Supervised Semantic Gradient Extraction Using Linear-Time Optimization

Shulin (Lynn) Yang
University of Washington
yang@cs.washington.edu

Jue Wang
Adobe Research
juewang@adobe.com

Linda Shapiro
University of Washington
shapiro@cs.washington.edu

Abstract

This paper proposes a new supervised semantic edge and gradient extraction approach, which allows the user to roughly scribble over the desired region to extract semantically-dominant and coherent edges in it. Our approach first extracts low-level edgelets (small edge clusters) from the input image as primitives and build a graph upon them, by jointly considering both the geometric and appearance compatibility of edgelets. Given the characteristics of the graph, it cannot be effectively optimized by commonly-used energy minimization tools such as graph cuts. We thus propose an efficient linear algorithm for precise graph optimization, by taking advantage of the special structure of the graph. Objective evaluations show that the proposed method significantly outperforms previous semantic edge detection algorithms. Finally, we demonstrate the effectiveness of the system in various image editing tasks.

1. Introduction

This work is motivated by gradient-domain image editing, which refers to the class of methods that achieve image editing by directly manipulating the gradient filed of the image. This requires transforming the image into the gradient domain, applying the desired gradient filter, and finally reconstructing the output image from the modified gradient filed. Recent work [17, 5] demonstrate that this framework is powerful and versatile enough for accomplishing various editing tasks such as blending, saliency sharpening, relighting, stylization, JPEG de-blocking, etc.

One major limitation of previous gradient-domain editing approaches is the lack of local user control for fine-level editing. For instance, the GradientShop system [5] used simple global criteria (such as the magnitude of the gradient value) to select pixels that will be processed by a filter. This often leads to artifacts when the desired and unwanted gradients cannot easily be separated by the global criterion. In the example shown in Figure 1, applying the global saliency sharpening filter boosts all details in the image (Figure 1b). However, bags under eyes are also enhanced, which may not be desired.

To improve its controllability, it is natural to adopt an interactive approach for local gradient editing. A straightforward approach is to ask the user to provide an accurate operational mask to define the application range of the filter. However, for natural images the gradient field is usually too complicated for accurate manual segmentation. As shown in the example in Figure 1d, it is tedious for the user to draw an accurate mask to only cover the contour of the eyelid without touching strong edges nearby. This is even harder to achieve in modern touch-based devices.

In this paper we propose a dominant gradient selection approach to solve the accurate gradient selection problem for interactive gradient-domain editing. The specific goal of our algorithm is the selection of a semantically dominant subset of edge/gradient points from all pixels in a user-marked area. There are two criteria for the selection:

- the selected pixels ensemble the semantically-dominant edges/gradients under the scribble, even if locally their gradient magnitudes are smaller than their nearby edges;
- The selection is well-contained and coherent so that applying filters to the selected pixels will not result in discontinuities in the final result.

We propose a new, graph-based approach to meet these criteria. We extract low-level edgelets (which are small clusters of edges) as primitives for graph construction, and jointly consider their geometric and appearance features in formatting energy terms. Given the complexity of the energy terms, we show that the proposed MRF cannot be effectively solved by commonly-used approximation methods such as graph cuts. Luckily, given the special structure of our graph, we can turn the MRF into a triangulated graph with limited maximum cliques, which is solvable in linear time. This leads to an efficient solution which gives the user rapid feedback for interactive editing. Objective evaluations show that our system significantly outperforms previous approaches in terms of accuracy. We also show how various image editing tasks, such as contour snapping, saliency sharpening and image stylization can benefit from using the proposed gradient selection method.

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.364

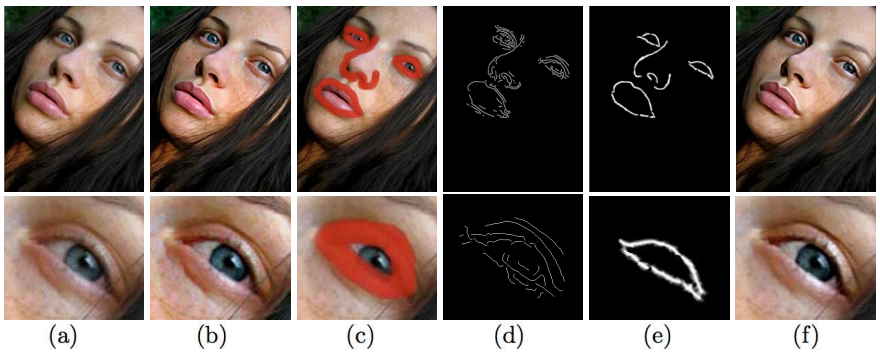


Figure 1. An example of saliency sharpening. (a) input image; (b) global filtering result; (c) user scribbles for gradient selection; (d) all edges in user selected region; (e) gradients selected by our approach; (f) locally filtering of the selected gradients. Note the color tone change and the enhanced eye bags in (b).

2. Related Work

We briefly review a few approaches that are closest to our work, in the area of contour extraction and gradient-domain editing.

Contour extraction. Global probability boundary (gPb) [13, 3] is the state-of-the-art on contour detection. It combines multiple local cues, such as brightness, color and texture, into a global optimization framework. However it is difficult to get long, clean, and coherent contour lines directly from the probability map, as shown in Fig. 5. In contrast our method is able to deliver contour lines with those desired properties.

Active contour model [9, 23], or Snake, is also a classic and popular choice for object segmentation. In this setting, the user specifies an initial contour close to the true object boundary, and the algorithm seeks for a final curve that minimizes an energy as a sum of both internal and external energy terms. However, this approach often converges to local minima that are far away from the desired selection, especially when the local gradient field is complex. Our approach uses precise graph-based optimization with proper energy formulations to avoid local minima.

Intelligent scissors [15] and magnetic lasso in photoshop are classic edge snapping tools. The former requires the user to select a few points accurately on image boundaries, then employ an algorithm to complete the gaps between the selected points. The latter selects image boundary by seeking the edge that is nearest to the current location of the mouse. Both methods require very precise user input for accurate selection. In contrast, our system only requires rough scribbles to cover the desired gradients, which is particularly an advantage on tablet devices where accurate finger input is impossible.

Beside curved-based tools, there are a large group of region-based interactive image editing systems [12, 11]. These tools aim at a different goal of selecting smooth image regions rather than gradients based on user scribbles.

Gradient-domain editing. There is a rich body of work on gradient-domain image manipulation in both vision and graphics field. Specifically, Orzan et al. [16] converted photographs into abstract renditions that capture their salient features. Zeng et al. [21] proposed a unified variational image editing model for image editing, which largely depends on gradient-domain adjustment. Agrawal et al. [2] used a gradient projection technique for a class of edge-suppressing operations. Other applications of gradient-domain image editing include tone mapping, image composition, image stitching, color interpolation, etc.. We refer the readers to a recent review [1] for these applications.

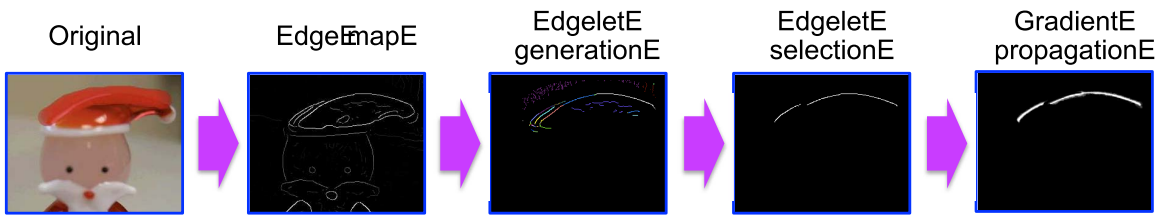
Our approach is largely motivated by the recent GradientShop system [5], which presented an optimization framework for exploring gradient-domain operators for various image and video editing tasks. Our gradient selection method can be integrated into this system to create an interactive and fully controllable gradient-domain image editing workflow.

3. Dominant Gradient Extraction

In this section, we first illustrate our pipeline for gradient extraction. Then, an energy minimization framework is presented, which captures the criteria for semantic dominancy of user scribbled region. Last, a linear time optimizer is proposed for minimizing the energy function, which is crucial to real-time image editing tasks.

3.1. Algorithm Overview

As illustrated in Figure 2, our gradient extraction method consists of four steps. Given the input image, we first apply a recent probabilistic edge detection method [13] to select a set of sparse edge points (local maxima in the edge probability map) as atoms for processing. This is because pixel gradients are affected by many factors such as noise and shading, and we are only interested in semantically-meaningful contours, i.e. edges separating different im-



(a) (b) (c) (d) (e)

Figure 2. The pipeline of the proposed gradient selection approach: the system starts by extracting an edge map of the input image using [13]; and it deploys a low level clustering to group these edge points into small clusters (namely *edgelets*); then an energy minimization framework is used to select from these edgelets the ones that are coherent with user interaction and semantically meaningful; last, a gradient mask is created by diffusing the selected edge points.

age regions, which can be effectively extracted by the detection method. We then employ a low-level clustering method to group edge points into small clusters, which we call *edgelets*, as shown in Figure 2c. These two steps are done once as preprocessing for each input image.

For online processing, given a new user scribble, we identify all edgelets that are covered by it, and build a graph by treating each cluster as a node, and defining affinities among them based on their geometric and appearance features. An efficient method is proposed to solve the graph labeling problem to provide the edgelets that belong to the semantically-dominant contour (Figure 2d). Finally, we diffuse the selected edge points to create the final gradient mask (Figure 2e).

The main reason we use edgelets instead of raw edge pixels for graph construction is that since each cluster represents a local edge segment, they are semantically meaningful, allowing us to embed high level semantical constraints into the graph. An additional benefit is the greatly reduced graph size which leads to high computational efficiency.

3.2. Pre-Clustering

We first identify all edge points using the detection method [13]. Features are extracted for each edge point, including its spatial location, gradient magnitude, gradient direction, and its appearance features (including color and texture cues [7]) of the local image patches around the point. For appearance feature we compute two means, one from the 3×3 patch along the gradient direction, whose center is 3 pixel away from the edge point, and the other from the patch on the opposite direction. Once features are extracted, the Mean Shift algorithm [8] is used to cluster the edge points into small clusters, denoted as $\{e_1, e_2, ..., e_n\}$. We call each cluster an *edgelet*, as it represents a small piece of coherent edge.

3.3. Energy Minimization for Edgelet Labeling

For extracting dominant gradients, we seek for a binary labeling function $L(i)$ for the ith edgelet e_i, where $L(i) = 1$ means that e_i belongs to the dominant gradient, and $L(i) = 0$ if e_i is not part of it. An energy minimization framework is adopted to represent the objectives of the edgelet labeling problem. Specifically, the total energy of a label assignment $E(L)$ is composed of a data energy E_d and a neighborhood energy E_r as:

$$E(L) = \sum_i E_d(i, L(i)) + \lambda \sum_{i,j} E_r(i, j, L(i), L(j)). \quad (1)$$

Data energy. For assigning the data energy $E_d(i, L(i))$, two factors are considered: 1) the strength of the edgelet, and 2) its consistency with the user scribble. The data energy thus can be decomposed into two terms:

$$E_d(i, L(i)) = E_d^s(i, L(i)) + \mu E_d^u(i, L(i)), \quad (2)$$

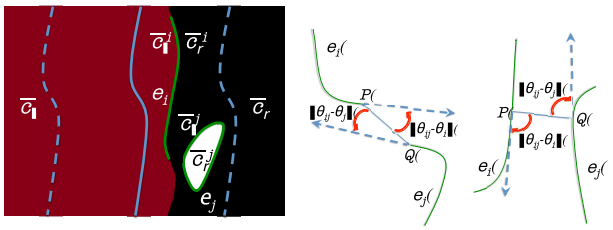
where $E_d^s(i, L(i))$ is the energy reflecting the strength of the edgelet:

$$E_d^s(i, L(i)) = \begin{cases} 1 -_\Delta p & : L(i) = 1 \\ {}_\Delta p & : L(i) = 0 \end{cases}, \quad (3)$$

where ${}_\Delta p$ is the average edge probability (gPb [13]) of all pixels in e_i:

$${}_\Delta p = \sum_{P \in e_i} P_b(P) / \|e_i\|. \quad (4)$$

$E_d^u(i, L(j))$ measures how consistent the edgelet is with the user-specified scribble. The idea behind this term is illustrated in Figure 3a. When the user overlays a scribble on a semantic contour, the two edges of the scribble usually have different appearances, denoted as $\bar{c}_l$ and $\bar{c}_r$, as they intersect with different objects. If the current edgelet belongs to the dominant contour inside the scribble, it is expected to separate the same two objects, thus the average appearance feature of local image patches on the two sides of the edgelet, denoted as $\bar{c}_l^i$ and $\bar{c}_r^i$, should be consistent with the $\bar{c}_l$ and $\bar{c}_r$. On the contrary, if $(\bar{c}_l, \bar{c}_r)$ is not consistent with $(\bar{c}_l^i, \bar{c}_r^i)$, then the edgelet probably belongs to the texture inside one of the objects, thus it should not be selected, even if it is a strong edge. Here, kernel descriptors [7] are used as appearance features.



(a) $_\Delta f$ in $E_d^u(i, L(i))$ (b) smaller $_\Delta\theta$ (c) larger $_\Delta\theta$

Figure 3. $_\Delta f$ and $_\Delta\theta$ in the energy terms. (a) comparing two edgelets e_i and e_j (the green curves) using appearance features; (b) and (c) illustration of $_\Delta\theta$ in defining $E_r^g(i, j, L(i), L(j))$.

We compute $(\bar{c}_l, \bar{c}_r)$ and $(\bar{c}_l^i, \bar{c}_r^i)$ for each edgelet from their local image patches, and define $E_d^u(i, L(j))$ as:

$$E_d^u(i, L(i)) = \begin{cases} 1 -_\Delta f & : L(i) = 1 \\ {}_\Delta f & : L(i) = 0 \end{cases}, \tag{5}$$

where

$$_\Delta f = 1 - (\|\bar{c}_l - \bar{c}_l^i\| + \|\bar{c}_r - \bar{c}_r^i\|)/2. \tag{6}$$

Neighborhood energy. In order to formulate the neighborhood energy $E_r(i, j, L(i), L(j))$, we first need to define the spatial distance between two edgelets. In this work it is defined as the closest distance between any points from these two edgelets, i.e., if $\exists P \in e_i, Q \in e_j, |PQ| < r$, then $d(i, j) < r$. If $d(i, j) > t_r$, a pre-defined threshold, then e_i and e_i are considered not in the same neighborhood ($E_r(i, j) = 0$). Otherwise the energy is defined as:

$$E_r(i, j, L(i), L(j)) = E_r^g(i, j, L(i), L(j)) + \nu E_r^c(i, j, L(i), L(j)), \tag{7}$$

where E_r^g measures the geometric compatibility of the two edgelets, and E_r^c measures the appearance compatibility between them. To define E_r^g, we first identify the pair of points $P \in e_i, Q \in e_j$ that has the minimal distance $d(i, j)$, and then measure the local curve direction at both points as θ_i and θ_j, as shown in Figure 3b and Figure 3c. We also compute the angle of the line PQ as θ_{ij}. E_r^g is then defined as:

$$E_r^g(i, j, L(i), L(j)) = \begin{cases} {}_\Delta\theta & if\ L(i)L(j) = 1 \\ -_\Delta\theta & if\ L(i) + L(j) = 1 \\ 0 & otherwise \end{cases}, \tag{8}$$

where

$$_\Delta\theta = \frac{|\theta_{ij} - \theta_i| + |\theta_{ij} - \theta_j|}{\pi/2} - 1. \tag{9}$$

The angle difference $|\theta_{ij} - \theta_i|$ is from 0 to $\frac{\pi}{2}$, thus $_\Delta\theta$ is normalized to $[-1, 1]$. The idea behind E_r^g is illustrated in Figure 3. If two edgelets belong to the same contour, then the angle differences will be small (Figure 3b), thus $_\Delta\theta$ is close to -1, and E_r^g encourages both edgelets to be selected. On the other hand, if they do not belong to the same contour, such as being parallel as shown in Figure 3c, then $_\Delta\theta$ is close to 1, and E_r^g penalizes selecting both of them and only allows one to be selected.

The appearance energy $E_r^c(i, j, L(i), L(j))$ penalizes the dissimilarity of the appearance features of two neighboring edgelets, if both are selected. It is defined as:

$$E_r^c(i, j, L(i), L(j)) = L(i)L(j)(1 -_\Delta f_{i,j}), \tag{10}$$

where $\Delta f_{i,j}$ measures the appearance difference between $(\bar{c}_l^i, \bar{c}_r^i)$ and $(\bar{c}_l^j, \bar{c}_r^j)$ by replacing $(\bar{c}_l, \bar{c}_r)$ with $(\bar{c}_l^j, \bar{c}_r^j)$ in Eq. 6.

3.4. Optimization

Despite that many optimization algorithms have been developed for energy minimization [20], the exact solution to the general MRF is NP-hard. Graph cuts is a commonly used approach, but it has strict requirements on the energy terms, such as the inequality $E(i, j, 0, 0) + E(i, j, 1, 1) \leq E(i, j, 0, 1) + E(i, j, 1, 0)$ should be satisfied for any pair of nodes [10]. Our energy function violates the requirements with the negative term in Equation 8. Other approximation algorithms exist, such as ICM [6], loopy belief propagation [6], etc. The solutions they provide however are intractable compared to exact optimization. The variable elimination [22] and the junction tree algorithm [6] are usually used for exact optimization of a graphical model. However they are computationally expensive, as both finding perfect elimination order and building a junction tree with minimal cluster size are NP-hard [4].

3.4.1 Our procedure.

Luckily, the graph structure of our model are characterized by an upper bound on the size of the maximal cliques of its triangulated graph. In our model, all edgelets are extracted from the image region under a user scribble which has limited bandwidth, and the nodes are connected based on the spatial distance between two edgelets. Our graph thus has a semi-chain shaped structure, as shown in Figure 4. (If the scribble forms a loop, the algorithm ignores the neighborhood energy between edgelets that is caused by scribble intersection.) Optimization of our model remains tractable, and we design an algorithm to find a variable elimination order and then use the variable elimination algorithm [22] for the exact optimal solution. The elimination order we find ensures that the variable elimination algorithm performs efficient (in linear time). This is critical for an image editing system that gives real-time feedback.

To better explain our algorithm, we introduce two definitions first. Defining a direction for a user scribble, for any two edge points P and Q in its brush mask, find the points

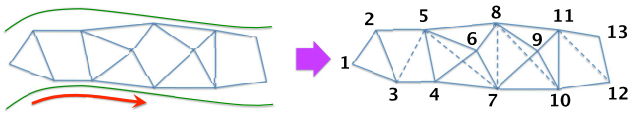


Figure 4. Illustration of the graph processing algorithm. Left: the original graph, where the region between the green curves is the scribble region, the red arrow represents the direction of the scribble. Right: the processed graph (triangulation and resultant elimination order).

that are closest to them on the scribble: P_s and Q_s. Following the direction of the stroke, if P_s is ahead of Q_s, we say that $P \prec Q$ (or $Q \succ P$).

Similarly, for any two nodes v_s and v_t in our graph (representing two edgelets), if $\forall P \in v_s, \forall Q \in v_t$, $P \prec Q$, we have $v_s \prec v_t$ (or $v_t \succ v_s$). $v_s \preceq v_t$ means that $v_s \succ v_t$ does not hold. This relationship is transmissive.

It can be proven that for any two vertices v_s and v_t in our graph, if there is no edge between them, $v_s \prec v_t$ or $v_t \prec v_s$; if there is an edge between them, $\exists P \in v_s, Q \in v_t$, $|PQ| < t_r$ (t_r is the threshold used in defining neighborhood energy E_r in Equation 7). We propose an algorithm that starts with a semi-chain shaped graph as shown in Figure 4 (left), triangulates it with edges whose lengths have an upper bound, and eliminates all vertices in an order that does not involves new edges, as shown in Figure 4 (right). The algorithm is formally defined as:

In the graph G' constructed in Algorithm 1, the distances of all node pairs are within $\sqrt{t_r^2 + b^2}$ where b is the width of the user scribble area. (proved in the supplementary material). The size of its largest clique in G', which means all edgelets are closer than $\sqrt{t_r^2 + b^2}$ to each other, is a limited number K because of the sparsity of detected edge points in an image. The elimination scheme used in Algorithm 1 guarantees that no additional edge is added (also proved in the supplementary material). After the graph is transformed, we use the variable elimination algorithm to solve the labeling problem. The computational cost of variable elimination is bounded by $O(2^K n)$ (K is the largest clique size of the graph, which is a limited small number in our problem; n is the number of edgelets).

3.5. Gradient Mask Generation

The output of the optimization procedure described above is a set of edgelets which together form the semantically-dominant contour in the scribbled area. Each edgelet is composed of edge points which are local maxima in the gradient map. To create the final gradient mask, we expand each edgelet based on the width of the local gradient. To estimate the local gradient width, starting from an edge point, we move along the gradient direction until the gradient magnitude is lower than a small threshold. We then move along the opposite direction to find the other end point, and compute the local gradient width as the distance between the two end points. The widths of nearby edge points are averaged to remove noise, and finally a mask is created by expanding the selected edgelets based on their widths.

Algorithm 1 Find variable elimination order of the semi chain-structured graph

input A undirected graph G
output An elimination order $\{v^1, v^2, ..., v^t\}$
  Construct a triangulated graph G', initiate $G' \leftarrow G$
  for any k-circle ($k > 3$) in G': $V = \{v_1, v_2, ..., v_t\}$ **do**
    find a node v_i, $\forall_{j \in \{1,2,..t\}, j \neq i \pm 1} v_i \prec v_j$; compare v_{i-1} and v_{i+1} (if $j > t$, $v_j = v_{j-t}$):
    if $v_{i-1} \prec v_{i+1}$ **then**
      consider $v_{i-2}, v_{i-1}, v_i, v_{i+1}$, we have $v_i \prec v_{i-2}$
      find $P_0 \in v_i$, $P_1' \in v_{i+1}$ that has $|P_0 P_1'| < t_r$
      find $P_1 \in v_{i-1}$, $P_2 \in v_{i-2}$ that has $|P_1 P_2| < t_r$
      if $|P_0 P_2| < |P_1 P_1'|$ **then** add an edge $v_i v_{i-2}$ to G'
      else add an edge $v_{i-1} v_{i+1}$ to G'
    else consider $v_{i-1}, v_i, v_{i+1}, v_{i+2}$, similar as when $v_{i-1} \prec v_{i+1}$
  end for
  Eliminate all vertices in V
  for $G' \neq \emptyset$ **do**
    find all $v_i \in V$, $\forall v_j \in V, i \neq j$, $v_i \preceq v_j$; select from them v_k with the least neighbors
    eliminate v_k, and remove it and all its edges in G'
  end for

4. Experiments

We evaluate the edge selection accuracy of our approach on a benchmark based on the Berkeley segmentation dataset [14]. The experiments suggest that, for the problem of supervised edge selection, our method outperforms the state-of-art algorithms used for edge detection and contour extraction.

4.1. Benchmark

Our benchmark dataset contains 100 images that were randomly chosen from the Berkeley segmentation dataset [14]. For each image, we manually drew a scribble over a dominant edge of the object, which overlays with the corresponding edge marked by human in the segmentation dataset. The human marked edges are treated as ground-truth. The proposed algorithm contains a few important parameters. Particularly, in Equation 1, 2 and 7, there are three weighting parameters λ, μ and ν which balance between different energy terms. We used Gaussian processes regression [18] to learn the best parameter setting. (Assume that the three weighting parameters (variables x), and the average edge detection accuracy on the training data (the fidelity value $f(x)$), are drawn from a GP distribution. By estimat-

Figure 5. Edge selection results compared with gPb thresholded by different values. The first image in each row is the original image with an overlay of user scribble; the second image is the edge probability map extracted by gPb; the third and fourth images are the results of directly using the output of gPb within the mask region, using different thresholds; the fifth image is the output of our method; and the last one is the groundtruth. While using gPb directly, the result may either include noises that are not coherent with the overall line, or leave out edge points that cause discontinuity of the line. Our results overcome the problem by formalizing the coherent relationship of the edgelets.

Table 1. Comparisons of edge extraction accuracy on the dataset.

Methods	gPb	Active contour	Graph-cut	Ours
F-measure	0.69	0.67	0.63	0.75

ing and sampling the GP model, the parameter setting with the highest responses is selected as our system parameter setting.) The parameters we found are: $\lambda = 1.02$, $\mu = 0.78$, $\nu = 0.23$.

4.2. Evaluation

We conducted an objective evaluation of the proposed approach using the dataset. Our results are compared against the active contour algorithm [9] and the Berkeley edge detection results (gPb) [13] by thresholding the edge points by their probability. We also compare against using the graph cuts algorithm to minimize our energy function (discarding the energy term that violates the triangular constraint for graph cuts).

Table 1 and Figure 5 shows the results. F-measure [19]

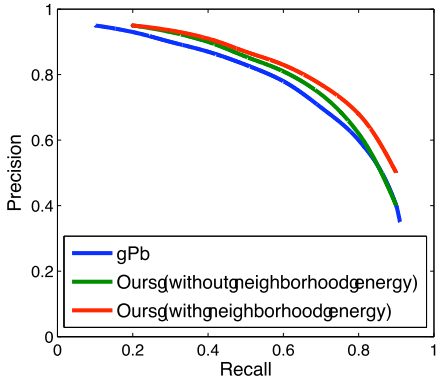


Figure 7. Precision recall curve

(the weighted harmonic mean of precision P and recall R) is used to evaluate the results. Table 1 clearly suggests that the proposed approach outperforms previous methods in terms of F-measure. Specifically, since the geometric compatibility term in our energy function is critical in find-

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6. Edge snapping results. (a) input image; (b) user scribble overlayed; (c) original edge map; (d) and (e) are selected edges by thresholding gPb using different values; (f) selected edgelets by our approach; (g) our edge snapping results (red is the initial curve, and green is the output); (h) edge snapping results using active contour.

ing non-conflicting edgelets, discarding this term results in a poor behavior using the graph cuts algorithm. The active contour has the problem of easily being stuck at local minima. gPb has the best results of all the three methods. However, it focuses only on the pixel level edge detection task. Relationship between the user scribble and edge points, as well as relationship among the selected edge points, is not well considered. Our approach significantly outperforms the result of thresholding gPb outputs, no matter what threshold is used (as in Figure 5). (Note that the score of gPb here is different from the score over the whole BSD dataset ([14],[13]). This is because the benchmark we created here evaluates edge detection results only in the masked regions of the images.)

Figure 7 shows the precision-recall curve of both our algorithm and gPb. We vary our algorithm in two ways: considering only data energy terms ($\lambda = 0$); and considering both data energy and neighborhood energy. When neighborhood energy is not taken into consideration, our algorithm behaves more close to thresholding gPb, especially when the recall is high, (meaning more edges are selected and constraints by user scribble is less weighted). The best result is achieved when all our energy terms are used. The neighborhood energy ensures that selected edge points are compatible with each other, thus have a better chance of laying on the same semantic contour.

It is also worth mentioning that our algorithm is very efficient, because optimization of the energy function can be done in linear time. It takes on average $2 \sim 3$ seconds to select gradients for each user scribble. In our system, the time between user interaction and results rendering is within 5 seconds, which differs a little for different applications.

4.3. Applications

Our algorithm can be used in various gradient-domain image editing tasks, including edge snapping, salient sharpening, non-photorealistic rendering, etc. Here we demonstrate the results on some them.

Edge snapping. Edge snapping tools, such as the classic magnetic lasso tool in photoshop, are popular choices for interactive object selection from images. However traditional methods seek for local minima of their energy functions, thus can be easily distracted by strong edges nearby the targeted contour. Our gradient selection method can be used the improve the performance of edge snapping, but forcing the final curve to snap to the selected gradients. Since our gradient selection procedure only picks out the dominant contour, the snapping result is more likely to be globally optimal, as shown in the examples in Figure 6.

Saliency sharpening. Sharpening is a natural application for gradient-domain editing, which is done globally in previous approaches [5]. Our system allows the user to mark specific contours to be sharpened, as in Figure 8. Systems like [5] can not perform such tasks without our gradient selection component.

Image stylization. The goal of stylizing an image is to abstract away the non-salient details in the image and only emphasize on salient contours. In gradient domain, this is typically done by suppressing small gradient values while magnifying large gradient values [5]. We provide a rendering interface that allows the user to control which gradient to magnify or suppress, thus provides a greater degree of freedom for artistic expression. Some examples are shown in Figure 9.

Figure 8. Saliency sharpening. For each example we show two sets of user scribbles and their corresponding results.

Figure 9. Stylization results. In each example, from left to right: input image; user scribbles; stylization result without gradient selection; our result (with gradient selection).

5. Conclusion

We propose a graph-based energy minimization approach for selecting semantically-dominant gradients in user-scribbled area. The graph is constructed by using low-level edgelets as nodes, and defining their affinities using both geometric and appearance features. The way we define them well considers various aspects of semantical dominancy of image edges, as well as the information from interactive user input. Furthermore, we show how to effectively transform the graph and find its precise optimal solution, thus the geometric and appearance features of selected gradients are ensured. The effectiveness of the proposed approach is objectively evaluated and demonstrated through various image editing applications. As future work we plan to explore its applications in other areas, such as video editing, making convenient annotation tool, etc.

References

[1] A. Agrawal and R. Raskar. Gradient domain manipulation techniques in vision and graphics. *ICCV course*, 2007.

[2] A. Agrawal, R. Raskar, and R. Chellappa. Edge suppression by gradient field transformation using cross- projection tensors. *CVPR*, pages 2301–2308, 2006.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans Pattern Anal Mach Intel*, 2011.

[4] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. on Algebraic and Discrete Methods*, 8:277–284, 1987.

[5] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM TOG*, 28, 2009.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. *NIPS*, 2010.

[8] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans Pattern Anal Mach Intell*, 24(5):603–619, 2002.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[10] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans Pattern Anal Mach Intell*, 26:147–159, 2004.

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

[12] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. *SIGGRAPH*, 25(3), 2006.

[13] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. *CVPR*, 2008.

[14] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proc. 8th Int'l Conf. Computer Vision*, 2:416–423, July 2001.

[15] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. *SIGGRAPH*, pages 191–198, 1995.

[16] A. Orzan, A. Bousseau, P. Barla, and J. Thollot. Structure-preserving manipulation of photographs. *NPAR*, 2007.

[17] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. *SIGGRAPH*, 2003.

[18] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

[19] C. V. Rijsbergen. Dept. of Computer Science, Univ. of Glasgow, 1979.

[20] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. *ECCV*, 2:16–29, 2006.

[21] Y. Zeng, W. Chen, and Q. Peng. A novel variational image model: Towards a unified approach to image editing. *Journal of Computer Science and Technology*, 2006.

[22] N. L. Zhang and D. Poole. A simple approach to bayesian network computations. *In Proc. of the 10th Canadian Conference on Artificial Intelligence*, pages 171–178, 1994.

[23] K. Zhanga, L. Zhanga, H. Songb, and W. Zhou. Active contours with selective local or global segmentation: a new formulation and level set method. *Image and Vision Computing*, 2010.