus they only use a subset of samples, and preclude a substantial portion of samples from further consideration. We point out that the step of building the sample set plays a role as a preliminary sample selection. Even worse, only spatial distance but no color fitness is taken into account in this preliminary stage. To avoid

missing the true samples, we propose to use a *global sample set* that contains all available samples. The spatial distance and the color fitness are then considered simultaneously for selecting the good samples from this set.

Global sample set. Our foreground (background) sample set consists of all known foreground (background) pixels on the unknown region's boundary (Fig. 1(e)). We also tested even larger sample sets comprising of all known pixels no more than k-pixel ($k > 1$) away from this boundary, and to the extreme of all known pixels in the image. We found that using the pixels on the unknown region's boundary ($k = 1$) is sufficient to produce good results.

Denote the size of the foreground/background sample set as n_{F} and n_{B}. For an 800×600 image, the typical value of n_{F} and n_{B} is $10^3 \sim 10^4$. So there exist about $n_{\mathrm{F}} \times n_{\mathrm{B}} = 10^6 \sim 10^8$ candidate foreground/background pairs. Our goal is to select a good pair of samples for any unknown pixel from all candidate pairs. We will address two challenging issues caused by this huge sample set: how to define a "good" sample pair (Sec. 3.1) and how to handle the computational complexity (Sec. 3.2).

3.1. Sample Selection Criteria

Our criteria for sample selection are based on both color fitness and spatial distance. Like previous methods, we favor sample pairs that can well explain the unknown pixel's color as the sample colors' linear combination (1). Formally, given a pair of samples $(\mathbf{F}^i, \mathbf{B}^j)$ with the sample indexes i and j, we first estimate the alpha value $\hat{\alpha}$ of an unknown pixel $\mathbf{I}$ by:

$$\hat{\alpha} = \frac{(\mathbf{I} - \mathbf{B}^j)(\mathbf{F}^i - \mathbf{B}^j)}{\|\mathbf{F}^i - \mathbf{B}^j\|^2}. \tag{2}$$

Then we define a color cost function $\mathcal{E}_c$ to describe how good a sample pair fits the matting equation (1):

$$\mathcal{E}_{\mathrm{c}}(\mathbf{F}^i, \mathbf{B}^j) = \|\mathbf{I} - (\hat{\alpha}\mathbf{F}^i + (1 - \hat{\alpha})\mathbf{B}^j)\|. \tag{3}$$

Intuitively, (3) is the color distance (in RGB space) between $\mathbf{I}$ and the color line spanned by $\mathbf{F}^i$ and $\mathbf{B}^j$. A similar cost function was first introduced in [19] and also used in [7, 12].

Due to the large size of the sample set, it is possible that a false pair of samples occasionally well explains the unknown pixel's color. Therefore, we further define a spatial cost function $\mathcal{E}_s$ that favors spatially close samples:

$$\mathcal{E}_s(\mathbf{F}^i) = \frac{\|\mathbf{x}_{\mathbf{F}^i} - \mathbf{x}_{\mathbf{I}}\|}{D_F}, \tag{4}$$

where $\mathbf{x}_{\mathbf{F}^i}$ and $\mathbf{x}_{\mathbf{I}}$ are the spatial coordinates of the foreground sample and the unknown pixel. The term $D_F = \min_i \|\mathbf{x}_{\mathbf{F}^i} - \mathbf{x}_{\mathbf{I}}\|$ is the nearest distance of the unknown pixel to the foreground boundary. The normalization factor D_F in eq. (4) ensures that $\mathcal{E}_s$ is independent from the absolute distance. The spatial cost $\mathcal{E}_s(\mathbf{B}^j)$ of a background sample is defined similarly.

Our final sample selection cost function $\mathcal{E}$ is a combination of the color cost and the spatial cost:

$$\mathcal{E}(\mathbf{F}^i, \mathbf{B}^j) = w\mathcal{E}_c(\mathbf{F}^i, \mathbf{B}^j) + \mathcal{E}_s(\mathbf{F}^i) + \mathcal{E}_s(\mathbf{B}^j), \tag{5}$$

where w is a weight which trades off the color fitness and spatial distance. We set $w = 1$ when $\mathcal{E}_c$ is computed using colors scaled in [0, 255]. Intuitively, a small $\mathcal{E}$ indicates that the sample pair can well explain the unknown pixel's color and is spatially as close as possible. Unlike previous methods that preclude spatially distant samples from the sample set, our method may pick them out if they have sufficiently better color fitness than the nearer samples.

3.2. A Randomized Search Algorithm for Global Sampling

Given the cost function, our next goal is to find a sample pair for each unknown pixel, which has the smallest cost. This is a challenging task because of the huge number of sample pairs. A brute-force algorithm would check all sample pairs for each unknown pixel, so is O($Nn_{\mathrm{F}}n_{\mathrm{B}}$) time in the number of the unknown pixels N. In this subsection, we reformulate this search as a correspondence problem, and generalize the "PatchMatch" algorithm [3] to handle it.

In a patch-matching problem [3], the task is to find the best matching patch in an image J_2 for each patch in the other image J_1. It is an O(M_1M_2) time task, where M_1 and M_2 are the sizes of J_1 and J_2. The PatchMatch algorithm [3] gives a fast approximate solution in O($M_1 \log M_2$) time.

Inspired by this scenario, we introduce a 2-D $n_{\mathrm{F}} \times n_{\mathrm{B}}$ "**FB** search space" which plays the role like J_2. All the unknown pixels play as J_1. A "good match" is measured by a small cost according to eq. (5). By analogy, we can expect to obtain an O($N \log(n_{\mathrm{F}}n_{\mathrm{B}})$) time approximation algorithm. We name this algorithm *SampleMatch*. In the next subsections, we describe the algorithm and analyze its validity in our global sampling problem.

3.2.1 The SampleMatch Algorithm

We sort all the foreground samples $\mathbf{F}$ by a certain criterion, and denote them as an ordered set $\{\mathbf{F}^i | i = 1, 2, ..., n_{\mathrm{F}}\}$. The background set $\{\mathbf{B}^j | j = 1, 2, ..., n_{\mathrm{B}}\}$ is sorted similarly. Notice that every sample has both color and spatial coordinates. We tested several kinds of sorting criteria, e.g. color/intensity and/or spatial coordinates, and found that the choice does not make a big difference. We have chosen to sort by intensity in this paper. The two ordered sets span a 2-D **FB** search space, in which each point represents a sample pair $(\mathbf{F}^i, \mathbf{B}^j)$, as illustrated in Fig. 2. For each unknown pixel, our target is to find a point in the **FB** search space which has the (approximately) smallest cost. Having defined the

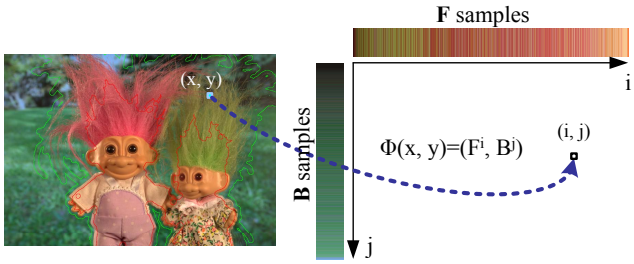


Figure 2. **FB** search space. For each unknown pixel, our target is to find a sample pair $(\mathbf{F}^i, \mathbf{B}^j)$ with the (approximate) smallest cost. We formulate it as a search problem in the 2-D **FB** space.

FB search space, our SampleMatch algorithm works analogously to PatchMatch. The algorithm is described below.

The SampleMatch algorithm maintains a current optimal (smallest cost) sample pair $(\mathbf{F}^i, \mathbf{B}^j)$ for each unknown pixel $\mathbf{I}(x, y)$. We denote this current optimal sample pair by $\Phi(x, y) = (\mathbf{F}^i, \mathbf{B}^j)$. We initialize $\Phi(x, y)$ by a random point in the **FB** search space. $\Phi(x, y)$ is updated while the algorithm runs, which we explain in the following.

After the initialization, the algorithm alternates between two steps: *propagation* and *random search*. Denote these two steps as P^k and S^k for each pixel $\mathbf{I}_k$. Then the algorithm scans the image and runs in the order (P^1, S^1, P^2, S^2,...,P^n, S^n). This process is iterated.

Propagation. For the unknown pixel $\mathbf{I}(x, y)$ being scanned, we update its $\Phi(x, y)$ by considering the current optimal sample pairs $\Phi(x', y')$ of its neighboring pixels (x', y'):

$$\Phi(x, y) \leftarrow \arg\min_{\Phi(x',y')} \mathcal{E}(\Phi(x', y')) \tag{6}$$

where $\mathcal{E}$ is defined in (5), (x', y') is in the first order neighborhood of (x, y) (including (x, y)), and '$\leftarrow$' is an assignment operator.

Intuitively, eq. (6) updates $\Phi(x, y)$ by a new sample pair $\Phi(x', y')$, if this pair gives a smaller cost. The propagation exploits that neighboring pixels tend to have similar foreground/background colors: if a pixel has found a good sample pair, this pair is also likely to be good for its neighboring pixels.

Random Search. For the unknown pixel $\mathbf{I}(x, y)$ being scanned, we update its $\Phi(x, y)$ by a sequence of random trials: $\{(\mathbf{F}^{i_k}, \mathbf{B}^{j_k}) | k = 0, 1, 2...\}$. We generate this sequence by:

$$(i_k, j_k) = (i, j) + \omega\beta^k \mathbf{R}_k, \tag{7}$$

where $\mathbf{R}_k$ is a uniform random number in $[-1, 1] \times [-1, 1]$, β^k is the k^{th} exponential of a ratio $\beta = 0.5$, and ω is the size of the search space. The sequence goes in the order of k=0, 1, 2... until the search window radius $\omega\beta^k$ is below 1. In this step, $\Phi(x, y)$ is updated if the new pair $(\mathbf{F}^{i_k}, \mathbf{B}^{j_k})$ has a smaller cost.

Intuitively, the random search step tests a sequence of random points (i_k, j_k) in a neighborhood (in the **FB** space)

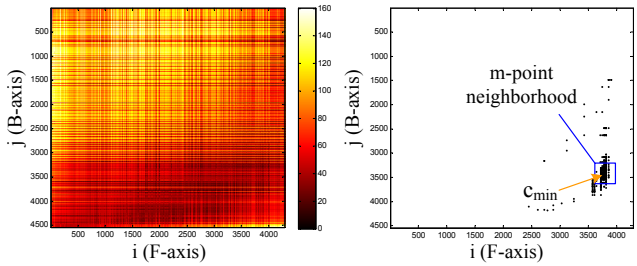


Figure 3. Left: a cost map in the search space ($n_F, n_B \approx 4000$). It is computed using an unknown pixel on the image in Fig. 2. Right: we marked the points (black) in this cost map that have values in $[c_{\min}, c_{\min} + \epsilon)$. The tolerance ϵ is set as 2 here. The rectangle is a m-point neighborhood near $c_{\min}$, in which a portion ρ(=0.1) are within the tolerance. See the text for details.

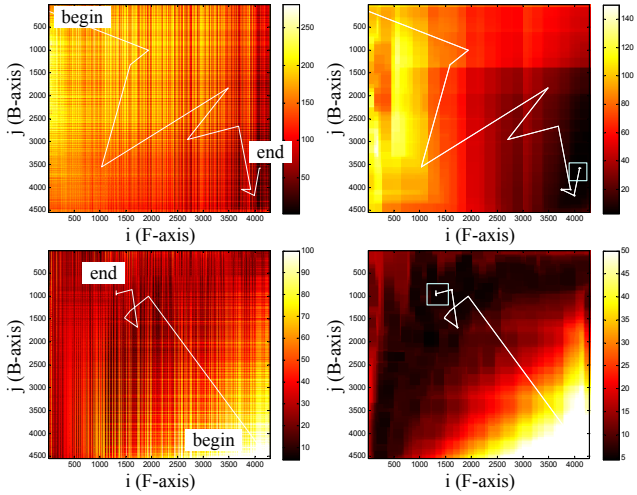


Figure 4. The random search process. Left: cost maps for two unknown pixels and their random search paths. Each line segment indicates one update of the current optimal Φ. On the right we display the same path on the min-filtered cost maps for better visualization. The rectangles denote the neighborhoods corresponding to some ϵ and ρ (for illustration only).

of the current optimal point (i, j). The neighborhood radius $\omega\beta^k$ decreases exponentially (in each iteration). Next, we explain why the random search can find an approximate global minimum.

3.2.2 Analysis of Random Search

Given any unknown pixel, a brute-force algorithm computes the cost values in eq. (5) for all sample pairs. These values generate an $n_F \times n_B$ cost map in the **FB** space (see Fig.3 left). Notice that each unknown pixel has its own cost map. As we will see, the random search step exploits the "local coherence" of this map, so avoids checking all points in this map.

By *local coherence* we mean that a point has many neighboring points (in a certain range) that have similar val-

ues with it. Formally, for a point with a value c in the cost map, given a tolerance ϵ and a fixed ratio ρ, we find a local region centered at c on the cost map, within which at least a portion ρ of the points have values in $(c-\epsilon, c+\epsilon)$. We denote m as the size of the local region that satisfies this condition. A larger m means a better local coherence. See Fig.3 for an example where c is the global minimum $c_{\min}$.

Note that locally coherent does not mean continuous (i.e. 1st-order neighbors have similar values). In fact, we found that the cost map from the real image is often not continuous (Fig.3 left) but still exhibits local coherence. This is because the foreground (background) samples that are similar in color are located nearby in the search space (due to the sample sorting), and so the color costs are locally coherent. Samples that have similar colors tend to have similar spatial coordinates (though not necessarily), so the spatial costs are also locally coherent. Hence, the entire cost map is locally coherent. We observed similar coherence if we would sort according to spatial proximity.

There is a good chance of finding a point inside the m-point region near the global minimum. Denote the size of the search space by $M = n_{\mathrm{F}} \times n_{\mathrm{B}}$. Suppose the random search uses the entire search space as search window. The probability of generating at least one point inside the m-point neighborhood is m/M for one random trial, and $1-(1-m/M)^n$ for n trials. This probability is quite good: *e.g.*, when $m/M = 10^{-2}$, the odds is 0.95 after $n = 300$ trials. Suppose the random search has fallen into the neighborhood m. Then it can use a small search window to examine the points inside m. It has a high probability of finding a point having a value within the tolerance (*i.e.*, in $[c_{\min}, c_{\min}+\epsilon)$) in some random trials. The probability depends on the ratio ρ.

In practice, the local region m is unknown to the algorithm, because the entire cost map is not explicitly computed. So the random search adopts a gradually decreasing search window and this process is iterated. When the search window is large, it plays the role as determining the range of the local region; when the search window is small, it plays as searching inside the local region.

To see the effect of the random search, we omit the propagation step and only run the random search iteratively for a single unknown pixel (100 iterations in this experiment). In Fig.4, we show the updating process of the random search (visualized by a path in the cost map). For a better visualization, we also display the same path on the cost map processed by a min filter[1] (but the algorithm is run in the original cost map). We found that the search path is gradually getting close to the global minimum.

The above experiment provides an intuitive view of how the random search approaches the global minimum. Next, we discuss the halting criteria and study the quantitative performance of how well it approximates the global minimum.

[1] Points in the cost map are replaced by the minimum in a local window.

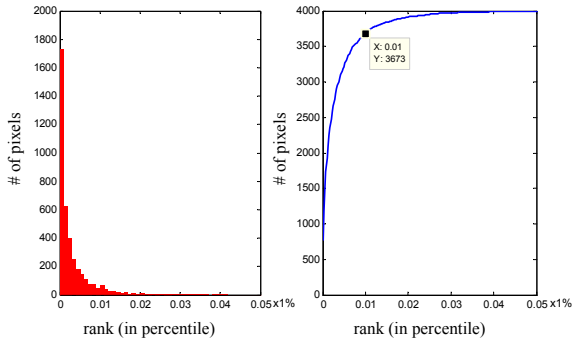


Figure 5. Performance of the SampleMatch algorithm approaching the global minimum. Left: distribution (histogram) of the cost ranks. The bin width is 0.001%. Right: cumulative distribution of the cost ranks. Please see the text for details.

3.2.3 Halting Criteria and Convergence

When running random search for a single pixel, the number of iterations required to approach the global minimum is not very small. Fortunately, the propagation step gives a better starting point for the random search step by first checking the neighboring pixels. This is helpful to reduce the iterations required.

Since we do not know the global minimum and the whole cost map, we halt the algorithm after a fixed number of iterations. We use 10 iterations, which we found to be sufficient in our experiments. Thus, the complexity per unknown pixel is $\mathrm{O}(10\log(n_{\mathrm{F}}n_{\mathrm{B}}))$, in terms of visited sample pairs. This complexity number is about 200 for $n_{\mathrm{F}} = n_{\mathrm{B}} = 10^3$. For comparison, this number is 400 for Robust Matting [19], 225 for Improved Color Matting [12], and 216 for Shared Matting [7], as reported in these papers.

We now test how good the SampleMatch algorithm approximates the global minimum. We run our algorithm on the benchmark images in [13] using the above halting criteria. The algorithm selects a sample pair for each unknown pixel. Then we compute the cost value c of this sample pair via eq. (5). This value is the approximate optimum given by the algorithm. Next we compute the entire cost map for each unknown pixel (in a brute-force manner). We sort all the cost values of this cost map in an ascending order, and find the rank (given in percentile) of the cost value c in these values. We randomly pick out 4,000 unknown pixels and investigate their ranks. Fig.5 shows the distribution of the ranks. We find that about 90% $(3673/4000)$ of the pixels have a cost value that is among the smallest 0.01% cost values in all candidate pairs (i.e. that ranks higher than 0.01%). This shows that we can find a quite good approximation of the global minimum.

3.2.4 A Probabilistic View

Many papers [14, 18] argue that for a large sample set it is important to represent the frequency or probability of the samples. A less frequent sample in the set is less likely to explain the unknown pixels. But existing methods using parametric probabilistic models like GMMs [6, 18] do not produce satisfactory quality (see [1]), whereas the recent non-parametric methods[19, 12, 7] do not consider the probability or frequency of samples.

Unlike the previous non-parametric sampling approaches, our method implicitly takes probability into account. As discussed in Sec. 3.2.2, our algorithm tends to find an approximate global minimum point when this point has sufficient similar points in its neighborhood in the search space. Since these points also have very small costs, the unknown pixel can also be well explained by any of them. On the other hand, a false sample pair that occasionally has the smallest cost (e.g. due to noise) is less possible to be picked out by our algorithm, if it is an isolated outlier.

3.2.5 Generalization of the PatchMatch Algorithm

We generalize the PatchMatch algorithm [3] by introducing the "search space" concept. In this viewpoint, the dimensions of the search space in [3] are simply the image coordinates of the patches. A recent extension of PatchMatch [4] takes into account patch rotation and scaling. Though a "search space" is not explicitly defined in their paper, this extension is searching in a higher-dimensional search space consisting of the sample patches' location, rotation, and scales,

A similar randomized search algorithm is possible for other features besides patches and samples. We believe that this concept can be leveraged to other computer vision applications.

3.3. Post-processing

Given a good pair of foreground and background samples, we can estimate the alpha value of each pixel independently using equation (2). In practice, applying a kind of post-processing (smoothing) by considering neighboring pixels can further improve the matting results. In this paper, we adopt two kinds of post-processing methods:

The first method is to solve a global optimization problem, as in [19, 12]. The $\hat{\alpha}$ in (2) is used as a data term, and the smoothness term is the matting Laplacian matrix L [11]. Specifically, the final alpha is computed by:

$$\alpha = \arg\min \alpha^{\mathrm{T}} \mathrm{L} \alpha + \lambda (\alpha - \hat{\alpha})^{\mathrm{T}} \mathrm{D} (\alpha - \hat{\alpha}), \qquad (8)$$

where λ is a weighting parameter and D is diagonal matrix. Its diagonal element is a large constant for the known pixel, and a confidence f for the unknown pixel. We set the confidence $f = \exp(-\mathcal{E}_c / 2\sigma^2)$, where $\mathcal{E}_c$ is defined in (3) and $\sigma = 1.0$. The solution to (8) can be obtained by solving a linear system [11].

The above method is relatively slow, taking several seconds or even minutes to compute. Alternatively, we can adopt the fast *guided filter* proposed in [10], which has been proven to be a good approximation of solving the matting Laplacian. The runtime of the filter is about 0.3 seconds per mega-pixel. In experiments, we found that our sampling results are often visually acceptable, so applying a local filter is sufficient in many cases.

4. Experimental Results

In the following, we compare our "Global Sampling Matting" approach with three other methods: Robust Matting [19], Improved Color Matting [12] and Shared Matting [7]. The results of [12] were provided by the authors, and the algorithms in [19, 7] were re-implemented by ourselves. For a fair comparison, we turn off any pre-processing steps (e.g. trimap expansion step in [7]) used by competing methods. Hence, the methods differ only in their sampling strategy. The only exception is Fig. 12, where we compare the original full implementations of all methods.

Comparisons of the Sampling Quality. To evaluate the sampling performance of various non-parametric methods, we compare the results of the sampling step alone (without any post-processing). This means that we calculate alpha by eq. (2), using the best sample pairs picked out by the different methods. To the best of our knowledge, this is the first comparison conducted purely on sampling results in the literature. We hope that this comparison gives a better understanding of non-parametric sampling methods.

Fig.1, Fig.6 (2nd row), Fig.7, and Fig.8 visually compare the sampling results. We also conducted a quantitative comparison on a benchmark data set [13] of 27 images with ground truth. The average SAD (sum of absolute difference) error of each method is plotted in Fig.9 (left). We also compare the accuracy of the foreground sample colors picked out by the different methods, using the ground truth foreground images provided by [13]. Since the ground truth foreground color is noisy in regions where alpha is close to zero, we compare the product $\mathbf{F}\alpha$ in Fig.9 (right).

Our experiments show that our method is performing the best. Shared Matting gives a comparable performance, while Robust Matting and Improved Color Matting perform less well. This is because Robust Matting and Improved Color Matting use only very locally distributed color samples. In contrast, our method and Shared Matting use considerably larger sample sets (Fig.6), thus can provide better results. However, the sample set of Shared Matting heavily depends on the trimap topology. This is due to its ray-based sampling strategy. So Shared Matting may still miss the true foreground samples (Fig.1 and Fig.7) or the true background samples (Fig.8). Though Shared Matting uses

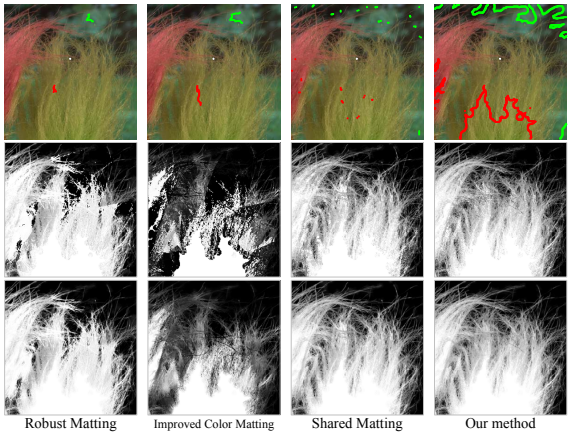


Figure 6. Visual comparisons of non-parametric sampling methods. 1st row: input image and the sample sets of an unknown pixel. 2nd row: the sampling results without post-processing. 3rd row: the results after solving the matting Laplacian.

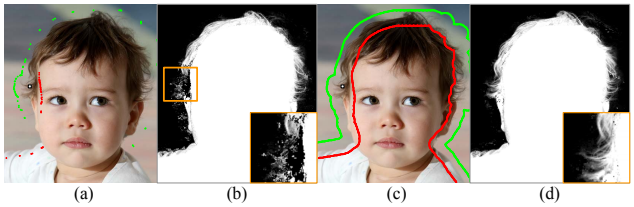


Figure 7. (a) The sample set of an example unknown pixel (marked as white) for Shared Matting. It does not cover any true foreground color. (b) The sampling result of Shared Matting. (c) The sample set of our method (the trimap boundary). (d) Our sampling result.

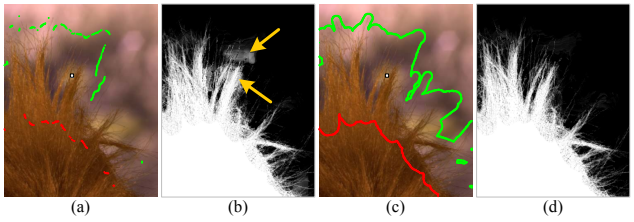


Figure 8. (a) The sample set of an example unknown pixel (marked as white) for Shared Matting. It does not cover any true background color. (b) The sampling result of Shared Matting. (c) The sample set of our method. (d) Our sampling result.

a complex cost function for sample selection, it is invalid if the sample set does not comprise the true samples. On the other hand, our method is less dependent on the trimap topology and can successfully find good samples by a simple cost function. Note that our cost function (5) has only one weighting parameter, while the cost function of Shared Matting has at least six parameters.

In summary, our main insight is that a large sample set is vital to achieve good results, while a simple cost function is sufficient to pick out the best sample pairs.

Comparisons of the Results after Post-processing. Next

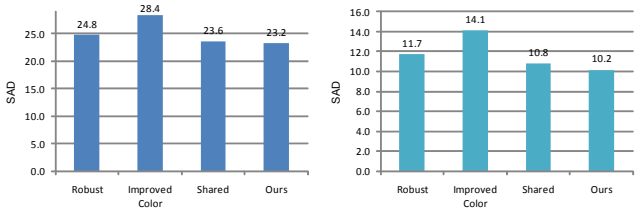


Figure 9. Quantitative comparisons on the sample results without post-processing. Left: average SAD error of α. Right: average SAD error of $\mathbf{F}\alpha$.

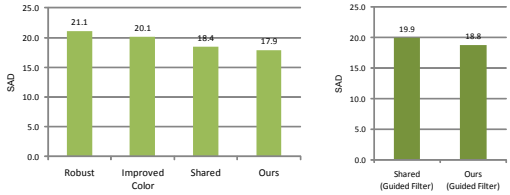


Figure 10. Quantitative comparisons on average SAD error of α, with post-processing. Left: post-processing via the matting Laplacian. Right: post-processing via guided filter.

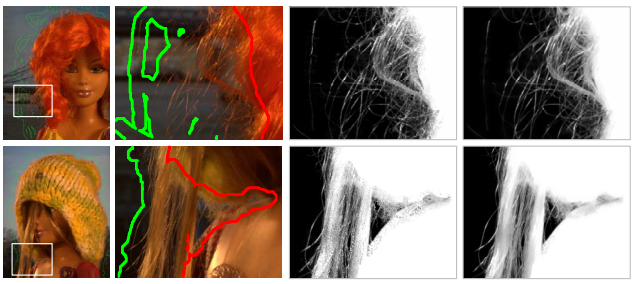


Figure 11. Our results using the guided filter as post-processing. From left to right: input, zoom-in, our sampling result, and our result processed by the guided filter.

we post-process (smooth) the sampling results and compare the quality.

When smoothing the results by solving the matting Laplacian, we set the weight λ in (8) to 0.1 for all methods. The confidence value f (defined in Sec. 3.3) is set as in the individual papers. Visual examples are in Fig.6 (3rd row) and a quantitative comparison on the 27 benchmark images in [13] is given in Fig.10 (left). Our method performs well.

We further compare the results using an alternative post-processing way: the guided filter [10] [2]. We compare the filtering-based results in Fig.10 (right). (Here, we only compare to the Shared Matting, because the other methods performed already worse when applying the full matting Laplacian). Because our sampling results are already quite accurate, our results processed by the filter have good visual quality (see also Fig.11).

Fig.12 shows a comparison on the independent matting benchmark of [1]. It reports the performance of the origi-

[2]Note that the Shared Matting proposed another filter-like "local smoothing" step. But we found it performs less well than the guided filter.

Avg. Rank (SAD)	
Shared	3.4
Global (Laplacian)	**4.3**
Segmentation-based	4.6
Improved Color	5.1
Global (filter)	**5.3**
Shared (Real Time)	6.1
Learning Based	6.2
Closed-Form	6.3
Large Kernel	8.0
Robust	8.9

Avg. Rank (MSE)	
Global (Laplacian)	**3.6**
Shared	3.9
Segmentation-based	4.7
Improved Color	4.8
Learning Based	6.3
Global (filter)	**6.5**
Closed-Form	6.6
Shared (Real Time)	6.9
Large Kernel	7.3
Robust	8.3

Figure 12. Average ranks w.r.t. SAD and MSE on the AlphaMatting evaluation website [1]. Our method, using the matting Laplacian as post-processing, is marked as red. The other non-parametric sampling methods are marked as green. Only the top 10 methods are displayed due to the limited space.

nal full implementation of all methods. The results of our method are denoted as "Global (Laplacian)" and "Global (filter)", based on the post-processing methods. The result of "Global (Laplacian)" ranks 2nd in SAD and 1st in MSE (mean squared error) out of 15 matting methods at the time of submission. Thus our performance is comparable to the original implementation of Shared Matting. Notice that the Shared Matting approach additionally uses pre-processing heuristics (*e.g.* trimap expansion) and requires a fair amount of parameters to be tuned. This can easily overfit the small test dataset. In contrast, our method has much fewer parameters and uses no pre-processing steps.

Moreover, the result of "Global (filter)" ranks 5th and 6th w.r.t. SAD and MSE. All methods ranked higher than our filter-based method solve a matting Laplacian, making them very slow. Hence our filter-based method is the highest ranked fast method. In comparison, the fast filtering-based Shared Matting (Shared Matting (Real Time)) ranks 6th in SAD and 8th in MSE.

Running Time. We now compare the runtimes of our method with those of Shared Matting (the other two methods are quite slow, so we do not take them into account). We run our C++ implementation on an Intel Pentium D 3.2GHz CPU with 2GB of memory. The total runtime on the 27 benchmark images in [13] is 170 seconds for our method (sampling only). It is slightly faster than our CPU implementation of Shared Matting (sampling only), which takes 220 seconds. This is mainly because our cost function (5) is much simpler than the one of Shared Matting. A GPU implementation of the Shared Matting can achieve real-time performance as reported in [7]. We believe that our method has potential to achieve similar performance, since the original PatchMatch algorithm [3] has been efficiently implemented on the GPU.

Limitation. Our method may fail when the sampling selection criteria (see eq.(5)) are not sufficient to resolve the color ambiguity. This may happen when an unknown pixel can be well explained as a linear combination of two false foreground/background color clusters.

5. Discussion and Conclusion

In this paper, we proposed a global sampling method for image matting. Using a simple cost function and a SampleMatch algorithm to handle the large sample set, our method is able to efficiently produce high quality results.

To the best of our knowledge this is the first attempt to generalize the (previously image-based) search space of the PatchMatch algorithm. We hope this generalization can be utilized in more computer vision applications matching other kinds of features.

References

[1] http://www.alphamatting.com.

[2] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. *ICCV*, 2007.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *SIGGRAPH*, 2009.

[4] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. *ECCV*, 2010.

[5] A. Berman, A. Dadourian, and P. Vlahos. Method for removing from an image the background surrounding a selected object. *U.S. Patent 6,134,346*, 2000.

[6] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. *CVPR*, 2001.

[7] E. S. L. Gastal and M. M. Oliveira. Shared sampling for real-time alpha matting. *Eurographics*, 2010.

[8] L. Grady, T. Schiwietz, and S. Aharon. Random walks for interactive alpha-matting. *VIIP*, 2005.

[9] K. He, J. Sun, and X. Tang. Fast matting using large kernel matting laplacian matrices. *CVPR*, 2010.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. *ECCV*, 2010.

[11] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *CVPR*, 2006.

[12] C. Rhemann, C. Rother, and M. Gelautz. Improving color modeling for alpha matting. *BMVC*, 2008.

[13] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. *CVPR*, 2009.

[14] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[15] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. *CVPR*, 2000.

[16] D. Singaraju, C. Rother, and C. Rhemann. New appearance models for natural image matting. *CVPR*, 2009.

[17] J. Sun, J. Jia, C. Tang, and H. Shum. Poisson matting. *SIGGRAPH*, 2004.

[18] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. *ICCV*, 2005.

[19] J. Wang and M. Cohen. Optimized color sampling for robust matting. *CVPR*, 2007.