Eurographics Symposium on Rendering 2008
Steve Marschner and Michael Wimmer
(Guest Editors)

Volume 27 (2008), Number 4

ScribbleBoost: Adding Classification to Edge-Aware Interpolation of Local Image and Video Adjustments

Y. Li[1], E.Adelson[1], and A. Agarwala[2]

[1]Department of Brain and Cognitive Sciences, and Computer Science and Artificial Intelligence Laboratory, MIT
[2]Adobe Systems, Inc.

Abstract

One of the most common tasks in image and video editing is the local adjustment of various properties (e.g., saturation or brightness) of regions within an image or video. Edge-aware interpolation of user-drawn scribbles offers a less effort-intensive approach to this problem than traditional region selection and matting. However, the technique suffers a number of limitations, such as reduced performance in the presence of texture contrast, and the inability to handle fragmented appearances. We significantly improve the performance of edge-aware interpolation for this problem by adding a boosting-based classification step that learns to discriminate between the appearance of scribbled pixels. We show that this novel data term in combination with an existing edge-aware optimization technique achieves substantially better results for the local image and video adjustment problem than edge-aware interpolation techniques without classification, or related methods such as matting techniques or graph cut segmentation.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing and Computer Vision]: Enhancement

1. Introduction

Local manipulation of color and tone is one of the most common operations in the digital imaging workflow. For example, to improve a photograph or video sequence an artist may increase the saturation of grass regions, make the sky bluer, and brighten the people. Traditionally, localized image editing is performed by carefully isolating the desired regions using selection tools to create mattes. While effective, this approach can be much more time-consuming than is necessary for color and tone adjustments, especially for video. Matting techniques are primarily designed for the challenge of cutting an object from one image and pasting it into another, in which case it is important to solve the matting equations and recover foreground colors de-contaminated of the background. In contrast, in the case of color and tonal adjustment everything is performed in place, within the original image. Thus, local edits can be interpolated directly and more easily without the need to solve the matting equations.

Recent experiments in *edge-aware interpolation (EAI)* [LLW04, LFUS06, YS06, CPD07] take this approach and offer the user a different interface to localized manipulation that does not require any explicit selection or masking from the user. Instead, a user simply draws rough scribbles on the image (e.g., one on the grass, one one the sky, and one on the people), and attaches adjustment parameters to each scribble. These adjustments parameters are then interpolated to the rest of the image or video in a fashion that respects image edges, i.e., the interpolation is smooth where the image is smooth. While EAI promises to be a powerful technique for localized image and video manipulation, there are a number of problems that currently limit its success in this context. At a high-level, EAI works by propagating the influence of each scribble along paths of pixels of similar luminance; image edges slow this propagation. One problem with this approach is that texture edges within an object also slow propagation. Texture edges may not be a problem if they are weak relative to object boundary edges, but this is often not the case. Another problem is the manipulation of fragmented appearances (such as blue sky peeking through the leaves of a tree, or a multitude of flowers) since the influence of scribbles will be stopped by the edges in-between; the user must therefore scribble each fragment. Finally, manipulating video is a challenge for EAI, since the time-axis tends to be much more aliased than the spatial axes, leading to strong temporal edges that slow propagation. Estimating video motion can sometimes address this limitation, but optical flow algorithms tend to be brittle and computationally-intensive.

In this paper, we significantly improve the performance of EAI for local image and video adjustment by taking advantage of an additional cue that is overlooked in existing EAI systems. Typically, the regions that a user wishes to adjust differently are not only separated by image edges, but they also *appear* different; that is, they have different distributions of color and texture. For this reason, many selection tools in commercial software (such as "select color range" in Adobe Photoshop) operate in color space, independent of a pixel's coordinates. Advanced users can often create a set of selections and rules in color space alone that accurately differentiate desired and un-desired regions [Eis05]. This option can be faster to specify than a spatial selection, and perform better in the presence of fragmented appearances and video motion. In this paper, we attempt to *learn* a good color space selection by training a discriminative classifier (gentleboost [FHT00]) to differentiate between the appearance of the pixels within different scribbles, and combine this per-pixel data term with the spatial regularization provided by the original smoothness term of EAI systems. Thus, in our interactive system, which we call ScribbleBoost, a scribble indicates that pixels *similar in appearance* to the scribbled pixels should be adjusted similarly, rather than only a continuous region containing the original scribble.

Of course, the combination of a per-pixel data term and a neighboring-pixel smoothness term is commonplace in algorithms for image segmentation [RKB04, Gra05] and matting [WC07]. In that light, our main contribution is to extend traditional edge-aware interpolation with a novel, discriminatively-learned and weighted data term that uses a boosting-based classifier. A key feature of our data term is a weighting scheme that considers the accuracy of the classifier over its continuous output range. As a result, the weighted data term creates "crisper" transitions between regions when the classifier is confident, while the smoothness term takes over when classification is more ambiguous. Our data term also significantly improves performance in the presence of texture edges and fragmented appearances. As we show with an extensive comparison to previous work, our approach yields substantially better results with just a few user-drawn scribbles.

2. Related work

Edge-aware interpolation was first introduced by Levin et al. [LLW04] for the purpose of colorizing a grayscale image from a set of user-defined scribbles. They demonstrated that a colorized image appears natural if the color parameters specified at scribbled pixels are interpolated in a fashion that respects luminance edges. Colorization from user-drawn scribbles continues to be an active topic of research for both natural images [YS06, LWCO*07] and hand-drawn illustrations [QWH06]; one significant difference from our problem is that these algorithms are not designed to take advantage of color input. Grayscale pixels are much harder to discriminate between than color pixels, and thus require the use of texture features in a neighborhood around each pixel [QWH06, LWCO*07]. In our experience (Section 4.2), color at a single pixel discriminates more reliably than texture in a pixel's neighborhood.

Edge-aware interpolation was first generalized beyond colorization by Lischinski et al. [LFUS06] for the purpose of interactive tone mapping. From the perspective of a user, our system is very similar to theirs; the primary difference for the user is that, in our system, adjustments can propagate not only to pixels that are spatially close, but also to pixels that are close in appearance. Their system also included a brush that allowed the user to select and scribble any pixel similar to the color or luminance of a specified pixel (similar to "select color range" in Photoshop). This brush is, in a sense, a simple appearance-based data term that can sometimes handle fragmented appearances. However, the appearance of many objects is not confined to a narrow enough color range for this approach to be effective; in our supplemental materials, we show that such a brush is not effective for any of our examples.

Edge-aware interpolation of color and tone parameters can be seen as scattered data interpolation; given a set of constraints specified at scribbles, interpolate those parameters to the entire image or video. For image manipulation the best results are achieved if the interpolation respects image edges. To that end, a number of EAI techniques have been developed, including smooth interpolation across a bilateral grid [CPD07], edge-weighted geodesics [YS06], and linear least squares optimization [LLW04, LFUS06]; our system utilizes the latter since the framework naturally accepts our novel data term. The bilateral grid approach is qualitatively different than the others, since strong edges do not necessarily stop propagation; we show better results on an example from their paper. The edge-weighted geodesics and least squares approaches both suffer in the presence of texture contrast and fragmented appearances. The colorization system of Luan et al. [LWCO*07] also addresses these same concerns. However, since they assume grayscale input, they first create a color labeling by executing a hard graph-cut segmentation of the image based on texture segmentation cues; as a result, they are not able to achieve the long-range, soft transitions that we believe are necessary for smoothly interpolating color and tone adjustments.

Several matting and segmentation systems create masks from user-drawn scribbles [LSTS04, Gra06, LLW06, WC07, BS07], and these masks can certainly be used for color and tone manipulation. However, it is not clear how to blend

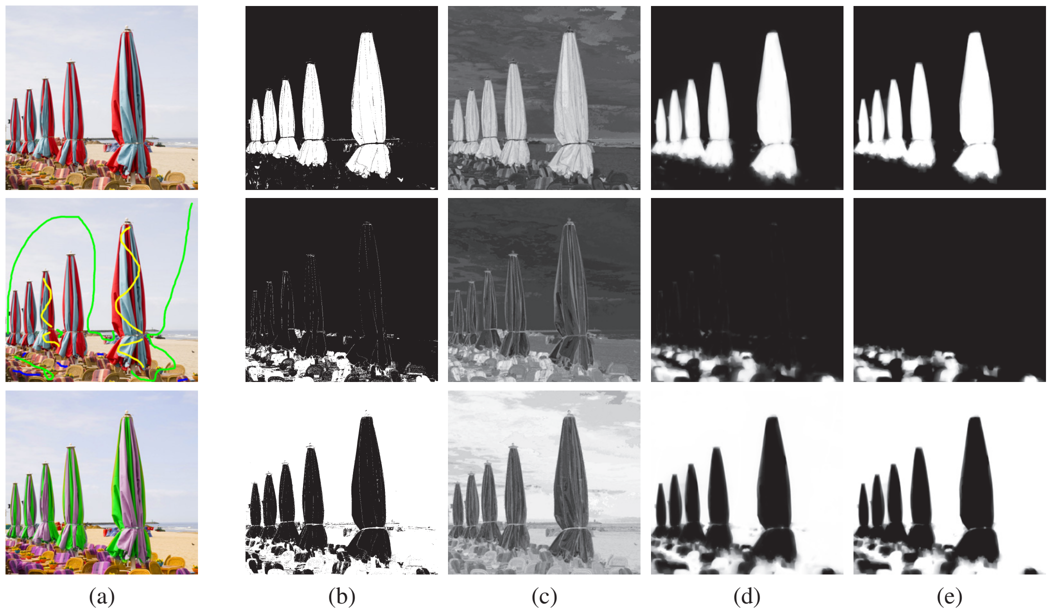

Figure 1: *One example of localized editing using our system. Column **(a)** shows the inputs and output of our system: the input images, the user-drawn scribbles to separate the image into three classes (umbrellas, chair upholstery, and everything else), and a manipulated result that changes the hue of the umbrellas and chairs differently and saturates the rest. The other columns show intermediate outputs with one row per class. **(b)** The binary output of the three classifiers, and **(c)** the continuous output of each classifier with zero mapped to gray. **(d)** The output of edge-aware interpolation of the scribble constraints and classifier outputs, and **(e)** the final blending weights after post-processing.*

more than two adjustment parameter constraints using mattes. Also, even for only two constraints, we find that our approach can better interpolate adjustment parameters with less user-effort than both matting and segmentation techniques, as we show in Section 7 with several comparisons. Algorithmically, the least squares problem solved in our system is similar to those used in both binary segmentation [Gra06] and matting [LLW06, WC07]. However, the smoothness term used by EAI systems is typically simpler and more efficient to compute than those used in matting algorithms, which involve a larger neighborhood that better captures the precise mix of foreground and background at each pixel. Also, if a data term is used for matting, it usually involves a foreground and background model local to each pixel [WC07], rather than our global data term. Finally, the combination of data and smoothness terms that we describe could be solved using graph cuts, which are used by several interactive segmentation systems [RKB04, LSTS04]. However, the long-range, soft transitions created by edge-aware interpolation are better suited to our problem than the discrete result of graph cuts, and our results compare favorably (Section 7).

Finally, both Protiere and Sapiro [PS07] and Wang [Wan07] have explored the use of texture cues and automatic feature selection in the context of interactive matting and segmentation. Also, a boosting classifier was used for binary image segmentation by Avidan [Avi06], though their focus was the incorporation of spatial priors into the Adaboost algorithm.

3. System overview

Our approach to local color and tone manipulation is implemented as a simple interactive prototype that allows the user to draw scribbles indicating the different classes of content that the user wishes to manipulate differently, as shown in Figure 1(a). In this example, yellow scribbles are drawn to indicate the umbrellas, blue scribbles indicate the chair upholstery, and green scribbles indicate everything else. The user chooses to adjust the hue of the umbrellas, adjust the hue of the chairs by a different amount, and increase the saturation of everything else (the edits might be more extreme than typical, but help to demonstrate the system). The result is shown at the bottom of Figure 1(a).

Our algorithm could have interpolated these hue and saturation parameters directly; instead, after the user clicks a button our system computes the blending weight masks shown in the right-most column (Figure 1e). As shown by Lischinski et al. [LFUS06], computing a set of per-pixel blending weights that linearly blend adjustments made to the different scribble classes is equivalent to directly interpolating the adjustment parameters themselves. So, our system computes these blending weights, which sum to one (pure white) at each pixel, and loads them as layer masks into Adobe Photoshop so that the user can adjust the different layers in real-time. (Ideally, these blending weights would never be exposed to the user, and scribbles and adjustments could be performed in a single interface.) If there are only two scribble classes, blending weight compositing is identical to alpha compositing (this equivalence is also true for the blending weights of Lischinski et al.; for more than two scribble classes, the compositing equations are different (they require "add" rather than "over" compositing [PD84]).

Our approach to calculating per-pixel blending weights consists of three simple steps (the intermediate results of each step are shown in Figure 1).

1.**Per-pixel classification.** In the first step, our system builds a boosting-based classifier to discriminate between the appearance of the different classes. In this example, the classifier attempts to learn whether a pixel more resembles the appearance of the umbrellas, the chairs, or everything else, given the training data of the scribbled pixels. The result of the classifier is a per-pixel, per-class scalar that is positive if the classifier believes the pixel belongs to the class, and negative if not (Figure 1(b,c)); the magnitude of the scalar represents the confidence of the classification.

2.**Edge-aware interpolation.** The second step computes an initial set of blending weights by performing edge-aware interpolation of both the scribbles and the per-pixel classification (Figure 1d). The interpolation is performed as a least-squares minimization of the sum of a per-pixel data term and a smoothness term per pair of neighboring pixels. Scribbled pixels are used as hard constraints, and the data term is weighted by the confidence of the classifier.

3.**Post-processing.** The third step improves the above-calculated weights in two ways. First, our system enforces a simple constraint; fractional weight values should only exist in a transition from a region of pixels of one class to a region of pixels in another class. Second, as in previous work [LLW06, CPD07], we apply a sigmoid to the weight values to bias the weights towards one or zero. An example of the final blending weights can be seen in Figure 1e.

4. Per-pixel classification

A user-drawn scribble in our system not only signifies a region that should be affected by the scribble, but also an *appearance*; regions of similar appearance to the scribbled region should also be affected. This appearance prior benefits our approach in two ways. For one, the user does not need to scribble every disconnected region of a fragmented class. For example, in Figure 1 all the chair covers are selected even though only a few are scribbled. The second benefit of the appearance prior, as we show in Section 7 with comparisons to results generated without it, is that it causes our masks to be much crisper than those generated solely through spatial interpolation. When a pixel is caught between the influence of two different scribbles, the classifier can use its appearance to disambiguate its class membership, whereas spatial interpolation alone might resort to an overly soft transition.

To accomplish this appearance selection, our system supplements edge-aware interpolation with a classifier that learns how to discriminate between the appearances of the different classes. We use the gentleboost classifier [FHT00], which is a member of the larger family of classifiers based on boosting [Sch90,HTF01]. (We expect most boosting variants would perform similarly; we choose gentleboost because it is simple and efficient). Boosting operates on the principle that a good classifier can be built as the weighted combination of many simple classifiers, each of which might perform just better than chance on the training data. One advantage of boosting-based classifiers is that they are discriminative rather than generative. That is, the classifier does not attempt to build a model that would generate the observed examples, but instead simply seeks to separate the data. Matting and interactive segmentation systems more commonly use the generative Gaussian Mixture Model (GMM) [CCSS01, RKB04, WC07] to describe appearance; when color distributions are not well approximated by a small number of Gaussians, our classifier performs better. In Section 7 we compare our results to ones generated by replacing gentleboost with GMMs; we also compare against the results of state-of-the-art matting and interactive segmentation algorithms. In Table 1 we compare the classification losses of a GMM-based classifier (with five Gaussians) and gentleboost.

Example	GMM loss (%)	Gentleboost loss (%)
Deer (Figure 2)	8.14	7.70
Deer, extra features	*	1.63
Chocolates (Figure 3)	0.11	0.00
Birds (Figure 4)	4.70	3.23
Buddha (supp.)	1.80	0.56
Girl (Figure 5)	3.55	2.64

Table 1: *A comparison of the classification loss on the training data as a sum of the percentage of positive pixels and the percentage of negative pixels misclassified by the GMM-based classifier and gentleboost. The classifiers use RGB values as features, except for the second row, where additional features were also used.*

If there are more than two scribble classes, we are faced with a multi-class classification problem [HTF01]. We therefore train one classifier per class in a one-versus-all framework. That is, we form the training data for the i'th class by simply aggregating the N scribbled pixels and setting label $z_j = +1$ if the j'th pixel belongs to the class, and $z_j = -1$ if not. Gentleboost then creates an ensemble classifier H_i for the i'th class as a sum of many simple weak classifiers. That is, it fits an additive model

$$H_i(v) = \sum_r h_r(v)$$

where v is the feature vector for the pixel being classified, $h_r(v)$ is a weak classifier, and r indexes over the weak classifiers. In our case, each weak classifier is modeled as a simple decision boundary in feature space. Such a decision boundary is often called a Perceptron [HTF01], and is represented by a hyperplane θ, where $\theta_r \cdot v$ splits the feature space; if the result is positive, the weak classifier believes that v belongs to the class, and vice-versa. Each training example v_j is associated with a weight w_j and label z_j. We fit each hyperplane as perpendicular to the axis of maximum separability of the weighted training data, which is computed using weighted Fisher's Linear Discriminant (FLD) [Fis36]. The offset of

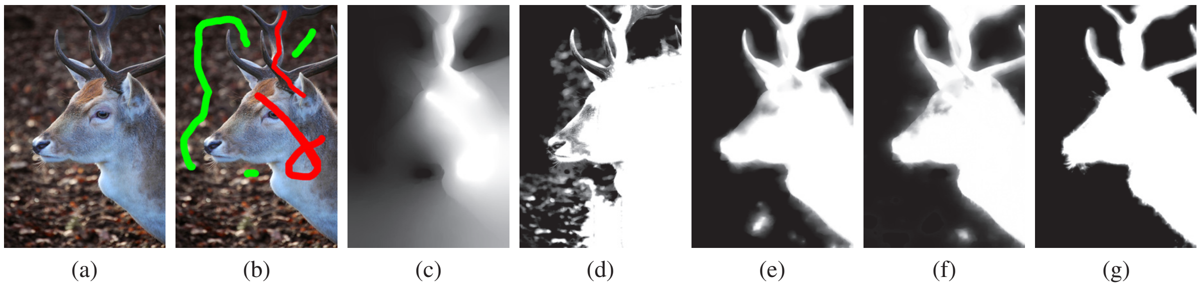

Figure 2: *(a) A challenging image with strong texture contrast and similar color distributions between foreground and background. (b) Image with scribbles, results of (c) Lischinski et al. [LFUS06], (d) RobustMatting [WC07], (e) using GMMs in the data term, (f) our result using RGB only, and (g) Our result using additional classification features as described in Section 4.2.*

the hyperplane along this axis is computed to minimize the weighted classification loss by a simple 1D search.

4.1. Gentleboost

Like most boosting algorithms, gentleboost (which we describe for completeness) adjusts the weights of the training data as each weak classifier is added to the ensemble so that new weak classifiers focus on the training data that is misclassified by the current combination of simple classifiers. The weak classifiers themselves also have weights that are proportional to their performance on the training data. The algorithm begins by first initializing the training data weights $w_j = 1$ and then normalizing so that the weights of the positive examples sum to 0.5, and the weights of the negative examples sum to 0.5. Let $\delta(\cdot)$ be the indicator function that is 1 if its argument is true, and 0 otherwise. Then, for each $r = 1, 2, \ldots, M$, where M is the number of weak classifiers,

1. Fit hyperplane θ_r to weighted training data using FLD.
2. Fit weak classifier

$$h_r(v_j) = a_r\delta(\theta_r \cdot v_j > 0) + b_r\delta(\theta_r \cdot v_j \le 0)$$

by calculating weak classifier weights a_r, b_r as

$$a_r = \frac{\sum_j w_j z_j \delta(\theta_r \cdot v_j > 0)}{\sum_j w_j \delta(\theta_r \cdot v_j > 0)} \quad b_r = \frac{\sum_j w_j z_j \delta(\theta_r \cdot v_j \le 0)}{\sum_j w_j \delta(\theta_r \cdot v_j \le 0)}$$

3. Update weights by $w_j = w_j e^{-z_j h_r(v_j)}$, and then renormalize.

The final classifier $H_i(v)$ classifies a pixel by a weighted sum of the beliefs of its weak classifiers; the more the weak classifiers agree with each other, the larger the magnitude of $H_i(v)$, and the higher the confidence of the classifier in its belief. We use 100 weak classifiers. Fewer classifiers yields a less continuous confidence measure, while more requires additional computation time; we found 100 to be a good compromise. To communicate this information to the next stage of our algorithm, we evaluate each classifier on each image pixel; that is, we compute each $m_{i,p} = H_i(v_p)$, where $m_{i,p}$ is the output of the i'th classifier on pixel p. The magnitude of $m_{i,p}$ can be considered the confidence of the i'th combined classifier for pixel p.

4.2. Features

All the results other than Figure 2(g) in this paper were generated simply using the RGB color as the feature vector v at each pixel. However, one of the benefits of boosting is feature selection, i.e., it can choose the best-performing features to train the next weak classifier $h_r(v)$ given the current weighting. We therefore experimented with using a wider set of features to measure appearance and texture at a pixel, including alternative color models such as LAB and HSV, texture features such as local derivatives and Laplacians, and even the spatial coordinates of the pixel. Figure 2 shows an example where these extra features did indeed help (see Table 1 for a numerical comparison). In this example, the color distributions of foreground and background are heavily overlapping, but the shallow depth of field allows the Laplacian to be highly discriminative. Overall, though, we found that extra features hurt as often as they helped, since the extra dimensionality allowed a greater possibility of over-fitting, and texture features often fail near object boundaries.

5. Edge-aware interpolation

The previous step of our approach calculates a measure of the belief that each pixel belongs to each stroke class, expressed as $m_{i,p}$. In this step, our system calculates per-class, per-pixel blending weights by performing spatial regularization, so that neighboring pixels of similar appearance are manipulated similarly.

Though the output of this step is a set of blending weights, the equations are easier to understand if we first present them as directly interpolating an adjustment parameter. That is, we assume that the user has already defined the desired values of some adjustment parameter (e.g., saturation or brightness) for each stroke class; we represent this scalar value as c_i for the i'th class. We then compute the value of this adjustment parameter f_p for each pixel p. To do so, we compute f_p that minimizes the sum of a per-pixel data term and smoothness term per pair of neighboring pixels,

$$\sum_{p \notin \Omega} D_p + \lambda \sum_{p,q} S_{p,q} \tag{1}$$

subject to the constraint that $f_p = c_i$ for all pixels $p \in \Omega_i$, where Ω is the set of stroked pixels, D_p is the data term on

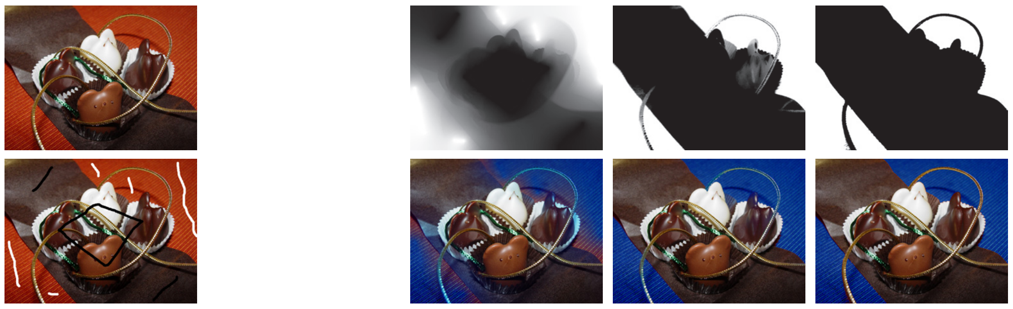

Figure 3: *(a) An example from the bilateral grid paper [CPD07] (top) and a similar set of scribbles separating the image into two classes (bottom). The mask and local image adjustment result of (b) bilateral grid, (c) Lischinski et al., (d) RobustMatting, and (e) ScribbleBoost.*

pixel p, $S_{p,q}$ is the smoothness term on neighboring pixels p and q, and λ weights the smoothness term relative to the data term (we use $\lambda = 1$ in all examples). Our formulation is similar to that of Lischinski *et al.* [LFUS06], except for the data term. Our smoothness term is nearly identical, i.e.,

$$S_{p,q} = \frac{(f_p - f_q)^2}{\nabla I_{p,q} + \varepsilon}$$

where $\nabla I_{p,q}$ is the magnitude of the color gradient between pixels p and q, and $\varepsilon = .001$ prevents division by zero. The smoothness term encourages neighboring pixels to have similar values of parameter f, but the strength of the term is weakened across image edges.

The data term, which is novel to our formulation, is designed to encourage the value f_p to be c_i if we believe pixel p belongs to the i'th class.

$$D_p = \sum_i w_{i,p}(f_p - c_i)^2$$

The most interesting aspect of this data term is the design of the weight $w_{i,p}$, which depends on the classifier output $m_{i,p}$ computed in Section 4.1. Obviously, if $m_{i,p}$ is less than or equal to zero, the classifier does not believe pixel p belongs to the i'th class, so $w_{i,p}$ should be zero. Otherwise, the weight should be proportional to the confidence of the classification. This confidence can be measured in two ways. The first is simply the absolute value of $m_{i,p}$, which measures the confidence of the i'th classifier specifically for pixel p. However, there is an additional and valuable cue for measuring confidence: the *overall* accuracy of the i'th classifier on its training data. In cases where the color distributions of the different classes are well-separated, the classifier may achieve no or almost no loss, in which case the data term weight should be higher. Otherwise, the color distributions may overlap significantly, and thus the classifier may perform poorly and offer almost no discriminative insight – in this case, the weight should be close to zero, and the overall optimization should revert to the original formulation of Lischinski et al. [LFUS06] that does not use classification.

One simple measure of overall accuracy is the classifier loss; however, this measure does not express how the classifier's performance varies over the range of classifier outputs. That is, the classifier might be quite inaccurate for low values of $m_{i,p}$, but very accurate for higher values. So, we instead ask a simple question: above what value of m_i does the i'th classifier perform perfectly on the training data? That is, what is the maximum value of $m_{i,p}$ for the negative training examples? For classifier outputs above this threshold (which we call m_i^*), we can be more confident of the classifier. For outputs below this threshold, we know that that classifier sometimes misclassifies, so confidence should be very low. We thus define a weighting function that decreases very rapidly below the threshold m_i^*, and increases less rapidly above it (as overly-strong weights can render the linear system that computes the minimum ill-conditioned).

$$w_{i,p} = \begin{cases} 0 & m_{i,p} \le 0 \\ \left(\frac{m_{i,p}}{m_i^*}\right)^4 & 0 < m_{i,p} \le m_i^* \\ \left(\frac{m_{i,p}}{m_i^*}\right)^2 & m_i^* < m_{i,p} \end{cases}$$

We add one additional caveat to the computation of m_i^*. If the classifier performs very well, m_i^* may be zero or even negative. Even positive values of m_i^* that are very small can be problematic, as the data term becomes too strong and the resultant masks almost binary. We thus do not allow m_i^* to be any smaller than $\frac{1}{10}$ of the overall range of positive classifier outputs.

The result of this weighting scheme is that the blending weights are softer in areas where the classifier has low confidence, and vice-versa. This effect can be seen by comparing Figures 2(f) and (g); the latter uses a better-performing classifier than the former, and so its transitions are much crisper. In effect, our scheme can minimize the negative effects of uncertainty by resorting to soft transitions that do not introduce new edges that attract the eye.

The minimization problem in equation (1) is quadratic, and its global minimum can be found by computing a linear system $Af = b$ with respect to the per-pixel adjustment parameter f. How can we, instead, compute a set of blending weights so that the linear system does not need to be resolved each time the parameters are changed? We again take inspiration from the approach of Lischinski et al., and separate the linear system into a set of per-class linear systems.

To do so, we assume the adjustment parameters are linear. Then, with a variable substitution $f_i' = \frac{1}{c_i} f_i$ and $b_i' = \frac{1}{c_i} b_i$, the linear system $Af = b$ can be expressed as $\sum_i A_i \sum_i f_i' = \sum_i b_i'$. We can compute each f_i' as $(\sum_i A_i) f_i' = b_i'$ and the final parameter vector f can be expressed as $f = \sum_i c_i f_i'$. We can therefore use each f_i' as a blending weight mask, and simply use "add" compositing to compute a final image. Note that this approach treats adjustment parameters as linear even though certain adjustments, such as hue, are not; none the less, treating these parameters as linear typically generates results that match our mental model of what we would expect to see.

There are a number of approaches to efficiently solving large, sparse linear systems of this form, including multigrid algorithms on the GPU [BFGS03]. For ease of implementation we use locally-adapted hierarchical basis preconditioning [Sze06]. Notice that each linear system (one per f_i') can be computed in parallel.

6. Post-processing

In the third and final step of blending weights calculation, the masks are improved in two ways. The first step is motivated by one of the artifacts that can be seen in the results of the previous step in Figure 1(d); there are occasional patches of soft, fractional values that are disconnected from any fully opaque pixels that definitely belong to the class represented by the mask. For example, in the second row of Figure 1(d) there are soft patches of pixels well above the covered chairs that this class represents. We make the observation that fractional values should only exist at the transition from one class to another,[†] and modify the masks to enforce this constraint. First, we assume that blending weights more than 95% opaque are definitely in the corresponding class, weights less than 5% opaque are definitely not in the class, and weights values in-between are transitional. Then, we compute a flood fill from in-class pixels to transitional pixels to identify those transitional pixels that are, in fact, connected. Any transitional pixel that are not connected to in-class pixels are set to zero. This operation is performed for each class, and then the weights are re-normalized to sum to unity. As can be see in Figure 1(e), this operation removes these errant regions.

The final post-processing step (which is also performed in other EAI systems [LLW06, CPD07]) simply biases the masks slightly towards zero and one; we scale the weights from the center of their range by a factor of 1.1, and clamp and re-normalize so that the weights sum to unity. The output is the set of final blending weights.

[†] This observation is not always true; counter-examples include partially transparent regions such as smoke, or structures thinner than one pixel for their entire extent. We ignore these cases.

7. Results

In Figures 2-7 we show a number of results created using our system as well as comparisons to results created using previous work with the same set of scribbles (we recommend zooming in on the image in the electronic version of this paper to better see the differences). In most examples we use only two scribble classes so that comparisons can be made to the output of matting and segmentation algorithms. In these cases we show blending weights from our technique and that of Lischinski *et al.*, as well as mattes from matting and segmentation algorithms. Comparing blending weights and mattes directly can be misleading, as mattes are computed to model the matting equations and produce precise foreground colors de-contaminated from the background, while blending weights are designed for in-place editing. If our blending weights were used for compositing onto novel backgrounds, the result would likely not be successful. However, these masks can be useful to bring attention to problematic areas in the final edited results, which were created in Adobe Photoshop by adjusting hue, saturation, contrast, and/or brightness of the differently masked layers. We often chose more drastic edits than might be typical since they better reveal the differences in the outputs of different systems. Finally, we show examples of multi-class edits for which matting algorithms cannot directly be compared in Figure 6, and a video result in Figure 7. Finally, several additional results and comparisons are shown in the supplemental materials.

Our comparison results of Lischinski et al. [LFUS06] were created using our system with the classification-based data term disabled; without this term, our systems are largely identical. In fairness, it should be noted that we are applying their method to a different problem than the one they were trying to solve; very soft masks that work well for HDR tone mapping might not work for color adjustments. For our application, we can see that their technique does not handle fragmented appearances where each fragment is not scribbled (e.g., several of the birds in Figure 4), and suffers in the presence of texture edges (e.g., the textured dress in Figure 5). An extreme example of a fragmented appearance can be seen in the lilypads in Figure 6; stroking each lilypad would be very time-consuming. The matting results of RobustMatting [WC07] and Bai and Sapiro [BS07] were created using the authors' systems (we manually drew similar strokes in their interfaces). Matting algorithms are challenged by the rather sparse set of scribbles used in these examples. Figure 3 shows a comparison using an image and result from the bilateral grid paper [CPD07]. Their result show significantly more color spilling than ours, which benefits from the rather easy separability of the colors in the separate classes of this example (Table 1). In Figures 4 and 5 we compare against the results of a publicly available implementation[‡] of Lazy Snapping [LSTS04], which uses graph cuts to create binary masks.

[‡] `http://www.cs.cmu.edu/~mohitg/segmentation.htm`

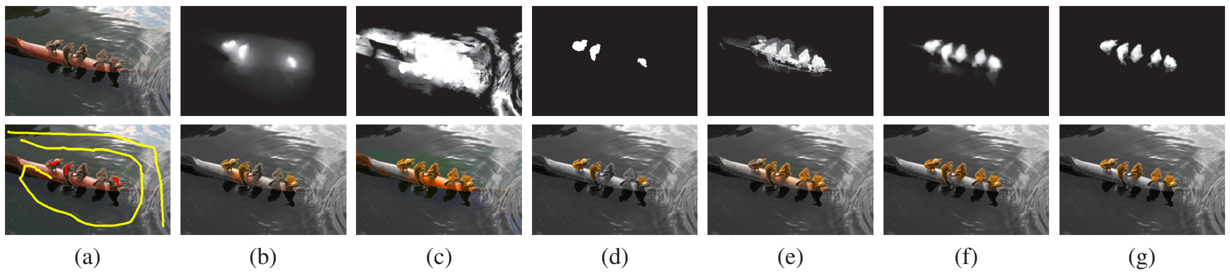

Figure 4: *(a) An example image (top) and a set of scribbles isolating the birds from the background (bottom). The mask and local image adjustment result of (b) Lischinski et al., (c) RobustMatting, (d) Lazy Snapping, (e) Bai and Sapiro, (f) our technique using GMMs in the data term, and (g) ScribbleBoost.*

We also show comparisons to results created using a GMM classifier instead of gentleboost. These results are still quite good, in part due to the other components of our technique which remain the same, such as edge-aware interpolation, the weighting scheme described in Section 5, and the post-processing stage. The classification loss comparison between GMMs and boosting in Table 1 varies from little difference up to a factor of 3.2 (gentleboost always performs better). In our experience, boosting also exhibits better accuracy in classification *confidence*; the effect of this difference can be seen in the generally softer masks from GMMs. Differences can also be seen in the editing results, most notably near the edges of the dress in Figure 5.

Most of our examples show blending weight masks that resemble alpha mattes and crisply separate different objects; however, users do not always apply different adjustments strictly to different objects. In Figure 8 we show an example from the paper of Lischinski et al.that has different scribbles on the same object (a tablecloth) to interpolate a depth-of-field effect. This example requires a longer-range, smooth transition, which our system can still produce.

Finally, our technique works well for video sequences, and we show an example in Figure 7 where scribbles are drawn on only 1 out of 123 frames (several more examples are shown in the accompanying video; for each, scribbles were drawn on just one frame). The spatial regularization and post-processing steps are performed independently for each frame. While we have not noticed any temporal coherence artifacts, more challenging video sequences might benefit from adding temporal smoothness terms to our EAI formulation. Our reliance on per-pixel classification benefits our video results, whereas pure EAI systems must depend solely on propagating information across time.

Failure cases. Our system does not always yield the desired result. One source of failure is when the color distributions of the layers that the user wishes to separate are very similar; an example can be seen in Figure 2. In this case, extra features can help. Also, we assume that the user wishes to manipulate pixels of similar appearance in the same fashion, which isn't always true. For example, if the user wished to edit only one umbrella in Figure 1, our system would hinder the user more than help. Perhaps the ideal system would involve two types of strokes; ours, and the scribbles of traditional EAI which only indicate a region and not an appearance.

Performance. Our system involves substantial computation. The bottlenecks, in decreasing order, are the solution of the sparse linear systems, the evaluation of the classifier (which involves 100 weak classifiers) on each pixel, and the training of the classifiers. However, our algorithms can easily benefit from recent GPU and multi-core processing models. For example, Szeliski [Sze06] points out that his solver easily maps to the GPU (our implementation is software-only), and the classification of each pixel can be performed in parallel. Each classifier can also be trained in parallel. While we have not experimented with GPU execution, we did achieve some parallelization with just a few lines of OpenMP (www.openmp.org) code. As a result, the one megapixel, three scribble-class example in Figure 1 took about 10 seconds on a multi-core machine to compute all weights. The 0.7 megapixel, two scribble-class example in Figure 5 took only 3 seconds. These execution times lead us to believe that a GPU-based implementation could respond in real-time to a new scribble at a preview resolution.

Sensitivity to user scribbles. Our comparisons show that our technique achieves significantly better performance than previous work. However, did we simply choose scribbles that favor our technique? To address this question, we performed an informal user study comparing the robustness of various methods to a variety of scribble styles. We asked five users to draw scribbles that separate a target object from the rest of the image; the resultant scribbles varied widely in terms of positioning and density, as shown in the supplemental materials. In spite of this variation, our method consistently performs better than the compared techniques.

8. Conclusion

Local color and tone manipulation is a very frequent task for image and video editors, and we believe that our technique has the potential to significantly reduce their burden. There are many ways that our approach could be further improved. One direction is improvements to our classifier, which is currently very simple. Classification is an actively researched topic and recent advances could be applied to our

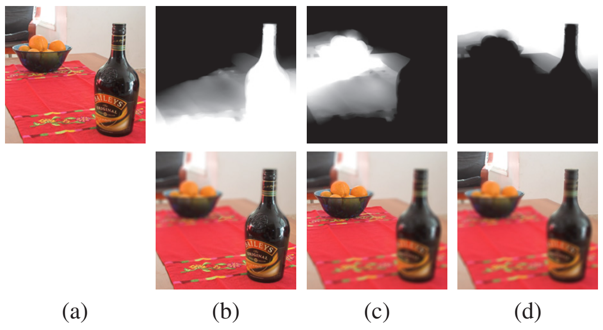

Figure 8: *An example from the paper of Lischinski et al.that requires long-range, smooth transitions that do not resemble object mattes. (a) Original image (top) and scribbles (bottom) indicating areas for spatially-varying blur. (b-d) Blending weights (top) computed using ScribbleBoost and the depth-of-field effects (bottom) achieved using these masks.*

problem. For example, our classification problem is semi-supervised, since the unlabeled pixel data is also available; applying semi-supervised methods could significantly improve results. Also, better classifiers may allow the use of extra features, if they can avoid the over-fitting that we sometimes observed.

Edge-aware interpolation offers an attractive and less effort-intensive alternative for local image and video adjustment. By augmenting existing methods with classification we are able to achieve significantly better results than previous work. Our algorithm is quite simple, consisting of a standard and easy-to-implement classifier, the setup and solution of a weighted linear system, and a few flood-fills. We hope to test our system with real users in the near future.

References

[Avi06] AVIDAN S.: Spatialboost: Adding spatial reasoning to adaboost. In *ECCV (4)* (2006), pp. 386–396.

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *ACM Transactions on Graphics 22*, 3 (July 2003), 917–924.

[BS07] BAI X., SAPIRO G.: A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE International Conference on Computer Vision (ICCV)* (2007).

[CCSS01] CHUANG Y.-Y., CURLESS B., SALESIN D. H., SZELISKI R.: A bayesian approach to digital matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2001), pp. 264–271.

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics 26*, 3 (2007), 103.

[Eis05] EISMANN K.: *Photoshop Masking & Compositing*. Peachpit Press, 2005.

[FHT00] FRIEDMAN J., HASTIE T., TIBSHIRANI R.: Additive logistic regression: a statistical view of boosting. *Annals of Statistics 2*, 28 (2000), 337–374.

[Fis36] FISHER R.: The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7* (1936), 179–188.

[Gra05] GRADY L.: Multilabel random walker image segmentation using prior models. In *2005 Conference on Computer Vision and Pattern Recognition (CVPR 2005)* (June 2005), pp. 763–770.

[Gra06] GRADY L.: Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (2006), 1768–1783.

[HTF01] HASTIE T., TIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning*. Springer, 2001.

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Transactions on Graphics 25*, 3 (July 2006), 646–653.

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 689–694.

[LLW06] LEVIN A., LISCHINSKI D., WEISS Y.: A closed form solution to natural image matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 61–68.

[LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 303–308.

[LWCO*07] LUAN Q., WEN F., COHEN-OR D., LIANG L., XU Y.-Q., SHUM H.-Y.: Natural image colorization. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)* (June 2007).

[PD84] PORTER T., DUFF T.: Compositing digital images. In *Computer Graphics (Proceedings of SIGGRAPH 84)* (July 1984), pp. 253–259.

[PS07] PROTIERE A., SAPIRO G.: Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing 16*, 4 (2007), 1046–1057.

[QWH06] QU Y., WONG T.-T., HENG P.-A.: Manga colorization. *ACM Transactions on Graphics 25*, 3 (July 2006), 1214–1220.

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 309–314.

[Sch90] SCHAPIRE R.: The strength of weak learnability. *Machine learning 5*, 2 (1990).

[Sze06] SZELISKI R.: Locally adapted hierarchical basis preconditioning. *ACM Transactions on Graphics 25*, 3 (July 2006), 1135–1143.

[Wan07] WANG J.: Discriminative Gaussian mixtures for interactive image segmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2007), pp. 386–396.

[WC07] WANG J., COHEN M. F.: Optimized color sampling for robust matting. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2007).

[YS06] YATZIV L., SAPIRO G.: Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing 15*, 5 (2006), 1120–1129.

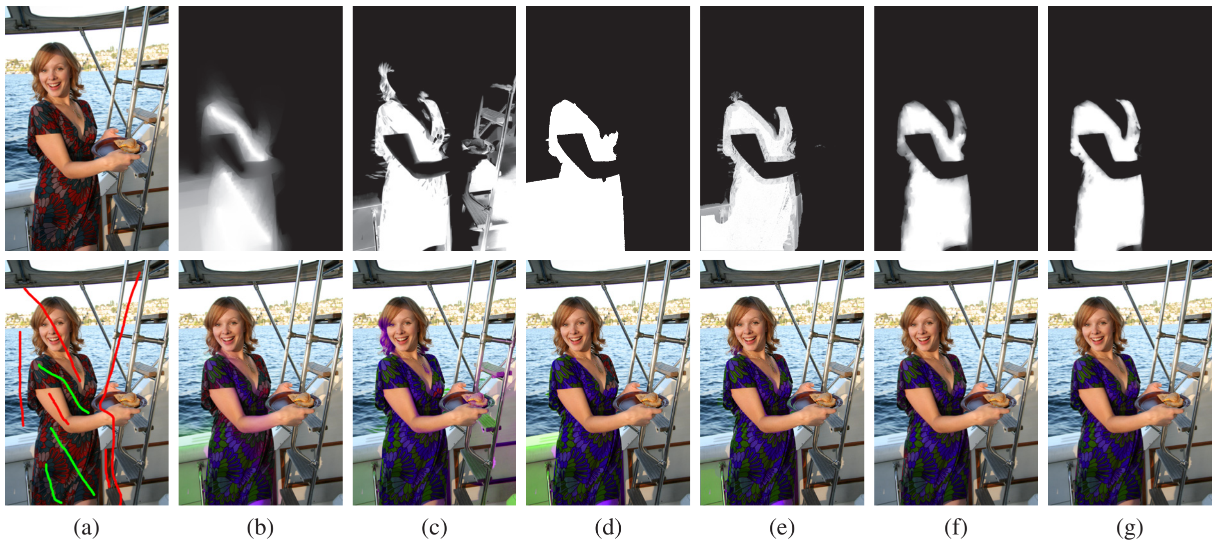

Figure 5: *(a) Example image (top) and scribbles isolating the dress (bottom). The mask and local image adjustment result of (b) Lischinski et al., (c) RobustMatting, (d) Lazy Snapping, (e) Bai and Sapiro, (f) our technique using GMMs in the data term, and (g) ScribbleBoost.*

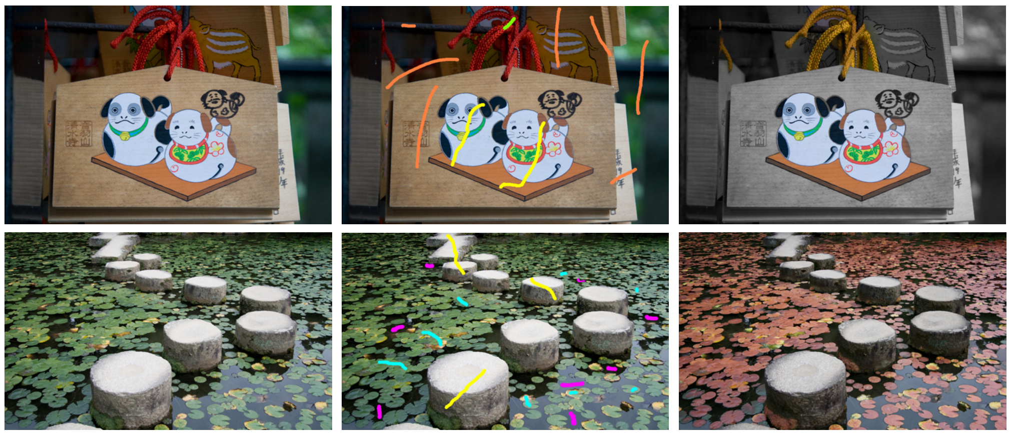

Figure 6: *Results from our system that involve more than two scribble classes. Left to right: original image, scribbles, and editing result.*

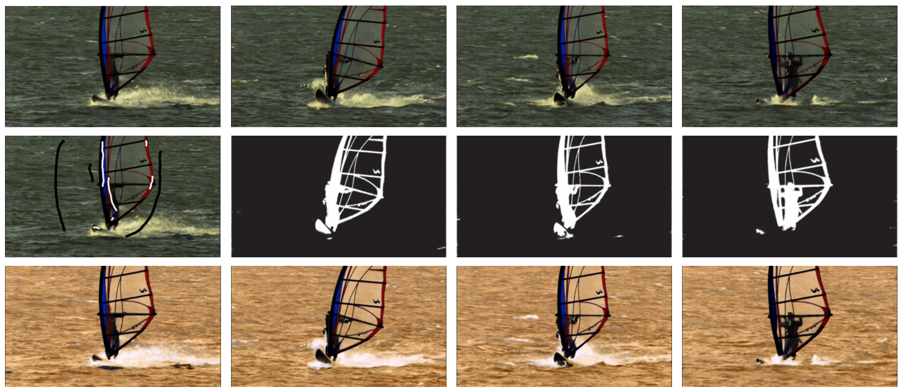

Figure 7: *A video example, where the water color is adjusted but the windsurfer is unchanged. One out of 123 frames was stroked. Row 1: frames; row 2: scribbles, masks; row 3: adjusted frames.*