Fast Adaptive Edge-Aware Mask Generation

Michael W. Tao*
University of California, Berkeley

Aravind Krishnaswamy†
Adobe Systems, Inc.

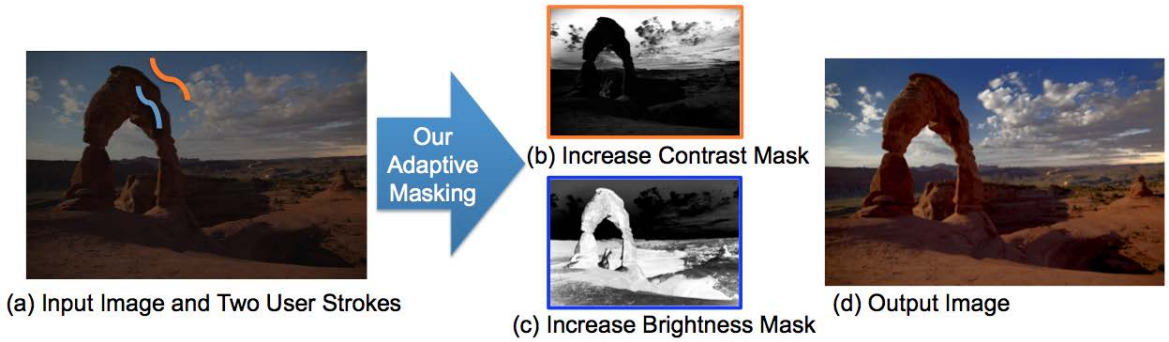


Figure 1: *An example of complex editing*. The user uses one stroke to increase local contrast and another to increase local brightness on the original image (a). With just two strokes, our adaptive masking computes editing masks to the respective user-strokes as shown in (b) and (c), resulting the output image (d). Our paper focuses on producing high quality results in an efficient framework with low user intervention.

ABSTRACT

Selective editing, also known as masking, is a common technique to create localized effects, such as color (hue, saturation) and tonal management, on images. Many current techniques require parameter tuning or many strokes to achieve suitable results. We propose a fast novel algorithm that requires minimal strokes and parameter tuning from users, segments the desired selection, and produces an adaptive feathered matte. Our approach consists of two steps: first, the algorithm extracts color similarities using radial basis functions; second, the algorithm segments the region the user selects to respect locality. Because of the linear complexity and simplicity in required user-input, the approach is suitable for multiple applications including mobile devices.

Index Terms: I.4.6 [Image Processing and Computer Vision]: Segmentation—Region growing, partitioning

1 INTRODUCTION

It is common in image editing for users to localize their edits such as color modification and tonal management in their digital artwork. Global edits offer less control and may cause undesirable effects in certain regions. Masking, which is used as an alpha channel to blend the original and desired image edits, assists users with localized edits. Two main components of image masking are finding color and local (spatial) correspondence. The goal of this paper is to introduce a novel algorithm that produces a mask with the two correspondences in an efficient and low-user intervention framework, suitable for mobile devices or quick-editing software. In previous works, techniques use coefficient fall-offs such as Gaussians to compute both correspondences. However, Gaussian fall-offs require the algorithm to choose one specific comparison color to calculate the coefficients of surrounding pixels and Gaussian sigmas for both color and spatial constraints. To extract one comparison color, algorithms usually use the mean of pixel values inside the user-selection. Because of the averaging, one specific color would require tolerance adjustments and many strokes to complete the desired selection. Spatial and color tolerances require the users to adjust these parameters to achieve desired effects. We present an algorithm that does not require parameter tuning and multiple user-strokes. We use radial basis functions (RBF) as described by Li et al. [16], which account for all pixels within a stroke, reducing the amount of user strokes required to obtain desirable results (Figure 2). For locality constraints, previous works change Gaussian locality sigmas to produce a smooth fall-off and some use multiple strokes to produce similar constraints. We compute the locality term using an efficient texture analysis. With our algorithm, we are able to produce soft masking with local contraints. The desired outputs require minimal strokes and the framework has a linear complexity relative to the number of pixels in the image, suitable for many graphics and editing applications that require real-time feedback.

*e-mail: mtao@berkeley.edu
†e-mail: aravind@adobe.com

Graphics Interface Conference 2012
28-30 May, Toronto, Ontario, Canada

2 RELATED WORK

Masking techniques are useful in many image editing applications, including re-coloring, image segmentation, and video modifications [2] [6] [23]. The authors have proposed different methods, which generally require parameter tuning and additional strokes to achieve desirable results. A summary of key benefits from our work is shown in Figure 3.

2.1 Gaussian Correspondence.

Gaussian correspondence is commonly used in many available tools today such as Apple® Aperture Edge-Aware Brush, Adobe® Photoshop Color Replacement, and Adobe® Lightroom Adjustment Brush Auto-Mask. The techniques are adapted from the edge preserving filters such as the bilateral filter introduced by [25]. The filter has two terms: spatial and color constraints, which are formulated in Equation 4. Calculating the Gaussian color constraint requires a single color target. To extract the target color, many of the tools use an average of the selected pixels and compute the Gaus-

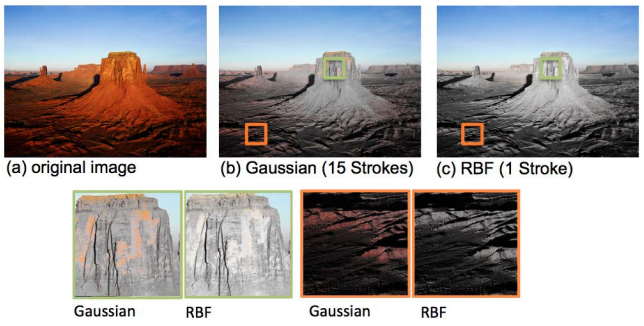


Figure 2: *Gaussian vs. RBF: desaturation example.* With a high textured region input (a), using a Gaussian fall-off causes misses in many colors and would require many strokes from the user (b). RBF accounts for all colors within the stroke, providing a more desirable result in textured areas (c).

Algorithm	Interactive (High Megapixels)	Low Parameter Tuning	Low User Strokes	Adaptive Feathered Result
Our Algorithm	✓	✓	✓	✓
Geodesics [Bai and Sapiro 2011] [Criminisi 2011]	✓		✓	
Scribbleboost [Li 2008]		✓		
Min-Cut [Boykov 2001]		✓		
Photoshop ® Color Replacement (Gaussian)	✓			
Photoshop ® Quick Select (Graph Cuts)		✓		
Photoshop ® Magic Wand (Flood Fill)	✓		✓	

Figure 3: *Algorithm comparisons.* Compared to current state-of-the-art algorithms, our algorithm shows several benefits.

sian coefficient for nearby pixels. Because textured regions consist of multiple colors, averaging user input stroke colors reduces effectiveness. The user would be required to make multiple strokes in textured regions to create a complete mask. Another solution to the problem is to increase the Gaussian color sigma, which corresponds to the tolerance of the color constraint; however, tweaking such parameters and changing brush properties require user intervention. In Figure 2, we illustrate the ineffictiveness of Gaussian coefficients in textured regions.

2.2 Gradient and Graph Cut Based.

Gradient and graph cut based techniques are commonly used for masking because of their ability to achieve high quality results. Because of the required linear solver and graph construction, these techniques are computationally expensive. These algorithms require many user strokes in selections with non-prominent edges (illustrated in Figure 11). Wang et al. [27] present a way for users to stroke at subject edges for producing an accurate matte. However, this technique requires the user to stroke around the subject. [4], [15], and [17] use similar concepts but require many user strokes to obtain desired results. More advanced techniques using graph cuts require high computation time or optimization ([5], [22], [23], Adobe® Photoshop Quick Select).

2.3 Flood Fill Techinques.

Flood fill techniques are known to be very fast and scalable for many applications as presented in the Photoshop magic wand tool and [18]. However, due to its binary nature, hard edges become very apparent and are not suitable in areas where smooth fall-offs (feathering) are essential. In Figure 8, the magic wand tool produces seams in sky regions. In many situations, the user is required to adjust the tolerance parameter to achieve a desired selection. Moreover, flood fill techniques perform poorly in textured regions, due to their limited ability of expanding only using color tolerances. We extend the flood fill techniques to improve our results.

2.4 Constraint Propagations.

Another method is to use constraint propagations such as radial basis functions or other defined domains to obtain quality matting. An and Pellacine [1] show an analysis of a Z matrix domain to find edges of an image. However, due to its dense matrix calculations, the method is computationally expensive and requires large amounts of memory. Farbman et al. [9] present a method that uses diffusion maps to obtain quality results. However, due to the sampling requirements of the diffusion maps, high quality results require more samples. The algorithm computation time scales linearly as we increase the number of samples. Li et al. [16] present an algorithm that propagates edits efficiently by using radial basis functions. However, the technique presents instabilities in outputs when the user selects textured regions (Figure 9). We expand on the technique to increase stability and reduce the number of samples needed to achieve consistent results.

2.5 Geodesics.

Geodesic frameworks have shown desirable results at interactive times [3] and [8]. These methods require a stroke within the background and foreground as opposed to our only requirement of the foreground stroke. Because of its path-finding use of Geodesics, these methods perform poorly with user selection error. In our paper, we alleviate user selection errors, which are common in touch devices, through our sampling method described in Section 3.2.2. We show that these methods still fail where colors are similar and when edges are not apparent. Because of our locality constraint and adaptive feathering, we are able to avoid such problems (Figure 11). To achieve the claimed performances, Geodesic frameworks rely heavily on parallelized architectures, which many mobile devices still lack.

2.6 Trimap Techniques.

Tripmap techniques require the user to stroke through the border between the background and foreground. Chuang et al. use a Bayesian probabilistic model for matting [7]. Due to its probabilistic propagation, such methods are computationally expensive. Recent probabilistic models produce great matting results at interactive times; however, these techniques require many long user-strokes to produce a trimap for matting [10], [12], and [21]. These algorithms are also complimentry to our work, as we can use our methods to produce trimaps with a small amount of strokes. We compare our results with [3] and [8] which use trimap techniques to achieve soft-matting.

2.7 Image Analogy and Training.

Many global training and other image techniques have been discussed [6], [11], [13], [20], and [26]. Because of the globalized nature of these edits, they are complimentry to our work as the users can use such results and blend them with the original image using our masking techniques. In this paper, we do not claim novelty in automated edits or color manipulation. We present a novel algorithm for masking that allows the users to make local edits.

3 Algorithm

3.1 Overview

In this paper, we introduce several contributions.

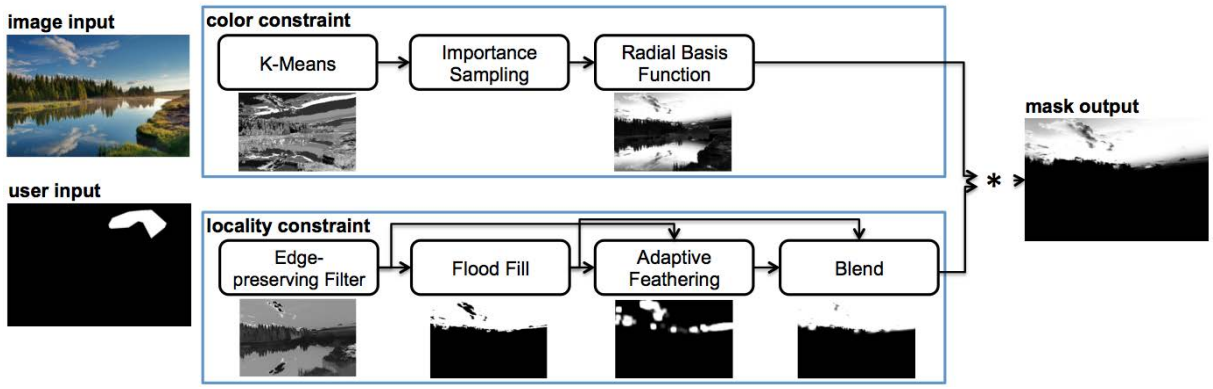
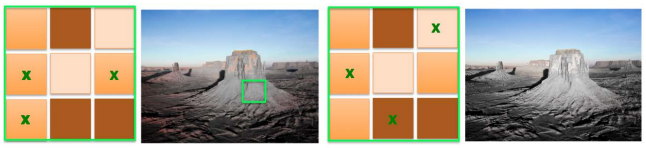


Figure 4: *Algorithm pipeline.* There are two inputs: the original image and the user stroke (mask input). The two main algorithm components consist of calculating a mask with color constraint and locality constraint. In the color constraint stage, we use k-means clustering to drive the importance sampling of the colors, which contributes to the radial basis function mask. For locality constraint, we reduce the impact of textured regions by using a bilateral filter and use the resulting hue channel for the remaining steps. We then apply a flood fill algorithm onto the bilateral filtered hue channel, which produces an initial locality mask. The binary output of the flood fill algorithm produces seams in low frequency regions. We use an adaptive feathering to reduce seams. Please refer to Section 3 for complete details and analysis on the pipeline.



(a) Random Sampling (3 green samples)

(b) Our Importance Sampling (3 green samples)

Figure 5: *RBF with too few samples.* The bright green area represents the user selected area. Performing the same desaturation operation as Figure 2, using too few samples in random sampling may cause under-represented results (a) and inconsistent variations as seen in Figure 9. With our importance sampling scheme, colors within textured regions are represented in the sampling, producing more consistent and favorable results (b).

Texture Reduction Technique for Segmentation. We describe an intuitive method to reduce impacts of texture for flood fill algorithms.

Fast Adaptive Feathering. We show how our fast adaptive feathering method removes hard seams in low-textured regions while maintaining sharp-edges in textured areas.

Radial Basis Function Importance Sampling. We show how importance sampling improves the use of RBF on image editing tools.

Efficient Framework. We present an efficient linear complexity framework that provides interactive color editing for mobile and touch devices.

The algorithm outline is shown in Figure 4. We have two stages: the first is to provide a color similarity constraint based off of radial basis functions [16] and the second is to provide a locality constraint to our mask based off of fast flood-fill techniques such as the method by Liu et al. [18]. The final mask output, M_f, is the element multiplication of the two outputs of our two stages. We use multiple color spaces for our algorithm and will specificy the color space in each step. All colors are represented in float form, ranging from 0 to 1.

3.2 Color Constraint with Importance Sampling.

The goal of this stage is to provide a smooth fall-out based on a color similarity metric between the user selected pixels and every pixel in the image. We want to minimize the number of strokes and parameter tuning required from the user.

3.2.1 Radial Basis Function and Limitations.

Using radial basis function for color selection was introduced and described in detail by Li et al. [16]. The main advantage of the work is that using radial basis functions accounts for all the pixels within the user selection. To summarize, the method samples the user selected pixels and the pixels within the image. With the samples, a linear solver is used to compute the coefficients of the interpolation formulation. We used the following formulation:

$$M(X_i) = \sum_{j \in C} a_i \Phi(||X_i - X_j||) \quad (1)$$

$$\Phi(r) = \exp(-\sigma * r^2) \quad (2)$$

Where M is the RBF mask output, X_i is the YCbCr vector at pixel i, C is the sampled YCbCr vector within the user selected region, and a_i are the coefficients we solved from the linear solver.

We used $\sigma = 42.5$.

The coefficients of the radial basis function represent the importance of a color being similar to a color within the set of selected colors. The radial basis function directly computes the value of the mask. Pixels with values of 1 are most similar to the selected colors while pixels with values of 0 represent dissimilar colors. Li et al. [16] formally describe this in Section 4 of their paper.

Although the algorithm does account for all pixels within the user selection, there are two main disadvantages to the algorithm. First, to construct the radial basis function, sampling plays a large role in determining the quality of the function. With too few samples, we can see that the RBF will behave inconsistently with the same selection (Figure 5). To alleviate this problem, the number of samples must increase to maintain consistent results. However, because the algorithm uses a linear solver to compute the RBF output of each pixel, each sample is very costly. The second disadvantage is that since all pixels are randomly sampled, user selection mistakes and noise affect the results.

3.2.2 K-Means and Importance Sampling.

To reduce the number of samples needed while maintaining consistent results, we sample the colors within the selected region using importance sampling, rather than random sampling. To determine which colors best represent our samples, we use K-means clustering to quantize the image [19]. We chose k-means because other quantization schemes such as uniform quantization introduces dithered

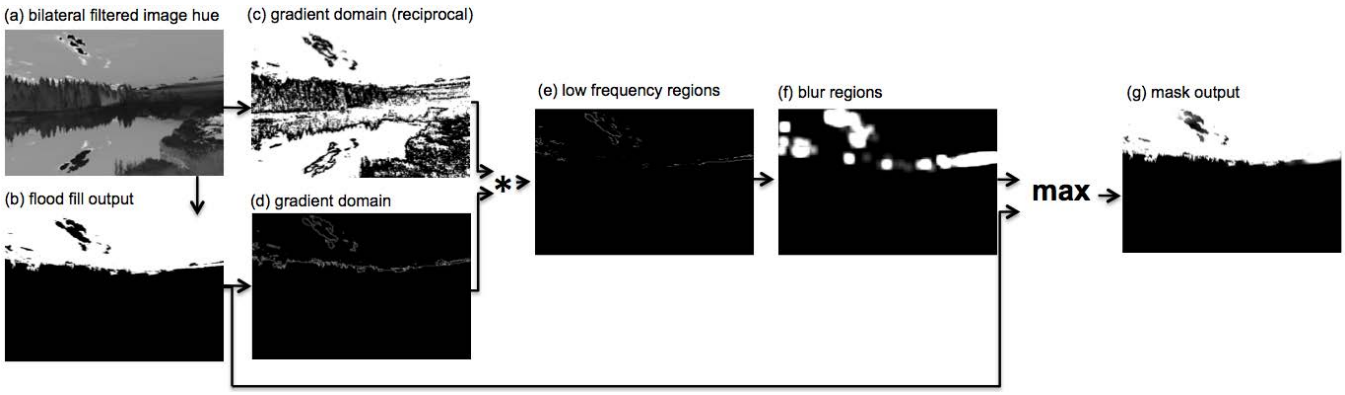


Figure 6: *Locality constraint pipeline.* The motivation is to feather regions in low frequency regions to prevent edge seams and prevent feathering in high frequency regions to reduce unwanted halos. With the bilateral filtered image hue and the magic wand output ((a) and (b)), we analyze their textures in the gradient domain ((c) and (d)). We then use the gradient outputs to determine which boundary regions of the selection are in low frequency regions (e). For selections in the low frequency regions, we perform a selective blur (f), softening the edges of the mask output, which shows smoothed selections in the low frequency regions while maintaining hard edges in high frequency regions (g).

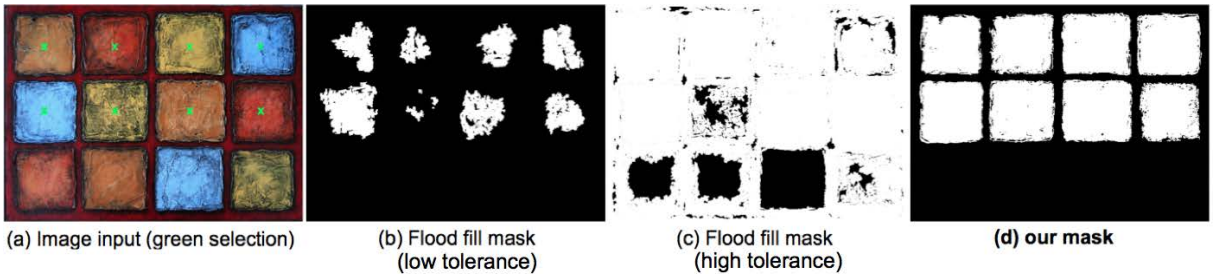


Figure 7: *Texture reduction.* We selected the eight tiles in the original input image (a). Without texture reduction, flood fill algorithms perform poorly in textured regions. The user has to modify the tolerance parameters, which shows a wide range of results ((b) and (c)). However, because of our texture reduction through the bilaterally filtered hue channel, we can see that the tiles are selected more appropriately (d).

results, creating unfavorable grouping of colors needed for our sampling. Moreover, there is more color resolution in groups within commonly appearing colors. This gives us an advantage of distinguishing colors with similar hue.

Within the user selected region, C contains clusters of G, which is a subset of all clusters within the image. For each cluster in G, we compute the number of occurances, N, of the cluster within the selected region. For each G_i, we then compute its weight.

$$W(G_i) = N_i / \sum_{j=0}^{n} N_j \tag{3}$$

where n is the number of unique clusters within the selected region. We then sample the pixels based on the weights. Effectively, all pixel colors that appear in the selection will be accounted for, which helps reduce the number of samples required to capture the important colors needed. This technique reduces the inconsistent output generated by small number of samples in random sampling (Figure 9).

In our implementation, we had 27 clusters, 54 samples within the selected region, and equal number of samples throughout the image.

3.3 Locality Constraint with Texture Analysis.

The goal of this part of the algorithm is to provide a locality constraint on the mask we computed through the radial basis function.

3.3.1 Flood Fill and Limitations.

The flood fill algorithm is an iterative algorithm that expands on the user selected area. The tool is commonly used in different products such as Photoshop Magic Wand. Flood fill is shown to be very efficient [18]. However there are two prominent limitations to the flood fill algorithm. It performs well in low textured regions, but because of its neighborhood comparisons, the tool is known to perform poorly in textured areas (Figure 7). The second limitation lies in its binary output. In areas of low frequency gradients, e.g. clouds, skies, and smooth backgrounds, flood fill creates binary discontunities, which produces seams in the selections (Figure 8). We inherit the properties of obtaining a fast locality constraint of the flood fill algorithm but use texture reduction and adaptive feathering to address these limitations (Figure 6).

The parameters we used for flood fill tolerance is 10 percent of the color space range.

3.3.2 Texture Reduction.

We now address segmentating out a textured area with a flood fill algorithm. Color-based selection such as the magic wand tool performs well in low-textured regions but poorly in high-textured regions. Therefore, to preserve the hard edges and reduce the effects from texture, we perform an edge preserving filter on the image. Edge preserving filters smooth out texture and noise regions while maintaining hard edges. In our implementation, we use the color-based bilateral filter, which is commonly used to remove texture and noise [25]. Given that we have an input image I and an output image of $\dot{I}$, we can formally write the fomulation as

Figure 8: *Adaptive feathering.* With the flood fill input, seams appear in the sky to due the flood fill's binary output (a). By globally feathering the output, we can see mask bleeding in the hard edges of the mountain (b). Using our adaptive mask filtering, we achieve smooth results in the sky and favorable masking on the mountains (c).

$$W(X_q) = \exp\left(-\frac{|X_p - X_q|^2}{2\sigma^2_{\text{spatial}}} - \frac{|I(X_p) - I(X_q)|^2}{2\sigma^2_{\text{color}}}\right)$$
$$\dot{I}(X_p) = \frac{\sum_{q\in N} I(X_q)W(X_q)}{\sum_{q\in N} W(X_q)} \quad (4)$$

where N is a $n\times n$ neighborhood around pixel p, and X_p and X_q represent the pixel location vector $< x, y >$. However, because of its texture and locality weighting terms, the bilateral filter is inefficient. In our implementation, we used an approximation of the bilateral filter by using histograms described by Igarashi et al. [14]. The histogram method reduces computation by reducing the locality term and performs in constant time relevative to the size of the filter. We use a large filter size for a more effective texture reduction. To further reduce impacts of shadows and dark regions, we use the hue channel of the bilateral filter output in the flood fill algorithm. We tried different color space channels such as ones in LAB, YCbCr, and RGB; the hue channel gave us the most consistent results.

The parameters we used for the bilateral filter are radius $= 60$ and $\sigma^2_{\text{color}} = 0.02$, assuming color channel values range from $[0,1]$.

3.3.3 Adaptive Feathering.

The goal of the adaptive feathering is to smooth the mask edges in low textured regions to reduce seams while preserving the hard mask edges in high textured regions to reduce haloing effects (Figure 8). Using the hue channel of the bilateral filtered image, we compute a locality mask using a flood fill neighbor expansion. To distinguish textured and untextured regions, for each pixel at the edge of the selection, we determine if the original image's corresponding pixel is in a high or low frequency region (Figure 6). To do so, we use a gradient domain approach, similar to what is proposed by Tao et al. [24]. We use a neighborhood region to determine the average of high gradient values. We take the absolute sum of the vertical and horizontal gradient of both the hue channel of the bilateral filtered image and the flood fill output. We blur the regions where the bilateral hue channel gradient is less than α and the flood fill output mask gradient is greater than 0. Using the bilateral hue channel gradient determines the low frequency regions while the flood fill output mask gradient determines where the selection edges are. For those pixels in low frequency regions, we perform Gaussian smoothing to those regions of the mask. The resulting mask will display smooth regions in low frequency regions while maintaining the favorable hard edges in high frequency regions.

In our implementation, we used $\alpha = 0.05$ and the Gaussian $\sigma = min(width, height) * 0.05$ pixels.

4 Validation

4.1 Importance Sampling.

We show the effectiveness of importance sampling. We can see the results in Figure 9. We used measurements that check the consistency of the output and checks for the convergence of the RBF output. We look at the mask value from the radial basis function and compute the following:

$$V(M(i)) = Var(\sum_{j=1}^{10} M_j(i)) \quad (5)$$
$$\bar{V} = \sum_{i\in I} V(M(X_i))/(width(I) * height(I)) \quad (6)$$

Where $\bar{V}$ is the average variance of all pixels, I is the mask of dimensions width and height, M is the mask value, i is the pixel index, and j is the iteration count. We used 10 iterations for all sample count in Figure 9. From the graph, we can see that using importance sampling produces more consistent results due to its low variance in the mask output values.

4.2 Algorithm.

We performed several different selections with over 120 diverse and realistic examples.

4.2.1 Evaluation and Performance.

To validate our results, we compared our output with the results with previous state-of-the-art works. To obtain our results, we first compute the mask, which ranges from 0 to 1. We blend two images: one is the original image, O, and the other is the globally modified, G, version (e.g. completely desaturated version of the original). With the computed mask, for each pixel, we compute a weighted average:

$$F(i) = G(i).*M_f(i) + O(i).*(1 - M_f(i)) \quad (7)$$

Where i is the index and M_f is our final mask output, and F, G, and O are the final, globally edited, and original images respectively.

In general, our results show smoother fall-offs in low-frequency regions and our segmentation performs well in many difficult scenarios. With multiple users, we recorded the average number of strokes, rounded to the nearest whole number, in our comparisons. The users are assigned with the desired parts of the image to be masked. We can see some of the comparisons in Figure 11. More examples are included in our supplementary files. Our supplementary video shows our iPad prototype running the algorithm at interactive rates.

5 Conclusion

In this paper, we presented a novel matting algorithm that adaptively feathers selections, provides smooth color correspondence masking, and performs well in textured regions. With its linear complexity and its simplicity of required user input, our algorithm is suitable for mobile and touch-based devices.

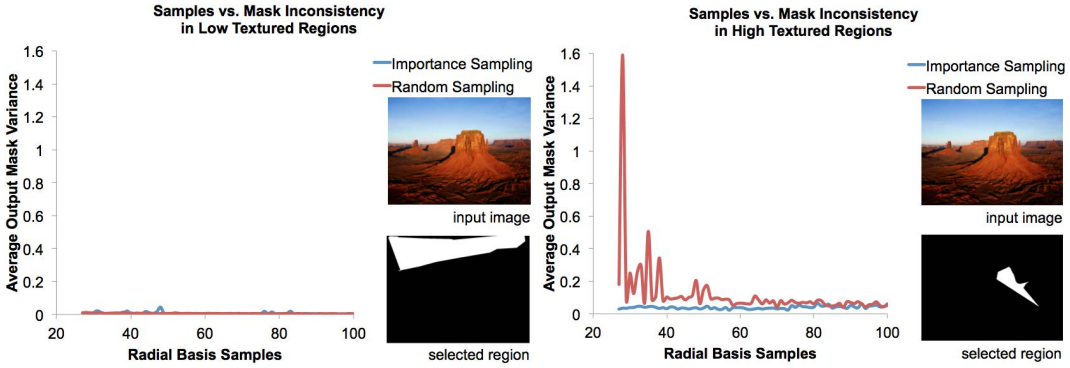


Figure 9: *Sampling convergence.* Using importance and random sampling to compute the radial basis function coefficients impacts the consistency of outputs. We vary the number of samples. For each number of samples, we repeat the operation ten times. We then measure the average pixel variance of the ten iteration output. The lower the variance means the results are more consistent, which is favorable in image editing. On low frequency selections, we show that the two methods show negligible difference. However, in high frequency selections, importance samping shows more consistent results with fewer samples. High variance results are shown in Figure 5. The computation penalty is linear as we increase the number of samples.

(a) Color-ambiguous stroke (orange selection)

(b) Over selected result

Figure 10: *Color limitations.* Regions with similar colors such as the out-of-focused leaves on the deers body can skew the results (a). The output becomes over selected; the result's corresponding mask is on the bottom left (b).

5.1 Limitations.

As with other color-based algorithms, we exhibit inaccurate selections when colors become too similar. In Figure 10, we see that the out-of-focus leaves blend with the deer in the center, creating an ambiguous color selection. The color similarity causes an over-selection throughout the image. Even with a constant time bilateral filter, this precomputation takes several seconds on mobile hardware such as an iPad.

5.2 Future Work.

By using the bilateral filter, we could explore different spaces. Color spaces show their limitations in color-similarity, but using texture-based metrics will help our algorithm perform better in different textures with similar color. We could explore more computationally expensive ways to estimate texture for our adaptive feathering. Because the goal of this paper is to provide an algorithm suitable for quick-editing on mobile devices, we constructed the framework with computation simplicity.

Acknowledgements

We would like to thank Xue Bai for helping us with the Geodesic comparisons, Ruth Archer for the tiles image in Fig. 7, and Sylvain Paris for initial thoughts and ideas.

References

[1] X. An and F. Pellacini. Appprop: all-pairs appearance-space edit propagation. *ACM SIGGRAPH*, 27(3), 2008.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011.

[3] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. pages 1–8, 2007.

[4] A. Bousseau, S. Paris, and F. Durand. User-assisted intrinsic images. *ACM SIGGRAPH Asia*, 28(5), 2009.

[5] J. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. pages 105–112, 2001.

[6] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustments with a database of input/output image pairs. pages 97–104, 2011.

[7] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proceedings of IEEE CVPR 2001*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[8] A. Criminisi, T. Sharp, and C. Rother. Geodesic image and video editing. *ACM Transactions on Graphics*, 29(5):134, 2010.

[9] Z. Farbman, R. Fattal, and D. Lischinski. Diffusion maps for edge-aware image editing. *ACM Transactions on Graphics*, 29(6):145, 2010.

[10] E. Gastal and M. Oliveira. Shared sampling for real-time alpha matting. *Computer Graphics Forum*, 29(2):575–584, May 2010. Proceedings of Eurographics.

[11] M. Gong, L. Wang, R. Yang, and Y. Yang. Real-time video matting using multichannel poisson equations. In *Graphics Interface*, pages 89–96, 2010.

[12] K. He, J. Sun, and X. Tang. Fast matting using large kernel matting laplacian matrices. *CVPR*, pages 2165–2172, 2010.

[13] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. *ACM SIGGRAPH*, pages 327–340, 2001.

[14] M. Igarashi, M. Ikebe, S. Shimoyama, K. Yamano, and J. Motohisa. O(1) bilateral filtering with low memory usage. pages 3301–3304, 2010.

[15] Y. Li, E. Adelson, and A. Agarwala. Scribbleboost: adding classification to edge-aware interpolation of local image and video adjustments. *Computer Graphics Forum*, 27(4):1255–1264, 2008.

[16] Y. Li, T. Ju, and S. Hu. Instant propagation of sparse edits on images and videos. *Computer Graphics Forum*, 29(7):2049–2054, 2010.

[17] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local manipulation of tonal values. *ACM Transactions on Graphics*, 25(3):646–653, 2006.

[18] J. Liu, J. Sun, and H. Shum. Paint selection. *ACM SIGGRAPH*, 28(3), 2009.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Berkeley Symposium of Mathematical Statistics and Probability*, 1967.

[20] S. Paris, S. Hasinoff, and J. Kautz. Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Transactions on Graphics*, 30(4):68, 2011.

[21] C. Rhemann, C. Rother, P. Kohli, and M. Gelautz. A spatially varying psf-based prior for alpha matting. In *CVPR*, pages 2149–2156, 2010.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[23] P. Sundberg, J. Malik, M. Maire, P. Arbelaez, and T. Brox. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, pages 2233–2240, 2011.

[24] M. Tao, M. Johnson, and S. Paris. Error-tolerant image compositing. pages 31–44, 2010.

[25] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *IEEE ICCV*, pages 839–846, 1998.

[26] B. Wang, Y. Yu, and Y. Xu. Exampled-based image color and tone style enhancement. *ACM Transactions on Graphics*, 30(4):64, 2011.

[27] J. Wang, M. Agrawala, and M. Cohen. Soft scissors: an interactive tool for realtime high quality matting. *ACM Transactions on Graphics*, 26(3):9, 2007.

Figure 11: *Desaturation comparisons.* We see that our algorithm provides smoother results and minimizes the visible artifacts in matting within minimized user-strokes. Using graph cut algorithms, many user strokes are usually needed in areas where gradients and edges are not very apparent. Geodesic framework algorithms exhibit the same issues. We tested both binary and soft-masking; Geodesic results show problems with inconvenient strokes and sharp mask fall-offs in gradient areas. The corresponding mask output for each algorithm and average stroke count are shown in the bottom of the images. Please refer to our supplementary material for more comparisons and full-sized image files.