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

Volume 29 (2010), Number 2

TouchTone: Interactive Local Image Adjustment Using Point-and-Swipe

Chia-Kai Liang, Wei-Chao Chen, and Natasha Gelfand †

Nokia Research Center Palo Alto

Abstract

Recent proliferation of camera phones, photo sharing and social network services has significantly changed how we process our photos. Instead of going through the traditional download-edit-share cycle using desktop editors, an increasing number of photos are taken with camera phones and published through cellular networks. The immediacy of the sharing process means that on-device image editing, if needed, should be quick and intuitive. However, due to the limited computational resources and vastly different user interaction model on small screens, most traditional local selection methods can not be directly adapted to mobile devices. To address this issue, we present TouchTone*, a new method for edge-aware image adjustment using simple finger gestures. Our method enables users to select regions within the image and adjust their corresponding photographic attributes simultaneously through a simple point-and-swipe interaction. To enable fast interaction, we develop a memory- and computation-efficient algorithm which samples a collection of 1D paths from the image, computes the adjustment solution along these paths, and interpolates the solutions to entire image through bilateral filtering. Our system is intuitive to use, and can support several local editing tasks, such as brightness, contrast, and color balance adjustments, within a minute on a mobile device.*

1. Introduction

Before a digital photograph is transmitted, displayed or printed, it often passes through a great number of manipulations, such as tonal adjustment, white balancing, and contrast enhancement. While global image adjustments suffice for many purposes, it is often necessary to perform local adjustments through region *selection*. Modern image editing software such as Adobe Photoshop includes various tools such as "magic wand", "magnetic lasso" and color range selection to facilitate this process. Despite their great utility for editing, however, the selection operation remains tedious even for professionals, in particular due to the intricate patterns of natural images.

On the other front, social networks and other online photo repositories are rapidly gaining popularity as a medium for sharing photos, and more and more photos are taken by camera phones and delivered through cellular networks. For example, Apple iPhone is becoming the most popular camera on Flickr [Fli]. Because many of these photos never enter the traditional download-edit-upload workflow, camera and mobile phone manufacturers have started to provide built-in image editing tools for users to quickly retouch photos before sharing. However, in the context of mobile devices, traditional desktop selection tools need to be reworked to accommodate for smaller screens where precise selection is difficult. The selection process also need to be fast and intuitive to encourage user adoption.

To this end, these on-camera editing tools may certainly benefit from stroke-based algorithms [AP08, LAA08, LFUS06]. Given a few roughly drawn strokes, these methods propagate the selection to the entire image through optimization. These methods, however, tend to require a great amount of memory and computation, making it rather difficult to adapt them to mobile devices. Moreover, it takes multiple conflict-free strokes to guide the optimization process. This interaction model requires users to iteratively draw or refine strokes to achieve desired selection.

† {liangck,weichao.chen,ngelfand}@gmail.com

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

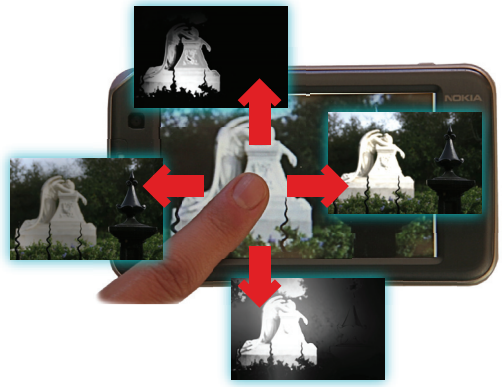


Figure 1: *TouchTone user interaction. Up and down swipes change the region size, while left and right swipes change the appearance of the image.*

In this paper we present *TouchTone*, a system for simple and efficient local image adjustments. Our interface, as illustrated in Figure 1, combines selection and adjustment into one single gesture. To perform image adjustment, the user simply *point* at a the region of interest, and *swipe* left and right to perform image adjustments, or up and down to change the size of the selection (Section 3).

To enable the proposed user interface on a computation-limited mobile device, we need a fast optimization algorithm for generating selection masks such that there is little to no perceived delay between pointing and swiping. Our key insight is that, while the optimization problem for propagating user strokes on 2D space in an edge-aware manner is expensive to solve, the *1D subproblem* with two endpoint-constraints is much cheaper. We therefore solve a collection of 1D subproblems along paths emitting from the clicked point and generate the selection mask, or influence map, through edge-aware interpolation. Our algorithm significantly improves the overall runtime and memory efficiency over full 2D optimization methods (Section 4).

Our system easily support various local editing operations, including brightness, contrast adjustments and recolorization (Figure 2). We further demonstrate our algorithm running on a mobile device with a touch screen and show that, even with its memory and computation limitations, our algorithm still proves to be efficient enough for interactive editing (Section 5).

2. Related Work

Tonal Adjustment The primary goal of tone mapping algorithms is to faithfully reproduce high-dynamic-range (HDR) images on low-dynamic-range (LDR) displays [RWPD05]. Such mapping could be a global curve or a local function with respect to the image contents in order to preserve details in the image [DD02, FLW02]. Tonal adjustment is useful for LDR images as well, since changing contrast and brightness can enhance the details or even convey different aesthetics and moods to the viewers. Bae et al. [BPD06] decomposed the image into two layers and transferred the photographic look from the model image. Farbman et al. [FFLS08] decomposed the image into multiple layers and adjusted details independently at different scales without causing halos. Even though these methods operate with local functions, they do not provide methods for localized image manipulations.

Stroke-Based Image Editing Instead of explicitly selecting regions of interest, one may indicate regions of interest by drawing a few rough strokes on the image and specifying their target values such as brightness and contrast on the strokes. These values are then propagated to the remaining pixels by solving an optimization problem. This concept has been adopted in many different applications including image appearance adjustment [LFUS06, AP08], colorization [LLW04, YS06, LAA08], image and video segmentation [LSTS04, CSB08], and image matting [BS07, LLW08, WBC*05, WC05]. When the target value is a segmentation or alpha matte the result is dubbed as a *selection mask* or *influence map*. Given the mask, the user can later apply several different editing operations to the region of interest.

Most of these methods are designed to retain the adjustment values on the strokes as closely as possible and propagate the adjustment either to nearby pixels where the image is locally smooth, or to pixels of similar appearance. By formulating these requirements as energy functions, one can obtain optimal adjustment values by minimizing the sum of those functions [LLW04, LFUS06]. To improve the results, [LAA08, WC05] added per-pixel classification into the energy function, and [AP08] incorporated long distance interaction penalties into the energy functions, both at the cost of additional computation.

Because most stroke-based algorithms solve a global optimization problem, whenever a new stoke is added, the energy function needs to be updated which triggers the optimization computation again. Although this process can be accelerated by exploiting the smoothness of solution [XLJ*09], it is still too expensive for mobile devices. Moreover, to avoid a trivial globally flat solution, the user has to draw at least two kinds of strokes, one positive and one negative, without any conflict. On the contrary, our system uses a new set of constraints at every edit operation. In practice, because of the predictability and fast response of our approach, most novice users can reproduce the examples included in our paper within well under one minute.

Geodesics Methods The geodesic methods [BS07, CSB08] set the influence value of a pixel based on the geodesic distances to the positive and negative constraints. While the results look similar to the falloff of our paths, our method is fundamentally different in that ours simultane-

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

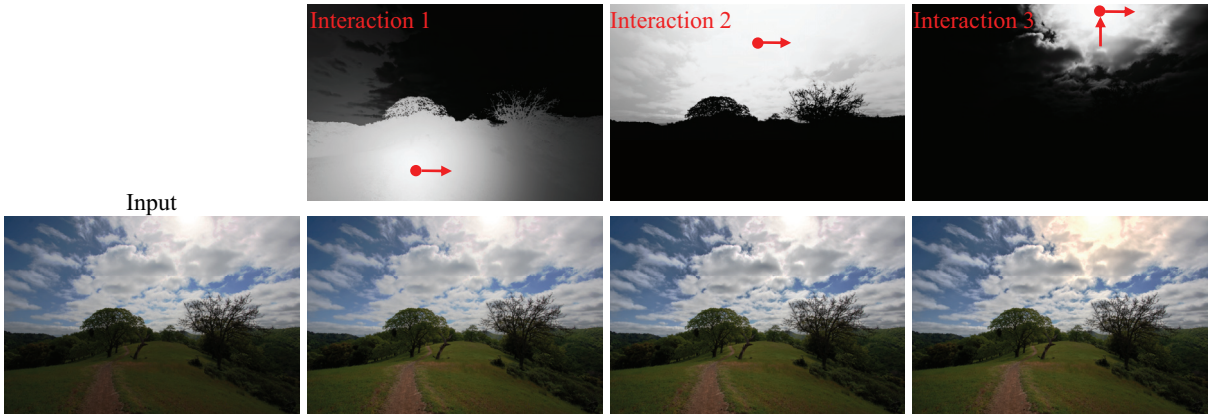


Figure 2: *An example of the editing process in TouchTone. The red dots represent click points and the arrows represent the swipe directions. From left to right, we sequentially adjust the brightness of the grass, the contrast of the sky, and the color temperature around the sun, by only three point-and-swipe interactions in 20 seconds.*

ously generates the paths and negative constraints. It may be possible to use geodesic distances with our proposed user interface, but further study is needed to convert these distances into influence maps.

Edge-Aware Brush Instead of inferring the selection mask from strokes, the edge-aware brush [CPD07, JH08] allows users to draw the selection mask or the editing effects by the simple brush tool, while the dissimilar regions are automatically avoided even if the brush touches them. To this end, the edge-aware brush can be viewed as complementary to our method since it provides the controllability for a more precise selection at the expense of more interaction time. However, it cannot adjust the scale of the selection mask, the editing strength, and the location of the mask in a single gesture, as in our user interface. Also, since the definition of edge-awareness is algorithm-dependent, the edge-aware brush may have similar limitations as our algorithm does.

Finally, we note that several commercial software packages have provided methods to simplify the selection process. For example, the adjustment brush in Adobe Lightroom 2 allows users to paint the adjustment without crossing object boundaries, similar to the method in [CPD07]. The U Point Technology [UP] can generate an content-aware selection mask through a user-clicked point. To our knowledge, there is no publicly available technical descriptions for these products.

3. User Interface

The primary goal of our user interface design is to enable quick image editing on mobile devices with small-sized touch screens. The interface should be intuitive while providing sufficient local region control. Therefore, we wish to hide the region selection process as much as possible away from the user. Because modern mobile devices have widely adopted point-and-swipe or flick gestures, we base our user interface on this basic finger gesture. That is, we wish to enable the user to *point* at an object and *swipe* to perform adjustments on the selected object.

However, one single point is not informative enough to communicate a user's intention regarding the shape and size of the selection. To avoid complicating the interface, we allow the user to control a scale parameter and let the algorithm resolve the exact selection mask in an edge-aware manner. The final design of our user interface is shown in Figure 1, where vertical finger swipes change the selection scale, and horizontal swipes apply the adjustment to the selected region.

Compared to multi-layer image editing frameworks on desktop computers, our design minimizes the switches between different operations and maximizes the region for image display. Also, the scale selection is sticky, meaning that the user can select a scale and apply adjustments over many different parts of the image without having to specify the scale again. Figure 2 shows a typical workflow using this interface. The overall editing time, including user and computation time, is well under a minute for novice users.

Given the input location and scale parameters, the system needs to compute a sensible selection through optimization. However, most optimization algorithms used by stroke-based approaches require both positive and negative constraints, which means that we need to set additional constraints if we were to adopt the those algorithms. In this respect, each user interaction should be equivalent to setting up an optimization process similar to [LFUS06], where two sets of constraints are generated implicitly - one at or close to the user-selected point, and another outside the intended region. Moreover, the full 2D optimization is approximately two orders of magnitude too slow on our target devices. In this paper we propose an algorithm that solves both the implicit-constraint and performance issues. The details of the proposed algorithm are presented in the next section.

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

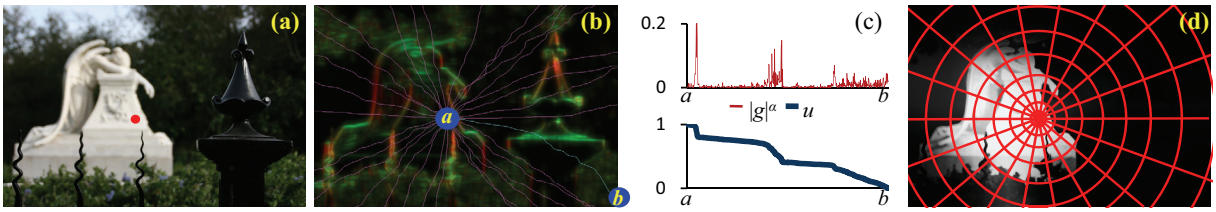


Figure 3: *Overview of our algorithm. From (a) the input image with a selected point (red dot), we compute its (b) diffused gradient field and use it to generate paths from the selected point. Red and green channels represent the gradients in x and y directions, respectively. (c) The gradients $|g|^\alpha$ (top) from the cyan-colored path in (b) are used to compute the influence values u (bottom). (d) The final influence map interpolated using a spatially-variant bilateral filter.*

4. Algorithm

Our algorithm is summarized in Figure 3. Given a single selected point in the image (Figure 3a), we use the diffused gradient field of the image to guide 1D paths originating from this point toward the image boundaries (Figure 3b). The endpoint constraints of the paths are automatically assigned. Then, we solve for the influence values along each path using a method modified from Lischinski et al. [LFUS06] (Figure 3c). Finally, we use a scattered bilateral interpolation to generate an influence map (Figure 3d). This influence map represents the selected region for image adjustments, and the adjustment strength to each pixel.

For this purpose, we begin out discussion using the algorithm in [LFUS06] as our basis. This algorithm starts with a set of user-drawn strokes and their associated adjustment values, and propagates them to other pixels in an edge-aware fashion by solving a large linear system $\mathbf{Af} = \mathbf{b}$. Let us denote the pixels of the k-th stroke as vector $\mathbf{w}_k$. The vector $\mathbf{b}$ incorporates the user constraints $\{\mathbf{w}_k\}$ as well as their corresponding scalar target values $\{\nu_k\}$, such that $\mathbf{b} = \sum_k \nu_k \mathbf{w}_k$. For an image with n pixels, $\mathbf{A}$ is an $n \times n$ sparse symmetrical matrix consisting of two components $\mathbf{A} = \mathbf{H} + \mathbf{W}$, where $\mathbf{H}$ depends only on the input image, and $\mathbf{W}$ is a diagonal matrix depending on the user strokes:

$$\mathbf{H} = [h_{i,j}] = \begin{cases} -\dfrac{\lambda}{(|g_{i,j}|^\alpha + \varepsilon)} & i \neq j \text{ and } j \in \mathcal{N}_i \\ -\displaystyle\sum_{k \in \mathcal{N}_i} h_{i,j} & i = j \\ 0 & \text{otherwise} \end{cases}, \qquad \mathbf{W} = \sum_j \text{diag}(\mathbf{w}_k). \tag{1}$$

For each pixel i, $\mathcal{N}_i$ denotes the indices of its four neighbors. $g_{i,j}$ denotes the gradient between two pixels i, j, computed as the log-luminance differences for HDR images, and luma differences for LDR images. A regularization term ε is added to ensure the stability of the linear system. The user-selected parameters, λ and α, control the sensitivity of solution $\mathbf{f}$ with respect to image gradient changes. Applying the adjustment map $\mathbf{f}$ to the input image yields the adjusted result.

As suggested by [LFUS06], one can solve for the contributions from each constraint separately as basis functions $\mathbf{u}_k$, $\mathbf{Au}_k = \mathbf{w}_k$, such that $\mathbf{f} = \sum_k \nu_k \mathbf{u}_k$. Vector $\mathbf{u}_k$ then defines the influence map for constraint $\mathbf{w}_k$, and a new image with different adjustment values can be obtained through simple linear combinations of $\{\mathbf{u}_k\}$ without solving the system again. However, all the basis influence functions $\{\mathbf{u}_k\}$ still need to be recomputed whenever a new stroke is added.

Next we show that, while solving this linear system requires expensive iterative methods, the solution can be computed very efficiently in 1D. As a result, we can compute the solution along some 1D paths emanating from the selected point, and then complete this partial solution through interpolation.

4.1. One-Dimensional Constraint Propagation

If we consider the case where each pixel contains only two neighbors in Equation 1, namely when the pixels form a continuous *path* within the original image, the matrices $\mathbf{H}$ and $\mathbf{A}$ both become symmetrical tridiagonal matrices. As a result this problem appears similar to a classic partial differential equation in 1D. Suppose we provide this new system of l pixels with two boundary conditions in the form of a single-pixel constraint at each end of the path, we have:

$$\mathbf{Au}_k = \mathbf{w}_k, \; k = \{0, 1\}, \tag{2}$$

where

$$\begin{cases} \mathbf{A} = \mathbf{H} + \mathbf{W} = \mathbf{H} + \text{diag}(\mathbf{w}_0) + \text{diag}(\mathbf{w}_1), \\ \mathbf{w}_0 = [1, 0, 0, ..., 0]^T, \text{and} \\ \mathbf{w}_1 = [0, 0, 0, ..., 1]^T. \end{cases} \tag{3}$$

For simplicity we denote $h_i \doteq h_{i,i+1}$, $g_i \doteq g_{i,i+1}$, and denote the solution on a path of length l as $\mathbf{u}_0 = [u_0, u_1, ..., u_{l-1}]^T$. Taking any two consecutive rows $(i, i+1), \forall i \in \{1, 2, ..., l-3\}$ from the matrix $\mathbf{A}$ and substituting Equation 1 into the system, we obtain this relationship:

$$h_i u_i - (h_i + h_{i+1}) u_{i+1} + h_{i+1} u_{i+2} = 0 \tag{4}$$

$$\Rightarrow h_i (u_i - u_{i+1}) = h_{i+1} (u_{i+1} - u_{i+2}), \tag{5}$$

$$\Rightarrow \Delta u_i |g_{i+1}|^\alpha = \Delta u_{i+1} |g_i|^\alpha, \tag{6}$$

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.



Figure 4: *Gradient renormalization. (a) Input image. (b,c) The path solutions and interpolated solution without renormalization. Notice the incorrect fall-off at the image boundaries. (d,e) Renormalizing against maximum gradient drop. Both results are generated with scale $s = 0.5$. We widen the path solutions in (b,d) for better visualization.*

which means that at every pixel i, the change of the influence map $\Delta u_i \doteq u_i - u_{i+1}$ should be inversely proportional to h_i, or after substituting Equation 1, proportional to the gradient raised to the power α. During the substitution process, we remove ε because Equation 6 is numerically stable. When $\lambda \to 0$, the solutions at the end points are dominated by user constraints and we can efficiently approximate $\mathbf{u}_0$ simply as a descent from 1 to 0 that respects local gradients,

$$\begin{cases} u_0 = 1, \\ u_i = u_{i-1} - \dfrac{|g_{i-1}|^\alpha}{\sum_i |g_i|^\alpha}, \quad i = \{1,2,...l-2\}, \\ u_{l-1} = 0. \end{cases} \tag{7}$$

With this formulation, we can efficiently compute the influence maps along paths within the image. Next we discuss how to generate suitable paths and content-adaptive boundary conditions.

4.2. Path Generation and Boundary Conditions

Our goal is approximating the full 2D optimization process through importance sampling along 1D paths. In our experiments we find that the distribution of paths plays a critical role. Generally, the distribution should follow three criteria:

1. The sampling density should be higher around the clicked point for higher accuracy.
2. The samples should distribute evenly away from the clicked point for good coverage.
3. The sampling bias should be minimized, that is, edges (i.e., strong gradients) belonging to the same object should not be crossed multiple times.

To satisfy Criteria 1 and 2, one may generate evenly distributed paths emanating outward from the selected point. However, this method is not stable when a path runs at a grazing angle to an image edge, since in this case a slight change of the selected point can cause a path to cross the edge slowly or even several times in rapid succession. This leads to over-sampling some edges and under-sampling others. We discovered that we can greatly improve the stability of the final solution map by cutting through the edges as orthogonal as possible, which satisfies Criterion 3. However, the distribution may no longer be evenly distributed.

We balance the conflict between the Criteria 2 and 3 by using a particle system. First, each particle is given an initial unit velocity $\mathbf{v}_0$ toward the image boundary. The radial directions of the particles are distributed evenly, which thus satisfies Criteria 1 and 2. We then use the image gradients to guide the particles such that it satisfies Criterion 3. One possibility is to treat the image gradients as a gravitational field such that stronger edges can attract the particles at greater distances. However, direct application of classic mechanics will cause the particles to incorrectly oscillate around strong edges. To solve this problem, we drop the signs when we sum up the gravitational force, which in turn allows us to precompute the field by diffusing the image gradient $\mathbf{g} = [g_x, g_y]$ at each point to its neighbors,

$$\mathbf{g}_d(x,y) = \sum_{u \neq x} \sum_{v \neq y} \frac{[|g_x(u,v)|, |g_y(u,v)|]}{(|u-x|)^\gamma + (|v-y|)^\gamma}. \tag{8}$$

We chose $\gamma = 1$ for our experiments. This diffused gradient field $\mathbf{g}_d$ (Figure 3b) allows us to simulate the edge attraction for Criterion 3. That is, when the particle passes through point (x,y), its speed is updated from $\mathbf{v}$ to $\mathbf{v}'$ as

$$\mathbf{v}' = \frac{\mathbf{v} + w_u \mathbf{v}_0 + w_g s(\mathbf{v}) \odot \mathbf{g}_d(x,y)}{\|\mathbf{v} + w_u \mathbf{v}_0 + w_g s(\mathbf{v}) \odot \mathbf{g}_d(x,y)\|_2}, \tag{9}$$

where $s(\mathbf{v})$ denotes the signs of the components in v, operator $\odot$ denotes per-element vector multiplication, and $\|\cdot\|_2$ is the ℓ_2 norm. The use of $s(\mathbf{v})$ means each particle is guaranteed to exit the image because its speed will not cross to a different quadrant. The parameters w_u and w_g balance Criteria 2 and 3, and we empirically set $w_u = 1.3$ and $w_g = 1$. The trajectories of the particles define our sampling paths.

The simple boundary conditions in Equation 7 always starts with one at the selected point and drop to zero at the image boundary. This approach introduces artifacts, in particular when the selected point and the image boundaries belong to the same visual region (Figures 4b and c). We address this problem by renormalizing the rate of influence value decay by the largest accumulated gradient over all M paths. We calculate $G_{max} = \max\{G_0, G_1, ..., G_{M-1}\}$, where $G_j = \sum_i |g_i^m|^\alpha$ is the accumulated gradient along the m-th

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

path. Equation 7 is then revised into:

$$\begin{cases} u_0 = 1, \\ u_i = \max(0, u_{i-1} - \frac{|g_{i-1}|^\alpha}{G_{max}}), \quad i = \{1,2,...,l-1\}. \end{cases} \tag{10}$$

If a path does not pass through any strong edge, it should belong to the same region that the user clicked. According to Equation 10, the solutions along this path would be close to one, thereby improving the overall quality of the influence map (Figures 4d and e).

4.3. Scattered Bilateral Interpolation

After computing solutions along the paths, we use them to generate an influence map for the whole image. Because all the paths are solved independently, solutions for different paths could be inconsistent, and proper filtering must be applied to remove this variation. Also, as we mentioned previously, the influence values of two pixels should be similar when they are close or similar to each other. To fulfill both requirements, we use a cross bilateral filter [ED04]. Given on-path pixels q's with solutions u_q's, the interpolated influence value u'_p at the pixel p is

$$u'_p = \frac{\sum_{q \in \mathcal{S}} f_d(\|p-q\|_2) f_r(\|I_p - I_q\|_2) u_q}{\sum_{q \in \mathcal{S}} f_d(\|p-q\|_2) f_r(\|I_p - I_q\|_2)}, \tag{11}$$

where I_p is the pixel value of p, $\mathcal{S}$ is the set of all pixels on the paths, and f_d and f_r are the distance and range filter kernels, respectively. This non-linear filter can be efficiently performed using the bilateral grid technique [PD09, CPD07].

However, we find the results not satisfactory because the paths are not evenly distributed across the image. The path density away from the clicked point is lower, and thus we have to use a larger filter kernel to fill the missing values. On the other hand, we should use a smaller kernel around the clicked point where the path density is higher. While a regular bilateral grid is efficient when the filter kernel is spatially invariant, a uniform filter kernel can result in leakage or holes in the solution. Although we can build dynamic structures such as a KD-tree in the affinity space [AGDL09, XLJ*09] to support spatially variant kernels, this turned out to be too expensive because the tree need to be updated every time a new new selection is made.

Upon closer inspection, the path density decreases linearly with the distance to the clicked point, and therefore the kernel width should increase linearly with that distance. That is, when f_d is a Gaussian filter,

$$f_d(\|p-q\|_2) = \exp(-\|p-q\|_2^2 / \sigma_p^2), \tag{12}$$

its variance σ_i^2 should be a function of the distance between p and the clicked point c,

$$\sigma_p^2 = \|p-c\|_2 \sigma_0^2. \tag{13}$$

This regular spatial variance structure allows us to develop a novel grid structure in polar coordinates centered at the clicked point (Figure 3d). This 3D grid spans the angular dimension, the radius dimension, and the range (intensity) dimension. While the grid width is constant along the angular and range dimensions, it grows linearly along the radius dimension. When we perform filtering on this grid, the result well approximates the result using Equation 12.

For each path, we first splat its solutions to the bilateral grid. Even though a path is continuous in the image, it can become scattered in the 3D grid along the range dimension when passing through strong edges. Therefore, we rasterize the 1D paths in the 3D grid to ensure there is no range discontinuity. We then perform three separable 1D low-pass filters along three dimensions for blurring followed by trilinear interpolation to obtain the filtered samples. For the final result, we optionally apply a sigmoid function as in [LLW08, LAA08] to enhance the contrast of the output map. Specifically, the influence value u'_p at point p is transformed to u''_p by

$$u''_p = (1 + \exp(-\beta(u'_p - 0.5)))^{-1}, \tag{14}$$

where we set $\beta = 10$ in all experiments.

Since the bilateral interpolation can propagate values to similar regions that are not spatially connected, the interpolated influence map no longer decreases monotonically as the original solutions on paths. This leak-through attribute is particularly useful when we wish to select similar regions without computing the all-pair affinity [AP08], as shown in Figure 5. In comparison, using similar constraints, the influence maps generated by [LFUS06] would include only one flower, and we find our results tend to match user intentions better.

We also note that our approach is related to the method in [CPD07] which splats the solutions on strokes to the bilateral grid. However, since the strokes are often localized in both the spatial and range domains, this method performs an additional optimization step to fill the empty grid nodes. In our method, because the paths span the whole image, the partial solutions are well distributed, and thus we can replace the optimization process with simple filtering.

4.4. Influence Region Control

So far our algorithm generates a fixed, deterministic influence map from each selected point. However, we mentioned earlier that we wish to provide users with the ability to adjust the scale or size of the influence map. We achieve this by adding a scale parameter s and replacing G_{max} with $G'_{max} = sG_{max}$ in Equation 10. The solutions along any individual path would tilt up for $s > 1$ or down for $s < 1$, effectively changing the size of the influence map. We combine this adjustment with the user interface described in Section 3. As illustrated in Figure 1, the user can change s by swiping upward or downward, and the system re-calculates the path solutions and interpolates a new influence map.

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.



Figure 5: *Selection propagation. Our approach is able to select nearby flowers using a single clicked point (red dot).*

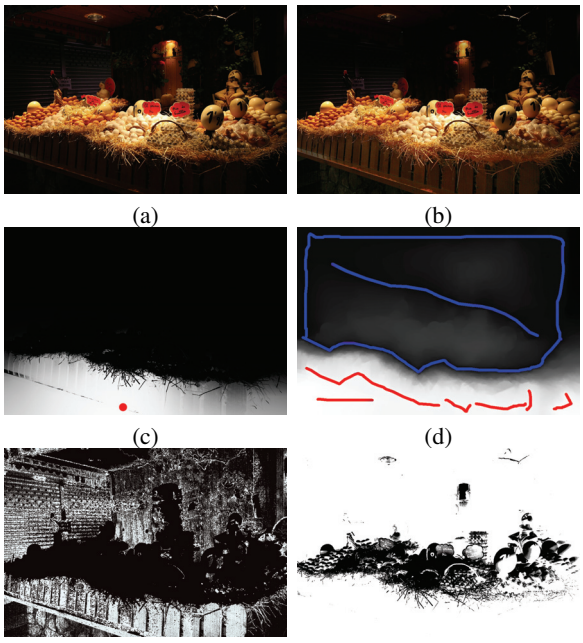


Figure 6: *(a) The input image, (b) the adjusted result, (c) one of the selections generated using our method, (d) the strokes and the selection generated by [LFUS06], (e) the Photoshop color range selection, and (f) shadow selection.*

5. Results

In this section we first demonstrate various editing results using our system, compare our method to previous ones in terms of speed and the characteristics of selection, and discuss the limitations in the end. All the following results are generated with 32 paths per image and $\alpha = 1$. In bilateral interpolation, we set the angular grid resolution to 20° and range grid width to 0.07 when the dynamic range is normalized to 1. Please refer to the accompanying video and supplemental materials for real-time interactive editing sessions and additional high-resolution results and comparisons to other methods.

The first example is shown in Figure 6, where the foreground is lightened and the background is darkened using only four interactions, and the total editing time is less than 10 seconds. In fact, all of the example images take less than one minute for novice users of our interface. Note that in this



Figure 7: *Left: the input image. Right: the contrast and brightness of dogs are selectively enhanced while the contrast of the grass is lowered.*



Figure 8: *Left: the input image. Right: the exposure of the sky is reduced and that of the subject is increased.*

image, because the background and foreground are similar, traditional selection tools such as *color range* and *shadow* in Photoshop would not perform properly for the desired output of our image. On the other hand, the stroke-based method of [LFUS06] blurs the boundary details, and requires more computation (Section 5.1).

Another application is local adjustment of contrast. Inspired by the two-scale decomposition approach [BPD06], we precompute the base and detail layers using bilateral filter (better decomposition methods, such as [FFLS08], are equally applicable here). The user then uses the point-and-swipe interface to adjust the weight of the detail layer. Additional results are shown in Figures 2, 7, 8, and 9.

Finally, we apply our algorithm for local white balance adjustment. The input images and our result are shown in Figure 10. This scene contains two light sources of different spectrums, and thus the images under global white balance coefficient appear either greenish or reddish. While the light mixture of each pixel can be estimated using complex algorithms [HMP*08], our result is generated through a few point-and-swipe interactions. In Figure 2 we use the similar method to adjust the color temperature around the sun.

5.1. Performance

The primary difference between our method and the stroke-based algorithms is in the construction of the influence maps. After the influence maps are generated, the processing speed depends on the type of adjustments. The three major steps in our algorithm are path generation, 1D constraint propagation, and scattered bilateral interpolation. Because the length of each path is bounded by the image width or height, the complexity of the first two steps are $O(M\sqrt{n})$, where M is

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.



Figure 9: *Left: the input image. Right: the adjusted image. The contrast of the sky and the foreground is increased, and that of the far grass is decreased. The flowers are lightened and the forests are darkened.*



Figure 10: *Local white balance. Left: the input image. Middle: globally adjusted image. Right: our result.*

the number of paths and n is total pixel number. The complexity of the third step is $O(n)$ where most of the computation is spent in the trilinear interpolation. The memory used for the selection algorithm is less than $1/5$ of the image size in our implementation.

Figure 11 shows the performance comparison between our algorithm and our implementation of the 2D optimization method [LFUS06] on two platforms, a Macbook Pro with Intel 2.5GHz Core 2 and 4GB RAM (only one core is used), and a Nokia N810 Internet Tablet with a TI OMAP at 400MHz and 128MB DRAM. Note that while the complexity of [LFUS06] grows linearly with the number of layers, we confine the comparison to the two-layer case. Our method is faster by a factor of $30-80$ on the laptop and $30-40$ on the tablet. Even on the mobile device, our method runs at interactive rates for images of moderate size.

Characteristics of the Influence Maps In Figure 12 we show the influence maps generated by our method and various other stroke-based and point-based methods. The stroke-based method of [LFUS06] is not designed for segmentation and has an overall smooth appearance (Figure 12b). Geodesic matting [BS07] (Figure 12c) and ScribbleBoost [LAA08] (Figure 12d) generate better results in terms of sharpness, but these approaches can incur additional processing complexity through the use of per-pixel classification. By clicking a single point, our method can generate comparable results for local adjustment purposes (Figure 12g). If we use the same end-point constraints as input strokes to [LFUS06], the results are always very fuzzy (Figure 12e). Finally, the commercial U Point Technology tends to generate masks with many holes (Figure 12f).

Even though our method may not always generate a influence map that perfectly matches user intentions, due to its fast processing speed, the user can quickly point to different

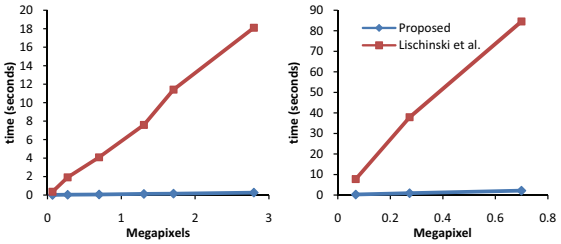


Figure 11: *The computation times on the Macbook Pro (left) and the Nokia N810 Internet Tablet (right).*

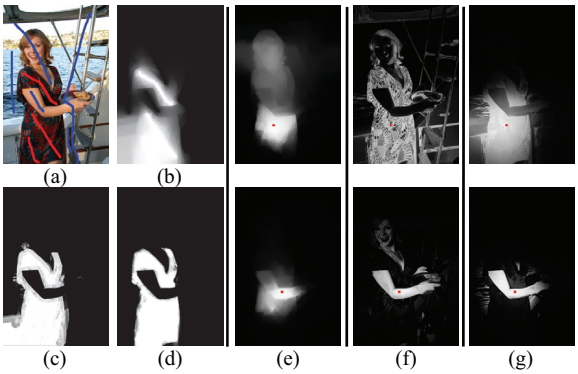


Figure 12: *(a) The input image overlaid with three positive (red) and three negative (blue) strokes. (b,c,d) The results generated by [LFUS06], Geodesic Matting [BS07], and ScribbleBoost [LAA08], respectively. (e) The results of [LFUS06] using the point-wise constraints (red dots). (f) The results of U Point Technology. (g) Our results.*

image regions to compensate for this. For example, in Figure 12g the lady's arm would be slightly included in the first adjustment. In this case, the user can point to the arm and reverse the changes. On the contrary, for the stroke-based methods, adding a stroke would often globally affect the influence map, making the addition and removal of strokes a sometimes difficult task.

5.2. Limitations

The inherent limitation of our system comes from our user interface construct that uses points as the selection mechanism. While a stroke-based interface can be used to select regions of various different appearances, ours is not expressive enough for this purpose. On the other hand, if the user points and samples different parts of the region, the combined effect should achieve the user's intention in many cases.

In addition, because the solution on each path is a smooth, monotonic function, our system can not produce sharp segmentation that contains many non-monotonic variations. An example is our algorithm can not simultaneously select all the white squares on a checker board. This can be addressed by optimizing the solution on paths using the all-pair affin-

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

ity [AP08], at the expense of extra computations. Finally, the 3D bilateral grid can not distinguish regions of similar intensities but different colors. We have experimented with the 5D bilateral interpolation [AGDL09] that incorporates color information, but the increase in computational cost does not warrant the limited improvement we have observed.

6. Conclusion

We have proposed *TouchTone*, a system for content-aware local image editing. It uses an intuitive interface which combines the region selection and image adjustment operations into a single point-and-swipe gesture. By dividing the 2D optimization problem into 1D subproblems and interpolating the 1D solutions, our algorithm significantly reduces computational costs for local image adjustment. As we have demonstrated, our method is efficient enough to run interactively on a commodity mobile device. We have applied our technique for a variety of editing operations and demonstrated its applicability to a wide range of input images.

In the future, we are looking forward to extending our algorithm to support multiple constraints through point or stroke based inputs. We would also like to explore more sophisticated interpolation algorithms that allow us to trade off the influences between gradient and spatial/range support. We believe that these research directions, when properly combined, will provide a more intuitive interaction model and make image editing both easy and fun.

Acknowledgements

The authors thank the reviewers and members of the Nokia Research Center for their insightful suggestions. We also thank Kari Pulli, Li-Yi Wei and Lance J. Williams for encouragements and proofreading. The Sunflower in Figure 5: from http://www.flickr.com/photos/33805306@N00/80386524 (Creative Commons license); the Angel in Figures 1 and 3: courtesy of Marc Levoy.

References

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph. 28*, 3 (2009), 21:1–21:12.

[AP08] AN X., PELLACINI F.: AppProp: all-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (2008), 40:1–40:9.

[BPD06] BAE S., PARIS S., DURAND F.: Two-scale tone management for photographic look. *ACM Trans. Graph. 25*, 3 (2006), 637–645.

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

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. 26*, 3 (2007), 103:1–103:10.

[CSB08] CRIMINISI A., SHARP T., BLAKE A.: Geos: Geodesic image segmentation. In *Proc. 10th European Conference on Computer Vision* (2008), Springer-Verlag, pp. 99–112.

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. 21*, 3 (2002), 257–266.

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (2004), 673–678.

[FFLS08] FARBMAN Z., FATTAL R., LISCHINSKI D., SZELISKI R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph. 27*, 3 (2008), 67:1–67:10.

[Fli] Flickr: Camera Finder. http://www.flickr.com/cameras.

[FLW02] FATTAL R., LISCHINSKI D., WERMAN M.: Gradient domain high dynamic range compression. *ACM Trans. Graph. 21*, 3 (2002), 249–256.

[HMP*08] HSU E., MERTENS T., PARIS S., AVIDAN S., DURAND F.: Light mixture estimation for spatially varying white balance. *ACM Trans. Graph. 27*, 3 (2008), 70:1–7.

[JH08] JR. D. R. O., HARRIS M. K.: Edge-respecting brushes. In *Proc. 21st Annual ACM Symposium on User Interface Software and Technology* (2008), ACM, pp. 171–180.

[LAA08] LI Y., , ADELSON E. H., AGARWALA A.: ScribbleBoost: adding classification to edge-aware interpolation of local image and video adjustments. *Computer Graphics Forum 27*, 4 (2008), 1255–1264.

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

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

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

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

[PD09] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. *IJCV 81*, 1 (1 2009), 24–52.

[RWPD05] REINHARD E., WARD G., PATTANAIK S., DEBEVEC P.: *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2005.

[UP] U point technology. Nik Software, http://www.upoint.com/.

[WBC*05] WANG J., BHAT P., COLBURN R. A., AGRAWALA M., COHEN M. F.: Interactive video cutout. *ACM Trans. Graph. 24*, 3 (2005), 585–594.

[WC05] WANG J., COHEN M. F.: An iterative optimization approach for unified image segmentation and matting. *Proc. ICCV* (2005), 936–943.

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using K-D tree. *ACM Trans. Graph. 28*, 5 (2009), 118:1–118:6.

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

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.