

Soft Scissors : An Interactive Tool for Realtime High Quality Matting

Jue Wang
University of Washington

Maneesh Agrawala
University of California, Berkeley

Michael F. Cohen
Microsoft Research



Figure 1: Our system computes a high quality matte (a) and a novel composite (b) in realtime as the user roughly paints the foreground boundary. Our system makes it easy to create new composites (c) very quickly.

Abstract

We present *Soft Scissors*, an interactive tool for extracting alpha mattes of foreground objects in realtime. We recently proposed a novel offline matting algorithm capable of extracting high-quality mattes for complex foreground objects such as furry animals [Wang and Cohen 2007]. In this paper we both improve the quality of our offline algorithm and give it the ability to incrementally update the matte in an online interactive setting. Our realtime system efficiently estimates foreground color thereby allowing both the matte and the final composite to be revealed instantly as the user roughly paints along the edge of the foreground object. In addition, our system can dynamically adjust the width and boundary conditions of the scissoring paint brush to approximately capture the boundary of the foreground object that lies ahead on the scissor’s path. These advantages in both speed and accuracy create the first interactive tool for high quality image matting and compositing.

1 Introduction

In the foreground matting problem an input image C is formulated as a convex combination of a foreground image F and a background image B as $C_p = \alpha_p F_p + (1 - \alpha_p) B_p$, where p refers to pixel locations, and α_p is the foreground opacity of the pixel. Once F_p and α_p are determined, a novel composite can be created by substituting B_p with a new background B'_p .

However, solving for both F_p and α_p from a single observation C_p is an underspecified problem. Thus, most previous matting algorithms require the user to roughly segment the image into a trimap in which pixels are marked as definitely belonging to the background, definitely belonging to the foreground, or unknown. These algorithms then use information from the known background and foreground regions to compute a matte. If the initial results are not satis-

factory (which is often the case), the user must then refine the trimap and run the algorithm again until the process converges. This process is usually very inefficient for the user.

Recent matting algorithms focus mainly on improving the quality of the matte by introducing more sophisticated analysis and optimization methods. However they are generally slow to compute a matte. As a result, the wait time between each iteration of the interactive loop described above can be very long. For instance, Bayesian matting [Chuang et al. 2001] takes 141 seconds of computation time to generate a result for the example shown in Figure 1. Also, these techniques recompute the whole matte on each iteration and there is no good strategy to update the matte incrementally. On the other hand, earlier approaches such as the Knockout 2 system [2002] are extremely simple and fast, but are not capable of generating high quality mattes for complex images.

Our aim is to provide a tool that can generate high quality mattes in realtime. In our system the user roughly specifies the foreground boundary using an intelligent paint stroke (or *soft scissor*). The system automatically updates the matte and foreground colors according to the newly-added information along the stroke to instantly reveal a local region of the final composite. The composite shown in Figure 1 took about 40 seconds of total interleaved user and computation time.

Our interactive system extends an offline robust matting algorithm we recently proposed [Wang and Cohen 2007], which is capable of extracting high quality mattes for difficult foreground objects such as furry animals¹. In adapting this algorithm to the realtime setting, we make three new contributions:

Incremental matte estimation. Based on newly-added user strokes, the system first determines the minimal number of pixels that need to be updated, and then computes their new alpha values. This is presented in Section 3.4.

Incremental foreground color estimation. In addition to alpha values, our system also incrementally computes the foreground colors for mixed pixels so the final composite can be updated immediately. This is presented in Section 3.3.

Intelligent user interface. The soft scissor width and the boundary conditions are automatically adjusted to approximately capture the boundary that lies ahead on the scissor’s path. This is presented in Section 4.

¹We will briefly describe the offline robust matting algorithm in Section 3.2.

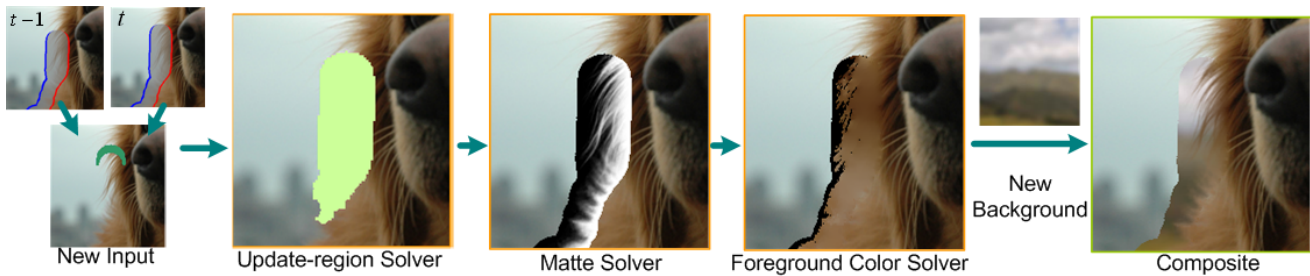


Figure 2: A flowchart of our system.

By combining these three novel elements and the robust matting algorithm, we demonstrate the first system to generate high quality mattes and composites in realtime.

2 Related Work

Binary Cutout. Classic binary segmentation approaches include region-based methods such as Photoshop’s magic wand [INCORP. 2002], and boundary-based systems such as intelligent scissors [Mortensen and Barrett 1995]. Recently the LazySnapping [Li et al. 2004] and GrabCut [Rother et al. 2004] systems have employed graph-cut optimization to achieve more coherent and higher quality foreground segmentation. However, none of these approaches deal very well with large amounts of partial foreground coverage.

Foreground Matting. Many matting techniques have been designed to deal with boundaries of fuzzy foreground objects such as hair and fur. Chuang et al. [2001] proposed Bayesian matting, which formulates the problem in a well-defined Bayesian framework and solves it using MAP estimation. The iterative matting system [Wang and Cohen 2005] solves for a matte directly from a few user specified scribbles instead of a carefully specified trimap. The Poisson matting algorithm [Sun et al. 2004] assumes the foreground and background colors are smooth. Thus, the gradient of the matte matches with the gradient of the image and can be estimated by solving Poisson equations. The closed-form matting [Levin et al. 2006] approach assumes foreground and background colors can be fit with local linear models, which leads to a quadratic cost function in α that can be minimized globally. Unlike our system, all of these approaches work in an offline fashion and generally require long processing times.

3 The Soft Scissors Algorithms

3.1 Overview

Our system updates the matte in realtime while the user roughly paints a scissor stroke along the boundary of the foreground object. A flowchart of each internal iteration of our system is shown in Figure 2. We assume that the scissor stroke implicitly defines a trimap, usually with the left edge of the stroke assumed to lie in the background (blue pixels), the right edge assumed to lie in the foreground (red pixels), and the middle of the stroke unknown (gray pixels). Both the boundary conditions and the width of the scissor stroke can be set manually by the user or dynamically adjusted by our system based on an analysis of the image statistics (see Section 4).

On each iteration the system determines which pixels were painted since the previous iteration. This new input region, M_t (shown in dark green) affects the alpha values of surrounding pixels in two ways. First, newly marked foreground and background pixels provide more foreground and background color examples, and also set new boundary conditions for the local image area. In addition, the newly marked unknown pixels are likely to be correlated with nearby pixels. Therefore the alpha values of the newly marked pixels should affect all of the correlated pixels which were previously marked as unknown. To determine the pixels that are affected by

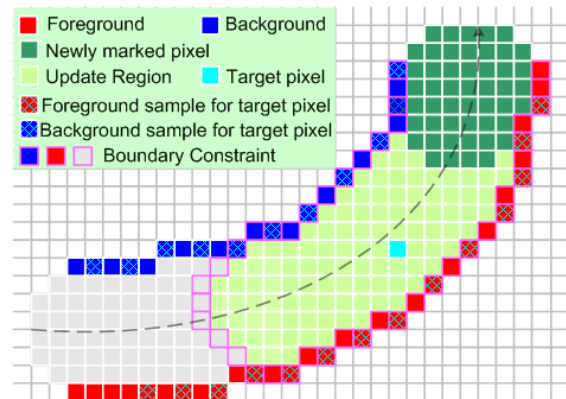


Figure 3: Our system quickly solves the matte under the leading edge of the soft scissors, constrained by boundary pixels.

the new input region, we use the input region to seed an update-region solver (see Section 3.4) which computes a small region of pixels (shown in light green) for which the alpha values need to be updated. The matting region Ω_t (including both the dark green and light green regions) is generally much smaller than the whole unknown region in the trimap and therefore solving the matte is significantly more efficient than re-calculating the whole unknown region in each iteration.

We estimate the alpha values for pixels in Ω_t using a robust matte solver (see Section 3.2). By treating pixels outside Ω_t as boundary conditions, the solution is guaranteed to be smooth across the boundary of Ω_t . Finally, the foreground colors of pixels in Ω_t are updated by a foreground color solver that uses the newly computed alpha values (see Section 3.3). We then display the updated composite in Ω_t .

3.2 Solving for the Matte

The central component of our soft scissors system is a robust matting algorithm we recently proposed [Wang and Cohen 2007]. For completeness we briefly summarize the algorithm here.

As illustrated in Figure 3, assume that we have already computed the matting region Ω_t (shown in light and dark green). We treat the problem of solving for α in this region as a soft graph-labeling problem. We use the graph structure shown in Figure 4(a), where Ω_F and Ω_B are virtual nodes representing pure foreground and pure background, white nodes represent unknown pixels in the image, and light red and light blue nodes are boundary nodes whose alpha values are fixed in this iteration. The boundary nodes for this graph include not only user marked foreground and background pixels, but also unknown pixels on the boundary of Ω_t whose alpha values have been estimated in previous iterations. In this way we ensure the matte is smooth across the entire boundary of the matting region.

We selectively sample a group of known foreground and back-

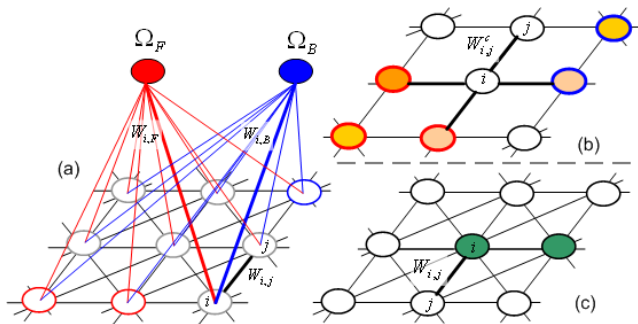


Figure 4: The matte (a), foreground colors (b) and the update region (c) are solved as soft graph-labeling problems.

ground pixels from the boundary of the trimap to compute non-parametric models of the foreground and background color distributions. We then assign *data weights* $W_{i,F}$, $W_{i,B}$ between pixel i and the virtual nodes based on these distributions. The data weights constrain pixels that are similar in color to the foreground(background) to have a stronger $W_{i,F}$ ($W_{i,B}$) and therefore make them more likely to have a higher(lower) alpha values. We use the formulation proposed in the closed-form matting paper [Levin et al. 2006] to set the *edge weights* $W_{i,j}$ between each pair of neighboring pixels i and j . Note that each pixel is connected to its 25 spatial neighbors in this formulation. The edge weights constrain nearby pixels to have similar alpha values. Once the graph is constructed, we solve the graph-labeling problem as a Random Walk [Grady 2006], which minimizes the total graph energy over real values.

Intuitively the Random Walk solver determines the alpha values by placing a random walker at pixel i that can walk to any neighboring node j (i.e. any node connected to i including the two virtual nodes) with probability $W_{i,j}/\sum_j W_{i,j}$. The walker then moves from j to another neighbor k in the same manner and this process iterates until the walker reaches one of the boundary nodes. The probability that the walker ends up at the foreground virtual node determines the alpha value of pixel i . This probability can be naively estimated by simulating the random walk process a large number of times, and counting how many times it arrives at the foreground node. In practice, however, we calculate the unknown alphas in closed-form by solving a large linear system using the random walk algorithm outlined in [Grady 2006].

3.3 Solving for the Foreground Colors

In addition to computing alpha values we also estimate the true foreground color F for each pixel in the unknown region. This allows the foreground to be composed onto a new background without bringing the colors of the old background into the new composite. Although we select a few foreground samples for each pixel in the matte estimation step, these samples are chosen individually without enforcing smoothness constraints. As a result, after the matte estimation step, the composite may contain visual artifacts, as shown in Figure 5.

To achieve higher quality composites, we refine the estimated foreground colors by solving a second graph-labeling problem using Random Walk, as shown in Figure 4(b). Only those pixels in Ω_t whose alpha values are strictly between 0 and 1 are treated as unknown pixels in this step, and each unknown pixel is connected to its 4 spatial neighbors. We define a color edge weight $W_{i,j}^c$ between two neighbors as $W_{i,j}^c = |\alpha_i - \alpha_j| + \epsilon$, where ϵ is a small value ensuring the weight is greater than zero. This edge weight encodes explicit smoothness priors on F , which are stronger in the presence of matte edges (where α_i and α_j have a larger difference).

The boundary pixels in this step are either foreground pixels($\alpha = 1$) or background pixels($\alpha = 0$). For foreground pixels(red out-

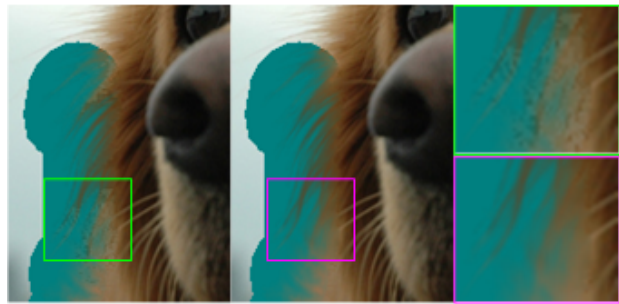


Figure 5: Left: Initial estimates of foreground colors after the matte estimation step; Right: Final foreground colors after optimization.

line nodes in Figure 4(b)), we use their true colors as boundary conditions, while for background pixels(blue outline nodes in Figure 4(b)), we use their initially estimated foreground colors in the matte estimation step as boundary conditions. The initial estimates are shown as the node colors in Figure 4(b). We then solve for the three foreground color channels individually using the Random Walk solver.

3.4 Solving the Update Region

A key feature of our system is that it is incremental – we only update the alpha and foreground colors for a small portion of the image on each iteration. Given a new input region we compute the set of pixels that might be affected by the new information as the update region Ω_t . To determine the update region, Ω_t , we again solve a graph-labeling problem as shown in Figure 4(c). All pixels that have been newly marked by the user in the current iteration are treated as boundary pixels with an assigned label of 1 (the dark green nodes in Figure 4(c)). Note that in this step the label does not correspond to the alpha value of the pixel, but rather represents the *impact* of the new input region on the pixel. All other pixels that were marked in previous iterations are treated as unknown pixels in this step (white nodes in Figure 4(c)). Similar to the alpha estimation graph in Figure 4(a), each pixel is connected to its 25 spatial neighbors with the same edge weights $W_{i,j}$ as we defined in the matte estimation step. We again solve the graph using the Random Walk solver. Each pixel is assigned a label measuring how much impact it receives from the new input region. Those pixels assigned non-zero (in practice greater than a small threshold $\delta = 1/255$) labels form the new update region Ω_t .

Intuitively, the Random Walk solver determines how far potential changes of alpha values due to the newly marked pixels should be propagated towards the boundary of the image. A smoother local image region will result in a larger Ω_t since the weights between neighboring pixels are high, and vice versa. This solver is similar in spirit to the region solvers employed in the interactive tone mapping system [Lischinski et al. 2006], but with different graph topologies and edge weights.

4 The Soft Scissor Interface

As the user paints along the boundary of the foreground object our system dynamically adjusts two properties of the Soft Scissors brush; 1) brush width and 2) boundary conditions for the trimap that is implicitly defined by the brush strokes. The adjustments are based on local statistics near the current brush stroke. In addition, users can manually adjust these parameters if necessary.

4.1 Choosing the scissor brush width

Wider scissors are appropriate for object edges that are very fuzzy while narrower scissors are better for sharper edges as they provide tighter bounds on the solution and greatly improve computation ef-

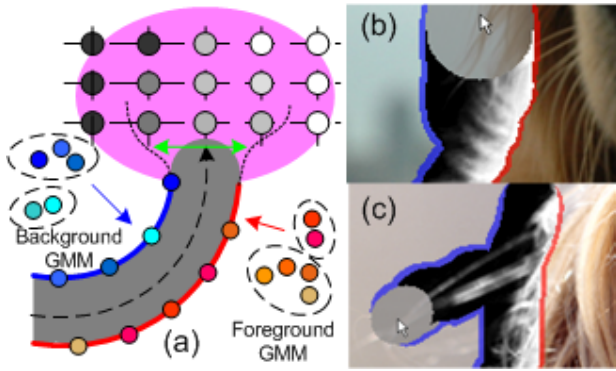


Figure 6: (a). Our system can automatically determine the soft scissor width and boundary conditions. (b). An example of enlarging the width to cover the mixed foreground/background region. (c). An example of changing the boundary condition.

efficiency. We automatically determine brush width as shown in Figure 6(a). At each time t , we first compute the current scissor path direction, then create a wide “look-ahead” region (shown in purple) by extending the current path of the scissor along that direction. The width of the “look-ahead” region is fixed to a maximum value set by the user (generally 60 pixels in our system) so it can capture almost all types of edges. We treat all pixels in this region as unknowns and include them in Ω_t for alpha estimation. Then, to estimate the matte profile we sample a group of pixels sparsely distributed along lines perpendicular to the current scissor path direction (shown as dash black lines in Figure 6(a)). The scissor width is set so that it covers all of the sample pixels with fractional alpha estimates.

Specifically, for each sampled point on a line we first compute a weight as $w_p = 0.5 - |\alpha_p - 0.5|$, where α_p is the estimated alpha value of the point. Then the center of the 1D distribution along the line is computed as $\bar{x} = \sum_p w_p x_p / \sum_p w_p$, where x_p is the distance from a sampling point to the extended scissor path. The width of the alpha profile is then estimated as $4\sqrt{\sum_p w_p (x_p - \bar{x})^2 / \sum_p w_p}$.

4.2 Determining the boundary conditions

We initially assume that the user orients the scissor brush strokes so that the left edge of the brush is in the background region and the right edge of the brush is within the foreground region. However, at times users need to follow thin structures (e.g. single hairs). In this case they require a brush for which both sides are marked as background as in Figure 6(c). In other situations users will paint back and forth over the foreground object and the brush must be able to reverse the background/foreground edges as the user reverses the brush direction.

We dynamically adjust the scissor boundary conditions by building color models of the foreground and background colors. Once the user has created a short stroke and marked enough foreground/background pixels under the initial assumptions we use Gaussian Mixture Models (GMM) to build foreground and background color models. Each GMM has 5 components with full covariance matrices. Then, for each new brush position we classify the brush edges based on whether their average color is closer to the foreground or background GMM. Specifically, as shown in Figure 6(a), we sample a group of pixels along the newly added left edge, and compute a foreground and background probability ψ_l^F and ψ_l^B by fitting samples with foreground and background GMMs, and normalize them so they sum to 1. We set the left edge to be foreground if $\psi_l^F > \psi_l^B + \delta_\psi$, or background if $\psi_l^B > \psi_l^F + \delta_\psi$, where δ_ψ is a difference threshold we typically set at 0.3. If the difference

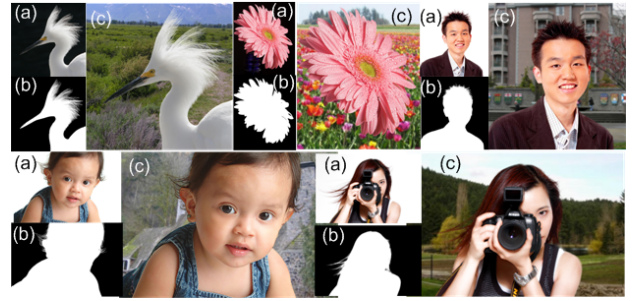


Figure 7: Test data set. (a). Original image. (b). Ground-truth matte. (c). Target image on which we apply matting algorithms.

between the two probabilities is smaller than δ_ψ , we then keep the current boundary condition unchanged. We classify the right edge in the same way.

In addition, the GMMs are updated periodically using recently marked foreground and background pixels. After a set number (typically 400) of new foreground and background samples are marked by the user, we re-compute the GMMs using the new samples. An example of dynamically changing brush condition is shown in Figure 6(c).

4.3 Automatic vs. Manual Parameter Selection

Automatic adjustment of brush parameters will not always be optimal, especially for images with high-frequency textures and complex foregrounds. To minimize abrupt, erroneous parameter changes we constantly monitor the automatically estimated brush width over a short period of time ($t - \delta_t, t$). If the variance of the estimated width is large, the estimate is not considered reliable and the width is left as is. The user can always manually adjust the brush width. Similarly, we discard the automatically determined boundary conditions if $|\psi^B - \psi^F| < \delta_\psi$ as described in previous section. Again, the user can set the appropriate conditions. At any time the user can disable either automatic algorithm to regain full control over the brush parameters.

5 Results and Evaluation

The images in Figure 1 show one example of the Soft Scissors in use. Figure 8 shows three more challenging examples of our system running on complex images and the resulting high quality mattes. We refer the readers to the accompanying video for a better demonstration of the realtime performance of our system.

The system not only runs in realtime, but also generates higher quality results than previous approaches. To evaluate our system, we constructed a test data set of 5 examples as shown in Figure 7. Each foreground object was originally shot against a solid colored background and we extracted a high quality matte using Bayesian matting. We used the resulting matte as the ground-truth and composited the foreground object onto a more complex background to synthesize a “natural” image as a test image. Finally we applied various matting approaches on these test images.

We compare mattes extracted using Soft Scissors with five previous matting approaches: Bayesian matting [2001], iterative BP matting [2005], closed-form matting [2006], knockout 2[2002] and global Poisson matting [2004]. We also compare the mattes with our offline Robust Matting approach [Wang and Cohen 2007]. All of these techniques were run using the same trimap created using our interactive system. Figure 9 shows the Mean Squared Error(MSE) of the extracted mattes against the ground-truth. Two visual examples are shown in Figure 10. These results suggest that our system extracts mattes with the highest quality. Note that Soft Scissors generates slightly better results than our previous offline robust matting approach [Wang and Cohen 2007]. The total processing time of



Figure 8: Three examples. From left to right: original image, snapshots of extracting the matte in realtime, and the new composite.

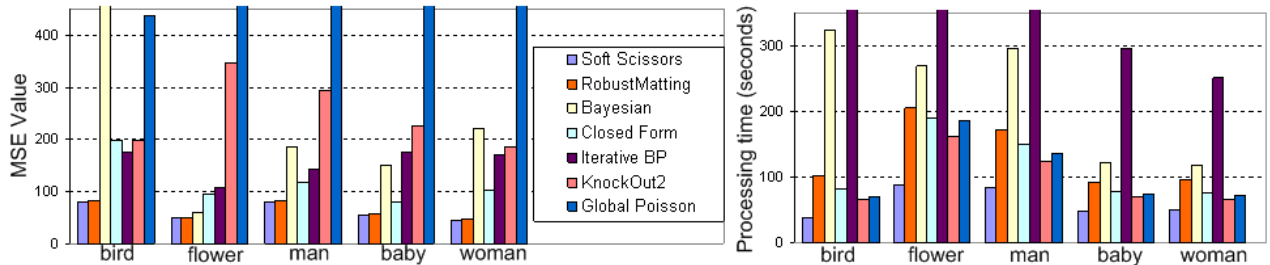


Figure 9: Comparing different algorithms on the data set in terms of matte errors and processing time.

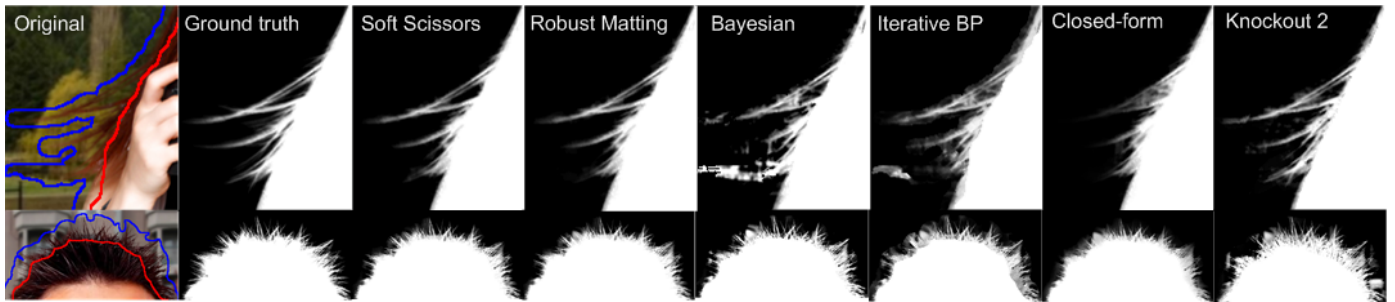


Figure 10: Partial results on test image “man” and “woman” in Figure 7.

different approaches are also shown in the figure. For fair timing comparisons we fixed the total number of iterations in the offline approaches to 4 (this means modify the trimap and run the algorithms 4 times). Not surprisingly, Soft Scissors takes the least amount of time to extract high quality mattes.

Our system also works well with more solid foreground objects. In Figure 11 we compare our algorithm with binary cutout tools on extracting the rabbit. Note that the results generated by Intelligent Scissors and GrabCut suffer from inaccuracies along the boundary

as well as “color bleeding”, where the boundary pixels represent a mixed foreground-background color due to partial pixel coverage (see the greenish boundary pixels in the binary results). In contrast, our system is able to fully extract the rabbit without such artifacts.

6 Preliminary User Study

We conducted an informal usability study comparing our system with Bayesian matting, Knockout 2 and our previous offline robust

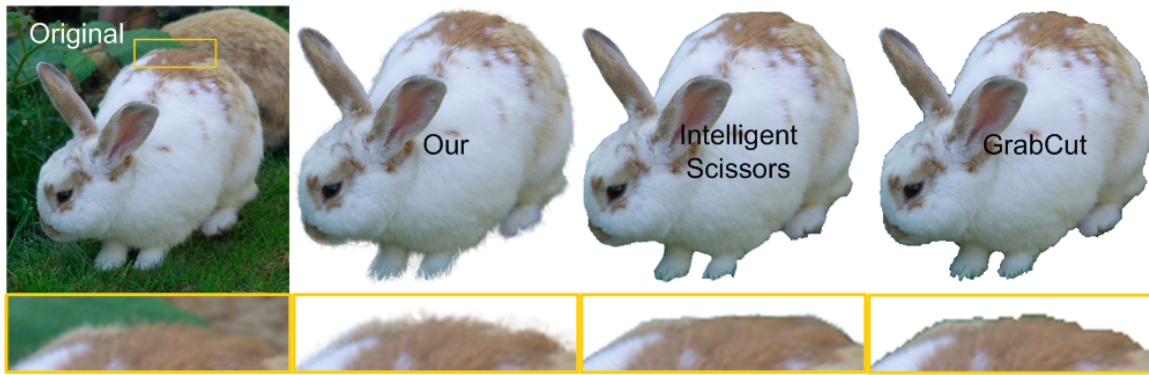


Figure 11: Comparing our system with Intelligent Scissors and GrabCut on extracting the foreground rabbit.

matting system. Three subjects, none of which had experience using any of the matting systems, were first instructed how to use each of the systems, and practiced using each of them for 10 minutes. These users were then requested to extract the foregrounds from three images: “bird”, “baby” and “man” (see Figure 7a-c). They worked for as long as they wanted until they felt they could not generate more accurate mattes.

We collected three types of data from these three users; the total time they spent, the error in the final mattes they generated, and their subjective preferences for each interface (a score for each interface on a scale from 1=worst to 5=best). The results are shown in Figure 12. Users found it difficult to achieve good results for the “bird” image using Bayesian matting and gave it the lowest subjective preference rating. Although soft scissors and offline robust matting generated similar quality results, all users gave soft scissors higher scores because it was faster to use and provided realtime feedback. While these results clearly suggest that soft scissors provides an effective interface for foreground matting, a more extensive and formal user study would be required to draw solid quantitative conclusions.

7 Conclusion

We have demonstrated the first realtime tool for generating high quality mattes and composites as the user roughly paints along the foreground boundary. The scissor brush width and boundary conditions adjust automatically as the user draws the scissor stroke. Our evaluation demonstrates that Soft Scissors outperform previous matting techniques both in quality and efficiency.

Currently we rely on the user to trace the foreground edge. One could imagine a hybrid system that first performs a quick binary segmentation to further guide the user and the underlying algorithms. However, for still images the speed and simplicity of the current approach may not warrant the complexity such an approach would add to the system. For video matting however, some hybrid of a fully automated and a user guided system will be needed. We feel the soft scissors approach can provide the basis for the user guided aspect of such a system.

References

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A bayesian approach to digital matting. In *Proceedings of IEEE CVPR*, 264–271.

CORPORATION, C. 2002. Knockout user guide.

GRADY, L. 2006. Random walks for image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

INCORP., A. S. 2002. Adobe photoshop user guide.

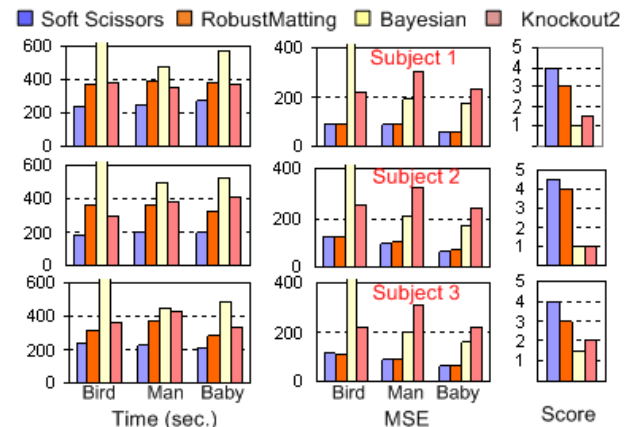


Figure 12: Results of the user study.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2006. A closed form solution to natural image matting. In *Proceedings of IEEE CVPR*.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. In *Proceedings of ACM SIGGRAPH*, 303–308.

LISCHINSKI, D., FARBMAN, Z., UYTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. In *Proceedings of ACM SIGGRAPH*.

MORTENSEN, E., AND BARRETT, W. 1995. Intelligent scissors for image composition. In *Proceedings of ACM SIGGRAPH*.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. Grabcut - interactive foreground extraction using iterated graph cut. In *Proceedings of ACM SIGGRAPH*, 309–314.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. In *Proceedings of ACM SIGGRAPH*, 315–321.

WANG, J., AND COHEN, M. 2005. An iterative optimization approach for unified image segmentation and matting. In *Proceedings of ICCV 2005*, 936–943.

WANG, J., AND COHEN, M. F. 2007. Optimized color sampling for robust matting. In *Proceedings of IEEE CVPR*.