*Interdisciplinary Information Sciences,* Vol. 13, No. 1, pp. 33–42 (2007)

# Image Segmentation Using MAP-MRF Estimation and Support Vector Machine

T. HOSAKA[1*], T. KOBAYASHI[1] and N. OTSU[1,2]

[1]*National Institute of Advanced Industrial Science and Technology, 1-1-1, Umezono, Tsukuba, Ibaraki 305-8568, Japan*
[2]*University of Tokyo, 7-3-1, Hongo, Tokyo 113-8654, Japan*

Received October 31, 2006; final version accepted December 28, 2006

Image segmentation has recently been studied in a framework of maximum *a posteriori* estimation for the Markov random field, where the cost function representing pixel-wise likelihood and inter-pixel smoothness should be minimized. The common drawback of these studies is the decrease in performance when a foreground object and the background have similar colors. We propose the likelihood formulation in the cost function considering not only a single pixel but also its neighboring pixels, and utilizing the support vector machine to enhance the discrimination between foreground and background. The global optimal solution for our cost function can be realized by the graph cut algorithm. Experimental results show an excellent segmentation performance in many cases.

KEYWORDS: image segmentation, MAP estimation, Markov random field, support vector machine, minimum cut algorithm

## 1. Introduction

Image segmentation, the technique of extracting an object in an image from the background, is the primary problem in image processing and computer vision. The task of image segmentation involves specifying a foreground object or background for every pixel in a given image, which can be regarded as binary classification problem[1]. This image cutout technique is widely used for applications such as object detection and tracking, and becomes important also for image composition.

Since the breakthrough of Geman and Geman [1], probabilistic inference has been a powerful tool for image processing, and some researches into image segmentation utilize the framework of probabilistic inference [2–5]. In particular, maximum *a posteriori* (MAP) estimation in a Markov random field (MRF) has recently been applied to find quality segmentation [3–5]. Our goal is to improve the MAP-MRF approach in order to accommodate "difficult" images for which their methods do not provide satisfactory results.

The image segmentation problem of the MAP-MRF strategy is formulated as follows. A given image can be represented as an array, $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$, where $c_i$ is the 1-dimensional and 3-dimensional pixel value(s) at pixel $i$ for gray-scale and RGB images, respectively, and $N$ is the number of pixels. Note that, for simplicity, we use a scalar in this expression to describe even RGB intensities of a color image. In this paper, the pixel value is defined as an integer from 0 to 255. A segmentation algorithm assigns a unique class label, $\alpha_i$, for each pixel, with $\alpha_i = 1$ if pixel $i$ is included in the foreground object, and $\alpha_i = 0$ if pixel $i$ is in the background. The solution $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ can be obtained by maximizing the posterior probability

$$P(\boldsymbol{\alpha}|\boldsymbol{c}) = \frac{1}{Z(\boldsymbol{c})} \exp\{-U(\boldsymbol{\alpha}; \boldsymbol{c})\}, \tag{1}$$

where $Z(\boldsymbol{c})$ represents a normalization factor, and $U(\boldsymbol{\alpha}; \boldsymbol{c})$ is the cost function, which is often expressed in the segmentation problem as

$$U(\boldsymbol{\alpha}; \boldsymbol{c}) = \lambda \sum_{i \in \mathcal{P}} U_1(\alpha_i; c_i) + \sum_{(ij) \in \mathcal{N}} U_2(c_i, c_j) \delta[\alpha_i \neq \alpha_j]. \tag{2}$$

The first term $U_1(\alpha_i; c_i)(\geq 0)$ is the likelihood term which expresses the cost when the label of pixel $i$ is estimated as $\alpha_i = 1$ (foreground) or $\alpha_i = 0$ (background). The symbol $\mathcal{P}$ represents the set of pixels. The second term in the right hand side of equation (2) is a local smoothing term, and $U_2(c_i, c_j)(\geq 0)$ is interpreted as a penalty for the discontinuity between pixels $i$ and $j$. The symbol $\mathcal{N}$ represents the set of adjacent pixel pairs, and the indicator function $\delta[\cdot]$ responds 1 when the argument is true, and 0 otherwise. The positive parameter $\lambda$ controls the balance between these two terms. Note that the MAP framework in this case is equivalent to minimizing the cost function $U(\boldsymbol{\alpha}; \boldsymbol{c})$.

* Corresponding author, E-mail: hosaka-t@aist.go.jp

[1] The term "image segmentation" could be used in other contexts. In this paper, image segmentation indicates fore/background binary classification.

Since we cannot foresee the foreground object which a user wants to extract, the user is required to impose certain hard constraints for segmentation by indicating some pixels that must be either part of a foreground object or the background. This marking gives clues to classify the remaining pixels based on the MAP-MRF approach. One typical method of providing these hard constraints is to enclose the foreground object that the user has in mind (Fig. 1(b)) to indicate that the region outside the lines must be in background. Another method is to draw few strokes in the foreground object and the background as illustrated in Fig. 1(d), where pixels on the red strokes are foreground and pixels on the blue strokes are in the background. The related work below was constructed according to this framework.

**Previous work**

The algorithms below can provide the global minimum of the corresponding cost functions (which is equivalent to the maximum of the posterior probability) using a minimum cut algorithm in practical time scale, developed in the graph theory [3, 6].

Boykov and Jolly [3] proposed a segmentation algorithm for a monochrome image, in which a user imposed the hard constraints by drawing a few strokes. To define the likelihood term $U_1(\alpha_i; c_i)$, they computed foreground and background gray-level distributions, namely histograms of intensities of user-input pixels. These were normalized to be used as a foreground probability $P_1(c)$ and a background probability $P_0(c)$, and provided the likelihood term with negative log-likelihoods as

$$U_1(\alpha_i; c_i) = -\ln P_{\alpha_i}(c_i). \tag{3}$$

They set the boundary penalties as

$$U_2(c_i, c_j) = \exp\{-\beta(c_i - c_j)^2\}, \tag{4}$$

where the parameter $\beta$ adjusts sensitivity for intensity difference. When the constant $\beta = 0$, the smoothing term is simply the well-known Ising interaction, encouraging smoothness everywhere. It has been shown, however, that it is usually far more effective to set $\beta > 0$ and relax the tendency of smoothness.

Rother *et al.* [4] adapted the above algorithm for color images, which is called *GrabCut* algorithm. Although the method of generating foreground and background probabilities is different, the formulation of $U_1(\alpha_i; c_i)$ and $U_2(c_i, c_j)$ is the same as in [3], except that $c_i$ represents the RGB intensities (3-dimensional vector) of each pixel:

$$U_1(\alpha_i; c_i) = -\ln P_{\alpha_i}(c_i), \tag{5}$$

$$U_2(c_i, c_j) = \exp\{-\beta\|c_i - c_j\|^2\}. \tag{6}$$

Instead of drawing a few strokes, a user is expected to roughly enclose the intended foreground object with a rectangle, a circle, or lines. As the increase in dimensions makes it difficult to construct reliable histograms in the RGB space, they establish the foreground and background probabilities using a full-covariance Gaussian mixture with $L$ components (typically $L = 5$). This algorithm initially sets $\alpha_i = 1$ inside the closed lines or curves and $\alpha_i = 0$ on or outside the closed lines or curves, and successively updates the probabilities $P_1(c)$ and $P_0(c)$ while the configuration $\boldsymbol{\alpha}$ is optimized, executing the minimum cut algorithm at each step.

The *Lazy Snapping* algorithm proposed by Li *et al.* [5] clusters the RGB intensities of pixels on user strokes by the $K$-means method (typically $K = 64$). The centroids of the foreground and background clusters are denoted as $\{D_n^1\}$ and $\{D_m^0\}$, respectively ($n$ and $m$ are cluster indices). Computing the minimum distance from color $c_i$ to foreground clusters as $d_i^1 = \min_n \|c_i - D_n^1\|$, and similarly $d_i^0 = \min_m \|c_i - D_m^0\|$, they define the likelihood term as

$$U_1(\alpha_i; c_i) = \frac{d_i^{\alpha_i}}{d_i^1 + d_i^0}. \tag{7}$$

The boundary penalty is defined as[2]

$$U_2(c_i, c_j) = \frac{1}{\|c_i - c_j\| + 1}. \tag{8}$$

Moreover, they execute certain preprocessing to improve computational efficiency.

**Goal of this paper**

The common drawback of the aforementioned algorithms is that performance tends to deteriorate when foreground and background regions have similar colors (Fig. 1). One solution for this difficulty is to provide an interactive user interface to modify imperfections, which has been adopted by the above authors [3–5]. In this paper, we examine a scheme to improve the performance itself, using the framework of Lazy Snapping. For constructing $U_1(\alpha_i; c_i)$, we focus on the configurations of each pixel and its neighbors, whereas Lazy Snapping uses intensities of each pixel alone. This

[2] They define a boundary penalty with $L2$-norm instead of $L1$-norm in their paper [5]. However, the smoothing term becomes too small to precisely adjust the parameter $\lambda$ for the optimal performance, and we found that the $L1$-norm-based formulation provides the qualitatively same or sometimes better segmentation results. Therefore, we adopt the boundary penalty (8) throughout this paper.

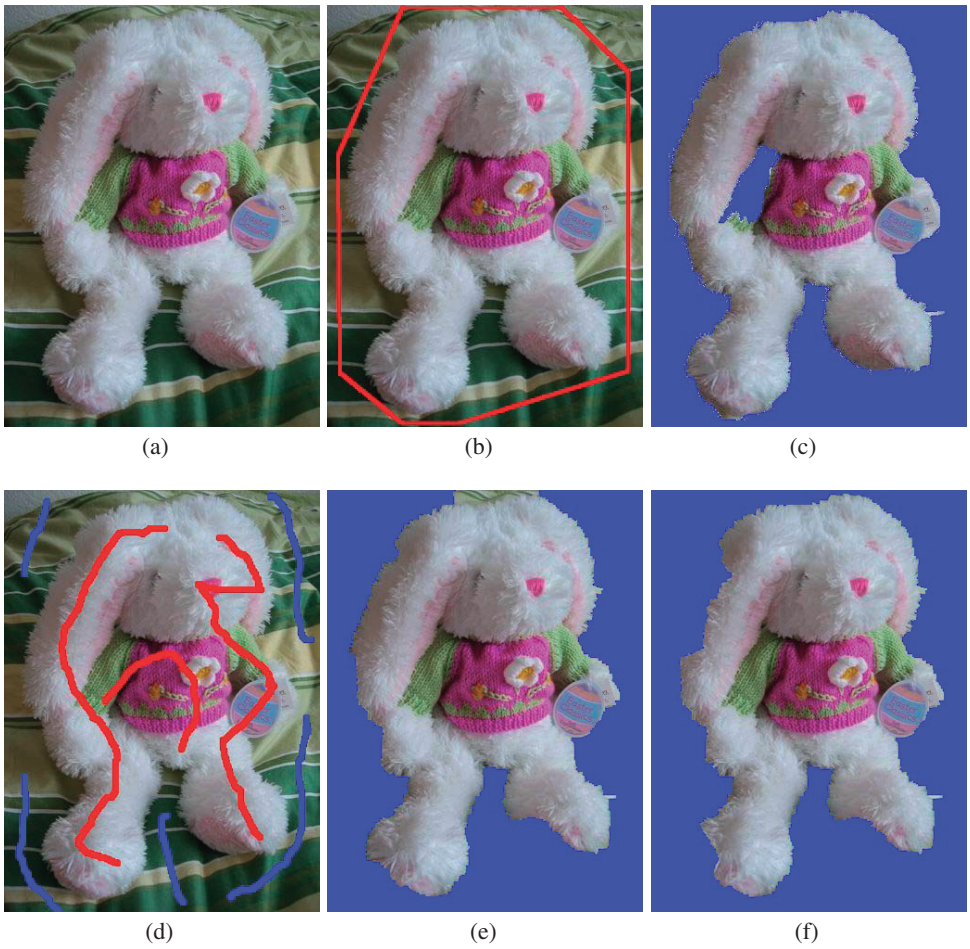


Fig. 1. Drawbacks of previous works. Traditional algorithms do not necessarily produce satisfactory outputs when a similar color is included in both the regions. (a) Original image of a stuffed rabbit. (b) User input enclosure. This indicates that the target object exists in the enclosed region, and therefore pixels outside the closed lines must be in the background region. (c) Result obtained by the GrabCut algorithm [4] using the program provided by the authors on their website. The right arm is misclassified as the background. (d) User input strokes. A user marks in the foreground (red strokes) and the background (blue strokes). (e) Result obtained with the Lazy Snapping algorithm [5] with $\lambda = 0.01$. The area above the head and left foot are erroneous. (f) Result obtained by the Lazy Snapping algorithm with $\lambda = 0.016$. Right ears and the left foot are erroneous.

extension incorporates neighboring information like texture into the segmentation to some extent. Furthermore, we enhance the discrimination between foreground and background by using support vector machine (SVM) [7].

This paper is organized as follows. In the next section, we introduce our cost function for image segmentation using the 15-dimensional feature vectors and SVM. Section 3 introduces the minimum cut algorithm which solves the minimization problem of the cost function. Experimental results are shown in Section 4. The final section is for concluding remarks.

## 2. Building Blocks of the Cost Function

We extend the framework of Lazy Snapping. This section describes our cost function.

### Extension of image vector

In Lazy Snapping, the minimum distance from the color of each pixel to the foreground and background cluster centroids is computed to construct a likelihood term. Including a similar color both in a foreground object and the background makes it difficult to classify these two regions based on pixel-wise RGB colors. One solution is to deal with neighboring information as well as pixel-wise colors and extract effective features for image segmentation.

Based on this perspective, we use the configuration of each pixel and its four nearest neighbors as one of the straightforward extensions. Although there are several alternatives for local image features such as HSV colors [8] and SIFT [9], we adopt the standard RGB colors and treat each color channel equally to facilitate comparison of our method with previous works. It has been also reported in the study of skin segmentation [10] that there is little difference in the average performances of the main color spaces (RGB, HSV, YCbCr, and so on). Therefore, we construct a 15-dimensional vector consisting of the RGB intensities of each pixel and its four nearest neighbors (Fig. 2). The array of

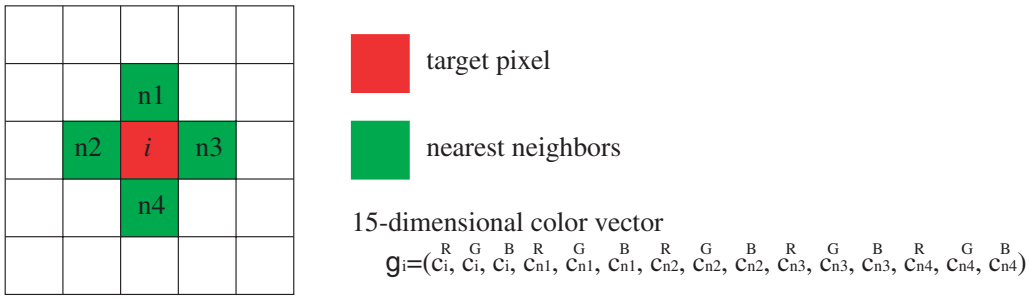


Fig. 2. Proposed 15-dimensional vector. We use a 15-dimensional vector consisting of the RGB intensities of each pixel (painted red) and those of its four nearest neighbors (painted green). Indices R, G, and B represent the color channel. The order of alignment is fixed and unchanged.

this vector is denoted as $\boldsymbol{g} = (g_1, g_2, \ldots, g_N)$, where each $g_i$ is a 15-dimensional vector at pixel $i$.

We expect this configuration to include some texture information. It is natural that the RGB color combinations among five pixels have more divergences than in the case of a single one, and therefore extending 3-dimensional RGB colors to 15-dimensional vectors provides additional information for more accurate classification. In particular, even though a similar color exists in the foreground and background, when the textures are different (*e.g.* Fig. 3(a)), the configuration of the 15-dimensional vectors exhibits patterns and regularities, which can be helpful in quality segmentation.

**Support vector machine**

We enhance discrimination between fore/background by using the above 15-dimensional vectors and learning from samples on user-input strokes to extract the effective information for image segmentation. The support vector machine (SVM) with the *kernel trick* [7] provides one of the schemes for carrying this task, which generates more discriminative nonlinear classification boundary between foreground and background.

SVM is a powerful machine learning technique for binary classification, supported by a strong theoretical foundation and excellent empirical success. We are given $M$ training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ $(\boldsymbol{x}_i \in \mathcal{R}^l)$ and their labels $\{y_1, y_2, \ldots, y_M\}$ $(y_i \in \{0, 1\})$. Vectors in input space $\mathcal{R}^l$ are mapped into a higher dimensional space by the nonlinear mapping function $\phi : \mathcal{R}^l \to \mathcal{R}^h (h > l)$. The SVM's output function is expressed as

$$f_{\mathrm{SVM}}(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b, \tag{9}$$

where superscript $T$ represents transpose, and $\boldsymbol{w}$ and $b$ are a weight vector and bias, respectively, which should be determined appropriately through learning. Input vector $\boldsymbol{x}$ is classified according to $y = \Theta[f_{\mathrm{SVM}}(\boldsymbol{x})]$, where $\Theta[z]$ is 1 for $z \geq 0$, and 0 otherwise. Geometrically, $\boldsymbol{w}$ indicates the normal vector of a hyperplane. All vectors $\phi(\boldsymbol{x}) \in \mathcal{R}^h$ lying on one side of the hyperplane are classified as 1 and all vectors on the other side as 0.

The goal of SVM is to find the parameters $\boldsymbol{w}$ and $b$ for the optimal hyperplane to maximize the margin $1/\|\boldsymbol{w}\|$ (distance between the hyperplane and the closest training instances termed *support vectors*), which is the solution of the following optimization problem

$$\min_{\boldsymbol{w}, b, \{\xi_i\}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + \gamma \sum_{i=1}^{M} \xi_i, \tag{10}$$

subject to

$$(2y_i - 1) \cdot (\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, 2, \ldots, M), \tag{11}$$

where the parameter $\xi_i$ represents how the $i$th instance gets into the wrong side, and the parameter $\gamma > 0$ controls the balance between two terms of the cost function (10).

This optimization problem can be transformed to a dual problem by the Lagrange method. Finally, the SVM's output function is obtained as

$$\begin{aligned} f_{\mathrm{SVM}}(\boldsymbol{x}) &= \sum_{i=1}^{M} \eta_i (2y_i - 1) \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + b \\ &= \sum_{i=1}^{M} \eta_i (2y_i - 1) K(\boldsymbol{x}_i, \boldsymbol{x}) + b, \end{aligned} \tag{12}$$

where $\{\eta_i\}$ is the Lagrange multiplier for the first inequality constraint in equation (11), which is the solution of the dual problem. We use the kernel function $K$,

$$\phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') \tag{13}$$

to obtain the inner product without explicit calculation in the higher dimensional space $\mathcal{R}^h$. In this study, we adopt the Gaussian kernel [11]

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right), \tag{14}$$

where $\sigma$ is a parameter.

**Likelihood term**

We construct the likelihood term based on the outputs of SVM. Note that the training data is the proposed 15-dimensional vectors at user-marked pixels. We restrict the values of likelihood and smoothing terms in the cost function (2) to the range between 0 and 1 in order to make scales of these two terms same and facilitate the adjustment of the parameter $\lambda$. The likelihood term $U_1(\alpha_i; g_i)$, which depends on $g_i$ instead of $c_i$, is defined by using the SVM's output function $f_{\mathrm{SVM}}(g_i)$ as

$$U_1(\alpha_i = 1 - k_i; g_i) = \frac{1}{1 + \exp\{-a_{k_i}|f_{\mathrm{SVM}}(g_i)|\}}, \quad U_1(\alpha_i = k_i; g_i) = 1 - U_1(\alpha_i = 1 - k_i; g_i), \tag{15}$$

where $k_i \equiv \Theta[f_{\mathrm{SVM}}(g_i)]$ is the classification result. The logistic function is widely used to convert the SVM outputs to values between 0 and 1, such as a probability [12]. The coefficients $a_1$ and $a_0$ should be optimally tuned for an appropriate conversion, and we empirically set these parameters as

$$a_k = \frac{4}{J_k}, \tag{16}$$

where $J_1$ and $J_0$ denote the absolute value of the averaged cost (12) for the foreground and background training data, respectively. The other parameters are fixed as $\gamma = 100$ in equation (10) and $\sigma^2 = 1000$ in equation (14) throughout the paper.

**Comparison of likelihood terms among techniques**

Figure 3 shows the effectiveness of the 15-dimensional vectors and classification by SVM. The figure describes the value of $U_1(\alpha_i = 1; c_i)$ or $U_1(\alpha_i = 1; g_i)$ with 256 gray-levels for the $K$-means method with the 3-dimensional pixel-wise RGB (Lazy Snapping, Figs. 3(b), (g), and (l)), the $K$-means method with the proposed 15-dimensional vectors (Figs. 3(c), (h), and (m)), SVM with the 3-dimensional RGB (Figs. 3(d), (i), and (n)), and SVM with the 15-dimensional vectors (our method, Figs. 3(e), (j), and (o)). Red and blue strokes indicate user inputs for foreground and background, respectively. Figure 3(a) is an artificial graphic produced to help understand the effectiveness of the proposed 15-dimensional vectors, in which a foreground object (yellow ball) exists in background texture of a striped pattern of width one pixel. When a similar color exists in both foreground and background, the performance of pixel-wise methods degrades. Figure 3 indicates that our method (15-dimensional vectors and SVM) provides favorable discrimination results.

**Smoothing term**

The smoothing penalty in our method is the same as equation (8), although there is little difference for the following experimental results when using equation (6).

## 3. Optimization by Graph Cut

The cost function (2) is defined so that its minimum corresponds to a high-quality segmentation, and the optimal configuration $\boldsymbol{\alpha}^*$ can be estimated as a global minimum

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, U(\boldsymbol{\alpha}; \boldsymbol{c}). \tag{17}$$

As mentioned before, this optimal assignment gives a maximum value for the posterior probability (1). Although it is difficult to find a global minimum of any type of cost function in practical time scale, minimization (17) for cost functions of the form (2) can be done using the minimum cut algorithm established in the graph theory [13, 14].

An undirected graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as a set of vertices $\mathcal{V}$ and a set of undirected edges $\mathcal{E}$ that connect these vertices. The correspondence between an image and a graph in this paper is shown in Figs. 4(a) and (b). The vertices expressed as circles correspond to pixels, and there are two additional nodes expressed with a square, a foreground terminal termed *source* $S$, and a background terminal termed *sink* $T$. The relationship between an image and a graph can be described as

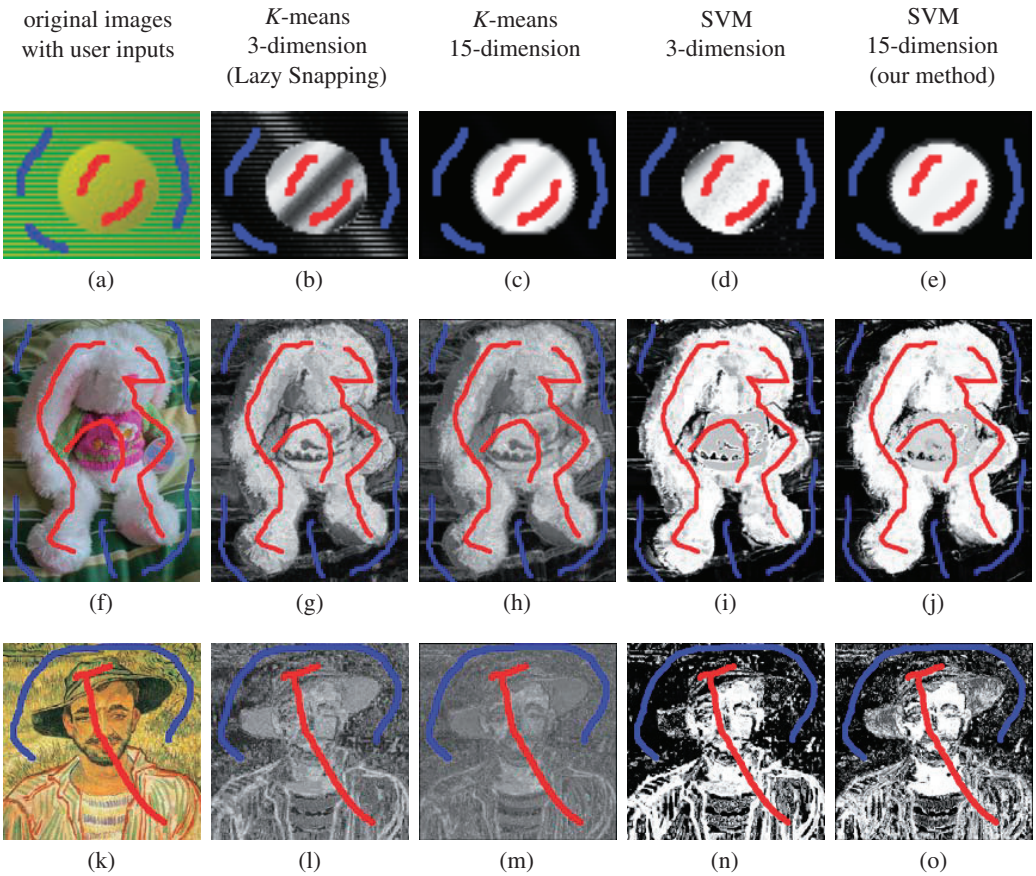


Fig. 3. Comparison of likelihood terms among several methods. The pixel in white indicates an affinity to the foreground and the pixel in black to the background. In the toy example shown in (a), the 3-dimensional pixel-wise RGB colors ((b) and (d)) are insufficient to separate the foreground object from the background texture, whereas the proposed 15-dimensional vectors ((c) and (e)) provide excellent classification. In the example of the stuffed rabbit, the $K$-means method with the 3-dimensional RGB vectors (Lazy Snapping, (g)) does not accurately separate the object from the background, especially for the right ear and the left foot. This does not significantly improve for the $K$-means method with the 15-dimensional vectors (h). However, SVM-based likelihood term (3-dimensional (i) and 15-dimensional (j)) shows better separation than those of the $K$-means method. In the case of the pictorial art example, the classification based on the $K$-means method ((l) and (m)) is not clear, compared with those of SVM ((n) and (o)). Nevertheless, desirable segmentation results are finally obtained as shown in Section 4. SVM with the 3-dimensional vectors misclassify pixels of the brim of the man's hat, which leads to an erroneous final segmentation.

$$\mathcal{V} = \mathcal{P} \cup \{S, T\},$$
$$\mathcal{E} = \mathcal{N} \cup \bigcup_{i \in \mathcal{P}} \{(iS), (iT)\}.$$

An edge between nodes $i$ and $j$ in the graph is assigned a nonnegative weight $z_{ij}$, which is reflected by the edge's thickness in the figure. A cut is defined as a subset of edges $\mathcal{C} \subset \mathcal{E}$ such that the terminals become separated on the induced graph $G_{\mathcal{C}} = \langle \mathcal{V}, \mathcal{E} \backslash \mathcal{C} \rangle$. In the combinatorial optimization problem, the cost of a cut is defined as the sum of the weights of the extracted edges, that is,

$$\text{cost} = \sum_{(ij) \in \mathcal{C}} z_{ij}. \tag{18}$$

The minimum cut problem is to find the cut with the globally minimum cost. It is well-known that minimum cut can be computed efficiently in low-order polynomial time with traditional algorithms [13, 14] and a subsequent sophisticated algorithm [6].

Graph cut formulation is well suited to image segmentation. As illustrated in Figs. 4(c) and (d), a cut separates a given image into two regions, one region including the foreground terminal $S$ and the other including the background terminal $T$. Although the image is divided into just one foreground object and one background region in the simplest example of Fig. 4, the algorithm in practice can provide object and background segments consisting of several isolated regions.

A minimum cut generates an optimal segmentation in terms of the properties built into the edge weights. We need to set adequate edge weights such that the resulting minimum cut provides the segmentation that minimizes equation (2)

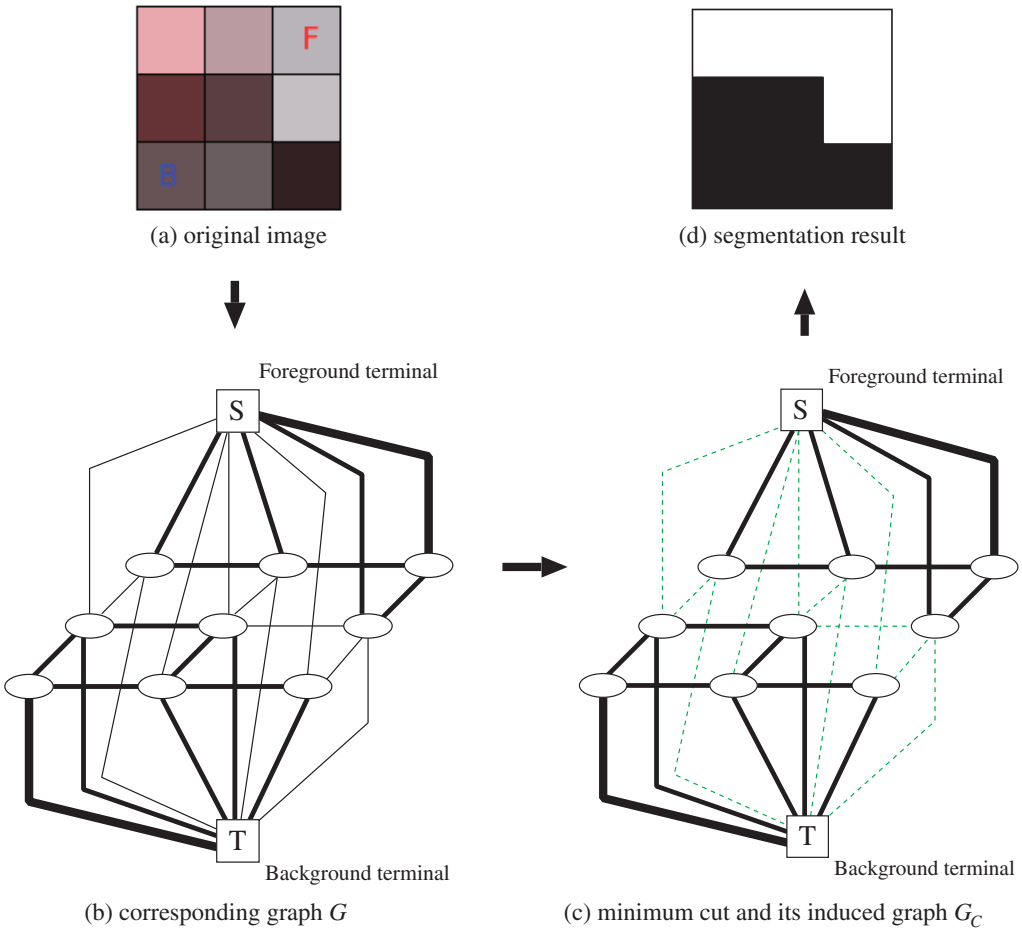


Fig. 4. Schematic profile of the structure of a graph and segmentation. (a) A $3 \times 3$ image. The letters F and B represent user-marked foreground and background pixels, respectively. (b) Each pixel in the image corresponds to a vertex, and an edge is put between the nearest neighbors. (c) A cut, expressed as green dashed lines, separates vertices into foreground and background regions. (d) Corresponding segmentation result.

among all configurations satisfying hard constraints of user inputs. It is mathematically proven [3] that this can be realized by setting weights of a graph as

$$z_{ij} = U_2(c_i, c_j) \qquad ((ij) \in \mathcal{N}), \tag{19}$$

$$z_{iS} = \begin{cases} \lambda \cdot U_1(\alpha_i = 0; g_i) & \text{if pixel } i \text{ is not a user input pixel} \\ 1 + \max\limits_{i \in \mathcal{P}} \sum\limits_{j:(ij) \in \mathcal{N}} U_2(c_i, c_j) & \text{if pixel } i \text{ is a foreground user input pixel} \\ 0 & \text{if pixel } i \text{ is a background user input pixel,} \end{cases} \tag{20}$$

$$z_{iT} = \begin{cases} \lambda \cdot U_1(\alpha_i = 1; g_i) & \text{if pixel } i \text{ is not a user input pixel} \\ 0 & \text{if pixel } i \text{ is a foreground user input pixel} \\ 1 + \max\limits_{i \in \mathcal{P}} \sum\limits_{j:(ij) \in \mathcal{N}} U_2(c_i, c_j) & \text{if pixel } i \text{ is a background user input pixel.} \end{cases} \tag{21}$$

Based on this fact, we implemented the minimization of the cost function introduced in the previous section using the new minimum cut algorithm [6]. The graph cut algorithm has also been used for problems other than image segmentation [15, 16].

One of the other well-known inference algorithms is the belief propagation (BP) [17]. BP is an iterative method that efficiently calculates the maximum posterior probability, or marginal posterior probability, on the basis of a factored cost function. The BP can be applied to our segmentation scheme. However, the BP solution is not necessarily optimal globally, which prevents us from using it. The segmentation results obtained by the BP are actually inferior to those obtained by the minimum cut algorithm in most cases.

## 4. Experimental Results

The proposed approach has been tested on various images. Figure 5 shows several results obtained by our method

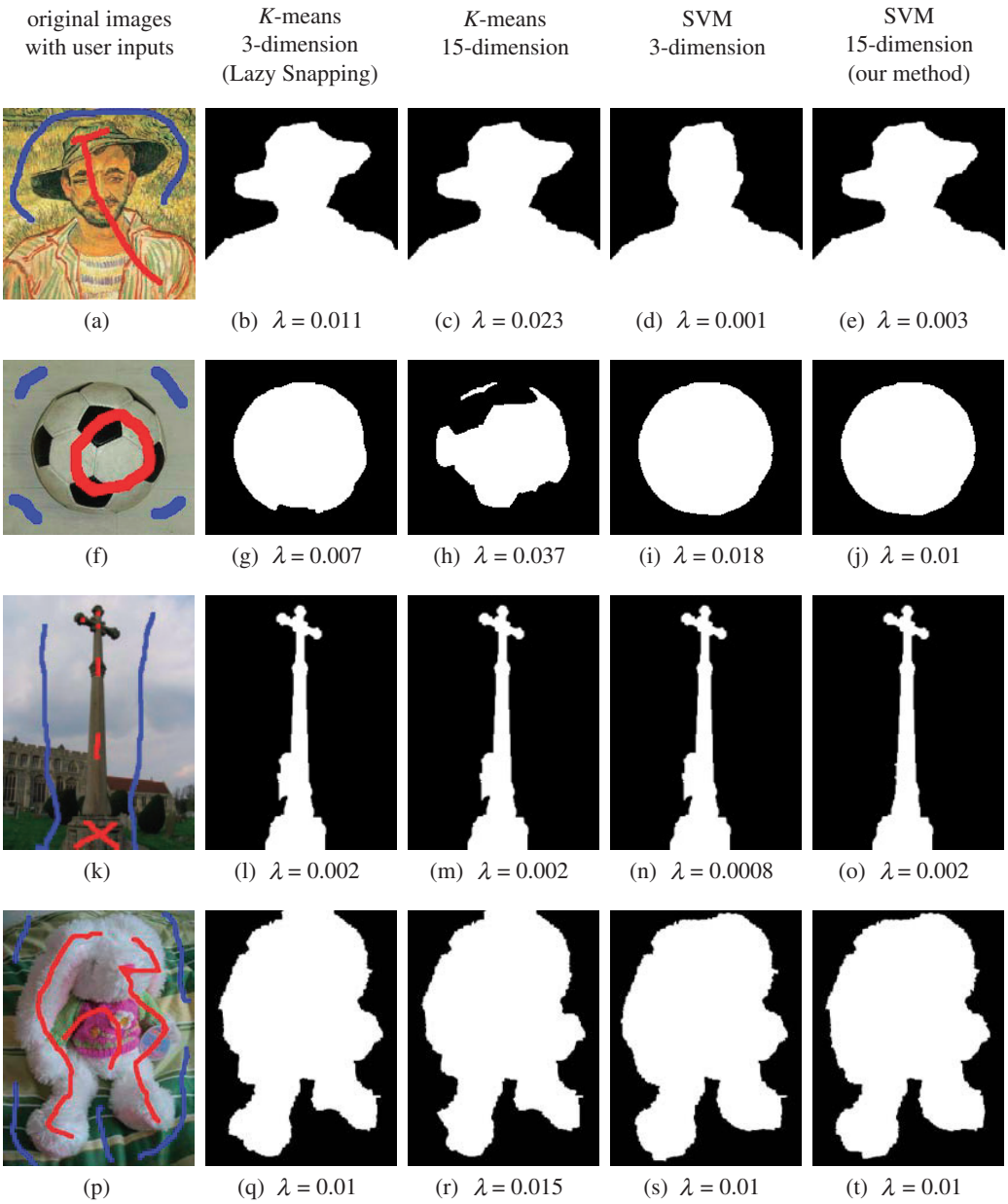


Fig. 5. Examples of experimental results. The first column shows original images with user-specified foreground and background strokes. The other columns show the results of the $K$-means method with the 3-dimensional pixel-wise RGB vectors (Lazy Snapping), the $K$-means method with the proposed 15-dimensional vectors, SVM with the 3-dimensional RGB vectors, and SVM with the 15-dimensional vectors (our method). The values of $\lambda$ are adjusted so that the performance is optimal by appearance.

with user-marked foreground and background strokes, and the results obtained by algorithms based on Lazy Snapping (the $K$-means method with the 3-dimensional pixel-wise RGB vectors), the $K$-means method with the 15-dimensional vectors, and SVM with the 3-dimensional RGB vectors. Since it is basically difficult to quantitatively evaluate performance in image segmentation, we have to resort to subjective evaluation. The parameter $\lambda$ was determined manually for every image and algorithm. Figure 6 shows composite images of a foreground object extracted by our method and a blue background.

Our approach compares favorably with other methods, indicating that the 15-dimensional color vectors introduced in this paper and classification by SVM are effective for image segmentation. However, for more difficult images where the foreground object and background have complicated boundary shapes, our approach may not always work well like other methods. Another common difficulty for segmentation algorithms is their dependence on positions and the quantity of user inputs. In particular, when a user draws only a few strokes, performance can deteriorate drastically.

The operational time is acceptable for personal use, although more computational time is usually needed for SVM implementation than for the $K$-means method. For instance, using a 3.8 GHz CPU with 4 GB RAM, an image size of

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

Fig. 6. Composite images with blue background. The foreground objects are extracted by our method.

341 × 455 pixels (the stuffed rabbit in Fig. 5) requires 19.6 sec for SVM operation and 1.8 sec for subsequent graph cut optimization. The Lazy Snapping algorithm requires 3.0 sec for $K$-means calculation.

## 5. Concluding Remarks

This paper has proposed an image segmentation algorithm for foreground extraction. One of our key contributions is that we use neighboring information with higher dimensional vectors, instead of considering the information in a single pixel alone. The other is that we enhance the discrimination between foreground and background using SVM. We obtained quality segmentation results even when a foreground object and background have similar colors.

Setting the parameter values influences segmentation results. In this study, we need to set an optimal value for $\lambda$, which ought to be deductively implemented in practice. Statistical inference methods such as the maximum of marginal likelihood [18, 19] could be used for this parameter estimation. For parameters $\gamma$, $\sigma$, and $a_k$ in the SVM formulation, we could not obtain the best performance for values mentioned in the former section. The cross-validation method is one promising solution for this difficulty [12].

When the image segmentation is used as a preprocessing for applications such as image recognition, object detection, and object tracking, the approach of binary classification in this paper is expected to be applicable. Nevertheless, it might be insufficient for image composition since binary classification tends to produce jaggies around the boundaries between the foreground object and its background. A simple solution for this difficulty is to implement the antialiasing for smooth boundaries. The other solution is to solve a problem similar to the image segmentation, called *matting* [20–22], in which a given image $\boldsymbol{c}$ is supposed to be a composite of a foreground image $\boldsymbol{f} = (f_1, f_2, \ldots, f_N)$ and background image $\boldsymbol{b} = (b_1, b_2, \ldots, b_N)$. The color of each pixel $c_i$ is assumed to be described by a linear combination of the corresponding foreground and background colors with its mixing rate $\alpha_i \in [0, 1]$ as $c_i = \alpha_i f_i + (1 - \alpha_i) b_i$, and the goal is to estimate mixing rates $\boldsymbol{\alpha}$, foreground colors $\boldsymbol{f}$, and background colors $\boldsymbol{b}$ for each pixel in the image. The segmentation problem in this paper corresponds to the case of $\alpha_i \in \{0, 1\}$. Our method holds one promising clues for quality matting, which will be detailed in a different paper.

## Acknowledgements

The study was supported by the advanced surveillance technology project of MEXT. The authors thank T. Kurita and N. Ichimura for their helpful discussions.

REFERENCES

[1] Geman, S., and Geman, D., "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. on PAMI*, **6**: 721–741 (1984).

[2] Mortensen, E. N., and Barrett, W. A., "Intelligent scissors for image composition," *Proceedings of ACM SIGGRAPH'95*, 191–198 (1995).

[3] Boykov, Y. Y., and Jolly, M. P., "Interactive graph cuts for optimal boundary & region segmentation of objects in N–D images," *Proceedings of ICCV 2001*, 105–112 (2001).

[4] Rother, C., Kolmogorov, V., and Blake, A., "GrabCut - interactive foreground extraction using iterated graph cuts," *Proceedings of ACM SIGGRAPH 2004*, 309–314 (2004).

[5] Li, Y., Sun, J., Tang, C. K., and Shum, H. Y., "Lazy Snapping," *Proceedings of ACM SIGGRAPH 2004*, 303–308 (2004).

[6] Boykov, Y. Y., and Kolmogorov, V., "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. on PAMI*, **26**: 1124–1137 (2004).

[7] Vapnik, V., Statistical learning theory, *Wiley: New York* (1998).

[8] Sobottka, K., and Pitas, I., "A novel method for automatic face segmentation, facial feature extraction and tracking," *Signal Processing: Image Communication*, **12**: 263–281 (1998).

[9] Lowe, D., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, **60**: 91–110

(2004).

[10] Phung, S. L., Bouzerdoum, A., and Chai, D., "Skin segmentation using color and edge information," *Proc. Int. Symposium on Signal Processing and its Applications*, 525–528 (2003).

[11] Müller, K. R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B., "An introduction to kernel-based learning algorithms," *IEEE Trans. on Neural Networks*, **12**: 181–201 (2001).

[12] Platt, J. C., "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, edited by Smola, A. J., Bartlett, P. J., Schölkopf, B., and Schuurmans, D., MIT Press (2000).

[13] Ford, L., and Fulkerson, D., "Flows in networks," *Princeton University Press* (1962).

[14] Goldberg, A., and Tarjan, R., "A new approach to the maximum flow problem," *Journal of the Association for Computing Machinery*, **35**: 921–940 (1988).

[15] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A., "Graphcut textures: Image and video synthesis using graph cuts," *Proceedings of ACM SIGGRAPH 2003*, 277–286 (2003).

[16] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. F., "Interactive digital photomontage," *Proceedings of ACM SIGGRAPH 2004*, 294–302 (2004).

[17] Pearl, J., "Probabilistic reasoning in intelligent systems," Morgan-Kauffman (1988).

[18] Zhou, Z., Leahy, R. M., and Qi, J., "Approximate maximum likelihood hyperparameter estimation for Gibbs priors," *IEEE Trans. on Image Processing*, **6**: 844–861 (1997).

[19] Tanaka, K., "Theoretical study of hyperparameter estimation by maximization of marginal likelihood in image restoration by means of cluster variation method," *Electronics and Communications in Japan*, **85**: 50–62 (2002).

[20] Chuang, Y. Y., Curless, B., Salesin, D., and Szeliski, R., "A Bayesian approach to digital matting," *Proceedings of the 2001 IEEE CVPR*, 264–271 (2001).

[21] Wang, J., and Cohen, M. F., "An iterative optimization approach for unified image segmentation and matting," *Proceedings of IEEE ICCV 2005*, 936–943 (2005).

[22] Levin, A., Lischinski, D., and Weiss, Y., "A closed form solution to natural image matting," *Proceedings of the 2006 IEEE CVPR*, 61–68 (2006).