The Journal of the Society for Art and Science, Vol.14, No.5, pp. 207–219

Iterative Refinement of Alpha Matte using Binary Classifier

Zixiang Lu[1)] **Tadahiro Fujimoto**[1)]

1) Graduate School of Engineering, Iwate University

luzixiang (at) cg.cis.iwate-u.ac.jp
fujimoto (at) cis.iwate-u.ac.jp

Abstract
Image matting is a technique used to extract a foreground object from an input color image by estimating its opacity mask, which is called an alpha matte. Many previous methods have estimated a poor-quality alpha matte when a foreground object and its background have similar colors. To overcome this problem, we treat an image matting problem as a binary classification problem to classify unknown pixels into foreground and background pixels. In this study, we propose a learning-based matting method based on binary classification. In our method, a binary alpha matte is first obtained by a binary classifier. The binary alpha matte is then iteratively refined to a high-quality alpha matte by repeat use of the classifier. Although we used a support vector machine classifier and the closed-form matting method, an important merit of our method is that it provides a general mechanism to refine an alpha matte by combining various binary classifiers and matting methods. The excellent performance of our method was revealed in several experiments, especially, for input images having similar foreground and background colors as well as objects with small holes.

1 Introduction

Image matting is a fundamental and useful technique for image and video editing. From a foreground region separated from a background region on a given input color image, image matting extracts the opacity mask, which is called an *alpha matte*, as well as foreground and background colors. Image matting is based on the following equation.

$$\boldsymbol{I}_i = \alpha_i \boldsymbol{F}_i + (1 - \alpha_i)\boldsymbol{B}_i. \quad (1)$$

The vectors $\boldsymbol{I}_i$, $\boldsymbol{F}_i$, and $\boldsymbol{B}_i$ represent the colors of an image color, foreground, and background of a pixel p_i in a given input color image. Each color is a three-dimensional (3D) vector with red (R), green (G), and blue (B) color values. The scalar $\alpha_i \in [0, 1]$ is an opacity value, *alpha*. If $\alpha_i = 1$ or 0, the pixel p_i is a *definite foreground pixel* or a *definite background pixel* respectively. Otherwise, the pixel p_i is a *mixed* pixel. An accurate estimation of α_i, $\boldsymbol{F}_i$, and $\boldsymbol{B}_i$ of mixed pixels p_i is required to separate foreground and background regions fully. Eq.(1) consists of three linear equations with three known variables R, G, B in $\boldsymbol{I}_i$ and seven unknown variables α_i and R, G, B in $\boldsymbol{F}_i$ and $\boldsymbol{B}_i$. This means that a matting problem is inherently ill-posed and does not have a unique solution to satisfy Eq.(1). Thus, estimating a likely solution of α_i, $\boldsymbol{F}_i$, and $\boldsymbol{B}_i$ for each pixel p_i according to an input image is essential.

Many matting methods have been proposed. Smith and Blinn [1] developed a blue screen matting method, that has been widely used in film production. In this method, foreground objects are photographed against a background with known colors to solve Eq.(1) easily by reducing unknown variables. Various methods have been developed to extract foreground objects on a natural image without a known background. These methods use either one or both of the two kinds of interfaces to offer clues to solving a matting problem. These interfaces are known as *trimap-based* and *scribble-based*. A trimap-based interface uses a *trimap* given by a user, and divides the entire image region into three subregions: a *definite foreground*, *a definite background* , and an *unknown region*, as shown in Figure 1(b). A definite foreground or background region has only definite foreground or background pixels, colors of which are used as clues. An unknown region has unknown pixels to estimate. These are not only mixed pixels but also definite foreground or background pixels. In a scribble-based interface, which was first proposed by Wang and Cohen [2], a user gives some *foreground* and *background scribbles* on an input image. These scribbles contain a few definite foreground and background pixels, respectively. The colors of these pixels are used to estimate alphas and the foreground or background colors of the remaining pixels. In addition, a matting method generally uses one of the following three approaches: *sampling-based*, *propagation-based*, and *hybrid approaches* [3]. Bayesian matting [4] is a well-known sampling-based method that is based on the maximum posteriori estimation in a Bayesian framework. Poisson matting [5] and closed-form matting (CF) [6] are propagation-based methods. Poisson matting obtains an alpha matte by solving a Poisson equation of alphas and image color gradients. Closed-form matting uses a matting Laplacian matrix based on a color line assumption. Most recent methods use hybrid approaches that combine sampling-based and propagation-based approaches such as robust color sampling matting [7], shared sampling matting [8], and large kernel matting [9]. In addition, some methods use *learning-based approaches*, such as learning based digital matting [10], support vector machine (SVM)-based matting [11], and support vector regression (SVR)-based matting [12]. In addition, additional excellent methods recently been developed, e.g., global sampling matting [13], KNN matting [14], local and nonlocal smoothness prior (LNSP) matting [15], color clustering matting (CCM) [16], and comprehensive sampling (CS) matting [17].

In this study, we propose a matting method founded on a learning-based approach. As described in Section 2.3 later, our method is completely different from the above-mentioned learning based methods [10] [11] [12]. The key idea of our method is the iterative refinement of an alpha matte. In our method, an alpha matte obtained by a basic matting method we employ is refined iteratively using a binary classifier that intelligently classifies foreground and background pixels. The alpha matte gets better gradually as the iteration proceeds. We use the *support vector machine* (*SVM*) [18] as a binary classifier in a manner different from the SVM-based matting [11], although we can use another binary classifier. The SVM classifier is a well-known excellent binary classifier that reliably classifies foreground and background pixels even if their colors are similar. In addition, we employ the closed-form matting (CF) method [6] as a basic matting method. Some methods, such as those in [8], [12], [13] and [17], use the CF method in post-processing to improve an obtained alpha matte. This usage of the CF method greatly inspired us to invent the idea of our method, although we can employ another method as our basic matting method. The iterative refinement mechanism proposed in this study is not limited to the combination of the SVM and the CF method. An important merit of our method is to provide a general mechanism to improve an alpha matte by combining various binary classifiers and matting methods.

2 Related work

2.1 Closed-form matting

Levin et al. proposed the CF method [6]. Its main assumption is the color line model: foreground or background colors in a small window lie on a single line in RGB color space. Based on this assumption, the alpha α_i of a pixel p_i in a small window is represented as a 4D linear model of $\alpha_i \approx \boldsymbol{a}^T \boldsymbol{I}_i + b$ using its color $\boldsymbol{I}_i$ and constants $\boldsymbol{a}$ and b. By minimizing the cost function $J(\boldsymbol{\alpha}, \boldsymbol{a}, b)$ defined using the 4D linear model, a quadratic function of $\boldsymbol{\alpha}$, which is a vector of the alphas α_i of all pixels p_i, is obtained.

$$J(\boldsymbol{\alpha}) = \min_{\boldsymbol{a},b} J(\boldsymbol{\alpha}, \boldsymbol{a}, b) = \boldsymbol{\alpha}^T \boldsymbol{L} \boldsymbol{\alpha}. \quad (2)$$

The matrix $\boldsymbol{L}$ is called matting Laplacian, which has been widely used in many matting methods. In addition, its elements are shown as follows.

$$L_{ij} = \sum_{k|(i,j)\in w_k} (\delta_{ij} - \frac{1}{|w_k|}(1 + (\boldsymbol{I_i} - \mu_k) (\Sigma_k + \frac{\epsilon}{|w_k|}\boldsymbol{I_3})^{-1}(\boldsymbol{I_j} - \mu_k))) \quad (3)$$

where δ_{ij} is the Kronecker delta, μ_k and Σ_k are the mean and covariance matrix of the colors in window w_k, $|w_k|$ is the number of pixels in w_k, and $\boldsymbol{I_3}$ is a 3×3 identity matrix.

To estimate an alpha matte to satisfy a constraint by a user for example, a trimap, the CF method solves the optimization problem.

$$\boldsymbol{\alpha} = argmin_{\boldsymbol{\alpha}}\ \boldsymbol{\alpha}^T \boldsymbol{L}\boldsymbol{\alpha} + \lambda(\boldsymbol{\alpha} - \boldsymbol{\beta})^T \boldsymbol{D}_s(\boldsymbol{\alpha} - \boldsymbol{\beta}), \quad (4)$$

where λ is a large number, $\boldsymbol{D}_s$ is a diagonal matrix whose diagonal elements are 1 for constrained pixels and 0 for others, and $\boldsymbol{\beta}$ is a vector that contains specified alphas for the constrained pixels and 0 for others.

After we estimate alphas, reconstructing foreground and background colors is typically necessary. However, this remains an ill-posed problem. Levin et al. proposed a reconstruction method to solve this problem using the 4D linear model, given as:

$$min \sum_{i\in I}\sum_{c}(\alpha_i F_i^c + (1-\alpha_i)B_i^c - I_i^c)^2 + |\alpha_{i_x}|((F_{i_x}^c)^2 + (B_{i_x}^c)^2) + |\alpha_{i_y}|((F_{i_y}^c)^2 + (B_{i_y}^c)^2), \quad (5)$$

where $F_{i_x}^c$, $F_{i_y}^c$, $B_{i_x}^c$, and $B_{i_y}^c$ are the x and y derivatives of F^c and B^c, and α_{i_x} and α_{i_y} are the matte derivatives. For a fixed α, the cost function (5) is quadratic and its minimum may be found by solving a sparse set for linear equations. In our method, we use the CF and reconstruction methods iteratively to refine an alpha matte.

2.2 Support vector machine

SVM [18] provides an excellent binary classifier based on the structural risk minimization principle. An SVM binary classifier separates samples into two classes according to their features. Its basic concept is to locate the optimal separating hyperplane to separate samples in the feature space so that the hyperplane has the maximal margin between the nearest samples in both classes.

The SVM classifier is first trained by means of a set of training data $\{(\boldsymbol{X}_i, y_i)\}$ of samples $\{s_i\}$, $i = 1, \ldots, M$, where $\boldsymbol{X}_i \in R^n$ is a feature vector and $y_i \in \{1, -1\}$ is a binary class label. To use the "kernel trick" technique, the feature vectors $\boldsymbol{X}_i$ are mapped to a higher dimensional space by a function $\boldsymbol{\phi}$. The function is used to define a kernel function $\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \boldsymbol{\phi}(\boldsymbol{X}_i)^T\boldsymbol{\phi}(\boldsymbol{X}_j)$, which is in turn used to define a classification function to realize the optimal hyperplane in the space. Many kernel functions have been proposed such as linear, polynomial, radial basis function [19], and sigmoid. In our method, we use the Gaussian radial basis function kernel.

$$\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{(-\frac{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2}{2\sigma^2})}, \quad (6)$$

where σ is a kernel parameter. After training using the training data, we obtain a classification function that can classify new samples using feature vectors given as a testing data. We used the LIBSVM [20] to develop our software.

2.3 Learning-based matting

Zheng et al. solved an alpha matting problem by treating it as a semi-supervised learning task in machine learning [10]. They proposed local and global learning based approaches. In the local learning based approach, it is assumed that the alpha of a pixel is represented not only by a linear combination of the alphas of its neighboring pixels but also by a linear combination of its color components. Based on the assumption in a local region (a 7 $\times$ 7 local patch), by using the ridge regression technique, a linear alpha-color local model is considered to estimate the alphas of unknown pixels by a coefficient matrix obtained from image colors of all pixels and the alphas of known pixels. The linear model is extended to a nonlinear model by the kernel trick. Then, the local learning based approach is extended to the global one by selecting for each unknown pixel its nearby known foreground and background pixels by the shortest Euclidean distances instead of its neighboring pixels.

Hosaka et al. proposed an optimization method to estimate an alpha matte by using the support vector machine (SVM) [11]. They defined a cost function to minimize on the basis of a Markov random field (MRF). The cost function consists of a matting term, a smoothing term, and a discrimination term. The matting term is defined for the fidelity to the matting equation on single pixels, and the smoothing term is defined for the local smoothness between neighboring pixels. The discrimination term is defined by using the SVM to discriminate between foreground and background pixels. The SVM uses a 15-dimensional feature vector consisting of the RGB colors of each pixel and its four nearest neighbors. The cost function is minimized by the belief propagation to estimate the alphas of unknown pixels and by a sampling method to estimate their foreground and background colors.

Zhang et al. solved an alpha matting problem as a supervised regression problem by using the support vector regression (SVR) [12]. In their method, first, an unknown region is segmented into small circular or fan-shaped pieces by a predefined pixel length r (they set $r = 30$ pixels), and the pieces are given an order to process according to their distances to definite foreground and background regions. In the process of each piece, first, candidate training samples are selected from definite foreground and background pixels and previously estimated pixels around the piece. Then, by using the similarity distance between two pixels defined by their pixel coordinates and feature vectors, m most similar and m most dissimilar pixels (they set

$m = 3$) among the candidate training samples are selected for every pixel in the piece as training samples for SVR to estimate the alphas of the pixels in the piece.

Our method proposed in this study is completely different from the above learning-based methods, although our method also takes a learning-based approach. The distinctive feature of our method is to refine an alpha matte iteratively by using a binary classifier which classifies foreground and background colors in order to finally obtain a high-quality matte. The binary classifier is first used to obtain an initial alpha matte in which the alpha of each unknown pixel is either 0 or 1. Then, the classifier is used in each iteration step to determine the confidence values of the current estimated alphas of the unknown pixels. The confidence values are used for the CF method, which is summarized in Section 2.1, to estimate better alphas. In this study, we use the SVM, which is summarized in Section 2.2, as a binary classifier.

3 Iterative refinement of alpha matte

3.1 Outline

Examples obtained by our method are shown in Figure 1 in which the binary classification (c) and final alpha matte (d) are obtained from the input image (a) and trimap (b). Our iterative refinement method estimates alphas of unknown pixels in an unknown region of a trimap according to the following three stages.

(1) Training of binary classifier

A binary classifier to classify unknown pixels into foreground and background pixels is trained by a set of training data obtained from an input color image and its trimap. We use SVM as a binary classifier, and other classifiers also can be used.

(2) Initial alpha estimation

By using the trained SVM classifier, initial alphas of unknown pixels are estimated.

Binary classification: All unknown pixels are classified into foreground and background pixels by the SVM classifier.

Confidence evaluation: Confidence values of the unknown pixels for the classification result are evaluated.

Alpha estimation: Using the confidence values, initial alphas of the unknown pixels are estimated by Eq.(12).

(3) Iterative refinement of alpha estimation

By starting with the initial alphas, alphas of the unknown pixels are refined by iterating the following three steps until the refinement converges.

Foreground and background estimation: Foreground and background colors of the unknown pixels are estimated from the latest estimated alphas.

Confidence evaluation: Confidence values of the estimated foreground and background colors for the unknown pixels are evaluated using the SVM classifier.

Alpha estimation: Using the confidence values, new alphas of the unknown pixels are estimated by Eq.(16).

The details are described as follows.

3.2 Training of binary classifier

We use an SVM classifier with a Gaussian kernel [18] to classify unknown pixels into foreground and background pixels. The classifier must be given appropriate feature vectors of pixels as training data in advance and testing data in application. We use the feature vectors described below and train the classifier as follows.

3.2.1 Feature vector

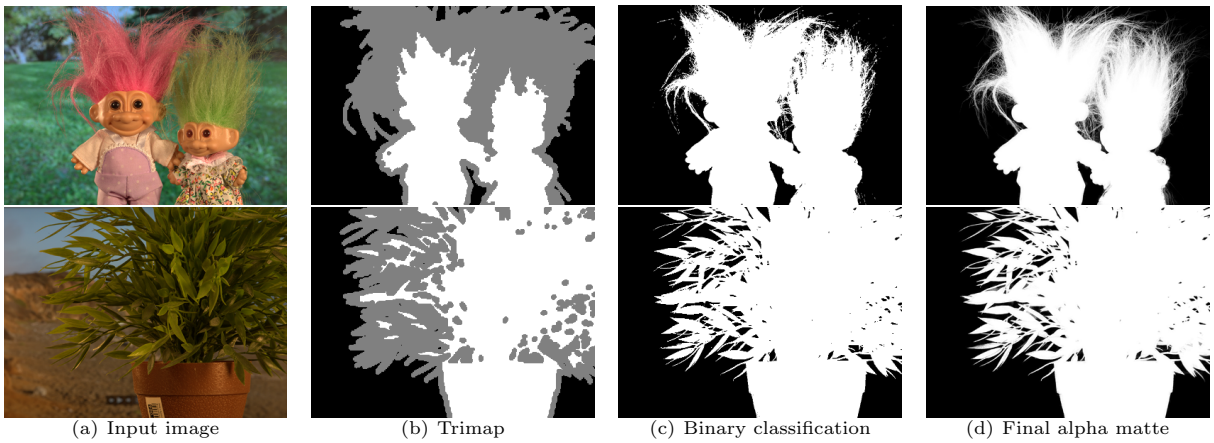
Many traditional segmentation methods distinguish foreground from background pixels on an input color image by using an RGB-color vector of each pixel as its feature vector. The RGB-color vector is a 3D vector. However, in some cases, such a feature vector does not contain adequate information for a high-quality segmentation result. In particular, distinguishing foreground and background pixels that have similar colors is very difficult. Therefore, several methods exist to enhance the information related to a feature vector. For example, the SVM-based matting method [11] uses not only a pixel's RGB-color vector but also RGB color vectors of its four neighboring pixels in order to construct a 15-dimensional feature vector.

For our SVM classifier, we enhance the information related to each pixel by using its position, RGB color, and RGB color gradient to construct its feature vector. For a pixel p_n, we define an 11-dimensional feature vector $\boldsymbol{X}_n = \{u_n, v_n, R_n, G_n, B_n, dRx_n, dGx_n, dBx_n, dRy_n, dGy_n, dBy_n\}$, where u_n and v_n are pixel coordinates; R_n, G_n, and B_n are color values; and dRx_n, dGx_n, dBx_n and dRy_n, dGy_n, dBy_n are the gradients of the color values in x and y directions, respectively.

3.2.2 Training of binary classifier using trimap

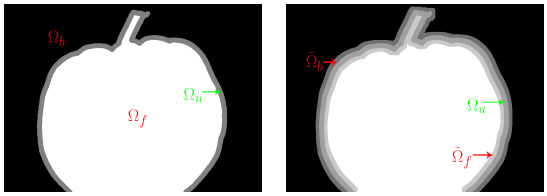
Our matting method uses a trimap to obtain an alpha matte from an input color image. A trimap divides the entire image region Ω into three subregions given by a user: a definite foreground region Ω_f, a definite background region Ω_b, and an unknown region Ω_u, as shown in Figure 1(b). A matting problem is solved in order to estimate a foreground color $\boldsymbol{F}_i$, a background color $\boldsymbol{B}_i$, and an alpha α_i for each unknown pixel $p_i \in \Omega_u$ in order to satisfy Eq.(1). Known pixels in Ω_f and Ω_b are used as clues for the solution. A pixel $p_j \in \Omega_f$ has a foreground color $\boldsymbol{F}_j$ that is the same as its image color $\boldsymbol{I}_j$ and its alpha α_j is 1. A pixel $p_k \in \Omega_b$ has a background color $\boldsymbol{B}_k$ that is the same as its image color $\boldsymbol{I}_k$ and its alpha α_k is 0.

Our SVM classifier uses an 11-dimensional feature vector $\boldsymbol{X}_n$ for a pixel $p_n \in \Omega$ to determine whether it is a foreground or background pixel. To train the classifier, a simple idea is to use feature vectors of all known pixels in Ω_f and Ω_b as training data. However, in most natural images, nearby pixels tend to have similar foreground and background colors and alphas. Thus, we select a subset $\tilde{\Omega}_f \subset \Omega_f$ as foreground training data. For every pixel $p_i \in \Omega_u$, we define a selection window w_i whose center is p_i. It has a small square shape (e.g., its size is 20×20 pixels). If a definite foreground pixel $p_j \in \Omega_f$ is in w_i, it is selected as a foreground training pixel, that is, $p_j \in \tilde{\Omega}_f$. In the



(a) Input image (b) Trimap (c) Binary classification (d) Final alpha matte

Figure 1: Examples obtained by our method. In (b), definite foreground regions are white, definite background regions are black, and unknown regions are gray. In (c), (d), intensities mean alphas in $[0, 1]$.



(a) (b)

Figure 2: Training data selected on a trimap. (a) shows a trimap. (b) shows the regions of foreground and background training data $\tilde{\Omega}_f$ and $\tilde{\Omega}_b$ along the unknown region Ω_u.

same manner, we select a subset $\tilde{\Omega}_b \subset \Omega_b$ as background training data. An example of selected subsets is shown in Figure 2. In our binary classification, for a pixel $p_n \in \tilde{\Omega} = \tilde{\Omega}_f \cup \tilde{\Omega}_b \cup \Omega_u$, we give a label $y_n = 1$ if p_n is a foreground pixel and $y_n = -1$ if p_n is a background pixel. Thus, we use foreground training data $\{(\boldsymbol{X}_j, y_j)\} = \{(\boldsymbol{X}_j, 1)\}$ of pixels $p_j \in \tilde{\Omega}_f$ and background training data $\{(\boldsymbol{X}_k, y_k)\} = \{(\boldsymbol{X}_k, -1)\}$ of pixels $p_k \in \tilde{\Omega}_b$.

After training is completed, the SVM classifier obtains an SVM function D that returns a decision value $D(\boldsymbol{X}_n)$ for a feature vector $\boldsymbol{X}_n$ of a pixel $p_n \in \tilde{\Omega}$. The function is trained to return a value $D(\boldsymbol{X}_n) \geqslant 0$ for a foreground pixel and $D(\boldsymbol{X}_n) < 0$ for a background pixel. When $|D(\boldsymbol{X}_n)|$ is large, its classification has high reliability. If $D(\boldsymbol{X}_n) \geqslant 1$ or $D(\boldsymbol{X}_n) \leqslant -1$, p_n is a foreground or background pixel, respectively, having high reliability. If $-1 < D(\boldsymbol{X}_n) < 1$, its classification reliability is low. Examples of classification are shown in Figure 1(c).

3.3 Initial alpha estimation

By means of the trained SVM classifier, initial alphas α_i^0 of unknown pixels $p_i \in \Omega_u$ can be estimated.

All unknown pixels p_i are first classified into foreground and background pixels by the signs of the decision values $D(\boldsymbol{X}_i)$. If a pixel p_i is classified to the foreground, its temporary alpha $\hat{\alpha}_i^0$ is set to 1. If p_i is classified to the background, $\hat{\alpha}_i^0$ is set to 0. In addition, $\hat{\alpha}_j^0$ of all pixels $p_j \in \tilde{\Omega}_f$ are set to 1, and $\hat{\alpha}_k^0$ of all pixels $p_k \in \tilde{\Omega}_b$ are set to 0.

A confidence value con_i^0 for each classified unknown pixel p_i is then evaluated as:

$$con_i^0 = e^{(d_i - 1)} \cdot e^{(g_i - 1)}. \tag{7}$$

The value d_i is calculated using the decision value $D_i = D(\boldsymbol{X}_i)$ as

$$d_i = \begin{cases} 1 & D_i \geqslant avg_f \\ D_i / avg_f & 0 \leqslant D_i < avg_f \\ D_i / avg_b & avg_b < D_i < 0 \\ 1 & D_i \leqslant avg_b \end{cases}, \tag{8}$$

$$avg_f = \frac{1}{|\tilde{\Omega}_f|} \sum_{p_j \in \tilde{\Omega}_f} D(\boldsymbol{X}_j), \tag{9}$$

$$avg_b = \frac{1}{|\tilde{\Omega}_b|} \sum_{p_k \in \tilde{\Omega}_b} D(\boldsymbol{X}_k). \tag{10}$$

The values $|\tilde{\Omega}_f|$ and $|\tilde{\Omega}_b|$ are the numbers of pixels in $\tilde{\Omega}_f$ and $\tilde{\Omega}_b$, respectively. The value g_i is calculated using K nearest neighbor pixels in $\tilde{\Omega}_f \cup \tilde{\Omega}_b$ to the pixel p_i on the feature vector space.

$$g_i = \begin{cases} K_f / K & D_i \geqslant 0 \\ K_b / K & D_i < 0 \end{cases}. \tag{11}$$

Among the K nearest neighbor pixels, K_f is the number of pixels $p_j \in \tilde{\Omega}_f$, and K_b is the number of pixels $p_k \in \tilde{\Omega}_b$.

Finally, using the temporary alphas $\hat{\alpha}_n^0$ of pixels $p_n \in \tilde{\Omega}$ and the confidence values con_i^0 of unknown pixels $p_i \in \Omega_u$, initial alphas α_n^0 are estimated by the CF method. The CF method transforms binary temporary alphas $\hat{\alpha}_n^0 \in \{0, 1\}$ into real initial alphas $\alpha_n^0 \in [0, 1]$ as a solution for minimizing the following optimization function.

$$\begin{aligned} \boldsymbol{\alpha}^0 = argmin_{\boldsymbol{\alpha}}\ & \boldsymbol{\alpha}^T \boldsymbol{L} \boldsymbol{\alpha} + \lambda (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^0)^T \boldsymbol{Q} \\ & (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^0) + \gamma (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^0)^T \boldsymbol{P}^0 (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^0). \end{aligned} \tag{12}$$

Let N be the total number of pixels $p_n \in \tilde{\Omega}$. The vector $\boldsymbol{\alpha}^0$ has N initial alphas α_n^0, and the vector $\hat{\boldsymbol{\alpha}}^0$ has N

temporary alphas $\hat{\alpha}_n^0$, $n = 1, \ldots, N$. The $N \times N$ matting Laplacian matrix $\boldsymbol{L}$ is obtained from the input color image given in Eq.(3). In the $N \times N$ diagonal matrix $\boldsymbol{Q}$, diagonal elements q_{nn} are 1 for $p_n \in \tilde{\Omega}_f \cup \tilde{\Omega}_b$, and 0 for others. In the $N \times N$ diagonal matrix $\boldsymbol{P}^0$, diagonal elements p_{nn}^0 are con_n^0 for $p_n \in \Omega_u$, and 0 for others. The parameters λ and γ are used for weighting and normalizing; usually, λ is large (e.g., 100), and γ is small (e.g., 0.1). The matrices $\boldsymbol{Q}$ and $\boldsymbol{P}^0$ constrain α_n^0 to remain $\hat{\alpha}_n^0$. The initial alphas α_n^0 of $p_n \in \tilde{\Omega}_f \cup \tilde{\Omega}_b$ are strongly constrained by $\boldsymbol{Q}$ and λ. The matrix $\boldsymbol{P}^0$ constrains α_n^0 of $p_n \in \Omega_u$ strongly if con_n^0 is large. As a solution to Eq.(12) for $p_n \in \tilde{\Omega}_f \cup \tilde{\Omega}_b$, α_n^0 can be neither 0 nor 1. However $\hat{\alpha}_n^0$ is either 0 or 1, and should remain as such. Thus, we do not accept the solution and instead set α_n^0 to $\hat{\alpha}_n^0$. We use the solution of α_n^0 only for $p_n \in \Omega_u$.

3.4 Iterative refinement of alpha estimation

The alpha matte of the initial alphas α_i^0 often possesses poor quality. Therefore, by starting with α_i^0, our matting method refines the alphas α_i of unknown pixels p_i iteratively by using the SVM classifier again until the refinement converges.

Let α_i^t be the estimated alpha of an unknown pixel p_i after t iteration steps, $t = 1, 2, \ldots, t_{max}$. In the t-th iteration step, foreground and background colors $\boldsymbol{F}_i^{t-1}$ and $\boldsymbol{B}_i^{t-1}$ of unknown pixels p_i are first estimated from the latest alphas α_i^{t-1} and the image colors $\boldsymbol{I}_i$. We use the reconstruction method proposed by the CF method [6] for this foreground and background color estimation as Eq.(5), although other methods can be used.

Next, we evaluate the confidence of the alphas α_i^{t-1} by evaluating the confidence of the estimated colors $\boldsymbol{F}_i^{t-1}$ and $\boldsymbol{B}_i^{t-1}$ instead. We construct the feature vectors of these colors and classify them using the SVM classifier. Ideally, these colors should be classified to foreground or background colors with high reliability. We consider that reliable foreground and background colors guarantee a reliable alpha. We calculate the confidence value con_i^t for each unknown pixel p_i as

$$con_i^t = conf_i^t \cdot conb_i^t, \tag{13}$$

where $conf_i^t$ and $conb_i^t$ are the sub-confidence values for the colors $\boldsymbol{F}_i^{t-1}$ and $\boldsymbol{B}_i^{t-1}$. These are obtained using the decision values $Df_i^{t-1} = D(\boldsymbol{X}(\boldsymbol{F}_i^{t-1}))$ and $Db_i^{t-1} = D(\boldsymbol{X}(\boldsymbol{B}_i^{t-1}))$ returned by the SVM classifier, where $\boldsymbol{X}(\boldsymbol{F}_i^{t-1})$ and $\boldsymbol{X}(\boldsymbol{B}_i^{t-1})$ are the feature vectors of $\boldsymbol{F}_i^{t-1}$ and $\boldsymbol{B}_i^{t-1}$ which include positions, RGB colors and RGB color gradients.

$$conf_i^t = \begin{cases} 1 & Df_i^{t-1} \geqslant 1 \\ \frac{1-cos(\pi Df_i^{t-1})}{2} & 0 < Df_i^{t-1} < 1 \\ 0 & Df_i^{t-1} \leqslant 0 \end{cases}, \tag{14}$$

$$conb_i^t = \begin{cases} 1 & Db_i^{t-1} \leqslant -1 \\ \frac{1-cos(\pi Db_i^{t-1})}{2} & -1 < Db_i^{t-1} < 0 \\ 0 & Db_i^{t-1} \geqslant 0 \end{cases} \tag{15}$$

Finally, we use the latest alphas α_n^{t-1} of pixels $p_n \in \tilde{\Omega}$ and confidence values con_i^t of unknown pixels $p_i \in \Omega_u$ to estimate new alphas α_n^t by the CF method.

$$\begin{aligned} \boldsymbol{\alpha}^t = argmin_{\boldsymbol{\alpha}}\ & \boldsymbol{\alpha}^T \boldsymbol{L} \boldsymbol{\alpha} + \lambda (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{t-1})^T \boldsymbol{Q} \\ & (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{t-1}) + \gamma (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{t-1})^T \boldsymbol{P}^t (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{t-1}). \end{aligned} \tag{16}$$

The matrices $\boldsymbol{L}$ and $\boldsymbol{Q}$ and the parameters λ and γ are the same as in Eq.(12). The vector $\boldsymbol{\alpha}^t$ has N alphas α_n^t, and the vector $\boldsymbol{\alpha}^{t-1}$ has N alphas α_n^{t-1}, $n = 1, \ldots, N$. In the $N \times N$ diagonal matrix $\boldsymbol{P}^t$, diagonal elements p_{nn}^t are con_n^t for $p_n \in \Omega_u$, and 0 for others. The matrix $\boldsymbol{P}^t$ constrains α_n^t to remain α_n^{t-1} by con_n^t for $p_n \in \Omega_u$. This constraint is stronger as con_n^t becomes larger. In the same manner as described in Section 3.3, we use the solution of Eq.(16) of α_n^t only for $p_n \in \Omega_u$, and retain α_n^t as 0 or 1 for $p_n \in \tilde{\Omega}_f \cup \tilde{\Omega}_b$.

As the iteration proceeds, the quality of the estimated alpha matte improves. The iteration stops when the difference between $\boldsymbol{\alpha}^{t-1}$ and $\boldsymbol{\alpha}^t$ is smaller than the predefined threshold.

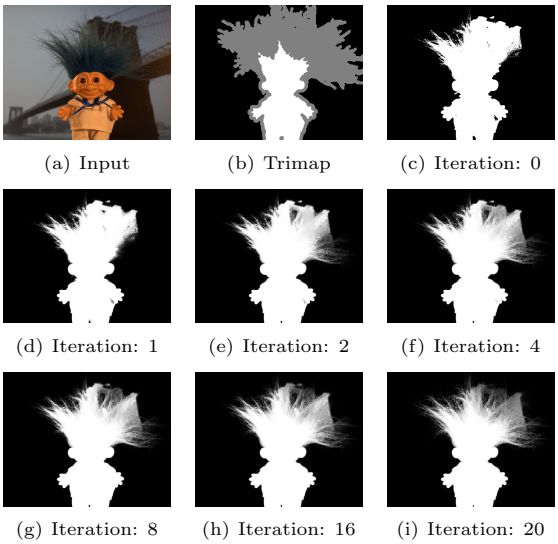
4 Experimental results

The algorithm of this study was implemented using MATLAB and executed with an Intel i7 3.4GHz CPU and 16GB memory. In addition, an SVM library, LIBSVM, was used, and the parameters of the svmtrain function were given the default settings introduced by [18].

Figure 1(c), Figure 3(c) and Figure 4(c) show examples of the binary classification by the SVM. These examples reveal the fundamental excellent ability of the SVM. All unknown pixels are classified into either foreground pixels or background pixels reasonably. These binary classification images seem to be rough approximations of accurate alpha mattes.

Figure 3 and Figure 4 show iterative refinement processes of alpha mattes obtained by our method. The input images, "Troll" and "GT25", are available on a well-known matting evaluation website [21]. In each figure, the alpha matte is gradually refined as the iteration proceeds, and the refinement converges. These results show that the iterative refinement mechanism based on the SVM in our method works well.

In the following, we compare our method with several well-known methods: CF matting [6], learning based digital matting (LB) [10], KNN matting [14], SVR matting [12], LNSP matting [15], color clustering matting (CCM) [16], and comprehensive sampling (CS) matting [17]. In order to evaluate resulting alpha mattes, we used four types of errors: Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient error (Grad.), and Connectivity error (Con.), which are discussed in [22] and used on the matting website [21]. Among these methods, the SVR, LNSP, CCM, and CS are current high-ranking methods on the matting website and on average provide high-quality results for the four error types. The LB and KNN are also well-known excellent methods, and their source codes are available.

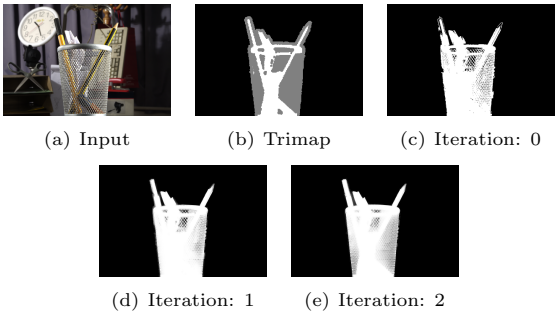


(a) Input (b) Trimap (c) Iteration: 0

(d) Iteration: 1 (e) Iteration: 2 (f) Iteration: 4

(g) Iteration: 8 (h) Iteration: 16 (i) Iteration: 20

Figure 3: Iterative refinement of "Troll". (c) is the classification result and (d)–(i) are the refinement results.



(a) Input (b) Trimap (c) Iteration: 0

(d) Iteration: 1 (e) Iteration: 2

Figure 4: Iterative refinement of "GT25". (c) is the classification result and (d)–(e) are the refinement results.

The matting website [21] provides useful benchmark input images, including "Troll" and "GT25" in Figures 3 and 4. Among these images, we used the images in the "GT dataset" to compare our method with the CF, LB, and KNN methods, whose source codes are available. Figures 5, 6 and 7 show the graphs for the comparisons of the SAD, MSE, and Grad. values respectively obtained by applying the four methods to the 27 input images in the GT dataset. In each graph, the horizontal axis indicates the index numbers of the images, and the vertical axis indicates error values. For each input image, the error values are normalized so that the error value by the CF method is one. Judging from the graphs, the CF and LB methods tend to provide similar error values in all the three error types. Compared to the CF and LB methods, the KNN method and our method tend to have large changes in error values. The KNN method tends to provide larger error values than the other methods while it provides small error values for some input images. As a whole, our method tends to provide the smallest error values among these methods. This means that our method has an ability to generate high-quality alpha mattes while it generates poor-quality alpha mattes for some input images.

Figures 8, 9, 10 and 11 show the resulting alpha mattes obtained by applying the CF, LB, and KNN methods and our method to several input images selected from the 27 images. The ground truth images in the figures are provided in the matting website [21]. Tables 1, 2 and 3 show the SAD, MSE, and Grad. values of the alpha mattes respectively. Our method achieves the smallest error values in all the three error types for the 8 input images used in Figures 8, 9 and 10. This means that, from the viewpoint of error values, our method generates the highest-quality alpha mattes for these input images. On the other hand, our method provides large error values for the 3 input images used in Figure 11. This means that our method generates poor-quality alpha mattes.

The "GT04" image in Figure 8 has a part with similar foreground and background colors given by green hair and green grass. The KNN method and our method successfully generate visually reasonable alpha mattes on the part, that is, the region including the hair of the two dolls in the middle of the zoomed images, while the CF and LB methods fail. The "GT11" image in Figure 9 also has a part with similar foreground and background colors in the zoomed region. Judging from the error values, in Figure 9 as well as Figure 8, our method achieves the best result although it is difficult to distinguish the results of the four methods visually. Figure 10 shows other cases in which the results of our method are better than those of the other methods from the viewpoint of error values. In particular, we find visually that our method works well for the "GT02" and "GT26" images having foreground objects with many small holes while the CF and LB methods fail to treat them. We consider that, in our method, the SVM succeeded in classifying foreground and background colors correctly even for the cases in which there were many small holes as well as the cases in which foreground and background colors were similar.

On the other hand, Figure 11 shows some cases in which our method generates poor-quality results from the viewpoint of error values although it is difficult to visually distinguish their qualities from those by the other methods. In order to easily compare the qualities achieved by the four methods, we show two kinds of error maps for the "GT08" image in Figure 12. The top error maps show, for each pixel, the difference dif_x between the alpha α_x obtained by the method x, which is CF, LB, KNN, or Ours, and the alpha α_{gt} of the ground truth: $dif_x = \alpha_x - \alpha_{gt}$. The red pixels mean $dif_x > 0$, the blue pixels mean $dif_x < 0$, and the black pixels mean $dif_x = 0$. A brighter red or blue color means a larger absolute difference. The bottom error maps show the difference $difg_x$ between the gradients of the alphas α_x and α_{gt}: $difg_x = grad\alpha_x - grad\alpha_{gt}$. The red, blue, and black pixels have similar meanings to the top error maps. We consider that, in these cases, it was difficult for the SVM to appropriately classify foreground objects (e.g., long and thin hair, and fine fur) and backgrounds because of the lack of training data to adequately learn their properties.

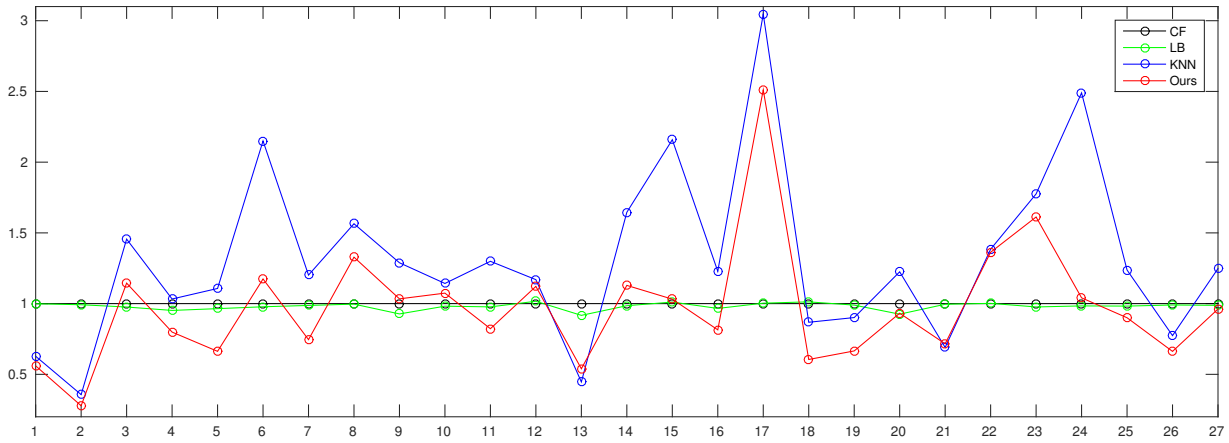


Figure 5: Comparison of SAD errors for GT dataset images. The horizontal axis indicates the index numbers of the images. The vertical axis indicates SAD_x/SAD_{CF}, $x \in \{CF, LB, KNN, Ours\}$.

Table 1: SAD errors for GT dataset images.

	GT01	GT02	GT04	GT05	GT07	GT08	GT11	GT17	GT18	GT23	GT26
CF	1131.2	4308.2	12095.5	1262.5	2150.7	11453.6	3630.5	1782.3	2188.1	1608.7	18056.2
LB	1130.1	4273.7	11513.5	1217.6	2122.9	11411.2	3546.8	1787.9	2215.3	1570.7	17881.2
KNN	710.1	1536.8	12490.4	1397.7	2584.3	17922.2	4716.5	5429.3	1899.6	2856.1	13971.4
Ours	633.5	1197.7	9661.8	836.4	1603.0	15212.8	2986.3	4476.3	1324.1	2593.8	11955.5

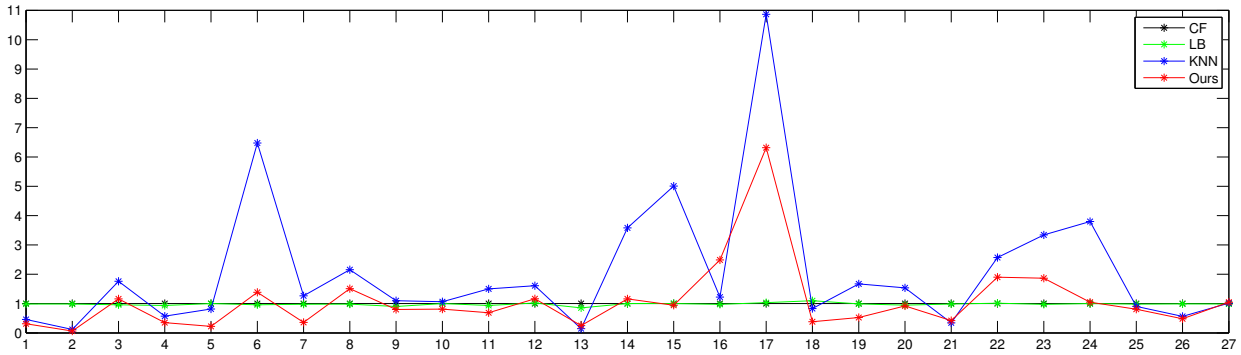


Figure 6: Comparison of MSE errors for GT dataset images. The horizontal axis indicates the index numbers of the images. The vertical axis indicates MSE_x/MSE_{CF}, $x \in \{CF, LB, KNN, Ours\}$.

Table 2: MSE errors for GT dataset images.

	GT01	GT02	GT04	GT05	GT07	GT08	GT11	GT17	GT18	GT23	GT26
CF	0.00054	0.0071	0.0091	0.0012	0.00087	0.0051	0.0016	0.0004	0.001	0.00025	0.0174
LB	0.00053	0.007	0.0085	0.0012	0.00085	0.005	0.0015	0.00041	0.0011	0.00025	0.0172
KNN	0.00025	0.00084	0.0052	0.00098	0.0011	0.011	0.0024	0.0043	0.00083	0.00084	0.0098
Ours	0.00017	0.00046	0.0032	0.00027	0.00031	0.0077	0.0011	0.0025	0.00038	0.00047	0.0084

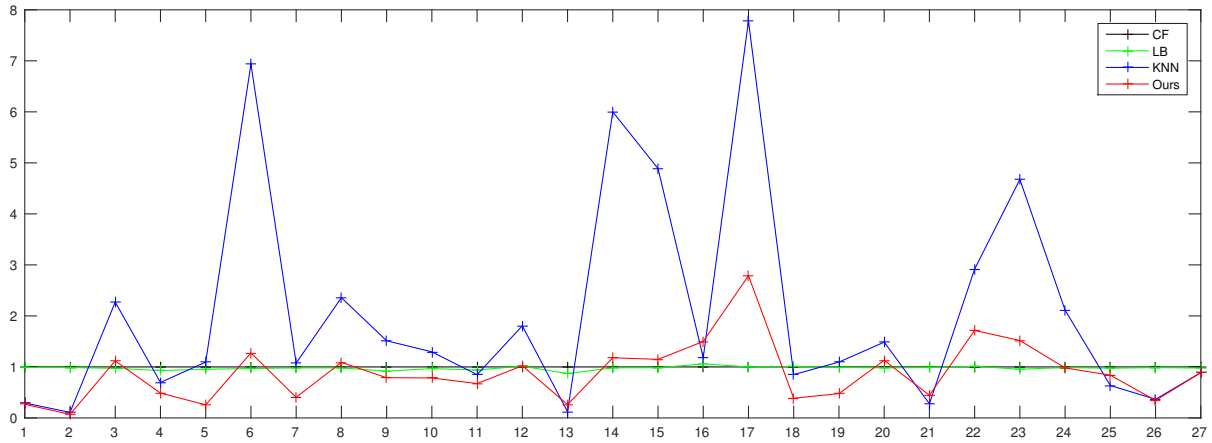
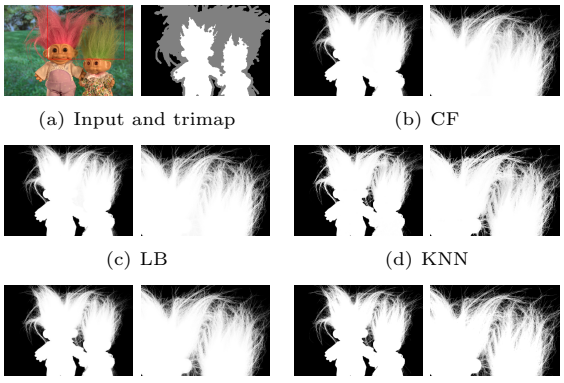


Figure 7: Comparison of Gradient errors for GT dataset images. The horizontal axis indicates the index numbers of the images. The vertical axis indicates $Grad_x/Grad_{CF}$, $x \in \{CF, LB, KNN, Ours\}$.

Table 3: Gradient errors for GT dataset images.

	GT01	GT02	GT04	GT05	GT07	GT08	GT11	GT17	GT18	GT23	GT26
CF	1776.6	21005	9610.8	2554.6	1868.8	5039.9	5046	764.1	2711.1	671.6	48368
LB	1754.7	20733	8925.3	2438.1	1834.3	4945.6	4744	764.2	2732.5	641.5	47720
KNN	521.9	2287.1	6686	2795.5	2010.7	11880	4281.6	5944.9	2304.3	3139.3	17682
Ours	476.7	1476.8	4680	657.5	756.7	5483.1	3401.9	2131.2	1048.3	1019.5	16564

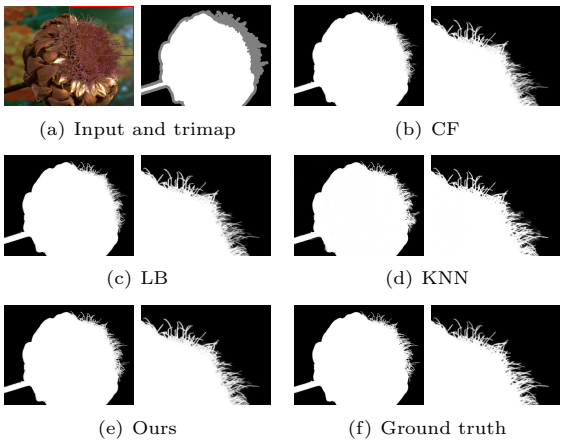


(a) Input and trimap (b) CF

(c) LB (d) KNN

(e) Ours (f) Ground truth

Figure 8: Alpha mattes for “GT04” image.

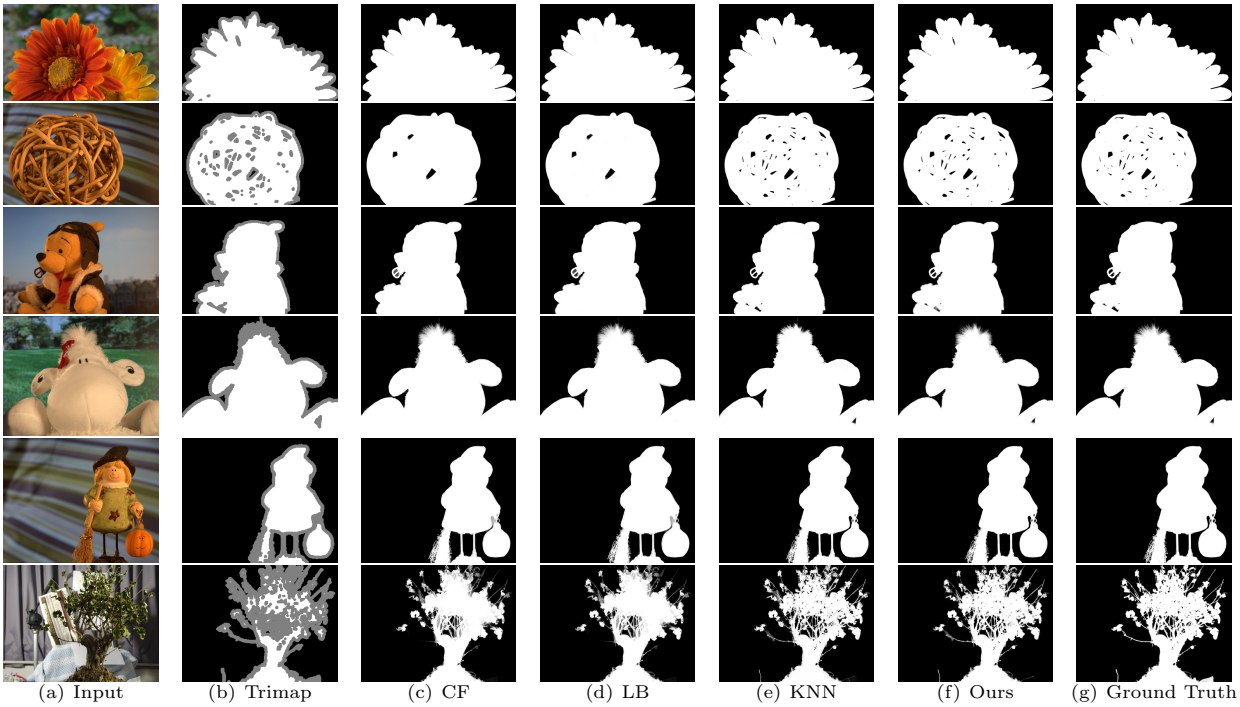


(a) Input and trimap (b) CF

(c) LB (d) KNN

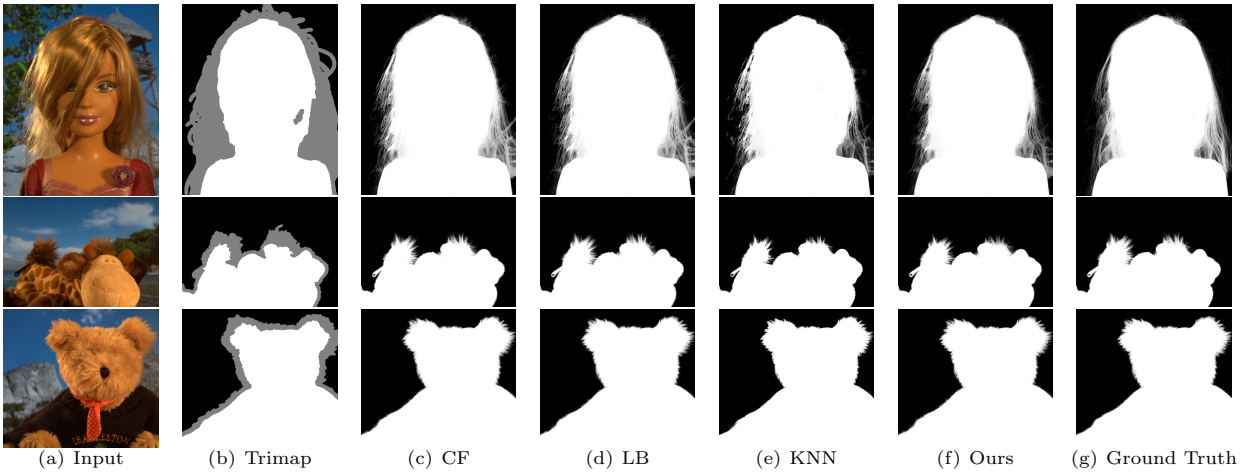
(e) Ours (f) Ground truth

Figure 9: Alpha mattes for “GT11” image.



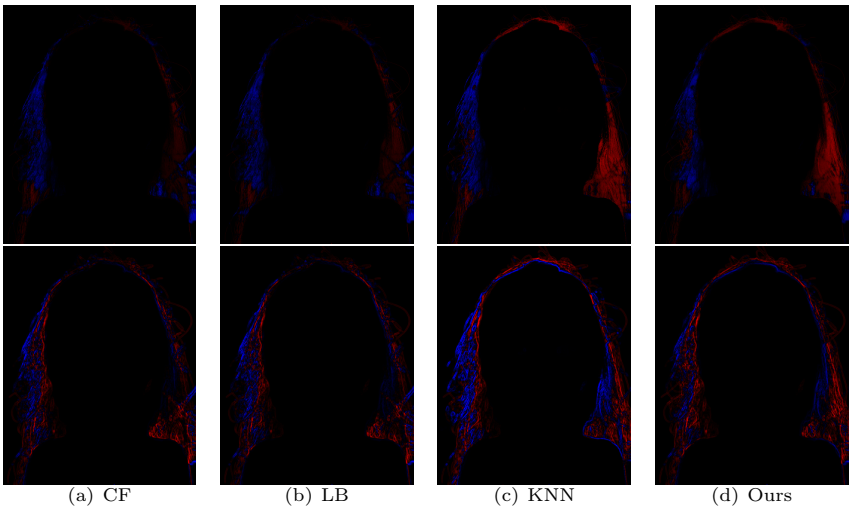
(a) Input (b) Trimap (c) CF (d) LB (e) KNN (f) Ours (g) Ground Truth

Figure 10: Alpha mattes for GT dataset images (from the top, GT01, GT02, GT05, GT07, GT18, and GT26) with small errors by our method.



(a) Input (b) Trimap (c) CF (d) LB (e) KNN (f) Ours (g) Ground Truth

Figure 11: Alpha mattes for GT dataset images (from the top, GT08, GT17, and GT23) with large errors by our method.



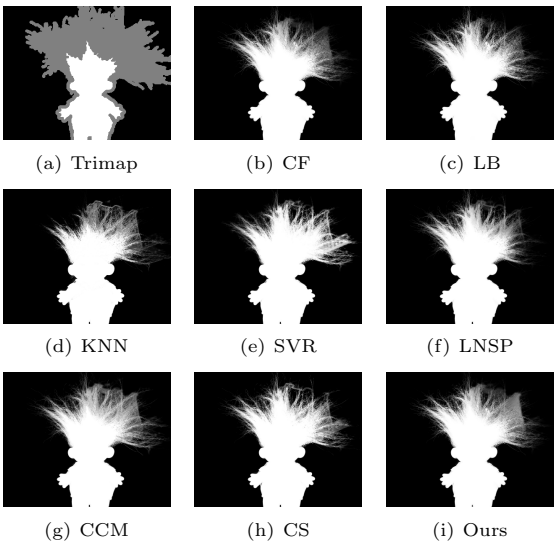
(a) CF (b) LB (c) KNN (d) Ours

Figure 12: Error maps for "GT08" image. The top maps show, for each pixel, the difference between alphas by the respective methods and the ground truth. The bottom maps show the difference between the gradients of alphas.

Figure 13 and Table 4 show the resulting alpha mattes and error values obtained by applying not only the CF, LB, and KNN methods and our method but also the SVR, LNSP, CCM, and CS methods to the "Troll" image in Figure 3. The alpha mattes and error values by the methods except for our method are available on the matting website [21]. On the website, the "Troll" image is provided as a typical image with "Strongly Transparent" objects. In addition, the hair part in front of the bridge is so difficult to treat to obtain its accurate alpha matte because of not only the fine detailed hair but also the similar foreground and background colors. As shown in Figure 13, no method achieves a high-quality result. The poor qualities of the resulting alpha mattes depend on methods. The results of some methods have artifacts on the hair part caused by the outlines of the bridge. The results of some methods including ours have larger alphas uniformly on the hair part. However, judging from the error values in Table 4, our method competes even with the current high-ranking methods (SVR, LNSP, CCM, and CS). In the same way as Figure 11, we consider that the poor-quality result by our method was caused mainly by the lack of training data. Figure 14 shows an alpha matte obtained by applying our method to the "Troll" image by using a trimap different from the trimap used in Figure 13. While the trimap in Figure 13 is provided in the matting website [21], the trimap in Figure 14 was created by us so as to give additional definite background regions inside the unknown region of the trimap in Figure 13. By comparing Figure 14(b) with Figure 13(i), we find that an improved alpha matte was obtained by the new trimap that provided more clues, which were used as more useful training data for the SVM.

Table 4: Errors for "Troll" image.

	SAD	MSE	Grad.	Con.
CF	12.7	0.5	0.3	0.8
LB	16.0	0.8	0.3	0.8
KNN	16.2	0.8	0.3	3.4
SVR	18.7	1.1	0.3	1.2
LNSP	12.2	0.5	0.2	0.8
CCM	13.8	0.5	0.2	0.7
CS	11.2	0.4	0.2	1.0
Ours	11.9	0.5	0.2	0.7

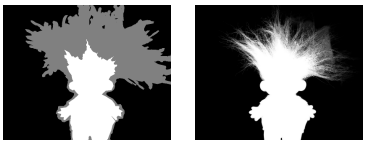


(a) Trimap (b) CF (c) LB

(d) KNN (e) SVR (f) LNSP

(g) CCM (h) CS (i) Ours

Figure 13: Alpha mattes for "Troll" image.



(a) Trimap (b) Alpha matte

Figure 14: An alpha matte for "Troll" image by our method using a different trimap.

From the above experiments, we find that our method generates high-quality alpha mattes particularly for input images having similar foreground and background colors and objects with small holes. For several input images, our method performs more effectively than other methods and competes even with high-ranking methods. Our method relies on the ability of a binary classifier. Thus, the quality of a resulting alpha matte depends on the performance of the classifier. When the learning of the classifier by training data is not enough, the quality of the alpha matte tends to become poor. The effective way of the learning of the classifier needs to be explored in future.

5 Conclusion

In this study, we proposed a learning-based matting method. In our method, an image matting problem is first treated as a binary classification problem, and a binary alpha matte is obtained through a binary classifier. The binary alpha matte is then iteratively refined to a high-quality alpha matte by repeat use of the classifier. Our method was performed better than other methods in experiments for several input images while it did not work well for some input images.

Although we used an SVM classifier and the CF method, an important merit of our method is that it provides a general mechanism to refine an alpha matte by combining various binary classifiers and matting methods. This represents a promising ability to achieve better results than those currently obtained, and we plan to investigate other combinations. Besides, an effective way of the learning of a classifier to classify foreground and background colors correctly needs to be explored. In addition, we plan to apply our method to high-resolution images efficiently. One idea is to use an effective trimap segmentation method.

References

[1] Alvy Ray Smith and James F. Blinn. Blue screen matting. *Association for Computing Machinery, Inc.*, pages 259–269, 1996.

[2] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. *Proc. 10th IEEE Int'l Conf. Computer Vision*, 2005.

[3] Jue Wang and Michael F. Cohen. Image and video matting: A survey. *Found. Trends. Comput. Graph. Vis.*, 3:97–175, 2007.

[4] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. *Proceedings of IEEE CVPR 2001*, 2:264–271, 2001.

[5] Jian Sun, Jiaya Jia, Chi-Keung Tangand, and Heung-Yeung Shum. Poisson matting. *ACM Trans. Graph.*, 23:315–321, 2004.

[6] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 30:228–242, 2008.

[7] Jue Wang and Michael F. Cohen. Optimized color sampling for robust matting. *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[8] Eduardo S. L. Gastal and Manuel M. Oliveira. Shared sampling for real-time alpha matting. *Computer Graphics Forum*, 29:575–584, 2010.

[9] Kaiming He, Jian Sun, and Xiaoou Tang. Fast matting using large kernel matting laplacian matrices. *Computer Vision and Pattern Recognition (CVPR)*, pages 2165 – 2172, 2010.

[10] Ye Zheng and Kambhamettu C. Learning based digital matting. *Computer Vision, 2009 IEEE 12th International Conference*, pages 889–896, 2009.

[11] Tadaaki Hosaka, Takumi Kobayashi, and Nobuyuki Otsu. Image matting based on local color discrimination by svm. *Pattern Recognition Letters*, 30:1253 – 1263, 2009.

[12] Z. Zhang, Q. Zhu, and Y. Xie. Learning based alpha matting using support vector regression. *ICIP*, 2012.

[13] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun. A global sampling method for alpha matting. *Computer Vision and Pattern Recognition (CVPR)*, pages 2049 – 2056, 2011.

[14] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 869–876, 2012.

[15] X. Chen, D. Zou, and P. Tan. Image matting with local and nonlocal smooth priors. *CVPR*, 2013.

[16] Y. Shi, O.C. Au, J. Pang, K. Tang, W. Sun, H. Zhang, W. Zhu, and L. Jia. Color clustering matting. *ICME*, 2013.

[17] E.Shahrian, D.Rajan, B.Price, and S.Cohen. Improving image matting using comprehensive sampling sets. *CVPR*, 2013.

[18] Chih wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[19] S.Keerthi and C. j. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15:1667–1689, 2003.

[20] C. Chang and C. Lin. Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[21] alpha matting evaluation website. http://www.alphamatting.com/index.html.

[22] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Zixiang Lu

Zixiang Lu is a master course student in Design and Media Technology at the Graduate School of Engineering, Iwate University from 2014. He received his bachelor degree from North West Agriculture and Forestry University in 2013. His research interest includes image processing, computer vision and data mining.

Tadahiro Fujimoto

Tadahiro Fujimoto is currently an associate professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics and computer vision. He received a BE in electrical engineering, and an ME and Ph.D. in computer science from Keio University in 1990, 1992, and 2000, respectively. He worked at Mitsubishi Research Institute from 1992 to 1995. He was a research associate in the Department of Computer and Information Sciences at Iwate University from 1999 to 2002, and a lecturer from 2002 to 2005. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE, and ACM.