Interactive Natural Image Segmentation via Spline Regression

Shiming Xiang, Feiping Nie, Chunxia Zhang, and Changshui Zhang, *Member, IEEE*

***Abstract*—This paper presents an interactive algorithm for segmentation of natural images. The task is formulated as a problem of spline regression, in which the spline is derived in Sobolev space and has a form of a combination of linear and Green's functions. Besides its nonlinear representation capability, one advantage of this spline in usage is that, once it has been constructed, no parameters need to be tuned to data. We define this spline on the user specified foreground and background pixels, and solve its parameters (the combination coefficients of functions) from a group of linear equations. To speed up spline construction, K-means clustering algorithm is employed to cluster the user specified pixels. By taking the cluster centers as representatives, this spline can be easily constructed. The foreground object is finally cut out from its background via spline interpolation. The computational complexity of the proposed algorithm is linear in the number of the pixels to be segmented. Experiments on diverse natural images, with comparison to existing algorithms, illustrate the validity of our method.**

***Index Terms*—Interactive natural image segmentation, K-means clustering, spline regression.**

I. Introduction

EXTRACTING the foreground objects in natural images is one of the most fundamental tasks in image processing and understanding. Generally, this task can be formulated as a problem of image segmentation. The efforts in segmentation have surged in recent decades, with the development of numerous approaches and proposals for real world applications [5], [24], [25], [42]. In spite of many thoughtful attempts, it is still very difficult to develop a general framework which can yield satisfactory segmentations for diverse natural images. The difficulties lie in the complexity of perceiving and modeling the numerous visual patterns in natural images and the intrinsic ambiguity of grouping them to be the needed objects.

Manuscript received May 14, 2008; revised February 16, 2009. First published May 12, 2009; current version published June 12, 2009. S. Xiang was supported by the Excellent SKL Project of NSFC (No. 60723005). This work was supported in part by the National Natural Science Foundation of China under Grants 60835002, 60721003, and 60705022. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sharathchandra Pankanti.

S. Xiang is with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China (e-mail: smxiang@gmail.com).

F. Nie and Chanshui Zhang are with the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: nfp03@mails.tsinghua.edu.cn; zcs@mail.tsinghua.edu.cn).

Chunxia Zhang is with the School of Software, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China (e-mail: cxzhang@bit.edu.cn).

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

Digital Object Identifier 10.1109/TIP.2009.2018570

To reduce the complexity and intrinsic ambiguity, one method is to design interactive frameworks, which can allow the user to specify the foreground and background according to her/his own understanding about the image. In such a work setting, the user can also act as a judge to accept or refuse the current segmentation results, or add more strokes to obtain better segmentation. In view of image perception, the user specified strokes give us the visual hints to model and group the visual patterns. With such supervised information, many existing algorithms developed in machine learning can be employed to formulate the task of image segmentation [2], [4], [13], [14], [19], [29], [32], [38].

The goal of interactive image segmentation is to cut out a foreground object from its background with modest user interaction [2], [4], [6], [14], [29], [30], [32], [38]. There are two main methods [14]: edge based and region based. Edge-based methods need the user to label the points near the object boundary. Examples include intelligent scissors [19], [20], snapping [9], and jet-stream [26]. Intelligent scissors approach is popularly used in Photoshop products as a plus tool. However, these methods still require the user to pay more attention to the edges between the foreground object and its background. Recently, researches mainly focus on region-based methods, for example, magic wand in Photoshop products, intelligent paint [1], [28], sketch-based interaction [34], interactive graph cut [2], [4], Grabcut [29], lazy snapping [14], segmentation via random walks (RW) [10], image matting [6], [13], [30], [32], [38], distance-based interaction [27], and so on. In these methods, the interaction style is largely improved. The user can label the regions of foreground object and its background by simply dragging the mouse. The main advantage is that it does not require the user to stare at and stroke along the object boundary. With the help of statistical inference or machine learning algorithms, the developed region-based interaction frameworks have achieved great successes, for example, those based on graph cut (GC) or belief propagation on Markov random field (MRF) defined on the image to be segmented [10], [14], [33], [38].

In view of machine learning, interactive image segmentation is a typical task of supervised or semi-supervised classification. Thus, the existing classification algorithms, inductive [8], [36] or transductive [43], [44], can be considered in this task. Unfortunately, the real practice shows that the commonly used classification algorithms, for example, "linear discriminant analysis (LDA) + K-nearest neighbor (KNN) classifier" [8], support vector machine (SVM) [36], label propagation (LP) via local and global consistency [43], and RW [10] may generate unsatisfactory segmentations in complex natural images. Fig. 1 illustrates an example. The goal is to cut out the leopard from the jungle. As can be seen in Fig. 1, there exist large gaps between

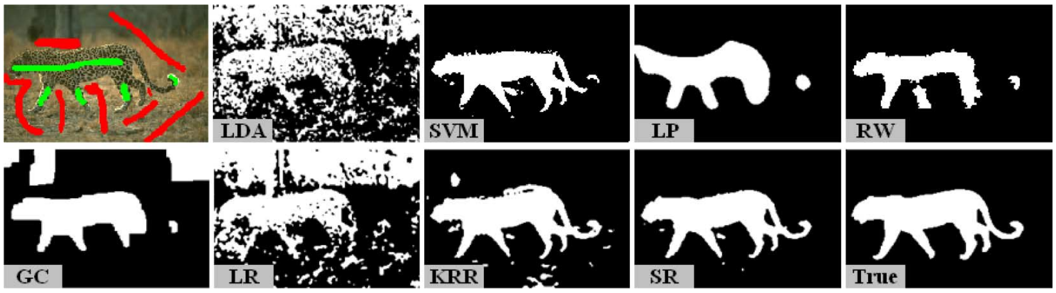


Fig. 1. Segmentations of the leopard. The first is the source image with the user specified strokes on the leopard and its background. From the second to the ninth are the segmentations obtained by linear discriminative analysis (LDA), support vector machine (SVM), label propagation (LP), random walks (RW), graph cut (GC), linear regression (LR), kernel ridge regression (KRR), and our spline regression (SR). The last is the ground truth for comparison.

the results generated by LDA, SVM, LP, and RW and the desired segmentation. Actually, one drawback in LDA is its distribution assumption. LDA is optimal in the case that the data distribution of each class is Gaussian. Obviously, the color distributions in this image are beyond Gaussian since there are more than one colors in the foreground and background regions. In addition, SVM may generates better results, but the kernel parameter should be well tuned to data. The optimal result with LP in [43] can be obtained under the condition that the data points are located in two sub-manifolds. Used here as a semi-supervised spectral segmentation, such an assumption on two separated sub-manifolds is difficult to be satisfied in natural images with similar foreground and background colors. This can also be employed to explain the performance of RW in view of equivalency between random walks and spectral segmentation [17]. Thus, to obtain better results with LP and RW, more user specified strokes should be supplied.

MRF formulation and graph-based algorithms have achieved great successes in many natural images. However, if there are many small regions with similar foreground and background colors, it is also difficult for GC algorithm [4] to obtain good segmentation. The reason is that similar colors will decrease the gaps between the likelihood values of the foreground and background pixels. Thus, uncertainty may be generated during label inference on data graph and the quality of segmentation could be degraded (see Fig. 1).

This paper presents a novel interactive natural image segmentation framework. The core idea is to formulate the interactive segmentation task as a problem of spline regression. This spline is a combination of linear and Green's functions developed in Sobolev functional space [22]. It has been proven to be suitable for the task of *scattered data interpolation* and thus popularly used in geometrical design [3]. There are two main advantages: it is smooth and able to approximate the interpolation values at the scattered data points with arbitrarily controllable accuracy. The smoothness guarantees that the spline can be used to predict the values of the unlabeled pixels. Additionally, with highly accurate approximation, the specified values for the user labeled pixels can be faithfully maintained. Thus, it could be adaptable to the situations that there exist similar foreground and background colors. Also, with controllable accuracy, one can avoid the over-fitting problem when using the estimated spline to predict the unlabeled pixels.

Specifically, the advantages or details of our interactive segmentation framework can be highlighted as follows.

1) The segmentation results achieved with spline regression on publicly available image segmentation databases are comparable to those with graph cut algorithm. Generally, it can yield satisfactory results where graph cut does. Experiments also indicate that spline regression shows better adaptability to most complex natural images, compared with LDA, SVM, LP, RW, linear regression (LR), and kernel ridge regression (KRR) [31]. Moreover, in contrast to Gaussian kernel functions used in KRR, the spline we use has no kernel parameters to be tuned to data.
2) By assigning "$+1$" to each of the user labeled foreground pixels and "-1" to each of the user labeled background pixels, the interpolation values of unlabeled pixels can be naturally classified into two classes by taking zero as a threshold. Thus, it is unnecessary to employ another classifier to make final label assignment as done in subspace learning algorithms [8]. Thus, the time consuming step of distance ranking is avoided.
3) During interactive segmentation, the user may label thousands of foreground and background pixels. This indicates that we should construct the spline on all of these pixels, and solve the linear equations with thousands of parameters. To reduce the computation, K-means clustering algorithm is employed to cluster the user specified pixels. By making use of the cluster centers, the spline construction is very fast.
4) The computational complexity of the proposed algorithm is linear in the number of the pixels to be segmented, and thus also linear in the number of cluster centers. Meanwhile, the main algorithm can be easily implemented. It can be run with only about 20 Matlab sentences.

The remainder of this paper is organized as follows. In Section II, the spline regression for interactive image segmentation is introduced. The connections of our algorithm to other algorithms will be given in Section III. In Section IV, three kinds of postprocessing methods are introduced for users to obtain better segmentations. Section V reports the experimental results and performs algorithmic comparisons. Conclusions will be drawn in Section VI.

II. Spline Regression for Interactive Image Segmentation

A. Problem Formulation

The problem we consider can be described as follows. Given an image $\mathcal{I}$ with n pixels to be segmented, namely,

$\mathcal{I} = \{p_1, \ldots, p_n\}$, and two labeled pixel sets, $\mathcal{F}$ $(\subset \mathcal{I})$ and $\mathcal{B}$ $(\subset \mathcal{I})$. Here, $\mathcal{F}$ contains the user specified foreground pixels, while $\mathcal{B}$ contains the user specified background pixels. The task is to assign a class label in $\mathcal{L} = \{\text{Foreground, Background}\}$ to each of the unlabeled pixels $p_i \in \mathcal{I}$.

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ collect the feature vectors of $\{p_i\}_{i=1}^n$, where d is the feature dimensionality. Further suppose that the user labeled $n_\mathcal{F}$ foreground pixels and $n_\mathcal{B}$ background pixels. Correspondingly, we can get two subsets of features: $\mathcal{U}_\mathcal{F} = \{\mathbf{x}_i^\mathcal{F}\}_{i=1}^{n_\mathcal{F}}$ $(\subset \mathcal{X})$ and $\mathcal{U}_\mathcal{B} = \{\mathbf{x}_i^\mathcal{B}\}_{i=1}^{n_\mathcal{B}}$ $(\subset \mathcal{X})$. Now we need to infer the labels of the unlabeled pixels in $\mathcal{X}$.

Our task is just a task of data classification. Many existing classification methods, such as "LDA + K-NN classifier" [8], SVM [36], semi-supervised classification [44], graph cut [4], random walks [10] can be applied to this task. In this paper, we use spline regression to formulate this task.

B. Spline Regression

Before constructing the spline, we assign "+1" to each of the data points in $\mathcal{U}_\mathcal{F}$ and "−1" to each of the data points in $\mathcal{U}_\mathcal{B}$ as interpolation values. Our goal is to construct a spline function f such that for each $\mathbf{x}_i^\mathcal{F} \in \mathcal{U}_\mathcal{F}$, $f(\mathbf{x}_i^\mathcal{F}) \approx 1$ and for each $\mathbf{x}_i^\mathcal{B} \in \mathcal{U}_\mathcal{B}$, $f(\mathbf{x}_i^\mathcal{B}) \approx -1$.

This task can be considered in a general regularization framework containing data fitting and function smoothness

$$J(f)=\sum_{i=1}^{n_\mathcal{F}}\left(1- f\left(\mathbf{x}_i^\mathcal{F}\right)\right)^2+\sum_{i=1}^{n_\mathcal{B}}\left(-1-f\left(\mathbf{x}_i^\mathcal{B}\right)\right)^2+\lambda S(f) \quad (1)$$

where $S(f)$ is a smoothness penalty in d dimensions on the function f, and λ is a regularization parameter.

The form of $S(f)$ will determine the form of function f. Specifically in Sobolev space [22], $S(f)$ can be defined as a semi-norm [7], [18], [37]

$$S(f) = \sum_{t_1+\cdots+t_d=s} \frac{s!}{t_1!\cdots t_d!} \int_{\mathbb{R}^d} \left(\frac{\partial^s f}{\partial x_1^{t_1}\cdots \partial x_d^{t_d}}\right)^2 d\mathbf{x} \quad (2)$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$, and $s, t_i, i = 1, \ldots d$, are positive integers. The larger the s is, the more the spline $f(\mathbf{x})$ is smooth.

Duchon [7] and Meinguet [18] demonstrated that under some constraints there is a unique spline function that minimizes $J(f)$ in (1)

$$f(\mathbf{x}) = \sum\nolimits_{i=1}^{l}\beta_i\, p_i(\mathbf{x}) + \sum\nolimits_{j=1}^{m}\alpha_j\, \phi_j(\mathbf{x}) \quad (3)$$

where $l = (d+s-1)!/(d!(s-1)!)$, $m = n_\mathcal{F}+n_\mathcal{B}$, $\{p_j(\mathbf{x})\}_{j=1}^l$ are a set of primitive polynomials spanning the space of polynomials in $\mathbb{R}^d$ of total degree less than s, and $\phi_j(\mathbf{x})$ is a Green's function [7]. In real applications, we can limit the polynomial space to be linear and take one form of the Green's functions, namely, $g(r) = r^2 \cdot \log(r)$. As one kind of radial basic functions, $g(r)$ has no parameters to be tuned to data. Then, we can obtain the following spline function:

$$f(\mathbf{x})=\beta_0+\sum_{i=1}^{d}\beta_i x_i+\sum_{j=1}^{n_\mathcal{F}}\alpha_j^\mathcal{F}\phi_j^\mathcal{F}(\mathbf{x})+\sum_{j=1}^{n_\mathcal{B}}\alpha_j^\mathcal{B}\phi_j^\mathcal{B}(\mathbf{x}) \quad (4)$$

where $\phi_j^\mathcal{F}(\mathbf{x})$ and $\phi_j^\mathcal{B}(\mathbf{x})$ take Green's function, namely, $\phi_j^\mathcal{F}(\mathbf{x}) = |\mathbf{x}-\mathbf{x}_j^\mathcal{F}|^2\log(|\mathbf{x}-\mathbf{x}_j^\mathcal{F}|)$ and $\phi_j^\mathcal{B}(\mathbf{x}) = |\mathbf{x}-\mathbf{x}_j^\mathcal{B}|^2\log(|\mathbf{x}-\mathbf{x}_j^\mathcal{B}|)$. In view of geometrical transformation [3], β_0 corresponds to the translation, $\sum_{i=1}^d\beta_i x_i$ reflects the affine transformation, while the last two terms record the locally nonlinear deformations near the $n_\mathcal{F}+n_\mathcal{B}$ scattered data points in $\mathcal{U}_\mathcal{F}\cup\mathcal{U}_\mathcal{B}$.

Now substituting the data points in $\mathcal{U}_\mathcal{F}\cup\mathcal{U}_\mathcal{B}$ into (4), we can obtain $n_\mathcal{F}+n_\mathcal{B}$ equations in terms of matrix form

$$\begin{pmatrix} \mathbf{K}_{\mathcal{F}\mathcal{F}} & \mathbf{K}_{\mathcal{F}\mathcal{B}} & \mathbf{e}_\mathcal{F} & \mathbf{X}_\mathcal{F}^T \\ \mathbf{K}_{\mathcal{B}\mathcal{F}} & \mathbf{K}_{\mathcal{B}\mathcal{B}} & \mathbf{e}_\mathcal{B} & \mathbf{X}_\mathcal{B}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_\mathcal{F} \\ \boldsymbol{\alpha}_\mathcal{B} \\ \beta_0 \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_\mathcal{F} \\ -\mathbf{e}_\mathcal{B} \end{pmatrix} \quad (5)$$

where $\boldsymbol{\alpha}_\mathcal{F} = [\alpha_1^\mathcal{F}, \alpha_2^\mathcal{F}, \ldots, \alpha_{n_\mathcal{F}}^\mathcal{F}]^T \in \mathbb{R}^{n_\mathcal{F}}$, $\boldsymbol{\alpha}_\mathcal{B} = [\alpha_1^\mathcal{B}, \alpha_2^\mathcal{B}, \ldots, \alpha_{n_\mathcal{B}}^\mathcal{B}]^T \in \mathbb{R}^{n_\mathcal{B}}$, $\beta_0 \in \mathbb{R}$, and $\beta = [\beta_1, \beta_2, \ldots, \beta_d]^T \in \mathbb{R}^d$, which are parameters to be solved. In the coefficient matrix, $\mathbf{K}_{\mathcal{F}\mathcal{F}} \in \mathbb{R}^{n_\mathcal{F}\times n_\mathcal{F}}$, $\mathbf{K}_{\mathcal{F}\mathcal{B}} \in \mathbb{R}^{n_\mathcal{F}\times n_\mathcal{B}}$, $\mathbf{K}_{\mathcal{B}\mathcal{F}} \in \mathbb{R}^{n_\mathcal{B}\times n_\mathcal{F}}$, and $\mathbf{K}_{\mathcal{B}\mathcal{B}} \in \mathbb{R}^{n_\mathcal{B}\times n_\mathcal{B}}$, which collect the values of Green's function. Obviously, $\mathbf{K}_{\mathcal{F}\mathcal{F}}$ and $\mathbf{K}_{\mathcal{B}\mathcal{B}}$ are symmetrical, and $\mathbf{K}_{\mathcal{F}\mathcal{B}} = \mathbf{K}_{\mathcal{B}\mathcal{F}}^T$. In addition, $\mathbf{X}_\mathcal{F}$ collects the data points in $\mathcal{U}_\mathcal{F}$, while $\mathbf{X}_\mathcal{B}$ collects those in $\mathcal{U}_\mathcal{B}$. That is, $\mathbf{X}_\mathcal{F} = [\mathbf{x}_1^\mathcal{F}, \mathbf{x}_2^\mathcal{F}, \ldots, \mathbf{x}_{n_\mathcal{F}}^\mathcal{F}] \in \mathbb{R}^{d\times n_\mathcal{F}}$, $\mathbf{X}_\mathcal{B} = [\mathbf{x}_1^\mathcal{B}, \mathbf{x}_2^\mathcal{B}, \ldots, \mathbf{x}_{n_\mathcal{B}}^\mathcal{B}] \in \mathbb{R}^{d\times n_\mathcal{B}}$. In (5), $\mathbf{e}_\mathcal{F} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n_\mathcal{F}}$ and $\mathbf{e}_\mathcal{B} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n_\mathcal{B}}$.

Note that in (5), there are $1 + d + n_\mathcal{F} + n_\mathcal{B}$ parameters (the combination coefficients of linear and Green's functions) to be solved. However, we have only $n_\mathcal{F} + n_\mathcal{B}$ equations. To make the problem determined, we need to introduce $d + 1$ new equations. Fortunately, these equations can be derived from the conditions of positive definite functions [7], [18], [37], [40], which are related to the uniqueness of the spline. Specifically, for the $n_\mathcal{F} + n_\mathcal{B}$ scattered data points in $\mathcal{U}_\mathcal{F}\cup\mathcal{U}_\mathcal{B}$, we have

$$\sum\nolimits_{j=1}^{n_\mathcal{F}}\alpha_j^\mathcal{F}\cdot p_i\left(\mathbf{x}_j^\mathcal{F}\right)+\sum\nolimits_{j=1}^{n_\mathcal{B}}\alpha_j^\mathcal{B}\cdot p_i\left(\mathbf{x}_j^\mathcal{B}\right)=0,\ i=1,\ldots,l. \quad (6)$$

In the case of linear polynomial space, (6) can be rewritten as the following $d + 1$ equations in matrix form:

$$\begin{pmatrix} \mathbf{e}_\mathcal{F}^T & \mathbf{e}_\mathcal{B}^T \\ \mathbf{X}_\mathcal{F} & \mathbf{X}_\mathcal{B} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_\mathcal{F} \\ \boldsymbol{\alpha}_\mathcal{B} \end{pmatrix} = \mathbf{0}. \quad (7)$$

Combining (5) and (7) together, it follows:

$$\begin{pmatrix} \mathbf{K}_{\mathcal{F}\mathcal{F}} & \mathbf{K}_{\mathcal{F}\mathcal{B}} & \mathbf{e}_\mathcal{F} & \mathbf{X}_\mathcal{F}^T \\ \mathbf{K}_{\mathcal{B}\mathcal{F}} & \mathbf{K}_{\mathcal{B}\mathcal{B}} & \mathbf{e}_\mathcal{B} & \mathbf{X}_\mathcal{B}^T \\ \mathbf{e}_\mathcal{F}^T & \mathbf{e}_\mathcal{B}^T & 0 & \mathbf{0} \\ \mathbf{X}_\mathcal{F} & \mathbf{X}_\mathcal{B} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_\mathcal{F} \\ \boldsymbol{\alpha}_\mathcal{B} \\ \beta_0 \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_\mathcal{F} \\ -\mathbf{e}_\mathcal{B} \\ 0 \\ \mathbf{0} \end{pmatrix}. \quad (8)$$

In the regularized case, $\mathbf{K}_{\mathcal{F}\mathcal{F}}$ and $\mathbf{K}_{\mathcal{B}\mathcal{B}}$ should be replaced by $\mathbf{K}_{\mathcal{F}\mathcal{F}} + \lambda\mathbf{I}_\mathcal{F}$ and $\mathbf{K}_{\mathcal{B}\mathcal{B}} + \lambda\mathbf{I}_\mathcal{B}$, where $\mathbf{I}_\mathcal{F} \in \mathbb{R}^{n_\mathcal{F}\times n_\mathcal{F}}$ and $\mathbf{I}_\mathcal{B} \in \mathbb{R}^{n_\mathcal{B}\times n_\mathcal{B}}$ are two identity matrices. The regularization parameter λ controls the amount of smoothness of the spline near

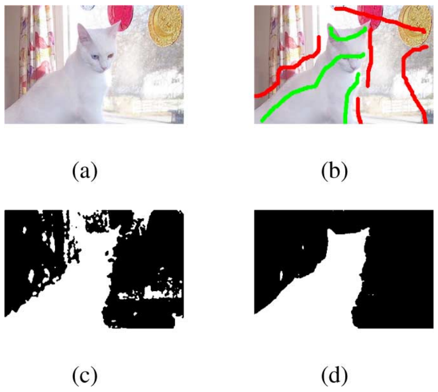


Fig. 2. Segmentations of the cat with different features. (a) source image; (b) the user specified strokes; (c) the segmentation by SR with color features; (d) the segmentation by SR with color features and spatial coordinates.

the scattered data points. In the limiting case, namely, $\lambda = 0$, these scattered data points will be exactly mapped. Here we select $\lambda > 0$ to avoid over-fitting.

Finally, the regression values of the unlabeled pixels can be directly obtained via (4). Since we use "+1" as the interpolation value of each foreground pixel and "−1" as that of each background pixel, the median value "zero" can be taken as a classification threshold. Thus, for each pixel p_i, its class label l_i can be assigned as follows:

$$l_i = \begin{cases} \text{Foreground}, & \text{if } f(\mathbf{x}_i) \geq 0 \\ \text{Background}, & \text{if } f(\mathbf{x}_i) < 0. \end{cases} \tag{9}$$

C. Feature Vectors of Pixels

Here, each pixel p_i is described as a 5-D feature vector, i.e., $\mathbf{x}_i = [r, g, b, x, y]^T$, in which (r, g, b) is the normalized color of pixel p_i and (x, y) is its spatial coordinate normalized with image width and height. The reason that we consider the spatial coordinates is that the discrimination between pixels with similar colors can be enhanced, especially when the foreground object and its background contain exactly identical colors. Fig. 2 shows an example. Our task is to cut out the white cat near the window. There exists similar white color in the foreground and background. Fixing the user specified strokes as shown in Fig. 2(b), the segmentation by SR with (r, g, b) is shown in Fig. 2(c), while that with (r, g, b, x, y) is illustrated in Fig. 2(d). We see that the performance with the normalized spatial coordinates is significantly improved.

Besides color, texture is another important image feature. There exist many texture descriptors, among which Gabor-based texture descriptors are popularly used in image segmentation [35]. Here the Gabor filter bank [15] is employed to describe the image to be segmented. Fig. 3 reports the segmentations of the leopard and the cat, with the user specified strokes in Figs. 1 and 2, respectively. The upper and lower bounder of interesting frequencies of Gabor filter bank are taken as 0.4 and 0.05, which indicates that a large frequency interval is considered. Other three parameters are the scale number (s) and orientation number (o) of the filter bank, and the filter size $w \times w$ in pixels. In Fig. 3, from the first to the last column are the results with $(s, o, w) = (3, 4, 9), (3, 4, 15), (3, 4, 21), (4, 6, 9,), (4, 6, 15)$, and $(4, 6, 21)$, respectively. To utilize the color information, three bit planes of red, green and blue are respectively filtered with this bank. In the case of $s = 3$ and $o = 4$, for example, the final feature dimensionality will equal to $36 (= 3 \times 12)$. To reduce the redundant information, principal component analysis [11] is employed to project the features into 10-D subspace (Note that worse results are obtained when we directly use the source features). Finally, the spline in Section II-A is constructed to segment the image.

Fig. 3 indicates that no better results are obtained with texture features. Actually, texture is perceptible only via spatial regions. If a texture pattern is not labeled by the user, it would be incorrectly segmented. However, different textures may have similar colors (see the scene outside the window in Fig. 2). Even if similar colors are contained in foreground and background, they may be segmented from each other by introducing spatial coordinates [see Fig. 2(d)]. Thus, "$\text{color} + \text{coordinate}$" is adequate to obtain satisfactory results for most natural images. This may be one reason why seldom interaction frameworks are developed with textures.

D. Clustering the User Specified Pixels

In the case that (r, g, b, x, y) are pixel features, in total, there are $6 + n_{\mathcal{F}} + n_{\mathcal{B}}$ parameters in (8) to be determined. Generally, there are thousands of pixels which may be scribbled by the user. Accordingly, we should to solve the linear equations with a large number of unknown parameters. Since the coefficient matrix in (8) is a dense matrix, the computation complexity of solving the linear equations will be up to about $O((6 + n_{\mathcal{F}} + n_{\mathcal{B}})^3)$. However, in most cases, the foreground object and its background only consists of a few number of different colors. Thus, we can cluster the user specified foreground and background pixels, and employ the cluster centers as their representatives. This idea is not new. Actually, in lazy snapping [14], K-means algorithm is used to cluster the user specified pixels, while in Grabcut [29] Gaussian mixture model is used to estimate the probabilistic density functions. Here, using cluster centers will significantly reduce the number of the unknown parameters in (8). Thus, the computation time can be largely saved. For example, in a PC with 2.4-G CPU and 1-G RAM, solving 126 parameters from linear equations in Matlab only needs about 0.002 s, while solving 2000 parameters may need about 2.7 s.

Specifically, we employ K-means algorithm to cluster the data points in $\mathcal{U}_{\mathcal{F}} = \{\mathbf{x}_i^{\mathcal{F}}\}_{i=1}^{n_{\mathcal{F}}}$ and $\mathcal{U}_{\mathcal{B}} = \{\mathbf{x}_i^{\mathcal{B}}\}_{i=1}^{n_{\mathcal{B}}}$. Let k be the number of clusters. After performing K-means algorithm, the data points in $\mathcal{U}_{\mathcal{F}}$ and $\mathcal{U}_{\mathcal{B}}$ will be respectively replaced by the k cluster centers. In this way, there are only $2k + 6$ parameters in (8) to be solved. The results in Figs. 1 and 2 are obtained with $k = 32$. Thus, we only need to solve 70 parameters from the linear equations.

E. Spatial Neighborhood Assignment

In our spline regression framework, once the spline is constructed, it can be used to map the unlabeled pixels one by one. Thus, the spatial relationship between pixels on the image grid will be simply ignored. To utilize the spatial structure of the

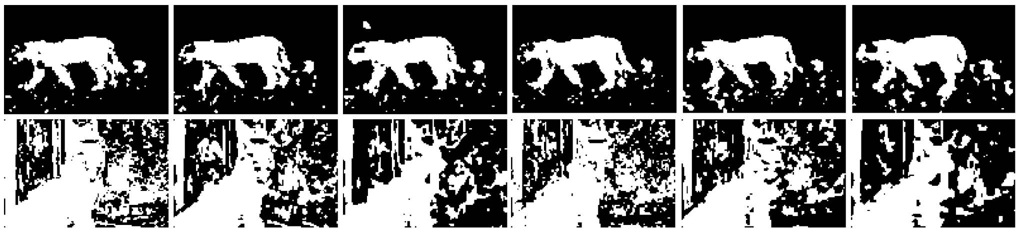


Fig. 3. Segmentations of the leopard and the cat with Gabor filter bank in different scales, orientations, and filter sizes.

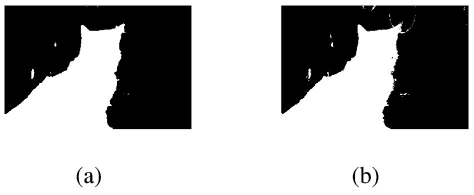


Fig. 4. (a) Segmentation of the white cat with SR, by performing the spatial neighborhood assignment; (b) the segmentation without performing the spatial neighborhood assignment.

image, we assign the regressed value of each pixel to its neighbors. In computation, the 3×3 neighborhood will be considered. For example, suppose pixel p_i is located in the rth row and cth column of the image. Then we assign its regressed value $f(\mathbf{x}_i)$ to its eight neighbors $(r-1, c-1)$, $(r-1, c)$, $(r-1, c+1)$, $(r, c-1)$, $(r, c+1)$, $(r+1, c-1)$, $(r+1, c)$, $(r+1, c+1)$ and itself (r, c). These values will be accumulated into the buffer and the results will be finally averaged at each of the pixels.

We take the cat image as an example. The segmentation result via the above spatial neighborhood assignment is shown in Fig. 4(a), which is identical to that shown in Fig. 2(d). For comparison, the result without performing this assignment is illustrated in Fig. 4(b). Clearly, in Fig. 4(b), more isolated pixels appear in the segmentation. Note that such isolated pixels may not be simply removed by the commonly used morphological approaches or median filters on image windows.

F. The Algorithm

Now we can give our algorithm of spline regression for interactive image segmentation in **Algorithm 1**. It has one parameter, namely, the number of the clusters k used in K-means algorithm to cluster the user specified foreground and background pixels. If $k \leq 0$, then the K-means clustering step will be simply skipped.

Note that there is a regularization parameter λ used in spline construction. It will be demonstrated in Section V that it is not sensitive in (0, 0.01]. Thus, we do not treat it as an important parameter and fix it to be 0.0001 in this algorithm.

It is worthy pointing out that this algorithm can be easily implemented. It can be run with only about twenty Matlab sentences, among which there are only a few sentences related to the core computations, namely, constructing and solving the linear system in (8) and mapping the pixels to be segmented via (4). Meanwhile, the most complex computation is to solve the linear equations in (8). Except these, there are no other complex calculations.

Algorithm 1 Spline Regression for Interactive Natural Image Segmentation

```
Input: The image $\mathcal{I}$ with $n$ pixels $\{p_i\}_{i=1}^n$ to be segmented;
the user specified stokes about the foreground object and
its background, $\mathcal{F}$ and $\mathcal{B}$; the number of clusters $k$ for
clustering $\mathcal{F}$ and $\mathcal{B}$.

Output: The segmentation of $\mathcal{I}$.

1: Construct the feature vector set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, in which
   $\mathbf{x}_i = [r, g, b, x, y]^T$ corresponds to the feature vector
   of pixel $p_i$.
2: Construct two subsets of feature vectors according
   to the user specified strokes about the foreground
   object and its background: $\mathcal{U}_\mathcal{F} = \{\mathbf{x}_i^\mathcal{F}\}_{i=1}^{n_\mathcal{F}} (\subset \mathcal{X})$ and
   $\mathcal{U}_\mathcal{B} = \{\mathbf{x}_i^\mathcal{B}\}_{i=1}^{n_\mathcal{B}} (\subset \mathcal{X})$.
3: if $k > 0$ then
4:    Cluster $\mathcal{U}_\mathcal{F}$ with K-means clustering algorithm,
      replace $\mathcal{U}_\mathcal{F}$ by the $k$ cluster centers, and let $n_\mathcal{F} \leftarrow k$.
5:    Cluster $\mathcal{U}_\mathcal{B}$ with K-means clustering algorithm,
      replace $\mathcal{U}_\mathcal{B}$ by the $k$ cluster centers, and let $n_\mathcal{B} \leftarrow k$.
6: end if
7: Construct the spline in (4) based on $\mathcal{U}_\mathcal{F}$ and $\mathcal{U}_\mathcal{B}$ and
   solve the linear equations in (8).
8: Allocate an array $S$ with $n$ zero elements.
9: for each pixel $p_i$, $i = 1, \ldots, n$, do
10:    Calculate the spline regression value $f(\mathbf{x}_i)$ with (4).
11:    Accumulate $f(\mathbf{x}_i)$ to $S[i]$: $S[i] \leftarrow S[i] + f(\mathbf{x}_i)$.
12:    Accumulate $f(\mathbf{x}_i)$ to the eight neighbors of the $i$th
       pixel and record them in the corresponding elements
       of $S$.
13: end for
14: for each pixel $p_i$, $i = 1, \ldots, n$, do
15:    Average $S[i]$, namely, $S[i] \leftarrow S[i]/9$.
16:    Assign class label via (9). Here, if $s[i] \geq 0$,
       then $s[i] \leftarrow 255$ (Foreground); otherwise, $s[i] \leftarrow 0$
       (Background).
17: end for
18: Output the binarized image $S$ by reshaping it to be an
    image with the same size of source image $\mathcal{I}$.
```

III. Connections to Other Algorithms

The regularization framework we use to develop the spline can be written in the following general form:

$$J(f) = \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2 + \lambda S(f) \tag{10}$$

where $\mathbf{x}_i$, $i = 1, 2, \ldots, m$, are m scattered data points and y_i, $i = 1, 2, \ldots, m$, are their objective values. Defining $S(f)$ as a semi-norm, we get the spline with form in (3).

Now we consider the linear function by introducing a projection vector $\mathbf{W} \in \mathbb{R}^d$

$$f(\mathbf{x}) = \mathbf{W}^T\mathbf{x} \tag{11}$$

and define $S(f) = \mathrm{tr}(\mathbf{W}^T\mathbf{W})$. It has been shown that this formulation is equivalent to the regularized LDA [41]. In the case of $\lambda = 0$, this model reduces to the standard linear regression (LR).

Previous work also shows that for the centralized data points LR is equivalent to the classic LDA [8], [39]. This constructs a bridge between LR and LDA. In real computation, we can add the translation term in linear regression

$$f(\mathbf{x}) = \beta_0 + \mathbf{W}^T\mathbf{x} \tag{12}$$

where β_0 is a scalar. β_0 and $\mathbf{W}$ can be solved via the following linear equations:

$$\begin{pmatrix} \mathbf{e}_{\mathcal{F}} & \mathbf{X}_{\mathcal{F}}^T \\ \mathbf{e}_{\mathcal{B}} & \mathbf{X}_{\mathcal{B}}^T \end{pmatrix} \begin{pmatrix} \beta_0 \\ \mathbf{W} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_{\mathcal{F}} \\ -\mathbf{e}_{\mathcal{B}} \end{pmatrix}. \tag{13}$$

Usually, the above problem is over-determined $(n_{\mathcal{F}} + n_{\mathcal{B}} > d + 1)$. In this case the pseudo-inverse matrix of the coefficient matrix can be employed to solve the above equations.

Then, we consider to define $S(f)$ as a norm in general reproducing kernel Hilbert space [31]. Thus, we have

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) \tag{14}$$

where α_i, $i = 1, 2, \ldots, m$, are the parameters to be determined. Note that the model in (14) is also named as kernel ridge regression (KRR) [31], and $k(\cdot,\cdot)$ is usually supplied as a Gaussian kernel

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/\sigma^2). \tag{15}$$

Taking the regularization parameter λ into account, the parameter α_i can be solved via the following linear equations:

$$(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{Y} \tag{16}$$

where $\mathbf{K}$ is an $m \times m$ matrix with element $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{I}$ is an $m \times m$ identity matrix, $\mathbf{Y}$ collects the objective functions, $\mathbf{Y} = [y_1, \ldots, y_m]^T \in \mathbb{R}^m$, and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T \in \mathbb{R}^m$.

The spline we use can be viewed as a combination of linear and kernel (or radial basic) functions (although the kernel forms are different). In contrast to Gaussian kernel functions, it is worthy pointing out that there is no kernel parameter σ in this spline. Furthermore, it has clearly geometrical property [3], that is, the m scattered data points are first mapped globally near to right positions with the translation and affine transform, and then dragged to the right positions with local kernel functions.

Both LP and RW can be explained in view of label inference on data graph. With Laplacian regularization on graph, $S(f)$ can be defined as $S(f) = \mathrm{tr}(\mathbf{Y}^T\mathbf{L}\mathbf{Y})$ in which $\mathbf{L}$ is a Laplacian matrix. Differing from SVM, LDA, LR and SR, here $\mathbf{Y}$ collects not only the values of the labeled data points, but also those of the unlabeled data points. Thus, it is developed in a transductive learning setting. It can be solved via the following unconstrained quadratic programming (QP): $\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda \cdot \mathrm{tr}(\mathbf{Y}^T\mathbf{L}\mathbf{Y})$.

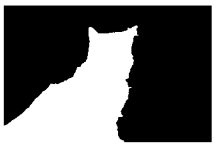


Fig. 5. Segmentation of the white cat by discarding small blobs via connectivity analysis. This result is obtained from the segmentation in Fig. 2(d) with an additional step of connectivity analysis.

With regularization on functional space or graph, SVM and GC on Markov random field (MRF) can also be cast on a regularization framework. QP (with linear constraints) is used to solve both SVM and GC on MRF [12]. However, GC on MRF is much faster than SVM.

IV. Postprocessing for Image Segmentation

In this section, three postprocessing methods will be introduced for user to obtain better segmentation. They are *connectivity analysis*, *edge fairing*, and *segmenting with more strokes*. In our framework, whether to perform these three steps will be decided by the user.

A. Connectivity Analysis

In segmentation, there may exist some small blobs which are incorrectly segmented [see Fig. 2(d)]. These blobs can be removed by performing connectivity analysis. That is, if the area of the blob is small, it can be discarded. For example, for the segmentation in Fig. 2(d), all the blobs with area less than 80 pixels will be deleted. Fig. 5 shows the cleared result. This step is considered in the design of software system in which the user can decide whether it will be performed or not.

B. Edge Smoothing

The object edge extracted from the binary image of segmentation may be not smooth. There are many curve fairing algorithms which can be employed to smooth the segmented edge [21], [23], [45]. Here, the algorithm based on discrete energy minimization for free-formed curves is used [45]. Fig. 6 shows an example. Based on the segmented binary image, the edge points of the cat is first extracted by tracking along the edge pixels. Then they are reduced uniformly by taking one third of all the edge pixels. In Fig. 6(a), the full contour is smoothed by discrete energy minimization approach. A drawback is that some corners which we hope to maintain may be rounded. For example, the corner in the bottom-left region is replaced by a rounded corner. This treatment can be avoided by only considering the user selected edge to be smoothed. In Fig. 6(b) three segments of the user selected edges are respectively smoothed and shown on the segmented image. As can be seen, the selected edges are all smoothed.

C. Segmenting With More Strokes

One advantage in an interactive segmentation setting is that the user can act as a judge to decide whether the current segmentation is acceptable or not. If the segmentation is not satisfactory, more strokes can be added through the user-computer

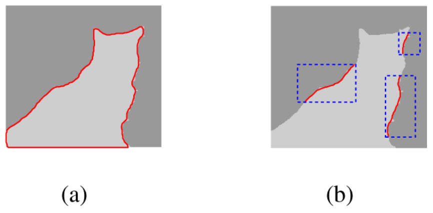


Fig. 6. Edge smoothing. (a) The smoothed whole edge and (b) the smoothed edges in the rectangle regions selected by the user.

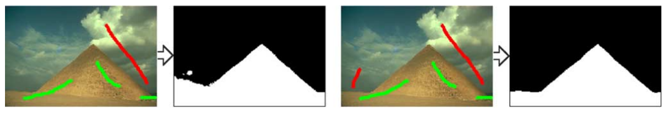


Fig. 7. Segmentations of the pyramid by SR with different strokes.

interaction interface. Fig. 7 shows an example. We see that with strokes as illustrated in the first image in Fig. 7, a part of the background is incorrectly segmented to be the pyramid (see the second image). To remove this background, a stroke is scratched on this background, as illustrated in the third image in Fig. 7. Now the background is successfully removed (see the fourth image). Note that when we use SR to segment the image once again, the old and new strokes are all employed. That is, we just re-run the steps of **Algorithm 1** and do not consider the previously-obtained results.

V. Experiments

In this section, we first evaluate the performance of spline regression (SR) algorithm for interactive image segmentation. Then we compare our algorithm with other algorithms.

The images we used are downloaded from the Berkeley segmentation dataset [16], the Grabcut database [29], and the visual object classes database of 2007 (Available at: http://www.pascal-network.org/challenges/voc/voc2007).

A. Performances of Spline Regression

One parameter in our algorithm is the number of the clusters, k, which is supplied to cluster the user specified foreground and background pixels. Fig. 8 shows the segmentations of the leopard and the pyramid images, with different k. From the first to the last column are the results obtained with $k = 3, 5, 10, 20, 30, 40, 50, 60$, respectively. For the leopard image, the user specified strokes in Fig. 1 are used, while for the pyramid image, the strokes in the third image in Fig. 7 are employed. To clearly show the initial results obtained by SR, we do not perform the postprocessing methods as introduced in Section IV. We see that the leopard can be cut out with k greater than 30. Actually, there exist similar foreground and background colors, and, thus, a small number of clusters are not enough to describe the difference between the foreground and background. In contrast, the segmentations of the pyramid indicate that when k is greater than 5, satisfactory results can be generated by SR. In fact, there are only a few colors in foreground and background regions and the colors of the pyramid and those of the sky are different from each other. We also tested other images in the databases. For example, the results in Fig. 2(d) is obtained by taking $k = 32$. As a conclusion, we recommend that k can be set to be about 30–60 in real applications.

Another parameter in SR algorithm is the regularization parameter λ. Experiments show that it is insensitive if it locates in (0, 0.01]. Fig. 9 illustrates an example, in which we fix k to be 40. From the left to the right are the segmented results with $\lambda = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$, and 10^{-2}, respectively. The user specified strokes are those used in Fig. 8. We see there are no significant changes between the results when we take different λ. Such an observation can also be obtained from other images. As pointed out in Section II-B, a larger λ indicates that the constructed spline will be more smooth, while a smaller λ indicates the given interpolation values of the scattered data points are more exactly satisfied. In interactive segmentation settings, the colors of the unlabeled pixels in general are similar to those of the user specified foreground or background. Thus, in real applications we can take a small λ. We fix λ to be 0.0001 in all experiments, and do not treat it as an important input parameter in **Algorithm 1**.

The computation complexity of SR algorithm can be given as follows. Given the number of the foreground strokes $n_{\mathcal{F}}$, the number of background strokes $n_{\mathcal{B}}$, and the dimensionality d of the feature vectors, the complexity of computing the coefficient matrix in (8) will be up to about $O(d\times(n_{\mathcal{F}}+n_{\mathcal{B}})+(n_{\mathcal{F}}+n_{\mathcal{B}})^2)$. The computational complexity of solving the linear equations in (8) is about $O((1+d+n_{\mathcal{F}}+n_{\mathcal{B}})^3)$. That of using (4) to map n pixels is about $O(n\times(1+d+n_{\mathcal{F}}+n_{\mathcal{B}}))$. In our SR algorithm, d equals to five. Thus, the computational complexity will largely depend on $n_{\mathcal{F}}$, $n_{\mathcal{B}}$, and n. Furthermore, when $n_{\mathcal{F}}$ and $n_{\mathcal{B}}$ are further decreased to be the number of the clusters, namely, $n_{\mathcal{F}} \leftarrow k$ and $n_{\mathcal{B}} \leftarrow k$, and when k is set to be about 30–60, then the computation scale will be largely determined by the number of the pixels to be segmented. The computational complexity is linear in n. Meanwhile, when n is fixed, the computation complexity is also linear in $n_{\mathcal{F}}+n_{\mathcal{B}}$. For the image with 337×225 pixels (e.g., Fig. 5), with $k = 3, 5, 10, 20, 30, 40, 50, 60$, on average SR will take about 2.30, 2.41, 2.65, 3.10, 3.52, 3.99, 4.42, and 4.85 s, respectively on a PC with 2.4-G CPU and 1.0-G RAM, using Matlab 7.0.

B. Comparisons With Other Algorithms

1) Details of Algorithms: In this section, we will compare our algorithm of SR with classical LDA, SVM, LP, RW, GC, LR, and KRR.

The K-means clustering algorithm is first performed to cluster the user specified foreground pixels and background pixels. Then the k cluster centers of the foreground are labeled as positive samples and the k cluster centers of the background are labeled as negative samples. These $2k$ samples will be trained by LDA and SVM. They will also be used in LR, KRR and SR to construct the linear, kernel and spline functions. In all experiments, k will be set to be 50.

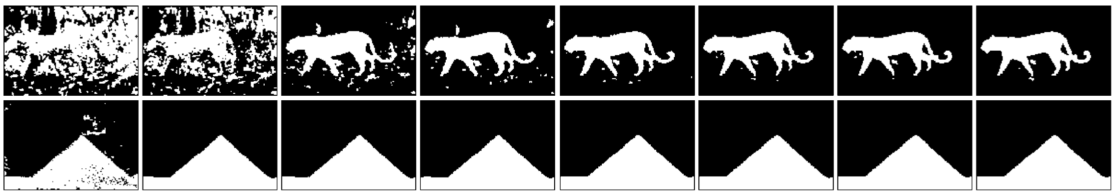


Fig. 8. Segmentations by SR with different k. From the first to the last column are the results with $k = 3, 5, 10, 20, 30, 40, 50, 60$, respectively.

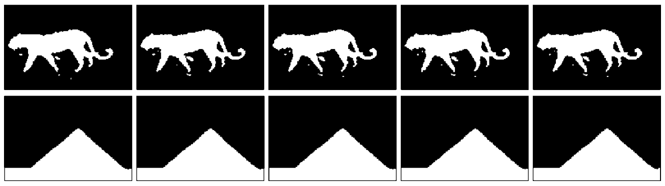


Fig. 9. Segmentations by SR with different λ. From the first to the last column are the results with $\lambda = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$ and 10^{-2}, respectively.

Since LDA is only a subspace learning algorithm, we use 1-NN as classifier to classify the pixels to be segmented. In addition, the kernel machine of SVM with Gaussian kernel function is employed in computation. The parameter σ (15) is evaluated as the median distance of all the distances between all the pair points among the $2k$ cluster centers. In experiments, the Matlab tool of *osu-svm* is employed (Available at: http://sourceforge.net/projects/svm/), in which we set the regularization parameter c to be 100.0 (Note that if c is very small, poor results may be generated).

LR and KRR follow the same steps as described in **Algorithm 1**. Differently, in LR (12) will be used and the parameters β_0 and $\mathbf{W}$ will be solved via (13). In KRR, (14) will be employed and the parameter vector α will be determined via (16). The parameter σ is also calculated as the median distance between the training data points.

In LP, the neighborhood of each pixel is defined as a 5×5 sub-window with its center at the pixel. Then Laplacian matrix is constructed. In experiments, the parameter α in [43] is set to be 0.99 and the number of iterations is fixed to be 50.

For RW, we downloaded the source codes from the author's homepage [10] and directly ran them for image segmentation.

In GC, the algorithm in [4] is implemented. We downloaded the source codes of graph cut from the homepage of the author and ran them for image segmentation. Differently, the data likelihoods of R_p("bkg") and R_p("obj") in [4] is calculated by (2) in [14], in which K-means clusters are employed.

All the above algorithms will be run on the same user specified strokes. In addition, to make fair comparisons, we do not perform the postprocessing methods in Section IV. That is, only **Algorithm 1** is run in experiments.

2) Comparisons: Figs. 10 and 11 illustrate the segmentations of different types of natural images. For comparison, the segmented white foreground object regions are replaced by the source visual objects. From the segmentations, we see that our method outperforms the commonly used LDA and SVM. Actually, for complex images where the foreground and background contain similar colors, LDA may fail to generate satisfactory segmentations. As a linear method, the solution of the LDA is optimal under the condition that the data is linearly separable. Thus, LDA will perform well in the situations where the data distribution of each class is Gaussian. For example, we see that LDA (and with 1-NN classifier) can roughly cut out the flower and the starfish in Fig. 10, but fails in other complex images. In contrast to LDA, kernelized SVM can generate better results. To this end, the kernel parameter σ should be well tuned to data. Our algorithm outperforms LR[1] and KRR. Actually, in function form, the spline we use is a combination of linear and Green's functions. In terms of geometrical transformation, the linear part captures the global affine transformation, while the nonlinear part captures the local nonlinear deformations [3]. Thus, in real applications, spline shows more adaptability to diverse natural images.

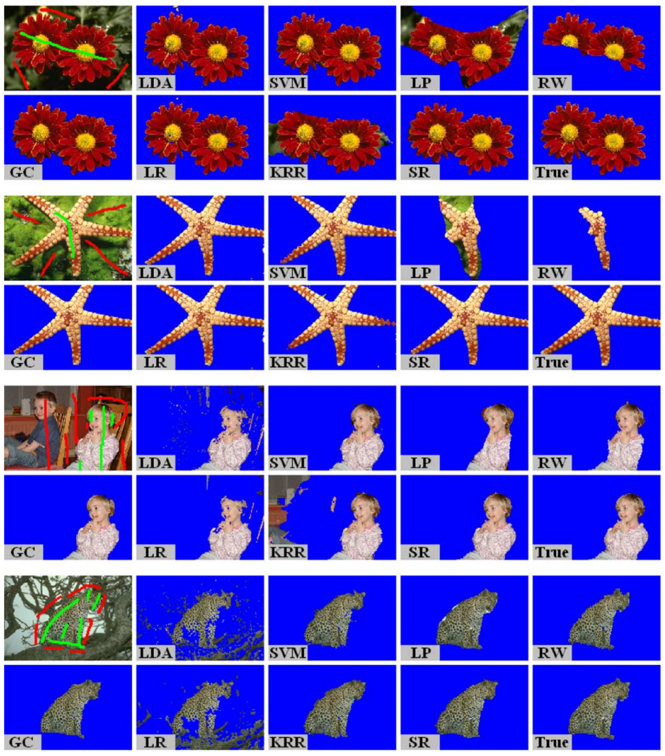


Fig. 10. Demo I: segmentations with LDA, SVM, LP, RW, GC, LR, KRR, and our algorithm SR.

Our algorithm also outperforms LP. In LP, the data points receive the class label information from their neighbors on the data graph. If there exists a region in which no points are labeled, it

[1]As mentioned in Section III, LR and LDA are equivalent to each other. Actually, the segmentations by these two algorithms are very similar to each other. There exist differences since in LR (9) is used to assign the class labels, while in LDA the commonly used 1-NN classifier is employed.

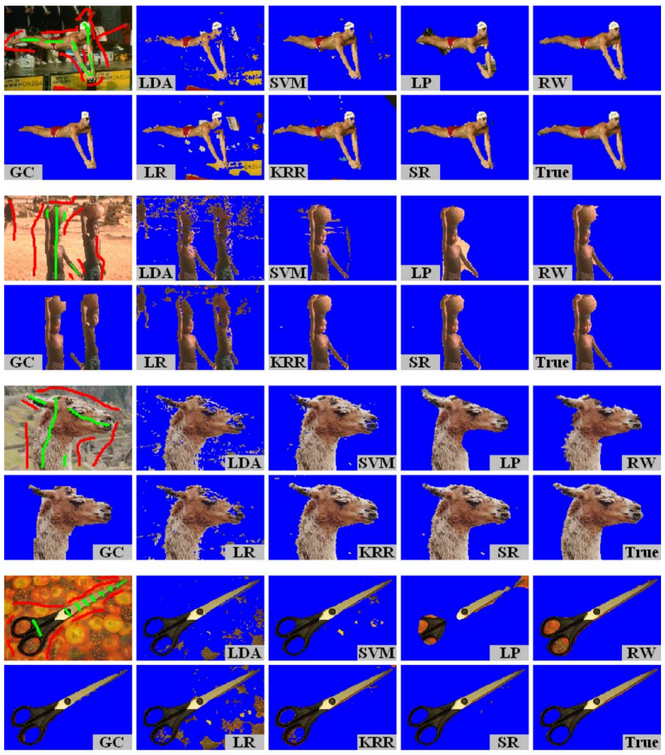


Fig. 11. Demo II: segmentations with LDA, SVM, LP, RW, GC, LR, KRR, and our algorithm SR.

may be incorrectly segmented. This can be observed in the segmentations in Figs. 10 and 11. As another graph-based method, RW generates better results, compared with LP. However, the performances of RW and LP are very similar to each other. This may be explained by the relations between random walks and spectral segmentation [17].

Our algorithm can also be comparable to the most successful algorithm of GC. Generally, the segmentations by GC are satisfactory, except that in the second image in Fig. 11. In contrast, our algorithm generates more accurate segmentation. This can be witnessed near the head of the baby and the ears of the leopard in Fig. 10, and near the head of the goat and the scissors in Fig. 11.

To further compare these algorithms, Table I gives a quantitative comparison between the algorithms. The number in Table I stands for the classification rate, which is calculated as the ratio (percent) of the number of correctly classified pixels to that of the total pixels in the image. Here the correctly classified pixels are identified by taking the ground truth as a reference. The first column in Table I indicates the indices of the images in Figs. 10 and 11 (the size of each image is 337 $\times$ 225 pixels). In contrast, in each segmentation our algorithm achieves the highest accuracy.

A main drawback of our algorithm is that after segmentation, there may still exist some small blobs which are incorrectly segmented. For example, the segmentations in Fig. 11 contain some incorrect blobs. These blobs could be deleted via connectivity analysis and area statistics.

In computation time, in the case of $k = 50$, and for images with 337 $\times$ 225 pixels, LDA, SVM, LR, KRR, and SR will take

TABLE I
CLASSIFICATION RATES OF THE EIGHT IMAGES IN FIGS. 10 AND 11, ORDERED WITH INDICES IN THE FIRST COLUMN

	LDA	SVM	LP	RW	GC	LR	KRR	SR
1	98.4	98.5	79.4	83.1	98.2	98.4	90.8	**98.5**
2	98.3	98.0	79.8	80.8	98.4	98.4	97.1	**98.6**
3	93.6	99.1	96.0	98.0	99.2	96.3	81.6	**99.5**
4	82.5	98.1	98.6	98.2	98.2	80.8	98.4	**98.7**
5	90.7	97.7	95.1	98.5	97.8	88.0	97.2	**98.6**
6	90.1	97.6	93.1	94.7	95.9	91.8	98.1	**98.8**
7	90.4	97.7	91.9	95.4	98.2	85.2	97.8	**98.7**
8	77.2	97.8	97.2	99.2	88.1	81.4	99.2	**99.3**

about 21.70, 1.22, 0.04, 3.38, and 3.99 s, respectively, on a PC with 2.4-G CPU and 1.0-G RAM, using Matlab 7.0. For the same input and also with Matlab setting, LP will take 207.81 s. We run GC in C++ setting. On average, it will take about 0.1 s. In C++ setting, on average SR will take about 1.1 s.

VI. CONCLUSION

In this paper, we proposed a novel algorithm for natural image segmentation. We formulated the task as a problem of spline regression. The spline is a combination of linear and Green's functions, with adaptability to diverse natural images. We also analyzed the connections of our spline regression algorithm to other algorithms, including those developed in inductive learning setting, transductive learning setting, regularization on graph and general functional spaces. Comparative experiments illustrate the validity of our method.

REFERENCES

[1] W. A. Barrett and A. S. Cheney, "Object-based image editing," in *Proc. SIGGRAPH*, San Antonio, TX, 2002, pp. 777–784.
[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive gmmrf model," in *Proc. European Conf. Computer Vision*, Prague, Czech Republic, 2004, pp. 428–441.
[3] F. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989.
[4] Y. Boykov and M. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *Proc. Int. Conf. Computer Vision*, Vancouver, BC, Canada, 2001, pp. 105–112.
[5] H. Cheng, X. Jiang, Y. Sun, and J. Wang, "Color image segmentation: Advances and prospects," *Pattern Recognit.*, vol. 34, no. 12, pp. 2259–2281, 2001.
[6] Y. Y. Chuang, B. Curless, and D. S. abd Richard Szeliski, "A bayesian approach to digital matting," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 264–271.
[7] J. Duchon, , A. Dold and B. Eckmann, Eds., "Splines minimizing rotation-invariant semi-norms in sobolev spaces," in *Constructive Theory of Functions of Several Variables*. New York: Springer-Verlag, 1977, pp. 85–100.
[8] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
[9] M. Gleicher, "Image snapping," in *Proc. SIGGRAPH*, Los Angeles, CA, 1995, pp. 183–190.
[10] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, Nov. 2006.
[11] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York: Springer, 2002.

[12] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Solving markov random fields using second order cone programming relaxations," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, New York, 2006, pp. 1045–1052.

[13] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 1–15, Jan. 2008.

[14] Y. Li, J. Sun, C. Tang, and H. Shum, "Lazy snapping," in *Proc. SIGGRAPH*, Los Angeles, CA, 2004, pp. 303–307.

[15] B. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837–842, Aug. 1996.

[16] D. R. 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. IEEE Int. Conf. Computer Vision*, Vancouver, BC, Canada, 2001, pp. 416–425.

[17] M. Meila and J. Shi, "A random walks view of spectral segmentation," presented at the 8th Int. Workshop on Artificial Intelligence and Statistics, Key West, FL, 2001.

[18] J. Meinguet, "Multivariate interpolation at arbitrary points made simple," *J. Appl. Math. Phys.*, vol. 30, 1979.

[19] E. Mortensen and W. Barrett, "Intelligent scissors for image composition," in *Proc. SIGGRAPH*, Los Angeles, CA, 1995, pp. 191–198.

[20] E. Mortensen and W. Barrett, "Toboggan-based intelligent scissors with a four-parameter edge model," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Fort Collins, CO, 1999, vol. 2, pp. 452–458.

[21] G. Mullineux and S. T. Robinson, "Fairing point sets using curvature," *Comput.-Aided Des.*, vol. 39, no. 1, pp. 27–34, 2007.

[22] A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science*. Berlin, Germany: Springer-Verlag, 1982.

[23] H. Nowacki, "Curve and surface generation and fairing," *Comput.-Aided Des.*, vol. 89, 1980.

[24] S. Olabarriaga and A. Smeulders, "Interaction in the segmentation of medical images: A survey," *Med. Imag. Anal.*, vol. 5, no. 2, pp. 127–142, 2001.

[25] N. Pal and S. Pal, "A review on image segmentation techniques," *Pattern Recognit.*, vol. 26, no. 9, pp. 1277–1294, 1993.

[26] P. Perez, A. Blake, and M. Gangnet, "Jetstream: Probabilistic contour extraction with particles," in *Proc. Int. Conf. Computer Vision*, Vancouver, BC, Canada, 2001, vol. 2, pp. 524–531.

[27] A. Protiere and G. Sapiro, "Interactive image segmentation via adaptive weighted distances," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1046–1052, Apr. 2008.

[28] L. J. Reese and W. A. Barrett, "Image editing with intelligent paint," in *Proc. Eurographics*, Saarbrucken, Germany, 2002, pp. 714–724.

[29] C. Rothera, V. Kolmogorov, and A. Blake, ""grabcut"—Interactive foreground extraction using iterated graph cuts," in *Proc. SIGGRAPH*, Los Angeles, CA, 2004, pp. 309–314.

[30] M. A. Ruzon and C. Tomasi, "Alpha estimation in natural images," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Hilton Head, SC, 2000, vol. 1, pp. 18–25.

[31] B. Scholkopf and A. J. Smola, *Learning With Kernels.*. Cambridge, MA: MIT Press, 2002.

[32] J. Sun, J. Jia, C. Tang, and H. Shum, "Poisson matting," in *Proc. SIGGRAPH*, Los Angeles, CA, 2004, pp. 315–321.

[33] J. Sun, L. Yuan, J. Jia, and H. Shum, "Image completion with structure propagation," in *Proc. SIGGRAPH*, Los Angeles, CA, 2005, pp. 861–868.

[34] K.-H. Tan and N. Ahuja, "Selecting objects with freehand sketches," in *Proc. Int. Conf. Computer Vision*, Vancouver, BC, Canada, 2001, vol. 1, pp. 337–344.

[35] M. Tuceryan and A. Jain, , C. Chen, L. Pau, and P. Wang, Eds., "Texture analysis," in *The Handbook of Pattern Recognition and Computer Vision*, 2nd ed. Singapore: World Scientific, 1998, pp. 207–248.

[36] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer Verlag, 1995.

[37] G. Wahba, *Spline Models for Observational Data*. Philadelphia, PA: SIAM, 1990.

[38] J. Wang and M. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *Proc. Int. Conf. Computer Vision*, Beijing, China, 2005, pp. 936–943.

[39] J. P. Ye, "Least squares linear discriminant analysis," in *Proc. Int. Conf. Machine Learning*, Corvallis, OR, 2007, pp. 1087–1094.

[40] J. Yoon, "Spectral approximation orders of radial basis function interpolation on the sobolev space," *SIAM J. Math. Anal.*, vol. 33, no. 4, pp. 946–958, 2001.

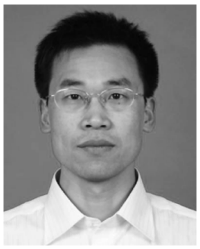
[41] P. Zhang and N. Riedel, "Discriminant analysis: A unified approach," in *Proc. Int. Conf. Data Minging*, New Orleans, LA, 2005, pp. 454–461.

[42] Y. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognit.*, vol. 29, no. 8, pp. 1335–1346, 1996.

[43] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," *Adv. Neural Inf. Process. Syst. 16* 2003.

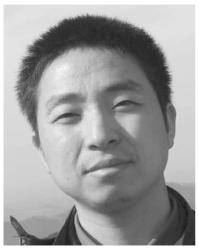
[44] X. J. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. Machine Learning*, Washington, DC, 2003, pp. 912–919.

[45] X. Zhu, *Free-Formed Curves and Surfaces Modeling Techniques*. Beijing, China: Science Press, 2000.



Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, China, in 1993, the M.S. degree from Chongqing University, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2004.

From 1996 to 2001, He was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a postdoctorate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently an Associate Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing. His interests include pattern recognition and machine learning.



Feiping Nie received the B.S. degree in computer science from the North China University of Water Conservancy and Electric Power in 2000 and the M.S. degree from Lanzhou University, China, in 2003. He is currently pursuing the Ph.D. degree in the Department of Automation, Tsinghua University, Beijing, China.

His research interests focus on machine learning and its applications.

Chunxia Zhang received the B.S. degree in mathematics from Shanxi University, China, in 1996, the M.S. degree from Yunnan Normal University, China, in 2000, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2005.

She is currently a Lecturer in the School of Software, Beijing Institute of Technology, Beijing, China. Her interests include information extraction, machine learning, etc.

Changshui Zhang (M'02) received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986, and the Ph.D. degree from Tsinghua University, Beijing, in 1992.

In 1992, he joined the Department of Automation, Tsinghua University, and is currently a Professor. His interests include pattern recognition, machine learning, etc. He has authored over 200 papers.

Dr. Zhang currently serves on the editorial board of the *Pattern Recognition* journal.