DOI: 10.1111/j.1467-8659.2009.01627.x

EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Shared Sampling for Real-Time Alpha Matting

Eduardo S. L. Gastal[1] and Manuel M. Oliveira[1,2]

[1]Instituto de Informática, UFRGS [2]Camera Culture Group, MIT Media Lab

Abstract

Image matting aims at extracting foreground elements from an image by means of color and opacity (alpha) estimation. While a lot of progress has been made in recent years on improving the accuracy of matting techniques, one common problem persisted: the low speed of matte computation. We present the first real-time matting technique for natural images and videos. Our technique is based on the observation that, for small neighborhoods, pixels tend to share similar attributes. Therefore, independently treating each pixel in the unknown regions of a trimap results in a lot of redundant work. We show how this computation can be significantly and safely reduced by means of a careful selection of pairs of background and foreground samples. Our technique achieves speedups of up to two orders of magnitude compared to previous ones, while producing high-quality alpha mattes. The quality of our results has been verified through an independent benchmark. The speed of our technique enables, for the first time, real-time alpha matting of videos, and has the potential to enable a new class of exciting applications.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classification

1. Introduction

Extraction and compositing of foreground objects are fundamental image and video editing operations. The process of digital matting aims at accurately extracting foreground objects from images and videos. For this, matting techniques need to estimate foreground (F) and background (B) colors for all pixels belonging to an image I, along with opacity (α) values. These values are related by the *compositing Equation* 1, where the observed color C_p of pixel p is expressed as a linear combination of F_p and B_p, with interpolation parameter α_p:

$$C_p = \alpha_p F_p + (1-\alpha_p) B_p. \tag{1}$$

For *natural images*, F and B are not constrained to a particular subset of values. Thus, all variables on the right-hand side of Equation 1 are unknown, making the problem of computing an alpha matte considerably harder.

Due to the highly ill-posed nature of the matting problem, most existing approaches require additional constraints in the form of user input, either as *trimaps* or *scribbles*. This user-supplied information identifies pixels for which the opacity value α_i is known to be 1 or 0, *i.e.*, *known foreground* and *known background* pixels, respectively. The remaining unconstrained pixels are marked as *unknown*. The goal of a digital matting algorithm is then to compute the values of α_p, F_p, and B_p for all pixels labeled as unknown.

Matting techniques can be classified according to the underlying method used for solving the matte [WC07a], which can be based on *sampling*, *pixel affinities*, or a combination of the two. Most recent matting algorithms [SRR09,LLW08, RRG08,WC07b] fall in one of the last two categories, where local affinities are employed in optimization steps for solving or refining the matte. This usually requires the solution of large linear systems, whose sizes are directly proportional to the number of unknown pixels in I and, thus, can become quite big. Furthermore, these optimization procedures solve for α independently of F and B, thus requiring an additional step for reconstructing F and, if necessary, also B. Consequently, state-of-the-art techniques take from seconds to minutes to generate alpha mattes for typical images (with about 1 Megapixels), making the matte creation process a tedious task. Long offline computations have also prevented the use of natural scenes in real-time matting applications, such as live broadcasting.

We present the first real-time matting technique for natural images and videos. Our approach is based on the key observation that pixels in a small neighborhood tend to have highly similar values for (α, F, B) triplets. Thus, a significant

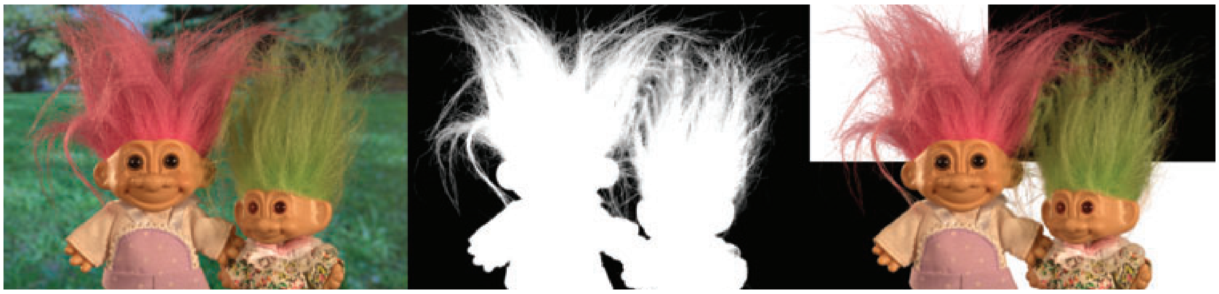

Figure 1: *Example of alpha matte extraction and compositing using our technique. (Left) Image (800 × 563 pixels) from the training dataset provided by [RRW*09]. (Center) Alpha matte computed with our technique in 0.043 seconds using a trimap with 40% of unknown pixels. (Right) Composite of the extracted foreground on a new background.*

amount of computation used to obtain the matte for neighboring pixels is in fact redundant and can be safely eliminated. We show how to avoid such unnecessary computation by carefully distributing the work among neighboring pixels, which will then share their results. Since the operations performed by the pixels are independent, they can be performed in parallel on modern GPUs. As a result, our approach can generate high-quality mattes up to 100 times faster than the previous techniques. The quality of our results have been confirmed by the independent image-matting benchmark of Rhemann *et al.* [RRW*09]. According to this benchmark, our technique ranked second among current state-of-the-art techniques. Figure 1 shows an example of an alpha matte extracted with our technique in 0.043 seconds for a challenging example taken from the training dataset provided by Rhemann *et al.* [RRW*09]. When extended with an additional optimization step described in Section 4.2, our approach ranks first in the same benchmark, while still performing at interactive rates.

We also introduce a new objective function for identifying good pairs of background and foreground colors for any given pixel p (Equation 7). Our new function takes into account spatial, photometric, and probabilistic information extracted from the image. Such a function allows our approach to achieve high-quality results while still operating on a considerably small discrete search space.

Due to its speed, our technique has the potential to enable new and exciting real-time applications that have not been previously possible. We illustrate this potential by showing the first real-time alpha matting demonstration for natural-scene videos, and by providing real-time feedback to users during interactive alpha-matting extraction sessions.

2. Related Work

Sampling-based approaches make the assumption that the true foreground and background colors of an unknown pixel can be explicitly estimated by analyzing nearby known pixels (*i.e.*, pixels in the trimap's known regions), which are called background or foreground *samples*. The first technique to use sampling for estimating the alpha values of unknown pixels was proposed by Mishima [Mis94]. Earlier systems also include the work of Ruzon and Tomasi [RT00], where alpha values are estimated using simple color distributions. Bayesian matting [CCSS01] models the foreground and background distributions with spatially-varying sets of Gaussians and solves the matte using a *maximum a posteriori* (MAP) estimation.

Affinity-based approaches solve for α independent of the estimation of foreground and background colors. The Poisson matting algorithm [SJTS04] observes that if the foreground and background colors are locally smooth, the gradient of the matte can be estimated from the gradient of the image. The matte is then found by solving Poisson equations, with boundary conditions defined by the trimap. The Random Walks method of Grady *et al.* [GSAW05] propagates user constraints to the entire image by minimizing a quadratic cost function. The Closed-form matting technique [LLW08] solves the matte by minimizing a cost function derived from careful analysis of the matting problem.

Combined approaches mix an initial sampling step with α-propagation methods. The recent Robust matting approach of Wang and Cohen [WC07b] uses an initial sampling step to adapt an energy function that is then minimized using random walks. The authors discuss a way to calculate the confidence of the collected samples, and the computation only uses high confidence samples. The work of Rhemann *et al.* [RRG08] improves on this idea by proposing new weights for the confidence metric. Furthermore, they improve the search for suitable foreground samples by assuming that the foreground object is spatially connected.

Some algorithms use additional information to constrain the matting problem. This extra information usually requires special conditions, such as capturing multiple images with different focal depths [MMP*05] or acquiring depth information using a time-of-flight sensor [CTP*08].

Interactive alpha matting of images is typically performed using a two-step iterative process: first, the user refines the needed constraints (*trimap* or *scribbles*), which will then be used for matte generation in a subsequent step. This process is repeated until the user is satisfied with the quality of the matte. Any delay between successive evaluations of the first step are enough to make this a time-consuming and tedious process. The system proposed by Wang *et al.* [WAC07] tries to avoid such a delay by noting that the user modifies the system constraints in a localized fashion. Therefore, the matte only needs to be (re)computed for a small portion of the image at a time. However, as noted by Rhemann *et al.* [RRRAS08], long or complex boundaries are still monotonous and time-consuming to trace.

For segmentation and matting of videos, Bai and Sapiro [BS07] use the geodesic distance — based on the shortest path on a weighted graph — to interactively make soft segmentation and matting of images and videos. The recent work by Bai *et al.* [BWSS09] uses local classifiers to propagate a user defined segmentation across time. The authors extend the work in [LLW08] by adding a temporal coherence term to the cost function for generating mattes for offline video sequences. None of these techniques, however, are suitable for real-time alpha-matting of videos.

3. Real-Time Alpha Matting

Our technique for real-time alpha matting is based on the fundamental observation that pixels in a small neighborhood often have highly similar values for their true (α, F, B) triplets. From this, it follows that: (i) the initial collection of samples gathered by nearby pixels differ only by a small number of elements; and (ii) close-by pixels usually select the same or very similar pairs for their best foreground and background colors. This leads to the conclusion that performing the alpha matte computation for each pixel independently of its neighbors results in a large amount of redundant work, which can be safely avoided without compromising the quality of the matte. In fact, as demonstrated in Section 4, it is possible to achieve speedups of up to two orders of magnitude while still obtaining high-quality results.

Our technique takes as input an image I (or video sequence) and its corresponding trimap(s), and consists of the following steps:

1. **Expansion of Known Regions**: extrapolates “known foreground" and “known background" regions of the trimap into the “unknown" regions;
2. **Sample Selection and Matte Computation**: tries to identify, for each pixel in the unknown regions, the best pair of foreground and background samples. It also computes an alpha matte from the obtained samples;
3. **Local Smoothing**: locally smooths the resulting matte while maintaining its distinct features.

3.1. Expansion of Known Regions

A trimap T segments an input image (or video frame) into three non-overlapping pixel regions: known foreground (T_f), known background (T_b) and unknown (T_u). The idea behind expanding known regions is to exploit the affinity of neighboring pixels to reduce the size of the unknown regions. Thus, let $D_{image}(p,q)$ and $D_{color}(p,q)$ be, respectively, the image-space and color-space distances between two pixels p and q. The expansion process consists of checking for each pixel $p \in T_u$ if there exists a pixel $q \in T_r$ ($r = \{f, b\}$) such that $D_{image}(p,q) \leq k_i$, $D_{color}(p,q) \leq k_c$, and $D_{image}(p,q)$ is minimal for p. In such a case, pixel p is labeled as belonging to region T_r based on its affinity to pixel $q \in T_r$. The value of the parameter k_i depends on the unknown region size; thus, larger images might require larger values of k_i. We found that $k_i = 10$ pixels and $k_c = 5/256$ units (measured as Euclidean distance in the RGB unit cube) produce good results for typical images. These values were used for all examples shown in the paper and supplemental materials.

3.2. Sample Selection and Matte Computation

For each remaining pixel $p \in T_u$, our goal is to find an (α, F, B) triplet that better models p. For this, a sampling strategy inspired by the work of Wang and Cohen [WC07b] is used, but it differs from theirs in some fundamental aspects. For any given pixel $p \in T_u$, Wang and Cohen's idea is to collect a large number of foreground and background samples in a neighborhood around p. Such samples are considered candidates for estimating the alpha value of p. Their assumption is that the true foreground and background colors should be close to the ones of some of the collected samples. This is a reasonable assumption when the initial sample set is large. For instance, in their approach, Wang and Cohen analyze 400 pairs of foreground and background colors for each pixel p [WC07b]. Unfortunately, the use of larger sample sets requires a significant amount of computation. The next sections show that this computational cost can be significantly reduced by exploiting affinities among neighboring pixels. Furthermore, a new and improved metric for electing the best samples is presented, which takes into account image parameters that were not considered in the sample-selection process described in [WC07b].

We minimize the amount of work involved in finding the best pairs of foreground and background samples for a set of neighbor pixels by leveraging their high affinity. For this, we first divide this task among the various pixels in the neighborhood (*sample gathering*), which then share and refine their results (*sample refinement*):

Sample Gathering: in this stage, each pixel $p \in T_u$ selects the best (F, B) pair from a small set of samples in its neighborhood. Those sets are gathered in a manner that guarantees that sample sets from neighboring pixels are disjoint. p gathers at most k_g background and k_g foreground samples

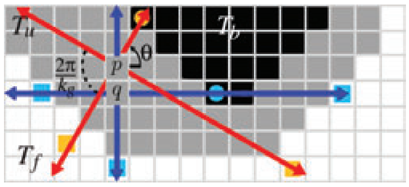

Figure 2: *The red arrows starting at p define the paths for searching for background and foreground samples for p. The selected (closest) samples are marked in orange. Pixel q explores a different path (in blue). Foreground samples are shown as squares, and background samples as circles.*

(Figure 2), resulting in at most k_g^2 tested pairs, from which the best candidate is selected.

Sample Refinement: in this stage, each pixel $p \in T_u$ analyzes the choices (without recombining them) of its (at most) k_r spatially closest pixels in T_u. Thus, while in practice p performs (at most) $k_g^2 + k_r$ pair evaluations, due to the affinity among neighbor pixels, this is roughly equivalent to performing (at most) $k_g^2 \times k_r$ pair comparisons. According to our experience, values of $k_g = 4$ and $k_r = 200$ produce very good results. For these values, the actual number of performed pair comparisons is 216 (*i.e.*, 16 + 200), while its net effect approximates a total of 3,200 (*i.e.*, 16 × 200) comparisons. Sections 3.2.1 and 3.2.2 present the details of the sample gathering and sample refinement sub-steps.

3.2.1. Sample Gathering

In this stage, each pixel $p \in T_u$ looks for possible foreground and background samples along k_g line segments starting at p (Figure 2). These segments divide the plane of the image into k_g disjoint sectors containing equal planar angles. The slope of the first line segment associated to p is defined by an initial orientation $\theta \in \left[0, \frac{\pi}{2}\right]$ measured with respect to the horizontal line. Such an angle takes a different value for each pixel $q \in T_u$ in a 3×3 window (Figure 2). The orientation of the other segments is given by an angular increment $\theta_{inc} = \frac{2\pi}{k_g}$. Starting from p and following a particular line segment yields at most one background and at most one foreground sample — the ones closer to p along the segment. Thus, p must find its best pair among, at most, k_g^2 sample pairs.

We introduce a new objective function that combines photometric, spatial, and probabilistic elements to select the best sample pair for a pixel from the initial set of k_g^2 pairs. The proposed approach is the first to comprehensively consider all these aspects. Thus, let $\mathbf{f}_i$ and $\mathbf{b}_j$ be a pair of foreground and background samples, whose colors are F^i and B^j, respectively. Next, we will derive an objective function (Equation 7) for identifying the best sample-pair for each pixel $p \in T_u$. Before describing this function, its required building blocks (*minimization of chromatic distortion* and *image space statistics*) will be presented.

Minimization of chromatic distortion: Similar to what has been described by Wang and Cohen [WC07b], our approach favors the selection of pairs of foreground and background colors that can model the color of pixel p as a linear combination of themselves. This is modeled by the *chromatic distortion* M_p, whose value should be small for a good pair of candidate colors:

$$M_p(F^i, B^j) = \| C_p - (\hat{\alpha}_p F^i + (1 - \hat{\alpha}_p) B^j) \| \qquad (2)$$

where C_p is the color of p, and $\hat{\alpha}_p$ is the estimated alpha value for p, obtained using the color space projection of C_p onto the line defined by F^i and B^j. Wang and Cohen [WC07b] further divide Equation 2 by $\|F^i - B^j\|$ — which we do not do — to enforce a wide separation of the foreground from the background colors. While their observation regarding linear interpolation is intuitively sound according to Equation 1, this second criterion is oversimplifying and does not address the fundamental issues involved in the computation of a matte. As such, it does not represent a good measure for comparison of candidate pairs. Thus, although a small $M_p(F^i, B^j)$ is necessary for accurately representing the alpha value of p, it is not a sufficient condition to elect a good sample pair. For this reason, we propose a new color metric derived from two previously made observations: (i) Omer and Werman [OW04] showed that small pixel neighborhoods tend to form locally linear clusters in color space. This is particularly true for small windows located over image edges; and (ii) Mitsunaga *et al.* [MYT95] showed that if the foreground and background gradient norms $\|\nabla F\|$ and $\|\nabla B\|$ are relatively small compared to $\|\nabla \alpha\|$, then the image gradient ∇I is directly proportional to $\nabla \alpha$.

Based on these observations, one concludes that in the unknown region of the trimap — where $\|\nabla \alpha\|$ is potentially very large — the locally-linear color variations observed by Omer and Werman [OW04] are primarily caused by variations in α. Thus, all colors from pixels in a small local window are situated approximately along the line in color space spanned by the true foreground and background colors F and B. This means that a good sample pair should minimize the chromatic distortion not only for p, but for all pixels in a small window around p. Thus, a good sample pair should minimize the least squares residual defined by the *neighborhood affinity* term:

$$N_p(\mathbf{f}_i, \mathbf{b}_j) = \sum_{q \in \Omega_p} M_q(F^i, B^j)^2 \qquad (3)$$

where Ω_p is the pixel neighborhood of p, consisting of all pixels in a 3 × 3 window centered at p, and M_q is the operator defined in Equation 2, evaluated at pixel q.

Image space statistics: In addition to color space information, image space statistics should play a key role in identifying good pair of samples. These image parameters were not considered in the sample selection process of [WC07b], where only color space metrics were used. We will now use such information to estimate the probability of pixel p belonging to the foreground, which will be later used in Equation 6 to enforce a meaningful alpha value. Thus, let $D_p(\mathbf{s}) = D_{image}(\mathbf{s}, p) = \|\mathbf{s} - p\|$ be the image space distance from a sample $\mathbf{s}$ to the current pixel p. We define the *energy* $E_p(\mathbf{s})$ to reach a foreground or background sample $\mathbf{s}$ from the current pixel p as the squared path integral of ∇I along the image space line segment L connecting p and $\mathbf{s}$:

$$E_p(\mathbf{s}) = \int_L \|\nabla I \cdot d\mathbf{r}\|^2 = \int_p^{\mathbf{s}} \left\| \nabla I \cdot \left(\frac{\mathbf{s} - p}{\|\mathbf{s} - p\|} \right) \right\|^2. \quad (4)$$

The energy $E_p(\mathbf{s})$ is directly proportional to the projected length of ∇I onto the normalized direction of integration $(\mathbf{s} - p)$. Thus, if the linear path from $\mathbf{s}$ to p crosses image regions where $\|\nabla I\|$ is large (*e.g.*, at image edges), greater energy will be required to reach $\mathbf{s}$.

An estimate of the probability of p belonging to the foreground, according to the energy $E_p(\mathbf{s})$, can be obtained as:

$$PF_p = \frac{min_j(E_p(\mathbf{b}_j))}{min_i(E_p(\mathbf{f}_i)) + min_j(E_p(\mathbf{b}_j))}. \quad (5)$$

Thus, if the minimum energy required to reach a foreground sample is much lower than the minimum energy required to reach a background sample, PF_p will be close to one — *i.e.*, pixel p has a high probability of belonging to the foreground.

Intuitively, we want the alpha value $\hat{\alpha}_p$ (computed from $\mathbf{f}_i$ and $\mathbf{b}_j$ — Equation 2) to correlate with the probability PF_p of pixel p belonging to the foreground. Thus, a good sample pair should also minimize the function A_p:

$$A_p(\mathbf{f}_i, \mathbf{b}_j) = PF_p + (1 - 2\, PF_p)\, \hat{\alpha}_p. \quad (6)$$

Indeed, for a given pair of samples $(\mathbf{f}_i, \mathbf{b}_j)$, when $PF_p = 0$, $A_p(\mathbf{f}_i, \mathbf{b}_j) = \hat{\alpha}_p$. Thus, minimizing $A_p(\mathbf{f}_i, \mathbf{b}_j)$ also minimizes the value of $\hat{\alpha}_p$. Likewise, when $PF_p = 1$, $A_p(\mathbf{f}_i, \mathbf{b}_j) = (1 - \hat{\alpha}_p)$, so minimizing $A_p(\mathbf{f}_i, \mathbf{b}_j)$ maximizes $\hat{\alpha}_p$. Finally, if $PF_p = 0.5$, $A_p(\mathbf{f}_i, \mathbf{b}_j) = 0.5$, and the value of $\hat{\alpha}_p$ has no effect on the minimization. $A_p(\mathbf{f}_i, \mathbf{b}_j)$ will be used as one of the terms in the final objective function (Equation 7).

The Objective function: The resulting objective function that combines photometric and spatial affinity, as well as probabilistic information for selecting good pairs of background and foreground samples can be expressed as:

$$g_p(\mathbf{f}_i, \mathbf{b}_j) = N_p(\mathbf{f}_i, \mathbf{b}_j)^{e_N} A_p(\mathbf{f}_i, \mathbf{b}_j)^{e_A} D_p(\mathbf{f}_i)^{e_f} D_p(\mathbf{b}_i)^{e_b}. \quad (7)$$

Here, $N_p(\mathbf{f}_i, \mathbf{b}_j)$ minimizes chromatic distortion in the 3×3 neighborhood around p. $A_p(\mathbf{f}_i, \mathbf{b}_j)$ enforces that the computed alpha matte values correlate with the probability of pixel p belonging to the foreground. $D_p(\mathbf{f}_i)$ and $D_p(\mathbf{b}_i)$ enforce the spatial affinity criterion: the background and foreground samples should be as close as possible to p. The constants $e_{\{N,A,f,b\}}$ define the penalties for having a high value in any of these metrics. In practice, we found that values of $e_N = 3$, $e_A = 2$, $e_f = 1$ and $e_b = 4$ tend to produce good results. Thus, the best pair of foreground and background samples $(\hat{\mathbf{f}}_p, \hat{\mathbf{b}}_p)$ for pixel p is obtained by evaluating $g_p(\mathbf{f}_i, \mathbf{b}_j)$ for all possible sample-pairs:

$$(\hat{\mathbf{f}}_p, \hat{\mathbf{b}}_p) = \operatorname{argmin}_{\mathbf{f},\mathbf{b}}\, g_p(\mathbf{f}_i, \mathbf{b}_j). \quad (8)$$

Let (F_p^g, B_p^g) be the corresponding colors of the best pair $(\hat{\mathbf{f}}_p, \hat{\mathbf{b}}_p)$ for p, obtained in the gathering stage. We then compute σ_f^2 and σ_b^2 as:

$$\begin{aligned} \sigma_f^2 &= \tfrac{1}{N} \textstyle\sum_{q \in \Omega_f} \left\| C_q - F_p^g \right\|^2, \\ \sigma_b^2 &= \tfrac{1}{N} \textstyle\sum_{q \in \Omega_b} \left\| C_q - B_p^g \right\|^2 \end{aligned} \quad (9)$$

where Ω_f and Ω_b are 5×5 pixel neighborhoods centered at $\hat{\mathbf{f}}_p$ and $\hat{\mathbf{b}}_p$, respectively, and $N = 25$. The values σ_f^2 and σ_b^2 measure the local color variations in the neighborhoods Ω_f and Ω_b assuming small univariate Gaussian color distributions around $\hat{\mathbf{f}}_p$ and $\hat{\mathbf{b}}_p$. We will use σ_f^2 and σ_b^2 in the sample-refinement step. Hence, the output of the sample gathering stage is a tuple $\tau_p^g = (F_p^g,\, B_p^g,\, \sigma_f^2,\, \sigma_b^2)$ for each pixel $p \in T_u$.

3.2.2. Sample Refinement

For small values of k_g, the number of samples analyzed by pixel $p \in T_u$ during the sample gathering stage is often not enough to reliably estimate either an alpha value or the true foreground and background colors. Thus, a more extensive search is performed by sharing the best results obtained by all pixels in a neighborhood around p in T_u.

In the sample-refinement stage, each pixel p compares its own choice of best sample-pair with the choices of its (at most) k_r spatially closest pixels $q \in T_u$. Among those, the three tuples with the lowest values of $M_p(F_q^g, B_q^g)$ are then averaged to create a new tuple $\tilde{\tau}_p^g = (\tilde{F}_p^g,\, \tilde{B}_p^g,\, \tilde{\sigma}_f^2,\, \tilde{\sigma}_b^2)$ for p. The purpose of this averaging is to reduce the occurrence of noise in the resulting alpha matte. This procedure is supported by the observation that neighbor pixels tend to have similar values of alpha, as well as background and foreground colors (*i.e.*, neighbor pixels tend to present high affinity). Therefore, by averaging the best few values in a given neighborhood, the occurrence of noise is reduced.

The output of the sample-refinement stage for pixel $p \in T_u$ is another tuple $\tau_p^r = (F_p^r,\, B_p^r,\, \alpha_p^r,\, f_p^r)$, where:

$$F_p^r = \begin{cases} C_p & \text{if } \left\| C_p - \tilde{F}_p^g \right\|^2 \le \tilde{\sigma}_f^2 \\ \tilde{F}_p^g & \text{otherwise} \end{cases}, \quad (10)$$

$$B_p^r = \begin{cases} C_p & \text{if } \left\| C_p - \tilde{B}_p^g \right\|^2 \le \tilde{\sigma}_b^2 \\ \tilde{B}_p^g & \text{otherwise} \end{cases}, \quad (11)$$

$$\alpha_p^r = \frac{(C_p - B_p^r) \cdot (F_p^r - B_p^r)}{\left\| F_p^r - B_p^r \right\|^2}, \quad (12)$$

$$f_p^r = \begin{cases} exp\left\{ -\lambda\, M_p(\tilde{F}_p^g, \tilde{B}_p^g) \right\} & \text{if } F_p^r \neq B_p^r \\ \varepsilon & \text{if } F_p^r = B_p^r \end{cases}. \quad (13)$$

Here, the superscript r represents quantities computed in the sample-refinement stage. The intuition behind the computation of F_p^r is that if the color C_p of pixel p is sufficiently close to the average color $\tilde{F}_p^g$ of the best three foreground samples computed during the gathering stage, then F_p^r should be taken as C_p, thus keeping the original color. The case for B_p^r is similar. The alpha value α_p^r is computed as the relative length of the projection of vector $(C_p - B_p^r)$ onto vector $(F_p^r - B_p^r)$, defined by the computed foreground and background colors. Thus, α_p^r represents the opacity of p. Finally, f_p^r expresses the confidence of p in its candidate foreground and background colors F_p^r and B_p^r. This confidence measure should decrease fast (but not too fast) as the foreground and background colors $\tilde{F}_p^g$ and $\tilde{B}_p^g$ fail to properly model the color C_p of p. According to our experience, a value of $\lambda = 10$ produces good results. Additionally, if C_p is close to both $\tilde{F}_p^g$ and $\tilde{B}_p^g$, the α value of p cannot be accurately estimated and the confidence f_p^r is set to a small value $\varepsilon = 10^{-8}$.

For completeness, output tuples for pixels outside the unknown region are also defined; thus, for pixels $v \in T_f$, we have $\tau_v^r = (F_v^r = C_v,\ B_v^r = C_v,\ \alpha_v^r = 1,\ f_v^r = 1)$; and for pixels $v \in T_b$, we have $\tau_v^r = (F_v^r = C_v,\ B_v^r = C_v,\ \alpha_v^r = 0,\ f_v^r = 1)$.

3.3. Local Smoothing

Although the sample-selection process takes into account affinities among localized groups of pixels, this is not enough to prevent discontinuities in the resulting matte. Thus, an additional step is used to ensure the local smoothness of the final alpha values, while maintaining its distinct features. This is achieved by computing, for each pixel $p \in T_u$, a weighted average of the tuples τ_q^r of the closest m neighbors of p in image space (we use a value of $m = 100$). Such neighbors can come from either T_u, T_f, or T_b. Let Ψ_p be such a neighborhood for pixel p. The weights are defined in such way that details in the matte are preserved.

The final foreground and background colors F_p and B_p of p are computed as:

$$W_c(p,q) = \begin{cases} G\left(D_{image}(p,q)\right) f_q^r \left|\alpha_p^r - \alpha_q^r\right| & \text{if } p \neq q \\ G\left(D_{image}(p,q)\right) f_q^r & \text{if } p = q \end{cases},$$

$$F_p = \frac{\sum_{q \in \Psi_p}\left[W_c(p,q)\, \alpha_q^r\, F_q^r\right]}{\sum_{q \in \Psi_p}\left[W_c(p,q)\, \alpha_q^r\right]}, \tag{14}$$

$$B_p = \frac{\sum_{q \in \Psi_p}\left[W_c(p,q)\, (1-\alpha_q^r)\, B_q^r\right]}{\sum_{q \in \Psi_p}\left[W_c(p,q)\, (1-\alpha_q^r)\right]} \tag{15}$$

where G is a normalized Gaussian function with variance $\sigma^2 = m/9\pi$ pixels. The set Ψ_p of the closest m pixels to p approximates an image space circle with area $m = \pi r^2$. The farthest pixels in Ψ_p have a distance of r to p and should have weights close to zero in the Gaussian (*i.e.*, $r = 3\sigma$). Thus, $m = \pi r^2 = \pi(3\sigma)^2$, which solves to $\sigma^2 = m/9\pi$.

The weight $W_c(p,q)$ blends the foreground (background) colors of pixels p and q taking into account: (i) *Spatial affinity* – the colors of two pixels that are far apart in image space should not be averaged. This is modeled by the Gaussian function; (ii) *Confidence values* – pixels with low confidence in their foreground and background samples should not propagate their uncertainty. (iii) *Difference in alpha values* – by minimizing $\|\nabla F\|$ and $\|\nabla B\|$ where the estimated $\|\nabla\hat{\alpha}\|$ is large, we make the final $\nabla\alpha \approx \nabla I$; Furthermore, whenever $C_q = B_q^r$, $\alpha_q^r = 0$ regardless the value of F_q^r; conversely, whenever $C_q = F_q^r$, $\alpha_q^r = 1$ regardless the value of B_q^r (Equation 12). Thus, we multiply F_q^r by α_q^r in Equation 14 to denote that the confidence of pixel q in its foreground color F_q^r is directly proportional to α_q^r. Similarly, we multiply B_q^r by $(1-\alpha_q^r)$ in Equation 15 to denote that the confidence of pixel q in its background color B_q^r is inversely proportional to α_q^r.

We can now compute the confidence f_p of pixel p in its final foreground and background colors F_p and B_p. To do so, we first define in Equation 17 the *mean foreground-background distance* D_{FB} (in color space) for the neighborhood Ψ_p. This mean is weighted by $W_{FB}(q)$ (Equation 16) which is directly proportional to the confidence f_q^r of q and is maximized for values of $\alpha_q^r = 0.5$ — where the confidence of q in both F_q^r and B_q^r is potentially maximal — while being zero for $\alpha_q = \{0,1\}$. D_{FB} will be used next to compute the final confidence f_p.

$$W_{FB}(q) = f_q^r\, \alpha_q^r\, \left(1-\alpha_q^r\right), \tag{16}$$

$$D_{FB}(\Psi_p) = \frac{\sum_{q \in \Psi_p}\left[W_{FB}(q)\, \left\|F_q^r - B_q^r\right\|\right]}{\sum_{q \in \Psi_p} W_{FB}(q)}. \tag{17}$$

The final confidence f_p of pixel p in its final foreground and background colors F_p and B_p is modeled by Equation 18. Here, the first term expresses the ratio of the distance $\|F_p - B_p\|$ to the mean foreground-background distance in the neighborhood Ψ_p (clamped to the range [0,1]). This ratio tries to detect pixels whose final foreground and background colors deviate from those in the neighborhood Ψ_p. The second term is analogous to Equation 13 ($\lambda = 10$).

$$f_p = min\left(1, \frac{\|F_p - B_p\|}{D_{FB}(\Psi_p)}\right)\, exp\left\{-\lambda\, M_p(F_p, B_p)\right\}. \tag{18}$$

Having F_p, B_p and f_p, we can now compute the final alpha value α_p of pixel p. In order to do so, we first define the *low frequency alpha* α_p^l (Equation 20) as the weighted average of alpha values in the neighborhood Ψ_p. The weights $W_\alpha(p,q)$ are proportional to the confidence f_q^r of q and inversely proportional to the image space distance of p and q. Additionally, greater weights are given for pixels lying in T_f or T_b (*i.e.*, known pixels).

$$W_\alpha(p,q) = f_q^r\, G\left(D_{image}(p,q)\right) + \delta(q \notin T_u), \tag{19}$$

$$\alpha_p^l = \frac{\sum_{q \in \Psi_p}\left[W_\alpha(p,q)\, \alpha_q^r\right]}{\sum_{q \in \Psi_p} W_\alpha(p,q)} \tag{20}$$

where δ is a boolean function returning 1 when $q \notin T_u$; or 0 otherwise. G is the Gaussian function from W_c (Equations 14 and 15).

The final alpha value α_p for p is given by Equation 21. This equation blends the alpha value computed using F_p and B_p with the low frequency alpha α_p^l, using as blending factor the final confidence f_p. Thus, pixels with a low final confidence will accept alpha values from higher-confidence neighbors (modeled by α_p^l) to preserve local smoothness.

$$\alpha_p = f_p \frac{(C_p - B_p)\cdot(F_p - B_p)}{\|F_p - B_p\|^2} + (1 - f_p)\,\alpha_p^l. \quad (21)$$

Finally, the output of the proposed algorithm for the matting parameters of pixel $p \in T_u$ is given by the tuple (F_p, B_p, α_p) with an associated confidence value of f_p. For completeness, the matting parameters for pixels outside the unknown region are also defined. Thus, for pixels $q \in T_f$ we have the tuple $(F_q = C_q, B_q = C_q, \alpha_q = 1)$ with confidence $f_q = 1$, and for pixels $w \in T_b$ we have the tuple $(F_w = C_w, B_w = C_w, \alpha_w = 0)$ with confidence $f_w = 1$.

4. Results

We have implemented the technique described in the paper using C++ and GLSL and used it to process a large number of images, and videos. Given that the search space associated with Equation 7 is both small and discrete, its minima is computed by evaluating this equation for its entire search space and by selecting the sample-pair with the smallest value. Since these operations can be performed independently for each pixel $p \in T_u$, we exploit the inherent parallelism of current GPUs to efficiently perform these searches in parallel. All the results reported in this paper were obtained on a 2.8 GHz Quad Core PC with 8 GB of memory and a GeForce GTX 280 with 1 GB of memory.

In order to assess the quality of our results, we used the benchmark provided by Rhemann *et al.* [RRW*09]. It evaluates and compares the accuracy of an image-matting technique against the results produced by the state-of-the-art. Such a benchmark is composed of eight test images publicly available at *www.alphamatting.com*, each accompanied by three trimaps (*small*, *large* and *user*). These images are designed to be challenging representations of natural scenes, containing examples of highly textured backgrounds, as well as images where background and foreground colors cannot be easily differentiated. The ground-truth alpha mattes for each of the test images are used to assess the quality of the results, but are not disclosed to the public. As such, Rhemann *et al.*'s benchmark provides an independent and reliable mechanism for evaluating the quality of digital image-matting algorithms. Tables 2 (a) and (b) show results from Rhemann *et al.*'s benchmark and summarize the accuracy of our technique (*Shared Matting - Real-Time*) in comparison to others. Our approach ranks second according to the sum of absolute differences (SAD) and mean squared error

Image	# of pixels	% Unknown	Time (sec)
elephant	536,800	16%	0.029
donkey	455,200	17%	0.028
pineapple	481,600	20%	0.032
doll	451,200	25%	0.034
plasticbag	529,600	28%	0.040
plant	425,600	35%	0.038
troll	512,000	40%	0.045
net	496,000	51%	0.056

Table 1: *Time to generate alpha mattes with our technique for the images in the benchmark (Figure 2), using the most conservative trimaps (*i.e.*, large).*

(MSE) metrics, when compared to previous techniques. It is, however, up to 100 times faster, allowing, for the first time, alpha matting in real-time applications. Such results indicate that our technique produce high-quality alpha mattes even for challenging images.

Since previous technique are not suitable for real-time applications, performance comparisons considering the time required to compute the alpha matte have been overlook in many previous publications. Our technique, on the other hand, can compute alpha mattes for typical images in real-time. Table 1 summarizes the times required to extract the alpha mattes for the test dataset available from Rhemann *et al.*'s benchmark [RRW*09] using the most conservative trimaps (*i.e.*, large). For each image, we provide its dimensions, the number of pixels in the unknown region of the trimap, and the time required to compute the matte.

Figure 3 shows the alpha mattes generated by our technique for some images from the training dataset provided by [RRW*09] (using the small trimaps). For such dataset, the ground-truth mattes are available and are shown next to our results for comparison. Versions of these images in their original resolutions are provided for closer inspections as supplementary material. For this dataset (average image size of 0.5Mpix, with 16% unknown pixels), our technique takes on average 0.024 seconds for the matte computation. The expansion-of-known-regions step (Section 3.1) reduced the number of pixels in T_u by 38% on average, but 4% of the pixels were erroneously considered to be in T_f or T_b.

In the gathering stage (Section 3.2.1), foreground and background samples are found using a linear search (Figure 2) with a step size of 6 pixels and a maximum of 300 steps. Fine details in the trimap might be "jumped over" by some unknown pixels; however, this is not a problem due to the sample-pair sharing in the sample-refinement stage.

Cost of the Algorithm: Let z be the number of pixels in T_u. Per pixel computation can be defined as follows: (i) the "expansion of known regions" step runs in constant time $O(\pi k_i^2)$; (ii) The "sample gathering" step has a worst cost of $O(z + k_g^2)$, and average cost of $O(\omega + k_g^2)$, where ω is the

Sum of Absolute Differences	overall rank	avg. small rank	avg. large rank	avg. user rank	Troll (Strongly Transparent) Input			Doll (Strongly Transparent) Input			Donkey (Medium Transparent) Input			Elephant (Medium Transparent) Input			Plant (Little Transparent) Input			Pineapple (Little Transparent) Input			Plastic bag (Highly Transparent) Input			Net (Highly Transparent) Input		
					small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user
Shared Matting	**1.8**	1.9	2.3	1.4	**10.8**$_1$	20.5$_2$	**15**$_1$	7.8$_3$	11.6$_3$	**8.1**$_1$	**4.2**$_1$	**5.3**$_1$	**4.2**$_1$	**2.1**$_1$	5.8$_3$	**2.9**$_1$	**5.9**$_1$	**9.2**$_1$	**11.4**$_1$	**5**$_1$	**8.8**$_1$	**6.8**$_1$	34.9$_5$	34.9$_4$	34.3$_4$	23.9$_2$	28.4$_3$	**25.7**$_1$
Improved color matting	**2.7**	2.9	2.6	2.6	14.9$_4$	24.5$_5$	20$_5$	6.7$_2$	9.5$_2$	8.5$_2$	4.6$_3$	6.1$_4$	4.3$_2$	2.6$_4$	5.4$_2$	3.4$_4$	7.5$_4$	9.9$_2$	12.5$_2$	6$_3$	10.1$_3$	8.4$_3$	26.1$_2$	26.7$_2$	**23.6**$_1$	**23.8**$_1$	**25.6**$_1$	26.7$_2$
Shared Matting (Real Time)	**3.3**	3.4	3.5	3.0	12.4$_2$	21.6$_3$	16.3$_2$	9.5$_5$	13.5$_4$	9.9$_4$	4.4$_2$	5.6$_2$	4.4$_3$	2.5$_3$	6.8$_5$	3.2$_2$	7.1$_2$	10.8$_3$	12.6$_3$	5.4$_2$	9.7$_2$	7.4$_2$	35.5$_7$	35.8$_5$	35.5$_5$	27.6$_4$	33.4$_4$	29.8$_3$
Closed-Form Matting	**3.6**	3.6	3.0	4.3	12.7$_3$	21.9$_4$	17.2$_3$	**5.9**$_1$	**8.5**$_1$	8.6$_3$	4.7$_4$	6$_3$	4.3$_2$	2.2$_2$	**4.6**$_1$	3.3$_3$	9.3$_6$	12.1$_4$	19.3$_6$	8.3$_6$	14.9$_6$	13.4$_7$	34.2$_4$	32.4$_3$	27.4$_2$	26.5$_3$	25.7$_2$	48.3$_8$
Robust Matting	**4.9**	4.3	5.4	5.0	17.3$_5$	28.4$_7$	21.1$_6$	10.1$_6$	16.9$_8$	11.4$_6$	4.8$_5$	6.5$_6$	5$_5$	2.8$_5$	7.3$_6$	4.4$_5$	7.3$_3$	14$_5$	18.1$_5$	6.8$_4$	14.6$_5$	10.6$_5$	**22.7**$_1$	**26.1**$_1$	32.1$_3$	34.4$_5$	37$_5$	38$_5$

(a)

Mean Squared Error	overall rank	avg. small rank	avg. large rank	avg. user rank	Troll (Strongly Transparent) Input			Doll (Strongly Transparent) Input			Donkey (Medium Transparent) Input			Elephant (Medium Transparent) Input			Plant (Little Transparent) Input			Pineapple (Little Transparent) Input			Plastic bag (Highly Transparent) Input			Net (Highly Transparent) Input		
					small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user
Shared Matting	**1.6**	1.5	2.0	1.4	**0.5**$_1$	1.6$_2$	**0.9**$_1$	0.5$_2$	0.9$_3$	**0.5**$_1$	**0.3**$_1$	**0.4**$_1$	**0.3**$_1$	**0.1**$_1$	0.4$_2$	**0.2**$_1$	**0.4**$_1$	**0.6**$_1$	**0.9**$_1$	**0.4**$_1$	**0.6**$_1$	**0.5**$_1$	2.9$_4$	2.8$_4$	2.7$_4$	**1**$_1$	1.3$_2$	**1**$_1$
Improved color matting	**2.0**	1.9	2.4	1.8	0.8$_3$	2.4$_6$	1.5$_5$	**0.3**$_1$	0.5$_2$	**0.5**$_1$	**0.3**$_1$	**0.4**$_1$	**0.3**$_1$	**0.1**$_1$	**0.3**$_1$	**0.2**$_1$	0.7$_4$	0.7$_2$	**0.9**$_1$	**0.4**$_1$	0.7$_2$	0.7$_3$	2$_2$	1.9$_2$	**1.4**$_1$	1.3$_2$	1.5$_3$	1.5$_2$
Shared Matting (Real Time)	**2.6**	2.3	3.5	2.1	0.6$_2$	1.7$_3$	1$_2$	0.7$_3$	1.2$_5$	0.8$_3$	**0.3**$_1$	**0.4**$_1$	**0.3**$_1$	0.2$_2$	0.6$_4$	**0.2**$_1$	0.5$_2$	0.8$_3$	1.1$_2$	**0.4**$_1$	0.8$_3$	**0.6**$_1$	3$_5$	3$_5$	3$_5$	1.3$_2$	1.9$_4$	1.5$_2$
Closed-Form Matting	**2.9**	2.5	2.8	3.5	**0.5**$_1$	1.8$_4$	1.1$_3$	**0.3**$_1$	**0.4**$_1$	0.6$_2$	**0.3**$_1$	**0.4**$_1$	**0.3**$_1$	**0.1**$_1$	**0.3**$_1$	**0.2**$_1$	1.2$_6$	1.4$_5$	2.3$_5$	0.8$_3$	1.6$_6$	1.6$_6$	3$_5$	2.7$_3$	1.9$_2$	1.3$_2$	**1.2**$_1$	5$_8$
Robust Matting	**3.3**	2.3	4.1	3.4	1.1$_5$	2.8$_8$	1.7$_6$	0.7$_3$	1.5$_6$	0.9$_4$	**0.3**$_1$	**0.4**$_1$	**0.3**$_1$	**0.1**$_1$	0.5$_3$	0.3$_2$	0.5$_2$	1.2$_4$	1.9$_3$	0.5$_2$	1.5$_5$	1.2$_4$	**1.5**$_1$	**1.8**$_1$	2.6$_3$	2.4$_3$	2.3$_5$	2.9$_4$

(b)

Table 2: *Results of Rhemann et al.'s benchmark. Our technique, Shared Matting (Real-Time), ranks second when compared against previous techniques. However, it is up to 100 times faster. A version of our technique extended with an optimization step (Shared Matting) ranks first when compared against the same set of techniques. The overal rankings were computed from the SAD and MSE error values supplied by Rhemann et al.'s benchmark. Only the top five techniques are shown.*

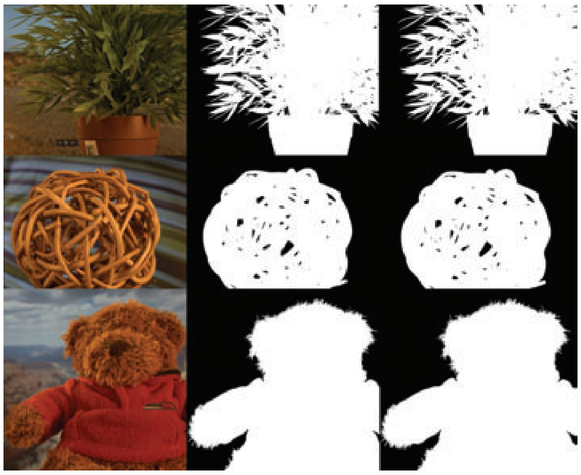

Figure 3: *Left: Images from the training dataset provided by [RRW*09]. Center: Alpha mattes extracted with our technique using the trimaps supplied in the dataset. Right: Ground truth alpha mattes provided for comparison.*

average number of pixels inspected before finding candidate samples; (iii) the "sample refinement" step runs in constant time $O(k_r)$; and (iv) the "local smoothing" step runs in constant time $O(4m)$. All this computation can be done independently for each pixel $p \in T_u$. Thus, let t be the number of pixels which are processed in parallel on a GPU. The average cost is $O(z/t)$. Worst cost is $O(z^2/t)$.

4.1. Applications

Video Matting Our method enables the use of alpha matting in real-time applications for the first time. One possible such application is real-time matte generation for videos. Since our approach uses a trimap as input, it needs to rely on other techniques for providing trimaps for each frame of the video in real-time. The supplementary video illustrates such an application for two video sequences. In these examples, the trimaps were created by dilating the boundaries of the binary segmented video frames. Such segmentation was obtained using a real-time foreground-background binary segmentation technique which models the background color distribution from color priors using the *kernel density estimation* method (as such, this technique is limited to videos where the foreground and background color distributions do not overlap). Thus, given an input video sequence, the results shown in the supplementary video were entirely computed in real-time. This means that the whole sequence of operations comprising binary segmentation, boundary dilation, and matte extraction was performed in real time. Figure 4 shows some frames from two of these video sequences with their corresponding extracted mattes.

Interactive Alpha Matting Another benefit of the improved performance of our technique is its ability to provide real-time feedback to users during interactive alpha-matting sessions. We demonstrate this feature using a simple trimap creation interface. Initially, all pixels in the image are labeled as belonging to the unknown region T_u. As one uses small scribbles over the image, a trimap is automatically computed and refined, also providing instant feedback on the resulting matte extraction. The scribbles are propagated using an iterative flood-filling procedure, limited by a simple edge detector. Figure 5 illustrates the concept using one of the images of the training dataset. On the left, one sees the scribbles superimposed onto the original image. The blue color is a label for foreground pixels, while red and yellow represent back-

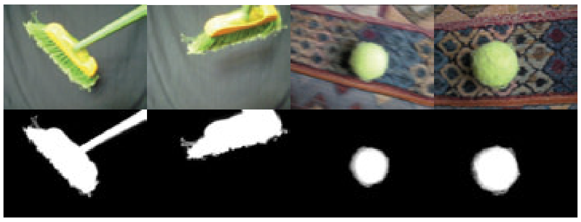

Figure 4: *Some frames extracted from video sequences processed by our technique for real-time matte extraction. The images on top show the original frames, while the extracted mattes are shown at the bottom.*

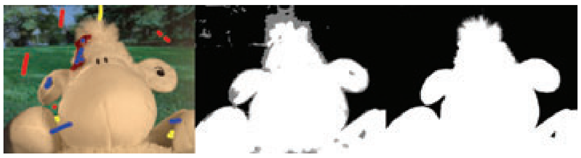

Figure 5: *Use of our technique to interactively segment images by alpha matting. As the user scribbles over the image (left), a trimap is automatically updated. The resulting trimap and alpha matte computed from the set of scribbles on the left are shown on the center and right images.*

ground and unknown pixels, respectively. The image on the center shows the computed trimap, for which a gray shade indicates uncertainty about whether a pixel belongs to the background (shown in black) or to the foreground (shown in white). The extracted alpha matte is shown on the right.

The speed of our method makes the matte creation process easier for the user, as there is no delay between input and matte refinement. This considerably reduces the time taken to interactively segment images by alpha matting. The supplementary video shows such an application in action. The resulting mattes are generated considerably faster for images with complex edges and topologies. Only in the worst case one needs to completely trace the border of the foreground object, which is always needed in the technique described in [WAC07].

High-resolution Matting Due to its improved speed, our method can generate alpha mattes for high-resolution images much faster than existing techniques. We ran our algorithm on the high-resolution training dataset from [RRW*09], where images are, on average, 6.7 Mpix with 16% unknown pixels. For these images, matte computation takes a mean time of 0.3 seconds, with a longest time of 1.1s and shortest time of 0.17s. We also tested our method on extremely high resolution images. For a 37 Mpix image, for which a trimap was created manually containing 11% unknown pixels, matte computation took 3.65 seconds (with $k_i = 30$).

4.2. Matte Optimization

Some users might want to obtain a matte with the best possible quality, even at the expense of extra computation time. This can be achieved by refining the matte obtained in Section 3 using an additional optimization step. This step is analogous to the one in [RRG08], where the final matte is obtained by minimizing a quadratic cost function in α. This cost function is comprised of a *smoothness term* and a *data term*. Here, we use the *matting Laplacian* of [LLW08] as smoothness term, but for the data term we use the matting parameters (α_p, f_p) obtained in Section 3. Thus, let $\hat{\alpha}^T = [\alpha_1, \ldots \alpha_p, \ldots \alpha_n]$ be a vector with the alpha values, obtained in Section 3, for all n pixels in the input image. Let $\hat{\Gamma}$ be a diagonal matrix where each diagonal element γ_p is defined as $\gamma_p = f_p$ if $p \in T_u$ or $\gamma_p = 0$ if $p \in T_f \cup T_b$. The final alpha matte is obtained by solving for:

$$\alpha = \operatorname{argmin} \ \alpha^T L \alpha + \lambda (\alpha - \hat{\alpha})^T D (\alpha - \hat{\alpha}) + \gamma (\alpha - \hat{\alpha})^T \hat{\Gamma} (\alpha - \hat{\alpha}) \tag{22}$$

where λ is some relatively large number when compared to to the values in $\hat{\alpha}$ and $\hat{\Gamma}$. $\gamma = 10^{-1}$ is a constant which defines the relative importance of the data and smoothness terms, L is the matting Laplacian and D is a diagonal matrix whose diagonal elements are one for pixels in $T_f \cup T_b$ and zero for all other pixels. Equation 22 yields a sparse linear system which we solve using the conjugate gradient method implemented on the GPU. For an image with 512,000 pixels and 25% unknown pixels, solving for α takes approximately 100ms. Thus, the system is no longer real-time but still interactive. For videos, the matting Laplacian L must be recomputed for every frame. However, such computation for frame k can be offloaded to the CPU and performed in parallel with the GPU optimization of frame $(k-1)$.

This optimization step induces an α-propagation from high confidence to low confidence pixels. Thus, while pixels with a high final confidence try to adhere to the alpha values computed in Section 3, pixels with a low final confidence will rely more on propagation for their final alpha values.

Tables 2 (a) and (b) show Rhemann *et al.*'s benchmark results for the version of our technique containing the additional optimization step (*Shared Matting*). This extended version ranks first both in the sum of absolute differences and mean squared error metrics.

5. Limitations

The proposed technique makes the assumption that the true foreground and background colors of unknown pixels can be explicitly estimated by analyzing nearby known pixels. For images where this assumption does not hold, computed alpha values will not be accurate (this will be apparent in the final, low, confidence values). Examples of such problematic images are those containing completely transparent foreground objects or those where the foreground and background color distributions strongly overlap.

For videos with highly textured backgrounds or greatly transparent foreground pixels, the produced matte might suffer from temporal noise, *i.e.* flickering. One can consider many ways for improving the temporal coherence of the matte, such as temporal blending of alpha values based on confidence estimates, or even selecting candidate samples along the time axis, in addition to the image space.

6. Conclusions and Future Work

We have presented the first real-time matting technique for natural images and videos. Our technique is based on the observation that pixels in a small neighborhood in image space tend to have highly similar values for (α, F, B) triplets. As such, independently computing such triplets for each pixel in the unknown region in a conventional way tends to result in a lot of redundant work. The amount of required computation can then be significantly and safely reduced by a careful selection of pairs of candidate background and foreground samples. The work required to perform this task can be distributed among neighbor pixels, leading to considerable savings in computation cost. Moreover, the required operations can be performed in parallel on programmable GPUs.

We have also presented a new objective function for identifying good pairs of background and foreground samples. Such a function takes into account spatial and photometric, as well as some probabilistic information extracted from the image. This improved objective function allows our approach to achieve high-quality results while still operating on a considerably small discrete search space. Our approach can achieve speedups of up to two orders of magnitude compared to previous approaches, while producing highly-accurate alpha mattes. We assessed the quality of our results by performing the independently-developed benchmark by Rhemann *et al.* [RRW*09]. In such a benchmark, our real-time technique ranked second with respect to both the sum of absolute differences and to the mean squared error metrics. A version of our technique that uses an additional optimization step to improve the matte quality ranked first according to both metrics, while still performing at interactive rates.

We have demonstrated that our technique can provide instant feedback to support interactive extraction of high-quality alpha mattes. It is also fast enough to, for the first time, support alpha-matte computation for natural videos in real-time, given that the corresponding trimaps are provided. This opens up exciting opportunities for new real-time applications and for improved real-time trimap generation for videos. We are currently exploring some of these directions. Finally, the problem of handling temporal coherence for real-time matte remains to be explored.

7. Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. Manuel M. Oliveira acknowledges the following CNPq-Brazil fellowships and grant: 200284/2009-6, 305613/2007-3, and 476954/2008-8. Eduardo S. L. Gastal acknowledges his PIBIC/CNPq/UFRGS fellowship.

References

[BS07] BAI X., SAPIRO G.: A geodesic framework for fast interactive image and video segmentation and matting. *ICCV 1* (2007), 1–8. 3

[BWSS09] BAI X., WANG J., SIMONS D., SAPIRO G.: Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graph. 28*, 3 (2009), 1–11. 3

[CCSS01] CHUANG Y.-Y., CURLESS B., SALESIN D. H., SZELISKI R.: A bayesian approach to digital matting. In *CVPR* (2001), vol. 2, pp. 264–271. 2

[CTP*08] CRABB R., TRACEY C., PURANIK A., DAVIS J., SANTA CRUZ C., CANESTA I., SUNNYVALE C.: Real-time foreground segmentation via range and color imaging. In *CVPR Workshop on Time-of-flight Computer Visionn* (2008), pp. 1–5. 2

[GSAW05] GRADY L., SCHIWIETZ T., AHARON S., WESTERMANN R.: Random walks for interactive alpha-matting. In *VIIP* (2005), pp. 423–429. 2

[LLW08] LEVIN A., LISCHINSKI D., WEISS Y.: A closed-form solution to natural image matting. *TPAMI 30*, 2 (2008), 228–242. 1, 2, 3, 9

[Mis94] MISHIMA Y.: Soft edge chroma-key generation based upon hexoctahedral color space, Oct. 11 1994. US Patent 5,355,174. 2

[MMP*05] MCGUIRE M., MATUSIK W., PFISTER H., HUGHES J. F., DURAND F.: Defocus video matting. *ACM Trans. Graph. 24*, 3 (2005), 567–576. 2

[MYT95] MITSUNAGA T., YOKOYAMA T., TOTSUKA T.: Autokey: Human assisted key extraction. In *SIGGRAPH* (1995), pp. 265–272. 4

[OW04] OMER I., WERMAN M.: Color lines: Image specific color representation. In *CVPR* (2004), vol. 2, pp. 946–953. 4

[RRG08] RHEMANN C., ROTHER C., GELAUTZ M.: Improving color modeling for alpha matting. In *BMVC* (2008), pp. 1155–1164. 1, 2, 9

[RRRAS08] RHEMANN C., ROTHER C., RAV-ACHA A., SHARP T.: High resolution matting via interactive trimap segmentation. *CVPR 1* (2008), 1–8. 3

[RRW*09] RHEMANN C., ROTHER C., WANG J., GELAUTZ M., KOHLI P., ROTT P.: A perceptually motivated online benchmark for image matting. In *CVPR* (2009). 2, 7, 8, 9, 10

[RT00] RUZON M., TOMASI C.: Alpha estimation in natural images. In *CVPR* (2000), vol. 1, pp. 18–25. 2

[SJTS04] SUN J., JIA J., TANG C.-K., SHUM H.-Y.: Poisson matting. *ACM Trans. Graph. 23*, 3 (2004), 315–321. 2

[SRR09] SINGARAJU D., ROTHER C., RHEMANN C.: New appearance models for natural image matting. In *CVPR* (2009). 1

[WAC07] WANG J., AGRAWALA M., COHEN M. F.: Soft scissors: an interactive tool for realtime high quality matting. In *SIGGRAPH* (2007), p. 9. 3, 9

[WC07a] WANG J., COHEN M. F.: Image and video matting: a survey. *Foundations and Trends in Computer Graphics and Vision 3*, 2 (2007), 97–175. 1

[WC07b] WANG J., COHEN M. F.: Optimized color sampling for robust matting. *CVPR* (2007), 1–8. 1, 2, 3, 4