Interactive Contour Extraction via Sketch-Alike Dense-Validation Optimization

Yongwei Nie, *Member, IEEE*, Xu Cao, Ping Li, Qing Zhang, Zhensong Zhang, Guiqing Li, and Hanqiu Sun

***Abstract*— We propose an interactive contour extraction method inspired by a skill often adopted in sketching: an artist usually sketches an object by first drawing lots of short, directional, and redundant strokes, then following these small strokes to draw the final outline of the object. Our method simulates this process. To extract a contour, our method relies on user interaction, which provides us with a narrow band containing the target contour. Then, we densely sample sub-bands from the whole band, with each sub-band containing a local segment of the target contour. We design a curve-centered coordinate system in which a dynamic programming algorithm is proposed to extract the local segment in each sub-band. The local segment is guaranteed to be as evident and smooth as possible, to mimic the strokes sketched by the artist. Finally, we integrate all local segments of all sub-bands together to obtain the whole target contour based on the weighted principal component analysis. Our method can extract high-quality object contours due to the dense validations among local segments. That is, even if one segment deviates from the right location, several other segments in its local neighborhood can correct it in the integration stage. Both quantitative experiments and a user study demonstrate the effectiveness of the proposed method.**

***Index Terms*— Sketching, image contour extraction, image segmentation, dynamic programming.**

I. Introduction

IMAGE edge/contour detection/extraction is one of the most fundamental topics in digital image processing communities, which has obtained a large amount of research during past decades [1]. Most of existing approaches focus on automatic edge/contour detection [1]–[13]. The ideas and strategies behind these automatic approaches have evolved much, e.g., from local filters to regression classifiers, from manually designed features to automatically learned features, from statistical learning to deep learning. However, the results of automatically detected edges/contours still have problems sometimes. For example, some important contours may be omitted or edges of unimportant objects may be preserved.

Another research line is to design methods that extract object contours assisted by users. For example, Kass *et al.* [14] have proposed the method of Active Contour Models, which iteratively moves an initial curve indicated by the user until it finally matches with the target object boundary. Level sets methods [15] work in a similar way which also require the user to provide an initial solution. Martin *et al.* [16] have developed a Java application called "segapp" with which the BSDS300 dataset was annotated manually. Later, the BSDS300 dataset has evolved to BSDS500 [7], by adding 200 more manually annotated images. Similarly, Russell *et al.* [17] have proposed an online labeling system called LabelMe, which allows all the users around the world to upload images and their annotations. In Computer Graphics community, interactive segmentation approaches have been proposed. For example, graph cuts based approaches [18], [19] demand the user to indicate some foreground and background pixels with which the boundary between foreground and background regions can then be automatically determined. In commercial software, such as Photoshop (https://www.photoshop.com/), AFFINITY (https://affinity.serif.com/), etc., interactive tools are usually provided to extract objects and their contours.

In this paper, we propose a novel interactive contour extraction method. All the approaches mentioned above (except the two commercial products with undisclosed technologies) try to find an object contour as a whole, which however is a process that is easy to introduce errors. Different from them, we propose to extract segments of the target contour first, and then integrate all the segments together to form the whole target contour. This idea is inspired by a skill often adopted in sketching. Fig. 1 shows such an example. An artist usually uses many short, directional, and overlapping strokes to sketch an object boundary (Fig. 1 left). Then, the accurate outline of the object can be easily traced out of all of the redundant strokes (Fig. 1 right). The reason behind this behavior is that there

Manuscript received July 5, 2018; revised October 24, 2018, January 6, 2019, and January 31, 2019; accepted February 1, 2019. Date of publication February 11, 2019; date of current version April 3, 2020. This work was supported in part by NSFC under Grant 61602183, Grant 61572202, and Grant 61802453, in part by the Science and Technology Project of Guangzhou City under Grant 201707010140, and in part by the Macau Science and Technology Development Fund under Grant 0027/2018/A1. This paper was recommended by Associate Editor W. Li. *(Corresponding author: Guiqing Li.)*

Y. Nie, X. Cao, and G. Li are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: nieyongwei@scut.edu.cn; xucao93@gmail.com; ligq@scut.edu.cn).

P. Li is with the Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China (e-mail: pli@must.edu.mo).

Q. Zhang is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China (e-mail: zhangq93@mail.sysu.edu.cn).

Z. Zhang is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: zszhang@cse.cuhk.edu.hk).

H. Sun is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: sunnyqiu24@gmail.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2019.2898691

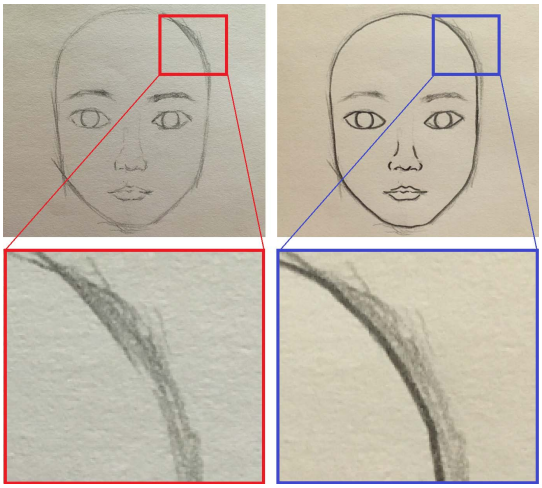


Fig. 1. Artist sketching example. Left: short and overlapping strokes are drawn first. Right: a contour is traced out of all of the local strokes.

exist dense cross validations among adjacent strokes. Even if a stroke contains error (maybe a wrong direction), many strokes at the same location can correct the error. This makes the sketching process more controllable and accurate.

Based on this observation, we propose a novel sketch-alike contour extraction method. Our method relies on user interaction, which provides us with a narrow band containing the target contour. Then our method extracts the target contour from the band in three steps. Firstly, we propose a multi-scale sub-band sampling algorithm to densely sample many sub-bands from the whole band, with each sub-band containing a segment of the target contour. Secondly, we propose a dynamic programming based algorithm to extract the local segment from each sub-band. Although the accuracy of each local segment cannot be guaranteed, many segments at the same location can together provide a more reliable estimate. Based on this, we expand wPCA in its first usage in the content of the paper to integrate all local segments of all sub-bands together to form the final target contour. Quantitative experiments on the BSDS500 dataset [7] and a user study demonstrate the effectiveness of the proposed method.

In summary, the contributions of this paper are:

- We propose a novel sketch-alike contour extraction method that densely extracts local and overlapping segments of a contour first and then integrate all the segments together to form the desired contour, which takes advantage of the redundancies of the local segments to enhance the correctness of the final contour.
- We propose a dynamic programming based local segment extraction method in a discretized curve-centered coordinate system.
- We propose a wPCA based global contour integration algorithm that can extract the final contour from all the local segments accurately.

The rest of this paper is organized as follows. In Sec. II, we review the related works of image edge/contour extraction. Sec. III describes the proposed method in detail. Sec. IV presents the experiments and comparisons. The paper is concluded in Sec. V.

II. Related Work

A. Interactive Contour Extraction

Previous interactive contour extraction approaches include the popularly used Active Contour Models [14], [20]–[22] and level sets methods [15], [23]. They require the user to indicate an initial curve, and then iteratively move the curve to its target place by optimizing an energy function [14] or by updating an implicit surface [15]. The Active Contour Models and level sets methods are often applied to domain-specific images, such as medical images [24] and sidescan sonar images [25], however they have not been found to be effective for natural images [26]. Martin *et al.* [16] have developed a Java application called "segapp" to segment images manually, based on which the dataset BSDS300 was built and also the BSDS500 dataset [7]. Russell *et al.* [17] have proposed an online labeling system called LabelMe, allowing users all around the world to upload images and their annotations. LabelMe only provides simple polygon and mask tools for users. In the computer graphics community, several interactive methods have been developed in the course of time. For example, the method proposed in [18] can segment foreground and background regions apart if some foreground and background pixels are indicated beforehand. With GrabCut [19], the user just needs to provide a rectangle enclosing the target object for GrabCut to cut the object out. Besides, nearly all commercial products such as Photoshop (https://www.photoshop.com/), AFFINITY (https://affinity.serif.com/), Sketch (https://www.sketchapp.com/), and GIMP (https://www.gimp.org/) provide tools that assist users to select objects and their contours in images.

B. Automatic Edge/Contour Detection

Approaches proposed in the early ages, such as Duda and Hart [2], Canny [3], Roberts [27], and Prewitt [28], use filtering and thresholding strategies to detect pixels with highest gradients in their local neighborhood. These works have been improved over time using three different strategies. First, sophisticated filters of different scales and orientations have been used [4], [5], [29], [30] to obtain richer descriptions of edges. Second, the feature vectors have been extended using colors and textures, in addition to brightness and gradients [6], [7]. In these approaches, low-dimensional features (Pb [6] and gPb [7]) are manually designed, then regression classifiers are trained to predict the probability of whether a pixel is located on an edge. Third, supervised learning approaches have been proposed [9]–[11], that learn boosting tree or random forests from thousands of simple features extracted from image patches.

Besides identifying individual edge pixels, many approaches connect the edge pixels together to obtain continuous contours [12], [31]–[35]. For example, high-gradient edge fragments have been directly linked together in [31] and [32]. Ren *et al.* [12] have constructed constrained Delaunay triangulation (CDT) based on local contours, then built Conditional Random Fields (CRF) on the CDT, and finally used the CRF to infer longer contours. Zhu *et al.* [34] have constructed a graph on local contours, and found long contours by

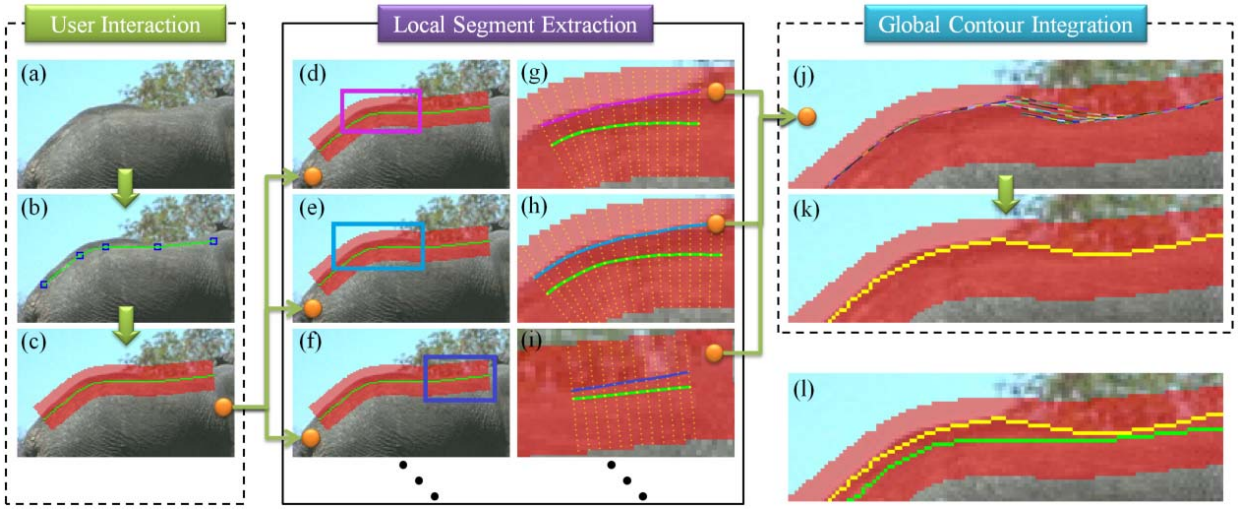


Fig. 2. Overview. (a) Input image. (b) The user just needs to indicate several key points (the blue squares) along the contour to be extracted. A Catmull-Rom spline (the green curve) is fitted from the user indicated key points. (c) The spline indicates a narrow band (the red one) which contains the target contour. (d)(e)(f) Sub-bands (contained in the squares) are densely sampled from the whole band. (g)(h)(i) Local contour segments of the sub-bands are extracted individually by a dynamic programming based algorithm. (j) All extracted local segments are shown together. (k) The target contour is extracted from all local segments by a global contour integration algorithm. (l) Both the spline and the extracted contour are shown.

computing eigenvectors of normalized random walk matrix. Arbeláez *et al.* [35] have aligned and combined segmentation results at different scales, and used the contours at coarser levels to depress small contour fragments at finer levels.

Recently, deep learning frameworks have been used to solve the problem of image edge/contour detection [1], [13], [36]–[39], achieving impressive results. N^4-fields [37] combines convolutional neural networks (CNN) with nearest neighbor search. DeepContour [38] partitions positive contour data into subclasses and fits each subclass by different model parameters. Multi-scale schemes which combine both low- and high-levels of information have been used [1], [13], [39]. The difference is that Bertasius *et al.* [39] have used multiple CNNs to process patches at different scales but centered at the same pixel, and then combined the outputs of the multiple CNNs together. Differently, fully convolutional networks (FCN) [40] has been used in [1] and [13] to achieve end-to-end contour detection.

C. Image Segmentation

Image contour extraction is highly related to image segmentation, as one can always recover closed contours from segmented regions. There is a broad family of image segmentation approaches which can be roughly divided into two categories: user interaction based methods, and fully automatic ones. The graph-based approaches, such as graph cuts [18], GrabCut [19], usually require users to draw some pixels of foreground and background objects, then they can segment the foreground and background regions. In contrast, clustering based methods, such as spectral clustering [41], normalized cuts [42], mean-shift clustering [43], can determine the number of objects in an image automatically. Another solution is to divide an image into superpixels [44], and then combine superpixels with homogeneous colors together [45]. Deep learning has also been adapted to perform image segmentation tasks nowadays [40], [46].

III. Our Method

Fig. 2 gives an overview of our method. Fig. 2 (a) is a source image where the contour of the back of the elephant is to be extracted. In our method, the user just needs to indicate several key points (the blue squares in Fig. 2 (b)) along the contour to be extracted. Our method then fits a Catmull-Rom spline, i.e., the green curve in Fig. 2 (b), from the user indicated points, which we call the user-indicated initial curve. Here, we choose to use the Catmull-Rom spline because it passes all the key points exactly. We then construct a narrow band (i.e., the red band in Fig. 2 (c)) with the initial spline as its medial axis, and assume the target contour is within the narrow band. Here, half of the width of the band is $r = 10$, where r is an important parameter of our method which has been proved from our analysis given in Section IV-B. Then we divide the band into many sub-bands. Fig. 2 (d), (e) and (f) show three examples of the sub-bands, contained in the squares. Each sub-band contains a segment of the target contour. We extract the local segment from each sub-band, as shown in Fig. 2 (g), (h) and (i), where the green curves are splines, while curves in other colors are extracted segments of the target contour. Fig. 2 (j) shows all the extracted local contour segments of all sub-bands together. Given Fig. 2 (j), we apply a wPCA based global contour integration algorithm to obtain the result of Fig. 2 (k), which is the final contour extracted by our method. Fig. 2 (l) shows both the initial spline and the final contour extracted.

In the following, we first introduce a curve-centered coordinate system. All algorithms of this paper are designed in this coordinate system. We then introduce our multi-scale sub-band sampling algorithm to sample sub-bands densely. After that, a dynamic programming based sub-band local segment

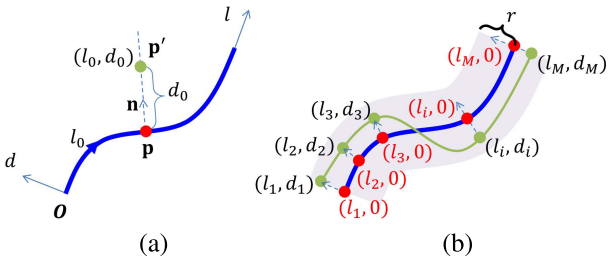


Fig. 3. (a) Illustration of the curve-centered l-d coordinate system. (b) The l-axis is discretized.

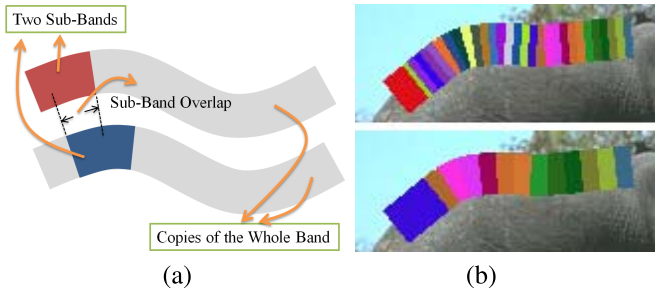


Fig. 4. (a) Illustration of the whole band and two sub-bands, and the overlap between the two sub-bands. (b) Top: all sub-bands of length 24. Bottom: all sub-bands of length 51.

extraction algorithm is presented. Finally, our global contour integration method is described.

A. Curve-Centered Coordinate System

We design a curve-centered coordinate system, or called l-d coordinate system. As shown in Fig. 3 (a), the blue curve indicates an initial Catmull-Rom spline. We define the origin $\mathbf{O}$ of l-d system as the beginning of the spline. The l-axis is along the spline, and the d-axis is always perpendicular to the spline. Denoting by $\mathbf{p}$ a point on the spline, and l_p be the distance from $\mathbf{p}$ to $\mathbf{O}$ along the spline, the coordinate of $\mathbf{p}$ in the l-d system is $(l_p, 0)$. Let $\mathbf{n}$ be the normal direction (in the image x-y coordinate system) of the spline at point $\mathbf{p}$, and $d_{pp'}$ be the distance from $\mathbf{p}$ to $\mathbf{p}'$ along the normal direction, then the coordinate of $\mathbf{p}'$ in the l-d coordinate system is $(l_p, d_{pp'})$. Let the coordinate of $\mathbf{p}$ in the x-y coordinate system be (x_p, y_p), then the coordinate of $\mathbf{p}'$ in the x-y coordinate system can be computed by transformation $\mathcal{T}$:

$$(x_{p'}, y_{p'}) = \mathcal{T}(l_p, d_{pp'}) = (x_p, y_p) + d_{pp'} \cdot \mathbf{n}, \tag{1}$$

where $(x_p, y_p) = \mathcal{T}(l_p, 0)$.

As shown in Fig 3 (b), we discretize the l-axis of the l-d coordinate system by evenly sampling M points on the spline: $\{(l_i, 0)|i \in \{1, 2, \cdots, M\}\}$, with $l_i = \Delta \cdot (i-1)$ where $\Delta = \frac{L}{M-1}$ is the sampling gap along the l-axis, and L is the total length of the spline. Then, the target contour can be discretely represented as a set of points:

$$\mathcal{S} = \{(l_i, d_i)|i \in \{1, 2, \cdots, M\}\}. \tag{2}$$

This representation of the target contour is much simpler than its form in the image x-y coordinate system, as l_i is known for any i and the unknowns are only the distances d_i of the points on the target contour to the spline. In other words, we reduce the variable space from 2D to 1D, i.e., from $\{(x_i, y_i)\}$ to $\{d_i\}$, by using of the curve-centered coordinate system. Note that d_i should be in the range of $[-r, r]$.

We use a two-pass sampling strategy to sample M nearly equidistant points of a Catmull-Rom spline. In the first pass, we sample 1000 points (which is dense enough) on each spline segment. This sampling is straightforward given four control points of a spline segment and 1000 floating numbers evenly distributed between 0 and 1. But note that the sampled points in this pass may be spaced at different distances. In the second pass, we choose from all the points sampled in the first pass M nearly equidistant points.

B. Multi-Scale Sub-Band Sampling

As shown in Fig. 2 (c), the user-indicated Catmull-Rom spline provides us with a narrow band. In this section, we densely sample sub-bands from the whole band. For example, Fig. 4 (a) illustrates a whole band and two sub-bands of it, and also illustrates the overlap of the two sub-bands.

We follow two principles to sample sub-bands. Firstly, the sub-bands should be densely sampled, as shown in Fig. 4 (b), covering the target contour redundantly. Secondly, the sampled sub-bands should be short. Otherwise, the longer the sub-band, the less reliable the segment extracted from the sub-band.

Based on the above thoughts, we propose a multi-scale sub-band sampling algorithm. The "multi-scale" here means we extract sub-bands of different lengths. Recall that we have evenly sampled M points on the initial spline. Let s and t be two natural numbers in $[1, M]$, and $s < t$. Then any pair of (s, t) determines a sub-band, and we use $t - s + 1$ to denote the length of the sub-band. Empirically, we sample sub-bands of length 24, 30, 39, 51, and 66. For each length, we evenly sample overlapping sub-bands from the whole band, and adjacent sub-bands have a $2/3$ percentage overlap relative to their length. Fig. 4 (b) shows evenly sampled sub-bands of length 24 and 51, respectively. The sub-bands of length 24 begin at $s = 1, 9, 17, \cdots$, and accordingly $t = 24, 32, 40, \cdots$. The sub-bands of length 51 begin at $s = 1, 18, 35, \cdots$, and accordingly $t = 51, 68, 85, \cdots$.

C. Sub-Band Local Contour Segment Extraction

Now we introduce how to extract the local segment of the target contour in a sub-band. Recall in Sec. III-B, we have used s and t to determine a sub-band. The local segment in the sub-band is also related to s and t, and is denoted as $\mathcal{S}_{st}$ (see the definition of $\mathcal{S}$ in Eq. 2):

$$\mathcal{S}_{st} = \{(l_i, d_i)|i \in [s, t], s \in [1, M], t \in [1, M], s < t\}. \tag{3}$$

The task is to determine d_i for each l_i. We first solve this problem by a continuous optimization, and then reformulate it as a dynamic programming problem.

To mimic strokes sketched by artists, the extracted local segment should be as evident and smooth as possible. Specifically, "evident" means the segement should have larger gradients. Our energy function is thus composed of two terms defined

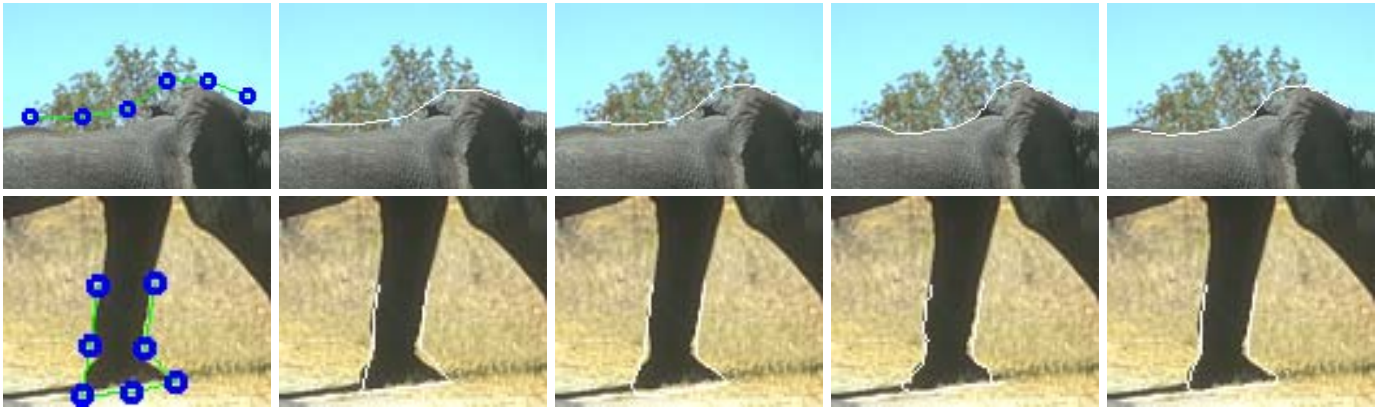


Fig. 5. (a) User edited initial curves. (b) Results by solving Eq. 4 with $\beta = 0$. (c) Results by solving Eq. 4 with $\beta = 3$. (d) Dynamic programming based segment extraction with $\beta = 0$. (e) Dynamic programming based segment extraction with $\beta = 3$.

on $\{d_i | i \in \{s, s+1, \cdots, t-1, t\}\}$:

$$\mathbf{E}(\{d_i\}) = \mathbf{E}_e(\{d_i\}) + \beta \cdot \mathbf{E}_s(\{d_i\}), \tag{4}$$

where $\mathbf{E}_e$ is evident term, and $\mathbf{E}_s$ is smooth term. β is a weight to balance the two terms. We define $\mathbf{E}_e$ as:

$$\mathbf{E}_e(\{d_i\}) = \sum_{i=s}^{t} gPb(\mathcal{T}(l_i, d_i), \mathbf{n}_i^{\perp}), \tag{5}$$

where $\mathbf{n}_i^{\perp}$ is the tangent direction perpendicular to the normal direction $\mathbf{n}_i$, and gPb is a operator from [7], which computes the gradient for pixel $\mathcal{T}(l_i, d_i)$ at orientation $\mathbf{n}_i^{\perp}$. We use gPb because it gives the gradients along the direction of the initial spline, which can better convey the user' intention than operators such as Sobel, Canny, etc. $\mathbf{E}_s(\{d_i\})$ is simple, which is the sum of the curvatures of all the points:

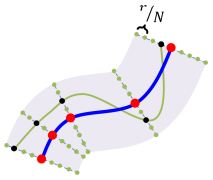
$$\mathbf{E}_s(\{d_i\}) = \sum_{i=s}^{t} curvature(\mathcal{T}(l_i, d_i)). \tag{6}$$

Finally, the variables should satisfy the following constraint:

$$\forall i, \quad -r \le d_i \le r. \tag{7}$$

The maximization of Eq. 4 is a constrained optimization problem, which can be solved by the function 'fmincon' of MATLAB® with 'Trust-Region-Reflective' solver. In default, we set $\beta = 3$, $\Delta = 2$ and $r = 10$. Fig. 5 (a), (b), and (c) show results of the above continuous optimization, where (a) shows user interactions. In (b), we set β as 0 to ignore the smoothness term. In (c), we set β as 3. It can be seen that contours in (c) are smoother than those in (b), which demonstrate the necessity of the smoothness term. However, the curves in (c) do not exactly match with the target object boundaries, as sub-optimal solutions are usually obtained when optimizing Eq. 4.

Reformulation by Dynamic Programming: It is not easy to obtain the global optimal solution when maximizing Eq. 4. Inspired by the shape context and chamfer matching works of Thayananthan *et al.* [47] and Thanh Nguyen [48], we reformulate the problem and solve it using a dynamic programming algorithm. Instead of viewing d_i as a continuous variable, we further discretize the d-axis by distance of $\frac{r}{N}$ to obtain $2N+1$ discrete coordinates: $\left\{d_{i,j} = \frac{r}{N} \cdot j | j \in \{-N, \cdots, 0, \cdots, N\}\right\}$, as shown below.



In this way, we obtain a regular lattice composed of $(t-s+1) \times (2N+1)$ points: $\left\{(l_i, d_{i,j}) | i \in \{s, \cdots, t-1, t\}, j \in \{-N, \cdots, 0, \cdots, N\}\right\}$. We demand our extracted segment to pass those discrete points. This dramatically reduces the solution space. More importantly, it allows us to find the segment by an efficient dynamic programming algorithm as following.

We construct a trellis graph $\mathcal{G}$ based on these regular points, as shown in Fig. 6. Each point $(l_i, d_{i,j})$ corresponds to a node $n_{i,j}$ of $\mathcal{G}$, and pairs of nodes $n_{i-1,j'}$ and $n_{i,j}$ are connected

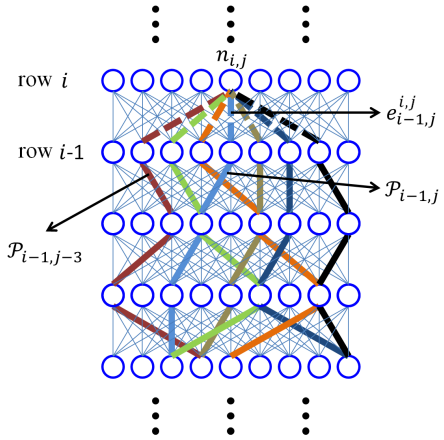


Fig. 6. Trellis graph for dynamic programming based local segment extraction algorithm. The blue circles are nodes of the trellis graph. Nodes of adjacent rows are linked by an edge if their difference in column is no more than $cmax$ (here $cmax = 3$). The solid lines are the best paths ending at nodes of row $i-1$. The dash edges together with the corresponding solid lines connected to them are candidates to be chosen as the best path ending at node $n_{i,j}$. We use $\mathcal{P}_{i,j}$ to denote the best path ending at node $n_{i,j}$, and $e_{i',j'}^{i,j}$ to denote an edge from node $n_{i',j'}$ to $n_{i,j}$.

by an edge $e^{i,j}_{i-1,j'}$ if $|j - j'| \leq cmax$ ($cmax$ is set as 3 in default). Let nodes of $\mathcal{G}$ encode evidence, and edges of $\mathcal{G}$ encode smoothness, the continuous optimization problem of Eq. 4 can be approximately re-formulated as finding the best path through the trellis graph.

We find the best path by dynamic programming. Firstly, we assume the best paths from row 1 to row $i-1$ have been found for all the nodes in row $i-1$, and let $\mathcal{P}_{i-1,j}$ denote the best path ending at node $n_{i-1,j}$. As shown in Fig. 6, for any node $n_{i,j}$ in row i, its best path must be one of: $\mathcal{P}_{i-1,j-3} \oplus e^{i,j}_{i-1,j-3}$, $\mathcal{P}_{i-1,j-2} \oplus e^{i,j}_{i-1,j-2}$, $\mathcal{P}_{i-1,j-1} \oplus e^{i,j}_{i-1,j-1}$, $\mathcal{P}_{i-1,j} \oplus e^{i,j}_{i-1,j}$, $\mathcal{P}_{i-1,j+1} \oplus e^{i,j}_{i-1,j+1}$, $\mathcal{P}_{i-1,j+2} \oplus e^{i,j}_{i-1,j+2}$, and $\mathcal{P}_{i-1,j+3} \oplus e^{i,j}_{i-1,j+3}$, where the symbol $\oplus$ connect the path and edge input to it. Let $\mathcal{E}(\mathcal{P})$ be the energy of path $\mathcal{P}$, and $\delta \in \{-cmax, \cdots, -2, -1, 0, 1, 2, \cdots, cmax\}$, we define the energy of path $\mathcal{P}^{\delta}_{i,j}$ (which is $\mathcal{P}_{i-1,j+\delta} \oplus e^{i,j}_{i-1,j+\delta}$) as:

$$\mathcal{E}(\mathcal{P}^{\delta}_{i,j}) = \mathcal{E}(\mathcal{P}_{i-1,j+\delta}) + E_e(i, j, \delta) + \beta \cdot E_s(i, j, \delta), \quad (8)$$

where $E_e(i, j, \delta)$ is the increment of evidence by adding point $(l_i, d_{i,j})$ to path $\mathcal{P}_{i-1,j+\delta}$, while $E_s(i, j, \delta)$ is the smoothness increment by adding edge $e^{i,j}_{i-1,j+\delta}$. Let:

$$\mathbf{u}(e^{i,j}_{i-1,j+\delta}) = \frac{\mathcal{T}(l_i, d_{i,j}) - \mathcal{T}(l_{i-1}, d_{i-1,j+\delta})}{||\mathcal{T}(l_i, d_{i,j}) - \mathcal{T}(l_{i-1}, d_{i-1,j+\delta})||} \quad (9)$$

be the unit vector along edge $e^{i,j}_{i-1,j+\delta}$. Then $E_e(i, j, \delta)$ is defined as:

$$E_e(i, j, \delta) = gPb(\mathcal{T}(l_i, d_{i,j}), \mathbf{u}(e^{i,j}_{i-1,j+\delta})), \quad (10)$$

i.e., the evidence increment is the gradient of point $\mathcal{T}(l_i, d_{i,j})$ along the direction $\mathbf{u}(e^{i,j}_{i-1,j+\delta})$. We define the smoothness increment $E_s(i, j, \delta)$ as:

$$E_s(i, j, \delta) = \mathbf{u}(e^{i,j}_{i-1,j+\delta}) \cdot \mathbf{u}(e^{i-1,j+\delta}_{i-2,\hat{j}}), \quad (11)$$

where $\hat{j}$ is the index that makes $n_{i-2,\hat{j}}$ be the second-to-last node on path $\mathcal{P}_{i-1,j+\delta}$. That means the smoothness increment measures the similarity of the directions of the last two edges before node $n_{i,j}$. With the well-defined Eq. 8, we can find:

$$\delta^* = \arg\max_{\delta} \mathcal{E}(\mathcal{P}^{\delta}_{i,j}). \quad (12)$$

Then the best path $\mathcal{P}_{i,j}$ ending at node $n_{i,j}$ is $\mathcal{P}_{i-1,j+\delta^*} \oplus e^{i,j}_{i-1,j+\delta^*}$. Based on equations from 8 to 12, we can compute the optimal path ending at every node of $\mathcal{G}$. Let:

$$j^* = \arg\max_{j} \mathcal{E}(\mathcal{P}_{t,j}), \quad (13)$$

then $\mathcal{P}_{t,j^*}$ is the finally detected local segment.

In this section, we have further discretized the d-axis and sampled $2N+1$ points along the d-axis, for the proposing of the dynamic programming based local segment extraction algorithm. In default, we set N as 100, i.e., 201 points being sampled. Fig. 5 (d) and (e) show the results extracted by the above dynamic programming algorithm, where (d) ignores the smoothness term by setting β as 0 while (e) considers the smoothness term by setting $\beta = 3$. The dynamic programming algorithm can obtain nearly global optimal solution as shown in (e), though it approximates the formulation of Eq. 4 by further discretizing the d axis. This may be due to two reasons. First, the discretization by $N = 100$ is dense enough to guarantee good results. And second, the dynamic programming algorithm does find the global optimal solution in the discretization space.

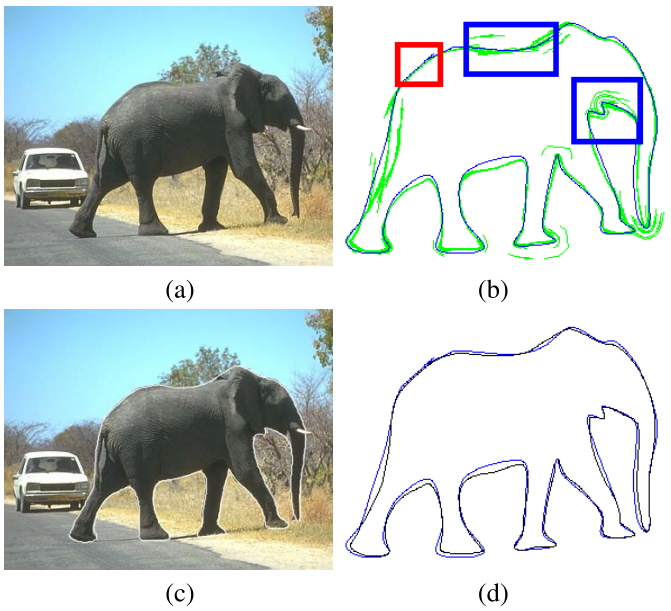


Fig. 7. (a) Source image. (b) The blue line is an initial curve indicated by a user, and the green lines are local segments extracted around the initial curve. The local segments make high agreement with each other in the red box, while exhibiting large variance in the blue boxes. (c) Our result imposed on the source image. (d) The blue line is the initial curve, and the black line is the global contour extracted by our method.

Fig. 7 (b) shows all the local segments for the input of Fig. 7 (a). The blue line is the initial curve and the green lines are the local segments extracted by our dynamic-programming based local segment extraction algorithm applied to all the sub-bands. It can be seen that the local segments make high agreement with each other in places with apparent gradients (see the red box in Fig. 7 (b)), but exhibit large variance in places with obscure contours (the blue boxes). In the later case, the extraction of a single segment is prone to error. But local segments with different lengths are complementary with each other, so as to enhance the correctness of the extraction.

D. wPCA-Based Global Contour Integration

Now, we have densely sampled local segments. The next task is to extract a single contour from all the local segments that best matches with the target object contour. We follow two rules to design a global contour integration algorithm. Firstly, if more local segments go through a pixel, the global contour should have higher probability to go through the pixel too. Secondly, the global contour should follow the overall direction of local segments. That means the direction at any point of the global contour should be consistent to the overall direction of the local segments nearby. PCA is well known for its capability in estimating principal direction for a set of points. We therefore use it in our global contour integration algorithm. More concretely, we use weighted PCA (wPCA) to

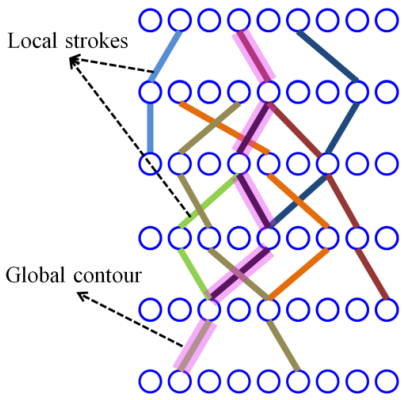


Fig. 8. Illustration of the wPCA-based global integration algorithm. The lines of different colors are local segments which determine the edge connections of the trellis graph. The wider transparent pink line, overlaid on local segments, illustrates the global contour detected.

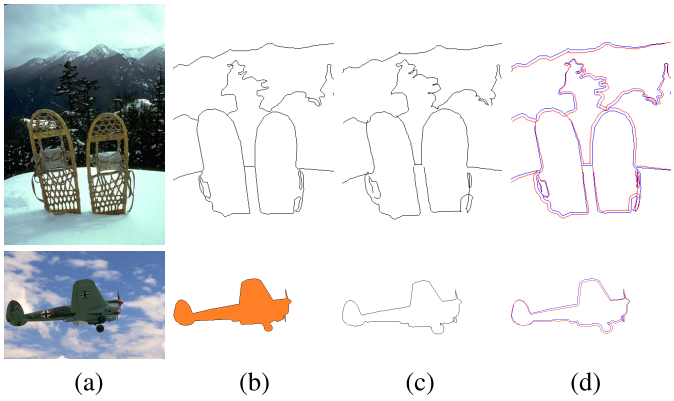


Fig. 9. (a) Testing images. (b) Top: ground truth boundary. Bottom: ground truth mask of main object. (c) Add sine noises to the ground truth of (b) by setting $f = 180$ and $A = 6$ (For mask, we extract its boundary first). (d) Blue curves: ground truth. Red curves: ground truth adding sine noises.

compute the locally overall direction of a point with respect to all the other points in the same set.

Before delving into our global contour integration algorithm, we review the theory of wPCA at first. Let $\mathcal{U}$ be a set of points with mean $(0, 0)$. For arbitrary point $\mathbf{p}_i$ of $\mathcal{U}$, the weighted covariance matrix of $\mathbf{p}_i$ with respect to all the other points of $\mathcal{U}$ is defined as (here $\mathbf{p}_i$ is a row vector):

$$Cov_i = \sum_j \theta(||\mathbf{p}_i - \mathbf{p}_j||)(\mathbf{p}_i - \mathbf{p}_j)^T(\mathbf{p}_i - \mathbf{p}_j), \quad (14)$$

where $\theta(r) = \exp^{-0.5\cdot(r/\sigma)^2}$ is a function reducing the influence of points that are far away from $\mathbf{p}_i$ (σ is set as 30 in default). Decomposing the covariance matrix by SVD, we obtain eigenvalues and eigenvectors. The eigenvector corresponding to the largest eigenvalue is the principal direction of point $\mathbf{p}_i$, which is denoted as $\mathbf{D}(\mathbf{p}_i)$.

Based on the above two rules that should be followed and the technique of wPCA, we design a dynamic programming based global contour integration algorithm. As shown in Fig. 8, we construct a trellis graph that is very similar to the graph in Fig. 6 except that two nodes of adjacent rows are now linked together only if at least one local segment passes through them simultaneously. Each node $n_{i,j}$ contains two values, $C(i, j)$ and $\mathbf{D}(\mathcal{T}(l_i, d_{i,j}))$, where $C(i, j)$ counts the number of local segments that go through point $(l_i, d_{i,j})$, and $\mathbf{D}(\mathcal{T}(l_i, d_{i,j}))$ is the principle direction of the point $\mathcal{T}(l_i, d_{i,j})$ with respect to all the other points of local segments. Our aim is to find a path across the whole trellis graph. This can be achieved by a dynamic programming algorithm similar to the one for local segment extraction.

Given a node $n_{i,j}$, and assuming $n_{i-1,j+\delta}$ be a node linked with $n_{i,j}$ by edge $e^{i,j}_{i-1,j+\delta}$, we compute the energy of path $\mathcal{P}_{i,j}$ by:

$$\mathcal{E}(\mathcal{P}_{i,j}) = \mathcal{E}(\mathcal{P}_{i-1,j+\delta}) + C(i, j) + \mathbf{u}(e^{i,j}_{i-1,j+\delta}) \cdot \mathbf{D}(\mathcal{T}(l_{i-1}, d_{i-1,j+\delta})). \quad (15)$$

As in Eq. 8, $\mathcal{E}(\mathcal{P}_{i-1,j+\delta})$ is assumed to have been computed. With Eq. 15, we can find the best global contour by using of the same way applied to Eq. 8.

Fig. 7 (c) shows the result of the above global integration algorithm. Fig. 7 (d) shows the difference between the initial curve and our extracted contour visually, where the blue line is initial curve and the black line is the final contour extracted by our method.

IV. Experiments

A. Experimental Setup

To evaluate our method and compare it with related approaches, we conduct quantitative experiments on BSDS500 dataset [7]. BSDS500 contains 500 images, in which 200 images are used for training, 100 images for validation, and 200 images for testing. Since our method is not learning based, we only use the 200 testing images. For each testing image, there are about 5 boundary images and 5 segmentation images, all are annotated by human as ground truth of the corresponding image. In our experiments, we do not need all the boundary or segmentation ground truth of each image. Instead, only one of them is needed, as input to our method or other approaches to be compared. Among the boundary ground truth, we choose the one with the moderate number of annotated boundary pixels. Among the segmentation ground truth, we choose the one with the fewest number of segmented regions and then manually select the region containing the main object of the image. In this way, we have 200 images, 200 boundary ground truth corresponding to these images, and 200 ground truth masks corresponding to the main objects of these images. Fig. 9 gives two examples of the testing images and the corresponding ground truth. In Fig. 9 (a) are two testing images. In (b), we show the ground truth boundary image on the top, while showing the main object mask at the bottom.

We add some noises to the ground truth curves. Specifically, we add a sine signal to them. For each point i of a ground truth curve, we first compute the normal direction of the point, and then move the point along the norm by a distance of $A \cdot sine(i \cdot \frac{\pi}{f})$, where A is the amplitude of the sine signal, and f determines the frequency of the signal. Fig. 9 (c) shows the noisy curves when $A = 6$ and $f = 180$, and (d) combines the ground truth and noisy curves together to illustrate their differences visually.

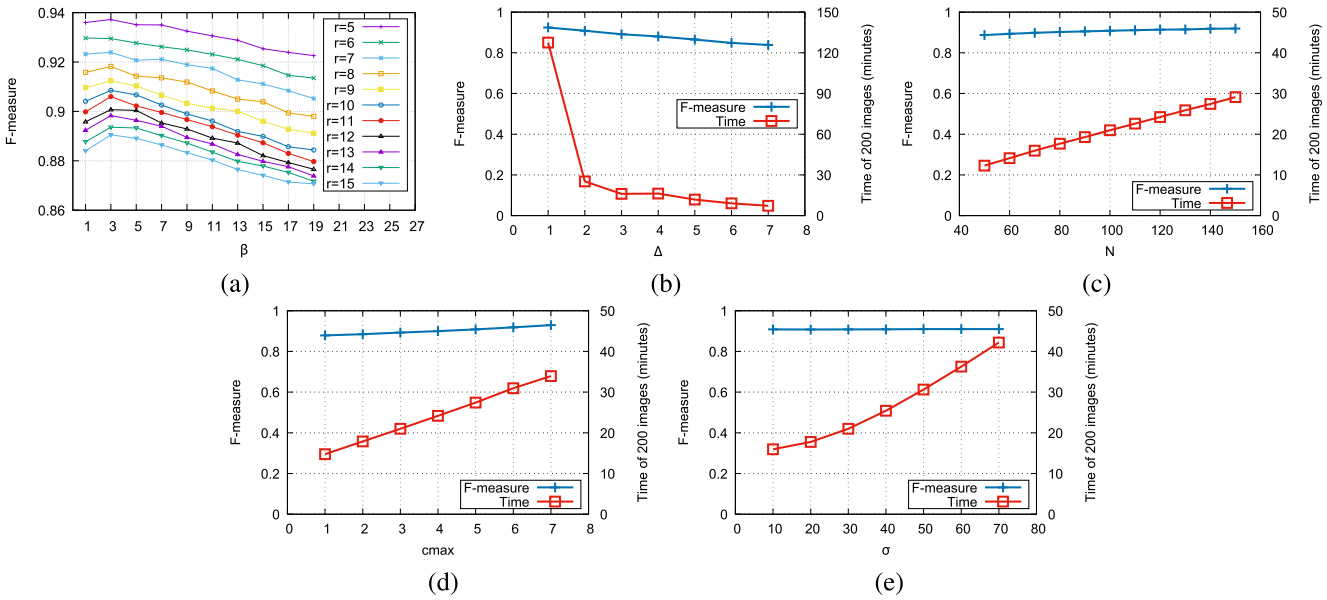


Fig. 10. Parameter analysis experiments. The noises are added by setting $A = 4$ and $f = 180$. In default $\beta = 3$, $r = 10$, $\Delta = 2$, $N = 100$, $cmax = 3$, and $\sigma = 30$. Each experiment is performed by changing some parameters while keeping the remaining parameters as their default values. (a) Experiments of combinations of β and r, where β is from 1 to 19, and r is from 5 to 15. (b) Experiments of Δ which is from 1 to 7. (c) Experiments of N which is from 50 to 150. (d) Experiments of $cmax$ which is from 1 to 7. (e) Experiments of σ which is from 10 to 70.

TABLE I

EXPERIMENTS BY CHANGING f AND A OF A SINE SIGNAL TO GENERATE DIFFERENT NOISY IMAGES. "F-MEA" STANDS FOR "F-MEASURE"

$f = 180$	A	4	5	6	7	8	9
	Noise F-mea	0.95	0.81	0.68	0.62	0.57	0.53
$A = 6$	f	22.5	45	90	180	360	720
	Noise F-mea	0.62	0.62	0.63	0.68	0.78	0.91

Following [6], we use **precision**, **recall**, and **F-measure** to measure the differences between two boundary images quantitatively. Precision is the fraction of detections that are true positives rather than false positives, while recall is the fraction of true positives that are detected rather than missed. F-measure is the harmonic mean of the precision and recall measures. The larger the F-measure, the better the edge/contour detection method. BSDS500 provide codes to compute these measures and we use their implementations.

In Table I, we show the quantitative differences between ground truth boundaries and the boundaries by adding noises to the ground truth. In the first case, we fix f as 180, and increase A gradually from 4 to 9. We compute the total F-measure of the whole 200 noisy images. As can be seen from the table, the F-measure decreases dramatically from 0.95 to 0.53. In the second case, we fix A as 6, and then gradually increase f which decreases the frequency of the noise signal. The F-measure increases from 0.62 to 0.91. We draw two conclusions from these experiments. First, the way we add noises is effective as it can reduce the F-measure appropriately. Second, larger amplitude or larger frequency of the sine signal cause larger noises to the ground truth curves.

B. Parameter Analysis

The proposed method contains several parameters. They are β that balances evidence and smoothness energies of a local segment, r that limits the distance of local segments from the user indicated initial spline, Δ, N and $cmax$ that determine the size and structure of the trellis graphs, and σ used to adjust wPCA algorithm. The default values of these parameters are $\beta = 3$, $r = 10$, $\Delta = 2$, $N = 100$, $cmax = 3$, and $\sigma = 30$.

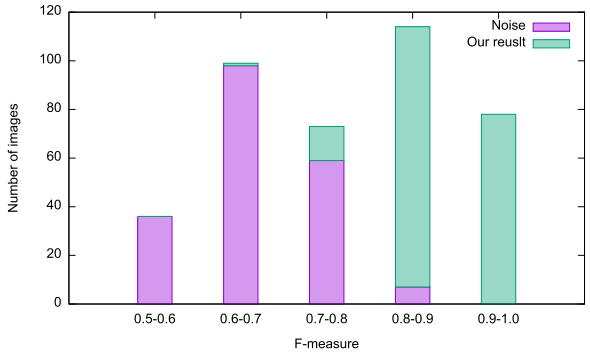


Fig. 11. F-measure distributions of noisy inputs ($f = 180$, $A = 6$) and our output results.

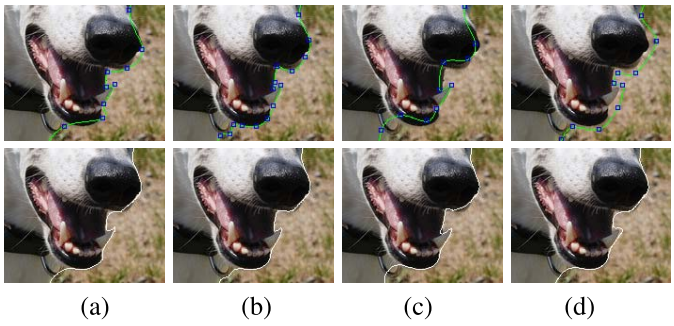


Fig. 12. The top row shows user interactions. The bottom row shows extracted contours by our method. (a) Fewer user interactions. (b) More user interactions. (c) User interactions on the left of the target contour. (d) User interactions on the right of the target contour.

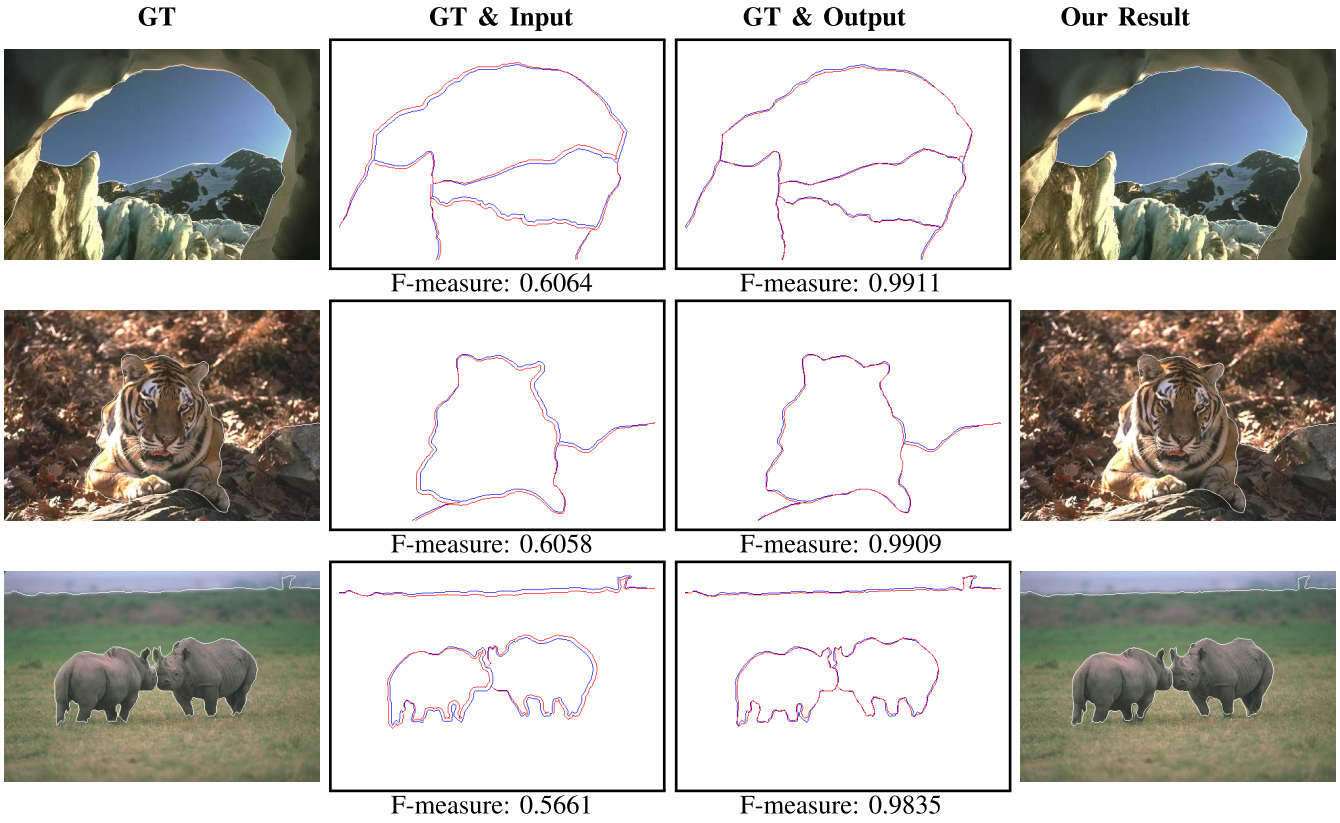


Fig. 13. First column: source images together with ground truth (**GT**) boundaries. Second column: blue curves are ground truth, red curves are noisy inputs by adding noise to ground truth ($f = 180$, $A = 6$). Third column: blue curves are ground truth, red curves are output contours generated from the proposed method. Fourth column: source images together with boundaries computed by our method. The numbers below images are F-measures of these red curves compared to blue ones.

In this section, we design experiments to show how we select the default values for these parameters.

The experiments are performed by first generating noisy boundaries, and then input the noisy boundaries to the proposed method. Finally, the F-measure between the outputs of the proposed method and the ground truth boundaries are computed. Since different parameters produce different outputs, we can obtain the influence of each parameter to the proposed method.

To generate noisy input, we set $f = 180$ and $A = 4$. We use small A here such that we can test small r which should be at least larger than A to include the ground truth contour in the search scope of the proposed method.

Our method has 6 parameters. It is not possible to test all the combinations of these parameters. Instead, we divide the parameters into two groups. r and β belong to a group, and Δ, N, $cmax$, and σ belong to the other group. We assume Δ, N, and $cmax$ to be the larger the better. This is reasonable because the larger the three parameters, the finer the discretization of the curve-centered coordinate system, and then more optimum local segments and global contour can be extracted. We thus directly set $\Delta = 2$, $N = 100$, and $cmax = 3$, which we assume to be appropriate to produce good results. We empirically set $\sigma = 30$ after several experiments with which the local principle directions of points can be correctly estimated. With the known Δ, N, $cmax$, and σ, we then test all the combinations of r and β by changing r from 5 to 15, and β from 1 to 19. Fig. 10 (a) shows the result of the experiments. It can be seen that for any r, we can obtain the best F-measure when $\beta = 3$. This tells the best value for β is 3. Therefore we set 3 as the default value of β. We also find that for any β, the F-measure decreases with the increase of r. That means r is very critical to the proposed method. It should not be too large, otherwise the quality of the outputs would decrease. Also, it should not be too small, otherwise the user has to provide very good initial curve. We find $r = 10$ is a good compromise.

With $r = 10$ and $\beta = 3$, we then test different Δ, N, $cmax$, and σ, as shown in Fig. 10 (b)-(e). When changing one of Δ, N, $cmax$, and σ, the other three are set as their default values as stated in the first paragraph of this section. The experiments meet our expectations. Increasing N or $cmax$ or decreasing Δ all increase the F-measure, i.e., we do obtain better results when discretizing the curve-centered coordinate system more finely. However, the finer the discretization, the more time consumed, as shown in Fig. 10 (b)-(e). We find that $\Delta = 2$, $N = 100$, $cmax = 3$ make good compromise between effectiveness and efficiency. We thus set $\Delta = 2$, $N = 100$, and $cmax = 3$ in default. As for σ, the experiments show that it does not influence the results much, but the larger the σ, the more time consumed in the computation. We set it as 30 in default which is the empirically found one.

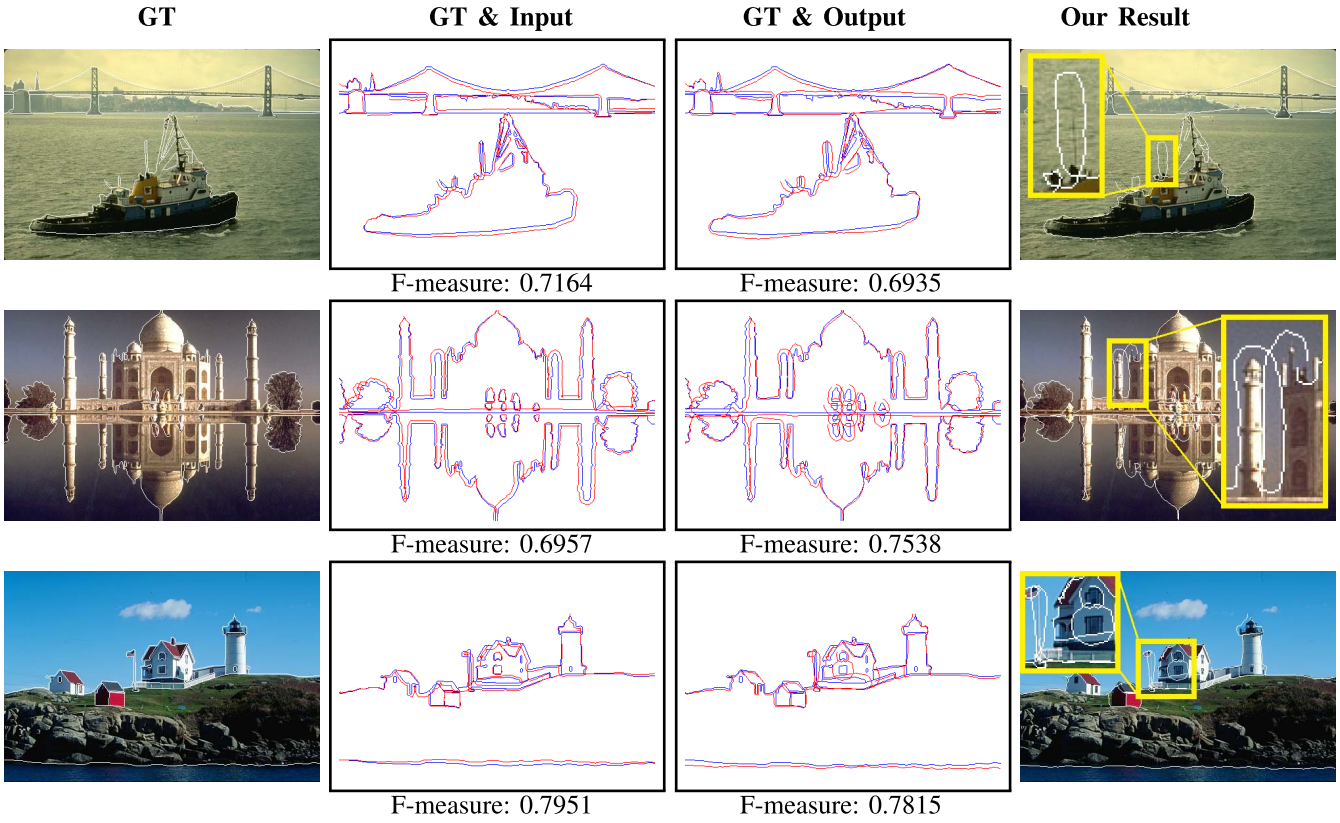


Fig. 14. First column: source images together with ground truth (**GT**) boundaries. Second column: blue curves are ground truth, red curves are noisy inputs by adding noise to ground truth ($f = 180$, $A = 6$). Third column: blue curves are ground truth, red curves are output contours generated from the proposed method. Fourth column: source images together with boundaries computed by our method with some failure places annotated. The numbers below images are F-measures of these red curves compared to blue ones.

C. Performance

We implement the proposed method in C++, and run it on Surface Pro 4. Given the default parameters, and setting $A = 4$ and $f = 180$, we run the proposed method on all of the 200 testing images 10 times, costing us 22 minutes and 42 seconds (or 22'42") on average per time. We then count the total length of all the boundaries of the 200 images, which is 633,679 pixels. Every 100 pixel cost us 0.21". Note that the images of BSDS500 are all of resolution of 481×321 or 321×481. Therefore, our algorithm is fast enough for the BSDS500 images, on which we can obtain real-time performance. However, for larger images such as 1080P images, the proposed method cannot work in real-time. We retain the further acceleration of the proposed method, such as by utilizing GPU, as a future work.

D. Robustness to User Interactions

In this section, we evaluate the robustness of our approach to different user inputs. We fix the parameters as their default values, and then run the proposed method with different noisy inputs by setting different A and f to the sine signal. Table II shows the experiment results. The table is similar to Table I, except that the F-measures of the output results of the proposed method by inputting different noisy inputs are also listed. From the table, we can see that the quality of the output results decreases with the increase of noises. But the decrease rate is slower than the increase rate. For example, when $f = 180$ and A changes from 4 to 5, the F-measure of noisy images is from 0.95 to 0.81, while the F-measure of our results changes only from 0.91 to 0.90. We also observe that even if the noises are very large, our method can output good results. For example, when $f = 180$ and $A = 8$, the F-measure of noisy images is 0.57, while our F-measure is 0.8 which is much higher.

TABLE II

EXPERIMENTS BY INPUTTING DIFFERENT NOISY IMAGES TO THE PROPOSED METHOD. "F-MEA" STANDS FOR "F-MEASURE"

$f = 180$	A	4	5	6	7	8	9
	Noise F-mea	**0.95**	0.81	0.68	0.62	0.57	0.53
	Our F-mea	0.91	**0.90**	**0.88**	**0.84**	**0.80**	**0.76**
$A = 6$	f	22.5	45	90	180	360	720
	Noise F-mea	0.62	0.62	0.63	0.68	0.78	**0.91**
	Our F-mea	**0.71**	**0.81**	**0.86**	**0.88**	**0.89**	**0.91**

Taking $f = 180$ and $A = 6$ as an example, the total F-measure of noisy images is 0.68, and the total F-measure of our outputs is 0.88. In Fig. 11, we show the F-measure distributions of noisy images and our outputs. The improvement is visually apparent. Most of noisy images have F-measures that are distributed from 0.5 to 0.7, while ours are mostly distributed from 0.8 to 1.0.

In Fig. 12, we give an example demonstrating the robustness of our method to different user inputs. The top row of Fig. 12 shows four different user inputs. The bottom row shows the corresponding contours extracted by our method. We provide fewer key points in Fig. 12 (a), and more key points in (b).

TABLE III

TOTAL RECALLS, PRECISIONS, AND F-MEASURES OF THE RESULTS OF GRABCUT [19] AND OUR METHOD

Method	Recall	Precision	F-measure
GrabCut	0.7586	0.7048	0.7307
Our	0.9385	0.9139	0.9261

The key points of (c) are on the left side of the target contour, while the key points of (d) are on the right side of the target contour. Although the four user inputs are very different from each other, our method produces very similar results in the bottom row.

In Fig. 13, we show several examples for which our method can produce good results. The images in the first column are source images together with ground truth boundary annotations. The images in the second column show both ground truth boundaries and noisy inputs generated by setting $f = 180$ and $A = 6$. The differences between the ground truth and noisy boundaries are visually apparent. This is confirmed by the F-measures under these images which are all smaller than 0.61. By inputting the noisy boundaries into our method as initial curves, our method can largely recover the ground truth information, as in the third column which shows both ground truth boundaries and the boundaries produced by our method. The differences are very small. Correspondingly, the F-measures are all larger than 0.98. The last column shows the source images together with our produced boundaries.

In comparison, we show several failure cases of our method in Fig. 14. The noises are also produced by setting $f = 180$ and $A = 6$. Among all the 200 images produced by our method, we totally have 15 images whose F-measures are less than 0.8. Fig. 14 gives 3 representative images among the 15 ones. From the 15 failure images, we find that our method fails mainly due to two reasons. The first reason is that our method cannot handle boundaries of skinny geometries, such as the mast of ship (last image of the first row) and the flagpole near the house (last image of the third row). Second, our method may be confused by very close boundaries. For example in the last image of the second row where two buildings are very close to each other, our method identifies the boundaries in the wrong order. To solve the two problems, an easy way is to reduce r at these places. We retain the task of solving these problems automatically as a future work.

E. Comparisons With Interactive Image Segmentation

In this section, we compare our method with the state-of-the-art interactive image segmentation approach GrabCut [19]. We use the GrabCut code provided at "https://grabcut.weebly.com/code.html". GrabCut requires the user to indicate a region of interest (ROI). Then it can automatically extract the object in the ROI. As described in Sec. IV-A, for each of the 200 testing image, we have indicated a main object of it and also the mask of the object. Now we perform a dilation to the mask to expand it by 10 pixels. The dilated mask is then input to the GrabCut.

To generate inputs to our method, we add sine noise to the ground truth object boundaries by setting $f = 180$ and $A = 6$. Table. III shows the total recalls, precisions, and F-measures

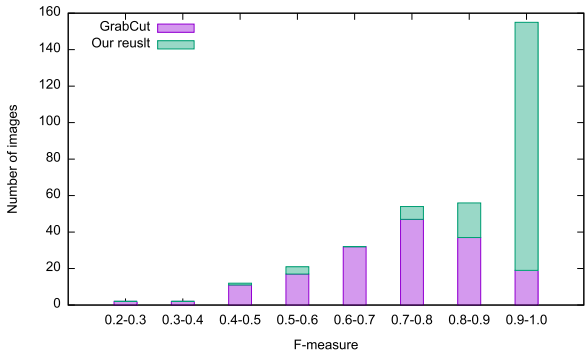


Fig. 15. Distributions of F-measures of the results computed by GrabCut [19] and our method.

of the results of GrabCut and our method. Our total F-measure on the 200 testing images is 0.9261, while the total F-measure of GrabCut is only 0.7307. Fig. 15 shows the distributions of the F-measures of images. As can be seen, the F-measures of our results are more distributed from 0.8 to 1.0, while the F-measures of GrabCut results are more distributed from 0.5 to 0.9. In Fig. 16, we give several examples of the results of GrabCut and our method. The first two rows show both good results. The third to fifth rows show failure results of GrabCut which however can be handled well by our method. In these examples, GrabCut tends to miss some parts of the target object. The six row shows an example GrabCut gives good results while our results are not that good, as our method cannot handle boundaries of skinny geometries well. The last row shows an example for which both GrabCut and our method fail. The comparisons demonstrate that our method outperforms GrabCut if a good initialization within 10 pixels to the ground truth is given. GrabCut has its own advantage. It can output satisfactory results even if the ROI is not that compact.

F. User Study

We use a user study to compare our method with the latest Photoshop CC 2018 and LabelMe [17]. We have invited 10 participants, and randomly chosen 12 images from the 200 testing images. We then randomly divided the 12 images into 3 groups, and sent the 3 groups of images together with their ground truth boundaries from BSDS500 to each participant. Each participant used Photoshop to process the images of the first group, used LabelMe to process the second group, and used our method to process the third group. The participant were asked to annotate boundaries as similar as possible to the corresponding ground truth boundaries of BSDS500.

Before performing the user study, we have conducted an hour-long hands-on session on Photoshop, LabelMe and our proposed method, for all participants in the user study. We have strongly recommended that the participants use the "pen", "quick selection", and "magnetic lasso" tools of Photoshop, along with the "layer" feature. At the same time, they were free to use other tools provided by Photoshop. By LabelMe, the participants can use mask and polygon tools to annotate object boundaries directly. As for our method,

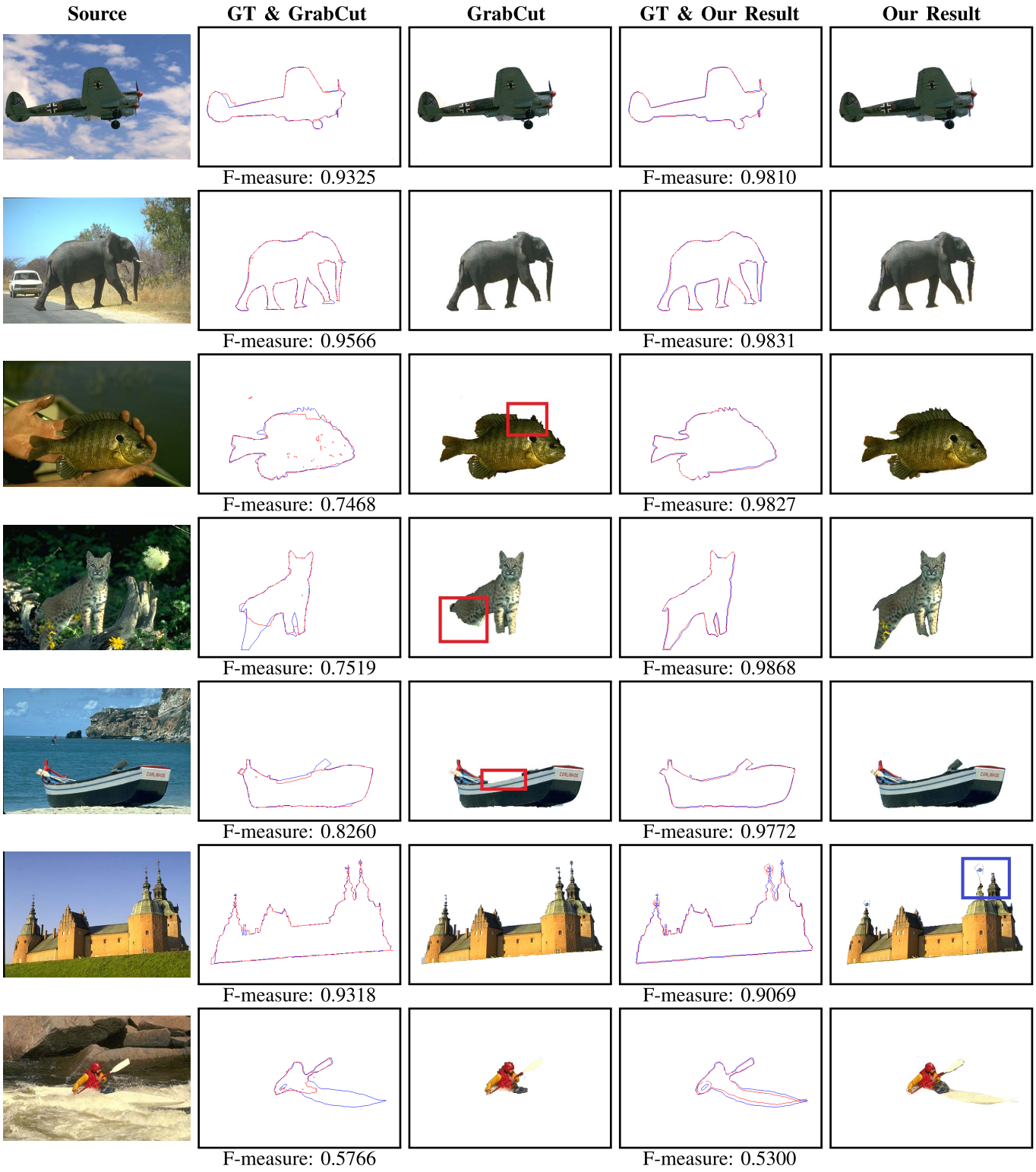


Fig. 16. First column: source images. Second column: blue curves are ground truth (GT) object boundaries, red are boundaries segmented by GrabCut [19]. Third column: results of GrabCut. Fourth column: blue curves are ground truth object boundaries, red are boundaries computed by our method. Fifth column: our results. The numbers below images are F-measures of these red curves compared to blue ones.

they only need to indicate key points along the contour to be extracted. "Undo" and "redo" operations can also be used. After the teaching, we had asked the participants to learn the tools by themselves for 30 minutes. Finally, the participants began to annotate the 12 images by the three tools.

For each method, we have 40 annotation images in total, annotated by 10 participants. For each annotation image, we have recorded the annotation time, and then computed the average time of the method used for each image. We also computed the total F-measure of the 40 annotation images

TABLE IV
USER STUDY

Method	Photoshop	LabelMe	Our method
Average time used	6'54"	1'44"	2'58"
Total F-measure	0.9232	0.8441	0.9187

to the corresponding ground truth boundaries of BSDS500. Table IV shows the results of the user study. The average time per image used by Photoshop to an image is 6 minutes 54 seconds (or 6'54"). LabelMe consumes 1'44". Our method uses 2'58". After querying the participants, we found that LabelMe is the fastest because it does not provide any feedback to users. Therefore, there was no trial-and-error operation when using LabelMe. But when using Photoshop and our method, there may exist some trial-and-error operations which increase the time. On the other hand, the total F-measure of Photoshop is 0.9232, the total F-measures of LabelMe is 0.8441, and the total F-measure of our method is 0.9187. LabelMe gives the lowest quality, though it is much faster. Photoshop gives the highest quality, but costs much more time. Our method costs much less time compared to Photoshop, while achieving nearly the same quality as Photoshop. This demonstrates the effectiveness of our method.

V. Conclusion and Future Work

In this paper, we proposed a novel user interaction based image contour extraction method. The method is composed of three stages: multi-scale sub-band sampling, sub-band local contour segment extraction, and global contour integration of the local segments. The proposed local-to-global method is sketch-alike, which utilizes the redundancies among the local segments to enhance the correctness of the global contour. We propose dynamic programming based algorithms to fulfills the tasks of the last two stages effectively. In the experiments, we conduct many quantitative comparisons and a user study. The experiments demonstrate that our method can save user's time when annotating image boundaries, and output high-quality boundary results.

Future works may fall in three aspects. First, our method cannot handle boundaries of skinny geometries well and may be confused when two boundaries are very close. We will try to solve the two problems. Second, we will accelerate the proposed method to make it fast enough for high-resolution images. Third, we can use the method to annotate a large ground truth boundary dataset.

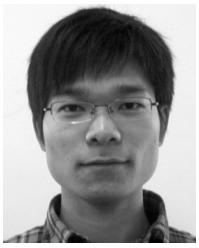
Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] Y. Liu *et al.*, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.
[2] R. O. Duda and P. E. Hart, *Pattern Classification And Scene Analysis*. New York, NY, USA: Wiley, 1973.
[3] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
[4] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, Sep. 1991.
[5] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *Int. J. Comput. Vis.*, vol. 30, no. 2, pp. 117–156, 1998.
[6] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.
[7] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
[8] R. Xiaofeng and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 584–592.
[9] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2006, pp. 1964–1971.
[10] J. J. Lim, C. L. Zitnick, and P. Dollar, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165.
[11] P. Dollár, C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.
[12] X. Ren, C. C. Fowlkes, and J. Malik, "Scale-invariant contour completion using conditional random fields," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Oct. 2005, pp. 1214–1221.
[13] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1395–1403.
[14] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.
[15] S. Osher and R. P. Fedkiw, "Level set methods: an overview and some recent results," *J. Comput. Phys.*, vol. 169, no. 2, pp. 463–502, 2001.
[16] 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," in *Proc. 8th Int. Conf. Comput. Vis.*, vol. 2, Jul. 2001, pp. 416–423.
[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A database and web-based tool for image annotation," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 157–173, 2008.
[18] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 109–131, 2006.
[19] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004.
[20] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.*, vol. 22, no. 1, pp. 61–79, 1997.
[21] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.
[22] J. Liang, T. McInerney, and D. Terzopoulos, "United snakes," *Med. image Anal.*, vol. 10, no. 2, pp. 215–233, 2006.
[23] S. Balla-Arabé, X. Gao, D. Ginhac, V. Brost, and F. Yang, "Architecture-driven level set optimization: From clustering to subpixel image segmentation," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 3181–3194, Dec. 2016.
[24] S. Zhou, J. Wang, S. Zhang, Y. Liang, and Y. Gong, "Active contour model based on local and global intensity information for medical image segmentation," *Neurocomputing*, vol. 186, pp. 107–118, Apr. 2016.
[25] G. Huo, S. X. Yang, Q. Li, and Y. Zhou, "A robust and fast method for sidescan sonar image segmentation using nonlocal despeckling and active contour model," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 855–872, Apr. 2017.
[26] K. Zhang, L. Zhang, K.-M. Lam, and D. Zhang, "A level set approach to image segmentation with intensity inhomogeneity," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 546–557, Feb. 2016.
[27] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 1963.
[28] J. M. Prewitt, "Object enhancement and extraction," *Picture Process. Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.
[29] M. C. Morrone and R. A. Owens, "Feature detection from local energy," *Pattern Recognit. Lett.*, vol. 6, no. 5, pp. 303–313, 1987.
[30] P. Perona and J. Malik, "Detecting and localizing edges composed of steps, peaks and roofs," in *Proc. 3rd Int. Conf. Comput. Vis.*, Dec. 1990, pp. 52–57.
[31] J. H. Elder and S. W. Zucker, "Computing contour closure," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 1996, pp. 399–412.
[32] L. R. Williams and D. W. Jacobs, "Stochastic completion fields: A neural model of illusory contour shape and salience," *Neural Comput.*, vol. 9, no. 4, pp. 837–858, 1997.

[33] P. Felzenszwalb and D. McAllester, "A min-cover approach for finding salient curves," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, Jun. 2006, p. 185.
[34] Q. Zhu, G. Song, and J. Shi, "Untangling cycles for contour grouping," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[35] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2014, pp. 328–335.
[36] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[37] Y. Ganin and V. Lempitsky, "N^4-fields: Neural network nearest neighbor fields for image transforms," in *Proc. Asian Conf. Comput. Vis.* Springer, 2014, pp. 536–551.
[38] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3982–3991.
[39] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015 pp. 4380–4389.
[40] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2015, pp. 3431–3440.
[41] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2005, pp. 1124–1131.
[42] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[43] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[44] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.
[45] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2141–2148.
[46] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
[47] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2003, p. 1.
[48] D. T. Nguyen, "A novel chamfer template matching method using variational mean field," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2425–2432.



Yongwei Nie received the B.Sc. and Ph.D. degrees from the Computer School, Wuhan University, in 2009 and 2015, respectively. He is currently an Associate Researcher with the School of Computer Science and Engineering, South China University of Technology. His research interests include image and video editing, and computational photography.

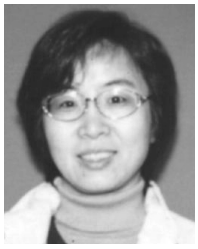
Xu Cao is currently pursuing the master's degree with the School of Computer Science and Engineering, South China University of Technology. His research interest includes image and video editing.

Ping Li received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong. He is currently an Assistant Professor with the Macau University of Science and Technology, Macau, China. He holds one image/video processing national invention patent. His current research interests include image/video stylization, GPU acceleration, and creative media. He has excellent research project reported worldwide by ACM TechNews.

Qing Zhang received the B.Sc. and Ph.D. degrees in computer science and engineering from Wuhan University in 2011 and 2017, respectively. He is currently a Research Associate Professor with the School of Data and Computer Science, Sun Yat-Sen University. His research interests include computer graphics and computer vision.

Zhensong Zhang received the B.E. degree from Xidian University in 2011 and the M.S. degree from the University of Chinese Academy of Sciences in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include image and video editing, and computational photography.

Guiqing Li received the B.Sc. degree from the University of Science and Technology of China in 1987, the M.Sc. degree from Nankai University, Tianjin, in 1990, and the Ph.D. degree from the Institute of Computing Technology, Beijing, in 2001. He was a Post-Doctoral Researcher with the State Key Lab of CAD&CG, Zhejiang University. In 2003, he joined the College of Computer Science and Engineering, South China University of Technology, where he is currently a Professor.



Hanqiu Sun received the M.S. degree in electrical engineering from The University of British Columbia and the Ph.D. degree in computer science from the University of Alberta, Canada. She is currently a Visiting Professor with Shanghai Jiao Tong University and Zhejiang University, and an Adjunct Professor with the University of ESTC. Her current research interests include virtual reality, interactive graphics, real-time hypermedia, virtual surgery, effective image/video synopsis, panoramic video displays, dynamics, and touch-enhanced simulations.