electronics

MDPI

Article

Interactive Trimap Generation for Digital Matting Based on Single-Sample Learning

Zhenpeng Chen [1,2], Yuanjie Zheng [1,2,*], Xiaojie Li [1,2,*], Rong Luo [3,*], Weikuan Jia [1,2,*], Jian Lian [4] and Chengjiang Li [4]

1 School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China; 18366108084@163.com
2 Key Lab of Intelligent Computing and Information Security in Universities of Shandong, Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Institute of Biomedical Sciences, Shandong Normal University, Jinan 250358, China
3 School of Light Industry Science and Engineering, Qilu University of Technology, Jinan 250351, China
4 Department of Electrical Engineering and Information Technology, Shandong University of Science and Technology, Jinan 250031, China; lianjianlianjian@163.com (J.L.); li_chengjiang@163.com (C.L.)
* Correspondence: yjzheng@sdnu.edu.cn (Y.Z.); xli162011@163.com (X.L.); lrcity@163.com (R.L.); wkjia@sdnu.edu.cn (W.J.)

Received: 19 February 2020; Accepted: 13 April 2020; Published: 17 April 2020

Abstract: Image matting refers to the task of estimating the foreground of images, which is an important problem in image processing. Recently, trimap generation has attracted considerable attention because designing a trimap for every image is labor-intensive. In this paper, a two-step algorithm is proposed to generate trimaps. To use the proposed algorithm, users must only provide some clicks (foreground clicks and background clicks), which are employed as the input to generate a binary mask. One-shot learning technique achieves remarkable progress on semantic segmentation, we extend this technique to perform the binary mask prediction task. The mask is further used to predict the trimap using image dilation. Extensive experiments were performed to evaluate the proposed algorithm. Experimental results show that the trimaps generated using the proposed algorithm are visually similar to the user-annotated ones. Comparing with the interactive matting algorithms, the proposed algoritm is less labor-intensive than trimap-based matting algorithm and achieved more accuate results than scribble-based matting algorithm.

Keywords: trimap; image matting; deep learning; one-shot learning; image segmentation

1. Introduction

Matting is a process of extracting a foreground object image F along with its opacity mask α (typically called alpha matte) from a given digital photograph I, which plays an important role in video editing and image processing. Specifically, the image matting problem is modelled as a convex combination of a foreground image F and a background image B as given in Equation (1).

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \tag{1}$$

where $i = (x, y)$ is the image lattice. The value of α is between 0 and 1; if $\alpha_i = 1$ or 0, then the pixel at location i belongs to the definite foreground or definite background, respectively. Otherwise, the pixel belongs to the opacity mask. This is the main difference in comparison with image segmentation, which has no pixels between the foreground and the background. The ill-posed problem in image matting is under-constrained since F, B and α are all unknown. Note that in Equation (1), if we consider a full color image (RGB), there are seven unknown parameters (F, B for each channel and

α). These under-constrained problems can be solved by adding more interactive information into it. This additional information is provided in the form of a trimap [1] or scribbles [2] (as shown in Figure 1b,e).

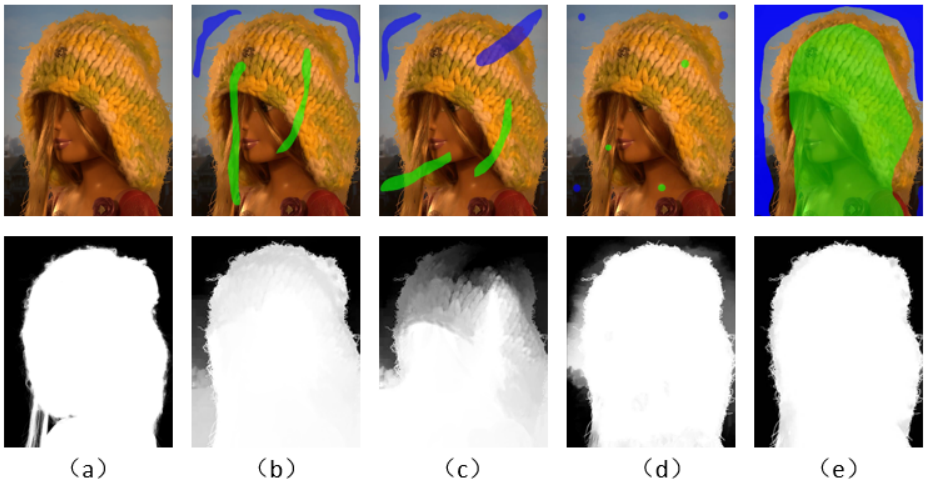

Figure 1. Existing image matting methods require the user to design additional constraints for specifying the foreground and background color samples. (**a**) Original image and groundtruth alpha matte. (**b**–**e**) The first row, from left to right shows different constraint information provided by the user. The second row shows their respective predicted alpha matte.

Some pixels are annotated as belonging to definite foreground or definite background. Scribbles only provide a small amount of interaction information, but trimaps can provide more complete interaction information. However, drawing such precise trimaps requires considerable human effort, which is often undesirable, particularly in the case of opaque objects. For example, drawing a trimap like the first row in Figure 1e will take about 50 s. Scribbles are easy to obtain, but they are error prone and inaccurate (Figure 1c) when the foreground scribble pixels are mixed with background pixels, then the generated alpha matte is inaccurate or even wrong. Relatively, user-clicks are more robust than scribbles. To fully extract meaningful foreground objects and minimize the user's annotation workload, our algorithm takes advantage of a few user-provided clicks and directly generates trimaps for image matting (Figure 1d) .

The spectral matting algorithm [3] and automatic trimap generation algorithm [4] automatically extract the matte from the input image without any user intervention. However, the limitation of these methods is that they assume that there is one single object present in the given scene. When multiple semantic targets appear in a scene, the generated result is not the user's interest region. For example, cats and dogs could appear in the same image. If we want to obtain the alpha matte for cats, the dogs should be seen as the background. The generated result cannot correctly address the user's interest in these cases. Hsieh et al. [5] proposed a method to automatically obtain trimaps, but their algorithm took the original image and the segmentation results as the input, which is the segmentation result obtained by the user's interaction. Therefore, ref. [5] requires similar constraint information for the base scribble algorithm. Computers cannot replace us when deciding which parts to target. Providing interactive information is a prerequisite for implementing the algorithm.

In actual image matting applications, we aim to obtain a large number of images' alpha mattes. Many of these images belong to the same semantic classes. The same semantic classes have similar characteristics. Previous methods needed to design trimaps for every image. Collecting these dense constraints for every image is another problem that is time consuming, tedious, and error-prone.

After the above analysis, we aimed to reduce the amount of user interaction while maintaining alpha matte accuracy. Our goal was to obtain the trimap for any image in an unknown class under, the condition of only one image, and the corresponding pixel level click annotation. We were inspired

by one-shot learning and propose a three-branched model to generate trimaps. Our model consists of three branches: the guided branch to extract the guidance from the annotated image, the inference branch obtains the segmentation results of the unlabeled image given guidance, and the generated branch converts the segmentation results into a trimap (Figure 2) .

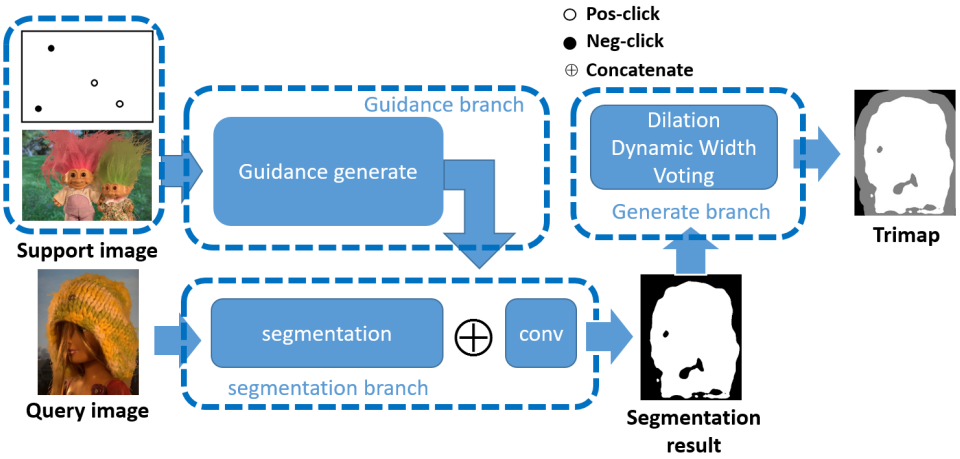

Figure 2. Proposed model overview.

To summarize, the main contributions of our work are three-fold: (1) To the best of our knowledge, this is the first algorithm to connect one-shot learning with trimap generation, achieving relatively accurate results. (2) We use an algorithm to generate trimaps, which further reduces user workload. By implementing the proposed algorithm, users do not need to design the trimaps; only a few clicks are needed. (3) Our algorithm can generate information to guide the segmentation process, which means we can obtain their trimaps based on the interactive information for one single image provided by users for any images from unseen semantic classes.

2. Related Work

Various digital matting approaches, like Bayesian matting [1], learning-based matting [6], and closed-form matting [7], require trimaps to be specified by the user. The GrubCut [8] and Lazy Snapping [9] algorithms use the graph-cut-based optimization approach to extract foregrounds from images according to a small amount of user input, such as a few strokes or a bounding box. Wang et al. [2] combined the segmentation and digital matting and presented a unified optimization approach based on belief propagation. Wang et al. [10] designed an interactive tool (Soft Scissors) to obtain high quality image matting. Cho et al. [11] built a deep neural network to learn the matching relationship between the inputs and predicted alpha mattes (obtained by [7,12]). Xu et al. [13] proposed a deep-learning-based model to improve the performance of the matting algorithm by tackling the colors and textures. Since digital matting is an under-constrained problem, all the matting algorithms require user interaction to solve matting problems. However, designing such a relatively accurate trimap is time consuming, which is unsuitable for practical applications. The other interactive approach, the scribble-based approach, lacks robustness. To fill this gap, we propose a method to generate trimaps from user clicks. The final result, accurate alpha mattes, is calculated by the generated trimap in the previous step.

A series of trimap generation studies [4,14–18] obtained trimaps through user interaction or automatically. Rhemann et al. [14] presented a new approach using parametric max-flow to generate trimaps; the final alpha matte was obtained by the gradient preserving prior. Shahrian et al. [15] presented a new sampling strategy to construct a comprehensive sampling set of known samples by sampling all the color distribution in the known region. Cho et al. [16] presented a trimap generation approach for light-field images. They took advantage of the binary segmentation result to obtain coarse trimaps. Then, they optimized the trimaps by analyzing the color distribution along the

boundary of the segmentation result. Gupta et al. [4] employed superpixels to over-segment the image. Then, they used a saliency map and local feature descriptor to help automatically generate trimaps. Juan et al. [17] proposed a segmentation algorithm to extract a relatively accurate trimap from coarse indication. Then, they took advantage of the generated trimap to design an improved matting method to produce a better alpha matte. Gastal et al. [19] presented a real-time matting algorithm for natural images and videos. The algorithm was based on the sampling technique, making full use of the similarity between adjacent pixels. The inherent parallelism of the algorithm was combined. The fundamental difference between our method and previous trimap generation algorithms is the guidance ability, which means that for any image of the same semantic class, we can obtain their alpha matte by providing only one scribbled image. A series of studies [20–22] refined the trimaps as a preprocessing step by expanding the foreground and background starting from their boundaries with the unknown region, which was proposed by Shahrian et al. [15]. Shen et al. [18] proposed a deep automatic matting method, which can generate trimaps for portrait images by deep network, however their system unable to handle other semantic classes.

Recently, digital image processing technology has developed rapidly, and many image processing algorithms have emerged. Zhu et al. [23] proposed a novel hashing approach to deal with scalable image retrieval problems. Fang et al. [24] solved imagery de-noising problems using discriminative representations. Zong et al. [25] proposed a novel multiple description image coding model to improve coding efficiency. Jiang et al. [26] proposed a matching-based method for aligning multimodal images. Zhao et al. [27] built an efficient image feature representation method (ASD) to detect images. They presented another multi-trend structure descriptor [28], which was built based on the local and multi-trend structures to further improve detection accuracy. Liu et al. [29] presented a novel computer-aided design system based on a computational approach to producing 3D images for stimulating the creativity of designers. Wang et al. [30] presented a novel just noticeable (JND) model that satisfies the visual perception characteristics of human eyes and matched the spread spectrum transform jitter modulation (STDM) watermark framework. Deng et al. [31] presented a graph-cut-based method for automated aorta segmentation.

Learning-based algorithms have achieved great success in the field of image processing. Li et al. [32] applied kernel learning to achieve face recognition. Some learning-based segmentation algorithms, such as U-NET [33], FCN [34], MASK R-CNN [35], and DeepLab [36], can accurately separate the foreground object from the background. However, these algorithms are trained with full annotations and require investments in expensive labeling tasks. To reduce the annotation workload, a promising alternative method is to apply weak annotations for learning, e.g., bounding boxes [37] and points [38]. The main disadvantage of weakly supervised methods is that they lack the ability to generalize unseen classes. For example, if a network is trained to segment cats using many images containing various breeds of cats, it will not be able to segment vehicles without fine-tuning the network using many images containing vehicles. Therefore, researchers extensively focused on generalizing new class objects so they can minimize labeling costs and improve the use efficiency of labeled samples.

Humans have relatively good cognitive abilities. Humans can recognize objects with little guidance. For example, a child can easily identify a dog species from an image of a dog, even though they had never seen a dog before. Inspired by this, one-shot learning focuses on imitating this ability. The goal of one-shot segmentation is to obtain the object regions of a query image with only one support image. Both support-image and query-image are sampled from the same unseen class. Sampling from an unknown class is the main difference between one-shot segmentation and traditional semantic segmentation. If we want to segment an unseen class object using a traditional learning-based algorithm, we need at least hundreds of labeled data and multiple iterations to achieve a good segmentation result for of objects. However, one-shot-based algorithms only takes one label–image pair as guidance, and optimization during the process of segmentation is not required. There are two major advantages for one-shot segmentation: (1) minimized annotation effort and (2) there is no need to fine-tune model, since the parameters are fixed after training, reducing time and computation costs.

One-shot semantic segmentation recognizes object regions from invisible categories with only one annotated sample serving as the supervision. Shaban et al. [39] proposed a pioneering applied one-shot learning to the semantic segmentation. They segmented new semantic classes requiring only an image and the corresponding densely annotated label. They constructed the two-branched model OSLSM, which is based on Siamese Network. The network is divided into conditioning and segmentation branches, where the conditioning branch supports image–label pairs and produces dynamic parameters for a segmentation branch. They used the conditioning branch to perform dense pixel-level prediction on a test image for the new semantic class. This process adds a convolutional layer after the FCN [34], and the parameters in this convolutional layer are provided by generated dynamic parameters. However, OSLSM still needs users to provide pixel-level annotation information for support images. It is also unstable during optimization; different support image–label pairs produce the same task parameters. Rakelly et al. [40] proposed the REVOLVER model, which requires an image and its pixel-level annotated label, minimizing user interaction workload. REVOLVER only need users to provide a few clicks for the support image (these clicks are located in the absolute foreground and the absolute background). Differing from the OSLSM, which generates dynamic parameters, REVOLVER adopts the distance metric method. It calculates the distance between query features with the foreground representation and background. REVOLVER achieved similar result as OSLSM with only a few pixel-level annotations. Xu et al. [41] introduced the state-of-the-art deep interactive object segmentation (DIOS). They transformed users' positive and negative clicks to Euclidean distance maps and train a full convolutional neural network to recognize "object" and "background" based on training samples. However, it was not designed to generate trimaps. In particular, DIOS cannot propagate annotations across different images (Figure 3). This is a bottleneck on annotation efficiency, since it requires at least two annotations for every input, whereas our method can segment new inputs independently.

Figure 3. The main differences between interactive methods and our method. Interactive-based methods cannot propagate annotations across different images, However, the proposed method can pass the annotations' information between images of the same semantic class.

3. Problem Setup

Suppose we have two datasets: a query set, $L_q = \left\{\left(I_q^i, Y_q^i\right)\right\}_{i=1}^{N_q}(l)$ and a support set: $L_s = \left\{\left(I_s^i, Y_s^i\right)\right\}_{i=1}^{N_s}(l)$ where I^i represents the original image; Y^i represents the corresponding groundtruth mask; N represents the number of images in each set, the indexes s and q represents the support-set and query-set; respectively; and l represents the semantic class. Our goal was to learn a model $f_\theta(L_s, L_q)$ that can precisely predict binary masks Y_q' according to the reference of the support set L_s, where θ represents the network parameters.

We cannot apply this model to generate trimaps directly because previous algorithms focused on semantic segmentation, which classifies each pixel. However, trimap generation focuses on classifying pixels to foreground, background, and opacity regions. For generating trimaps, we require a groundtruth trimap to optimize the network. Existing dataset labels only include the foreground and background (PASCALVOC [42]). We still trained the model using a binary mask to generate

binary segmentation results. We used the Dilation method to produce an initial estimate of the trimap, inspired by [5]. We used dynamic width and voting steps to optimize the trimap. The implementation details are provided in Section 4.

During the training process, the support image is fed into the network with its sparse pixel-level annotations, which are obtained from their corresponding groundtruth mask. We simulated manual labeling and randomly selected some points from the foreground and background to guide network training. The query image is fed into the network with its dense mask, which is used for loss calculation and parameter optimization. In the test process, there is no label to exploit, there are only the sparse annotations collected from user interaction. Notably, L_{qry} and L_{sup} share the same types of objects, but no categories are the same between the training set and the test set $\{l_{train}\} \cap \{l_{test}\} = \emptyset$. This is the main difference between one-shot segmentation and traditional image segmentation. The traditional training process splits the dataset into a training set and a test set; the training set images never appear in the test set. However, the training set and the test set have overlaps in terms of categories. So, when training data are processed, we turn the target into the background if its categories appear in the test set.

State-of-the-art algorithms for image segmentation [36] use networks pre-trained on ILSVRC.

4. Proposed Method

The network in this paper consists of three branches: the first branch generates task representation and guides the second segmentation branch to generate the segmentation results. Finally, the generated branch converts the segmentation result into a trimap. Our model is able to make predictions on its own, and, with expert guidance, can direct the task or correct errors. The process of self-prediction can be regarded as interaction segmentation when the support image and query image are the same. Note that interactive is a special case of one-shot segmentation. In particular, we used the guidance branch to extract guidance z from support set: $z = g(i_s, Y_s)$. Afterward, the segmentation branch was combined with guidance z and query image features to jointly predict the output results $y = f(i_q, z)$. We discuss how to design $z = g(i_s, Y_s)$, and $y = f(i_q, z)$ in the following sections.

4.1. Guidance Branch

Guidance branch fused user interaction involves clicks with the support image, and guidance information z_s is generated. The guidance process can be expressed as :

$$z_s = \mu(\lambda(I_s), m(Y_+), m(Y_-)) \tag{2}$$

where z_s includes target features and background features. We assume that the user can provide a positive click (+) and a negative click (−). We match pixel-level clicks to the same coordinate scale as the support image $I_s \in R^{3*w*h}$, where w, h represent the length and width of support image respectively; the pixels under the strokes are set to values of 1, and 0 otherwise. Then, we can obtain two annotation maps (Y_+) and (Y_-), $Y \in \{0,1\}^{w*h}$. We used fully convolutional networks as the feature extractor to extract visual features from the support image I_s using λ. The feature extractor λ of our model is VGG-16 [43], pre-trained on ILSVRC [44] and converted into fully convolutional form [34]. The $\lambda(I_s) in R^{c*w'*h'}$ and c, w', h' represent the channels, length, and width of feature maps, respectively. To ensure that they have the same scale, the positive map and negative map are down-sampled to the same scale using bilinear kernel $m(Y_+)$, $m(Y_-)$. Then, we fuse the support features with the positive map and negative map using the element-wise product μ. The spatial relationship between the support image and annotations can be well defined by fusing features from the visual and annotation branches. Using m to interpolate and μ to fuse, the visual representations of the support and query can be obtained using a unified feature extractor λ. The guidance process is shown in Figure 4c.

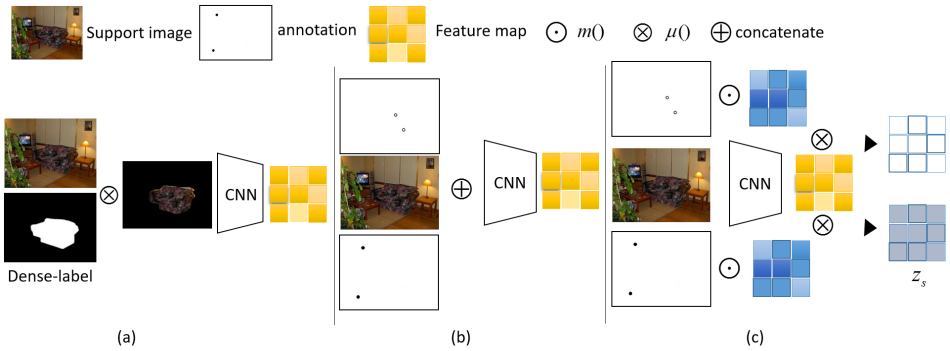

Figure 4. The main differences in guidance branch: (**a**) Shaban et al. [39] used multiple support images and directly dense labelled, and the background was omitted. (**b**) Xu et al. [41] concatenated on the support image and annotation maps, which breaks the input structure of the net. (**c**) Our proposed fusion method: the guidance branch combines annotation and support image in feature level, which maintains the identical net structure and complete background information.

In contrast to Shaban et al. [39], in which an element-wise multiplication was directly applied to the support image and the dense label annotation (foreground value 1, background value 0), the background is omitted as a result (Figure 4a). Our method saves the background information for the support image. In addition, by integrating annotations with feature-level information through the factorization method, the spatial dependency between them is more clearly defined. Xu et al. [41] proposed an early fusion method that concatenates the positive map and negative map with the support image to five channels, as shown in Figure 4b. However, the disadvantage of this method is that concatenating the support image with their maps breaks the input structure of the network, which also prevents the implementation of a unified network. The features of the support image and query image need to be extracted through different network structures. The model proposed in this paper (Figure 4c) maintains the identical input structure of the network, which enables us to process both the support and query images within a unified network.

When multiple foreground objects appear in the image (Figure 5a), we need to obtain the segmentation mask for all objects (Figure 5d) in the image instead of obtaining one object (Figure 5c). In addition, when the support and query images are completely different, no spatial information is available. The only mapping between the support and query should be achieved through characterization. We chose global pooling g_z to merge the local task representations and discarding the spatial dimensions, which can be represented as $z_s \in R^{c*w'*h'} \rightarrow v_s \in R^{c*1*1}$. However, when the support and query images are the same (e.g., interactive segmentation), location information can be used and global pooling g_z procedures can be omitted.

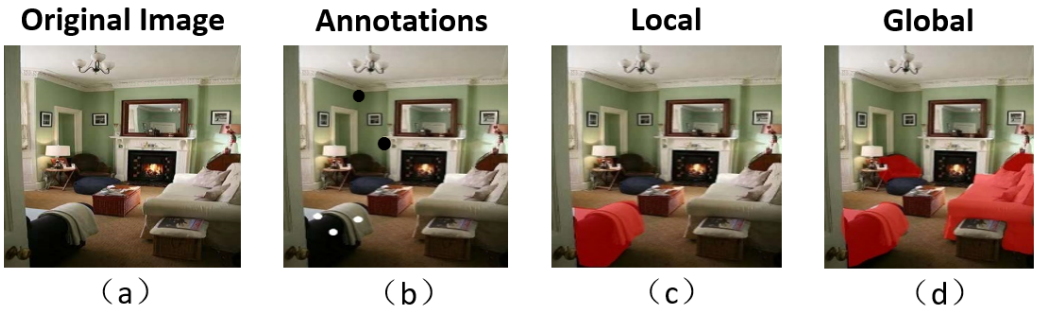

Figure 5. Global pooling the task representation in the guidance branch. In this case, we annotate a single sofa (**b**) on the original image (**a**), but global guidance causes all the sofas to be segmented (**d**) instead of obtaining one sofa (**c**).

4.2. Segmentation Branch

We obtain the foreground target of the image by segmentation branch and generated rough segmentation results, the segmentation model define as:

$$\overline{y} = f_\theta\left(\lambda\left(I_q\right) \oplus req\left(v_s\right)\right) \tag{3}$$

The same as for the guidance branch, we extract visual features from the query image I_q using a deep convolution network $\lambda()$, as with the support image, where $I_q \in R^{c*w'*h'}$ and w', h' represent the length and width of support image respectively. v_s is the globalized task representation obtained by the guidance branch. $\oplus$ represents the channel number stack. We repeat guidance vector v_s until its spatial dimension is equal to query features maps $\lambda(I_q)$ to ensure the parameters have the same dimension. f_θ is a small convolutional network that fuses the query-support feature and decodes to a binary predicted segmentation result. f_θ can be interpreted as a learned distance metric for retrieval from support to query. The distance metric part consists of two components. The first component fuses query-support features through one combination of the convolution layer (1×1 kernel size), rectified linear units (ReLU), and drop-out, and the parameters in the convolution layer are used to compute the distance between pixels from support to query. The outputs of this component are coarse distance metric maps. The second part only includes one layer of convolution (1×1 kernel size) with a channel dimension of 2 to predict scores for foreground and background classes at each of the coarse distance metric maps. The second part is followed by bilinear upsampling for end-to-end learning by back-propagation from the pixel-wise loss. The segmentation process is shown in Figure 6.

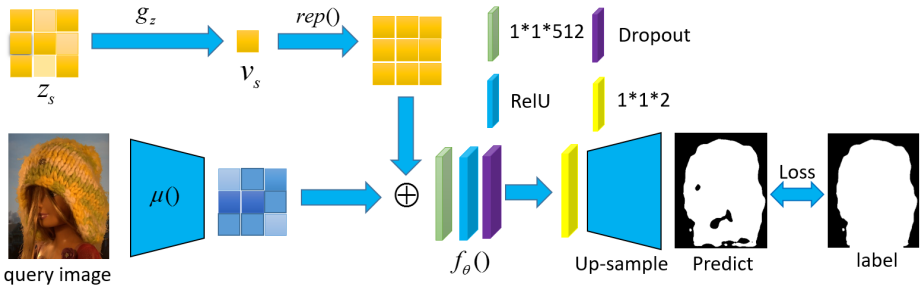

Figure 6. The model structure of the segmentation branch, where the convolution component f_θ consists of the convolution layer, activation layer and Dropout, which is used to compute the distance between pixels from support to query. Where the z_s represents the guidance, g_z represents global pooling and v_s represents the pooled guidance vector.

Inspired by REVOLVER's [40] training episode, we first sampled a task. Then, we sampled a subset of images containing the task, which we divided into support and query. Given inputs and targets, we trained the network by cross-entropy loss:

$$L = \frac{1}{N}\sum_i -\left[y_i * log\left(\overline{y}_i\right) + \left(1 - y_i\right) * log\left(1 - \overline{y}_i\right)\right] \tag{4}$$

where the $\overline{y}$ represents the predicted segmentation results, the y represents the corresponding dense labels. Notably, the optimization process of the model during training is different from the one-shot learning process, where the parameters are not optimized during the one-shot learning process, and one-shot learning is achieved via guidance and guided inference.

4.3. Generate Branch

After the training, the model parameters were fixed. The model predicts segmentation results using several clicks. However, the predicted segmentation results have relatively rough edges, which do not satisfy the digital matting requirements. Therefore, we used three main steps to further

process the segmentation results. Firstly, we optimized the segmentation results by conditional random field (CRF) to increase the precision of the target edge region. Secondly, inspired by [5], we obtained the initial trimap by dilating and eroding the binary segmentation result. We set the foreground region of the ablation image as the foreground region in the trimap, the background region of the expansion image as the background region in the trimap, and the rest of the image as the unknown region. Finally, we used the deep matting model to obtain the final alpha matte, and the model was trained by the Adobe image matting database [41].

5. Dataset and Experiments

5.1. Dataset

We chose the PASCALVOC2012 [42] dataset D_{train} to train the model. The PASCALVOC2012 dataset includes 21 classes (aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person potted plant, sheep, sofa, train, tv/monitor, and background). In the first experiment, the performance of the proposed matting method was evaluated on the standard benchmark [45]. The dataset consists of 27 images with corresponding groundtruth alpha and trimaps. The trimaps were generated by an experienced user given a paint tool with different size brushes. The generated trimaps included two types that are represented as trimap-1 and trimap-2. In the second experiment, to verify the generalization of unseen classes, we selected 20 images from the Adobe image matting dataset [13] with their corresponding groundtruth alpha matte, as shown in Figure 7. Each row of pictures represents the same category. The criteria of our experiment include Mean Absolute Error (MAE), the formula is expressed as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i| \tag{5}$$

n represents the number of pixels, x_i and y_i represent predicted alpha matte and groundtruth alpha matte, respectively, and the Intersection-over-Union (IoU) of unknown areas in the trimap, the formula is expressed as:

$$IoU = \frac{U_{pred} \cap U_{gt}}{U_{pred} \cup U_{gt}} \tag{6}$$

U_{pred} and U_{gt} represent the unknow-region area of predict and user-input trimap, respectively.

We split the training dataset into four parts as shown in Table 1. We used the $D_{train}(l)$ training network separately, where l represents the semantic categories. We used l_{split} to represent the partition of the dataset, e.g., l_0 means we use $l_{i=1,2,3}$ for training and $l_{i=0}$ for testing. For comparison with the standard benchmark and obtain more accurate results, in the first experiment, we selected all categories for training the model $D_{train}(l_{oracle})$. To verify the guidance ability of the model, in the second and third experiments, we selected part of the categories for training the models $D_{train}(l_{split})$.

Table 1. Classes for each fold of PASCAL-5i. To ensure the class disjoint $\{l_{train}\} \cap \{l_{test}\} = \emptyset$, we split the dataset into four parts. We used three of them for training and the remainder for testing.

$l = 0$	$l = 1$	$l = 2$	$l = 3$
aeroplance, bicycle bird, person	diningtable, car, cat chair, cow	bus, dog, horse motorbike, bottle	plant, sheep, sofa train, tv/monitor

Figure 7. We select twenty images from [13]. Each row has the same category corresponding to Table 1, from top to bottom represents persons 0–4, cats 5–9, plants 10–14 and dogs 15–19, respectively. Subscripts represent image numbers.

5.2. Experiments

Experiment 1: We compare trimap precision directly in this section. We selected all images from the alpha matting dataset [45] to test our model. To obtain more accurate results, we set the support image and query image as the same. The trimap generation process can be understood as one-shot interactive segmentation. We used all classes $D_{train}(l_{oracle})$ to train our model. A partial visualization of the experimental results is provided in Figure 8. We estimated the performance of the trimap generation method using two criteria. Firstly, we performed the deep matting [13] for the different trimaps (proposed method, user input trimap, and scribble) and compared their matting results. From Figure 8, the generated trimaps are visually similar with the user-input trimaps. However, there is still a performance gap between the generated trimap and the user-input trimap. Since the user-input trimaps select from benchmark matting dataset, which drawed by professional users, the generated trimap is difficult to achieve this accuracy. The advantages of our algorithm lie in processing time and interactive workload. Besides, the proposed algorithm is much more accurate than the user scribble (Table 2). We think that MAE comparison between the proposed and scribble-based map is fair, because our method takes images and user clicks as inputs; the labeling time consumption between clicks and scribbles are basically the same. Secondly, we compared the unknown region intersection over union (IOU) between the generated trimap and the user input trimap. As shown in Table 3, the mean unknown region IOU was close to 50% for both trimap-1 and trimap-2. The above experiments proved the feasibility and robustness of our algorithm.

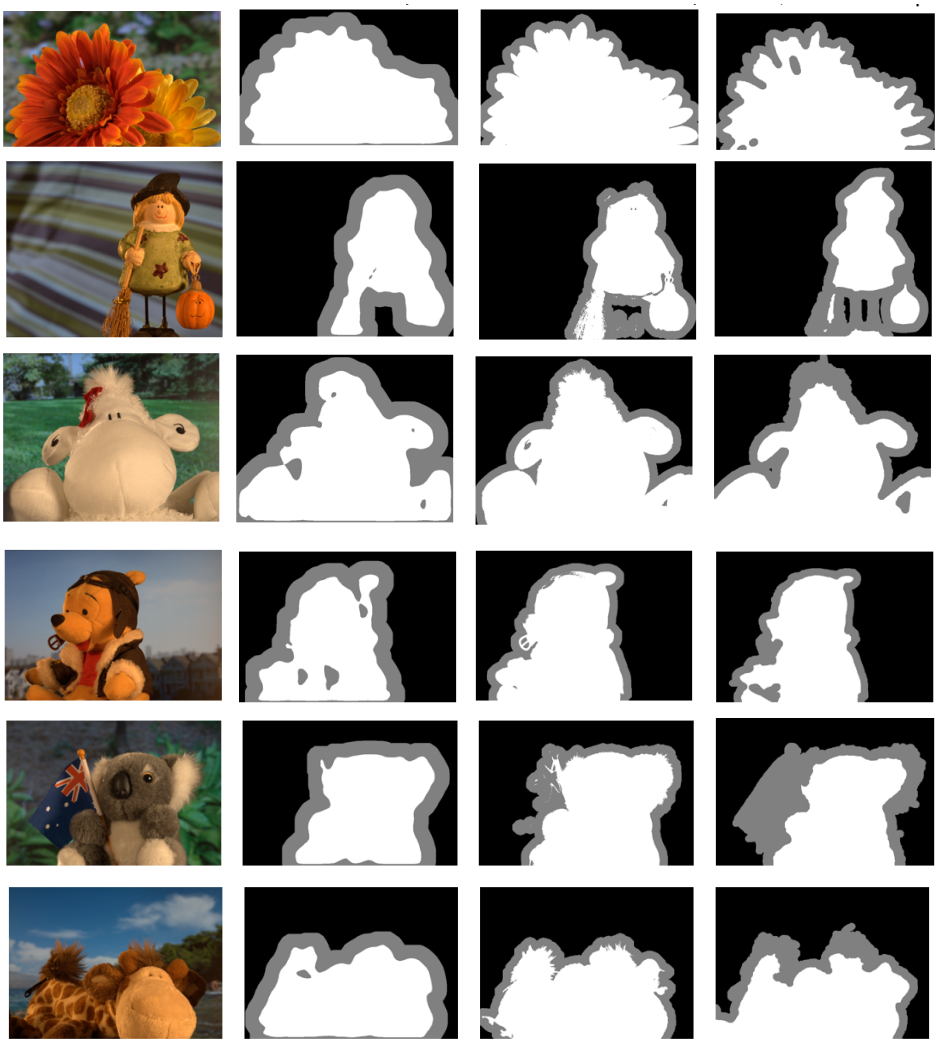

Figure 8. Partial trimap comparison results of experiment 1. The first column represents the original image (selected from [45]). The last column represents the user input trimaps (selected from [45]). The second column represents the trimaps generated by the proposed method. The third column represents the optimized trimaps (by trimap trimming [15]).

Table 2. Mean absolute error (MAE) statistics of alpha matte computation on [45] (using [13]). The proposed method obtains a approximate MAE to the trimap-based image matting, and is much more accurate than scribble-based image matting.

Image	Proposed	Trimap	Scribble
GT01	28.715	8.271	156.781
GT02	25.474	7.792	118.215
GT03	24.662	35.854	139.766
GT04	67.918	51.443	334.628
GT05	22.414	5.555	127.115
GT06	29.505	10.736	171.267
GT07	26.378	12.405	136.459
GT08	43.088	34.998	220.026
GT09	33.668	19.962	181.406
GT10	26.914	10.842	150.306
GT11	30.473	14.962	158.58
GT12	20.902	9.351	99.622
GT13	54.067	25.136	311.447
GT14	33.329	10.390	159.199

Table 2. *Cont.*

Image	Proposed	Trimap	Scribble
GT15	27.823	11.347	155.263
GT16	37.342	22.030	204.322
GT17	24.297	11.356	125.960
GT18	32.171	10.183	162.286
GT19	22.662	4.938	121.062
GT20	25.018	13.519	132.551
GT21	30.688	16.915	170.912
GT22	26.687	12.586	124.645
GT23	25.184	12.544	114.109
GT24	28.701	12.341	136.212
GT25	33.181	11.302	191.416
GT26	49.373	25.675	230.560
GT27	77.891	56.878	450.836

Table 3. Intersection over union (IOU) statistics of the unknown region between the generated trimap and the user input trimap. trimap-1 and trimap-2 represent the trimaps painted with different size brushes.

Image	Trimap-1	Trimap-2
GT01	0.368	0.345
GT02	0.25	0.271
GT03	0.466	0.408
GT04	0.418	0.411
GT05	0.363	0.467
GT06	0.386	0.458
GT07	0.396	0.476
GT08	0.557	0.519
GT09	0.56	0.569
GT10	0.441	0.508
GT11	0.402	0.49
GT12	0.51	0.574
GT13	0.325	0.343
GT14	0.359	0.437
GT15	0.373	0.448
GT16	0.569	0.577
GT17	0.501	0.518
GT18	0.385	0.504
GT19	0.326	0.449
GT20	0.476	0.527
GT21	0.259	0.27
GT22	0.504	0.581
GT23	0.516	0.577
GT24	0.307	0.35
GT25	0.229	0.288
GT26	0.461	0.49
GT27	0.407	0.454

Experiment 2: We verify the guidance ability of our proposed method in this section. We used part of the classes $D_{train}(l_{split})$ to train our model. So, we obtained four models with different parameters. Following the principle of category disjunction ($\{l_{train}\} \cap \{l_{test}\} = \emptyset$), we used different rows in Figure 7 to verify the different models, e.g., we used the first row (person semantic class) to verify the model trained by $D_{train}(l_0)$ (without person semantic class). To obtain more accurate results, we still set the support image and query image to the same. As with experiment 1, we compared the MAE and IOU for the 20 test images. The experimental results are partially depicted in Figure 9. Even though most details in the margin are captured by the proposed algorithm (third column in Figure 9), the generated

trimap is visually similar to user input trimap (fourth column in Figure 9). The statistical results are shown in Tables 4 and 5. The experimental results showed that although the test class does not appear in the training class, our model can still identify foreground targets by pixel-level annotations and generate relatively accurate trimaps. The results showed that our model has guidance ability and has the potential to generalize unknown semantic classes. However, the main problem with the proposed approach is that one-shot segmentation calculates the distance metric for every pixel in the query image to the foreground and background using support image as guidance. So, the model will misjudge when the foreground and background have similar representations. The fourth row in Figure 9b,c shows that the white hair of the dog is similar to the color of the beach, causing the model to judge part of the white hair belonging to the foreground as the background.

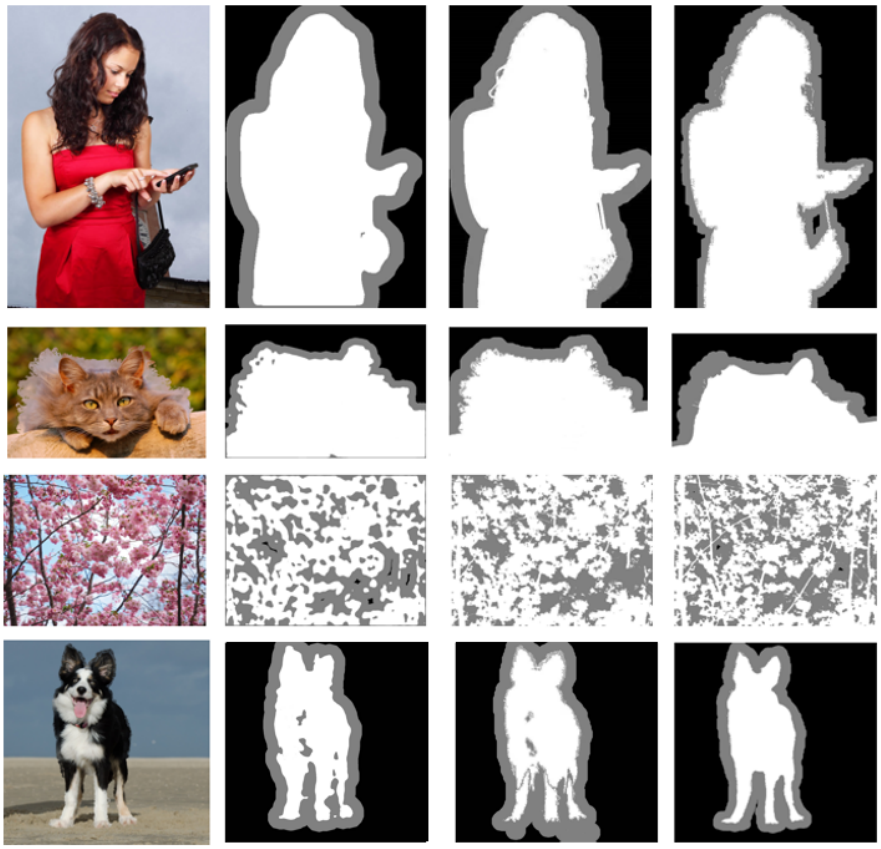

Figure 9. Partial trimap comparison results of experiment 2. Although the test class does not appear in the training class, the generated trimaps are still accurate enough (see column 3), the experimental results showed that our model has strong guidance ability.

Table 4. Experiment 2: MAE statistics of the 20 test images; the matte obtained by [13].

Image	Proposed	Trimap	Scribble
image0	44.199	8.271	156.781
image1	47.724	7.792	118.215
image2	51.2	35.854	139.766
image3	46.745	51.443	334.628
image4	33.635	5.555	127.115
image5	39.439	10.736	171.267
image6	40.58	12.405	136.459
image7	40.58	34.998	220.026
image8	41.584	19.962	181.406
image9	27.275	10.842	150.306

Table 4. *Cont.*

Image	Proposed	Trimap	Scribble
image10	68.825	14.962	158.58
image11	49.825	9.351	99.622
image12	60.375	25.136	311.447
image13	74.821	10.390	159.199
image14	117.501	11.347	155.263
image15	22.508	22.030	204.322
image16	51.626	11.356	125.960
image17	34.524	10.183	162.286
image18	23.09	4.938	121.062
image19	54.023	13.519	132.551

Table 5. Experiment 2: IOU statistics of the 20 test images. The trimap-1 and trimap-2 obtained by dilating the groundtruth matte with different expansion sizes.

Image	Trimap-1	Trimap-2
image0	0.3	0.22
image1	0.629	0.67
image2	0.32	0.296
image3	0.42	0.318
image4	0.362	0.269
image5	0.227	0.272
image6	0.529	0.527
image7	0.531	0.521
image8	0.368	0.635
image9	0.441	0.538
image10	0.653	0.662
image11	0.594	0.406
image12	0.626	0.533
image13	0.355	0.352
image14	0.435	0.556
image15	0.393	0.414
image16	0.676	0.442
image17	0.45	0.322
image18	0.43	0.28
image19	0.42	0.396

Experiment 3: To verify the time values of our algorithm, we compared the proposed method with classical matting algorithms, Bayesian matting [1], closed-form matting [7], and Soft Scissors [10]. The experimental results are shown in Figure 10a. Our method is the fastest. The main reason for the speed lies in the following two points. Firstly, our system generates trimaps using a few clicks. Compared with previous algorithms used to design a complete trimap, the click time is negligible. Second, we use deep convolutional neural networks to obtain the final matte results. Compared with sample-based algorithms or propagation-based algorithms that need many iterations, our algorithm requires one forward propagation of the model.

Experiment 4: At first glance, the proposed method is similar to interactive matting algorithms like graph-cut. So, in this experiment, we proved the differences and further verified the model's ability to transmit guidance in the same semantic class. We selected more complex foreground images from [42] and set the support image and query image are the completely different (semantic category consistency), as shown in Figure 11a. The first row to the last row represent different spatial positions, different foreground objects, and multiple foreground objects, respectively. The query image is overlaid with our predicted mask in green (Figure 11b). The experimental results showed that when the difference is huge in the morphological and spatial position of the foreground target

between the query image and the support image, the task representation still can guide the query branch to obtain a relatively accurate result. However, graph-based methods cannot transmit task representation between different images like our proposed method. To verify the advantages of our method from the previous methods, we compare our system with interactive trimap segmentation methods (e.g., [14,17]), a partial visualization of the experimental results is provided in Figure 12, the trimaps are covered by RGB color. Our results (second row in Figure 12) is visually similar to the interactive segmentation methods (first row in Figure 12), since our deep learning based model better combines the depth features, and the CRF further optimizes the segmentation results. Besides, the results further prove guidance ability of our model, which delivers the semantic representation between different query and support images.

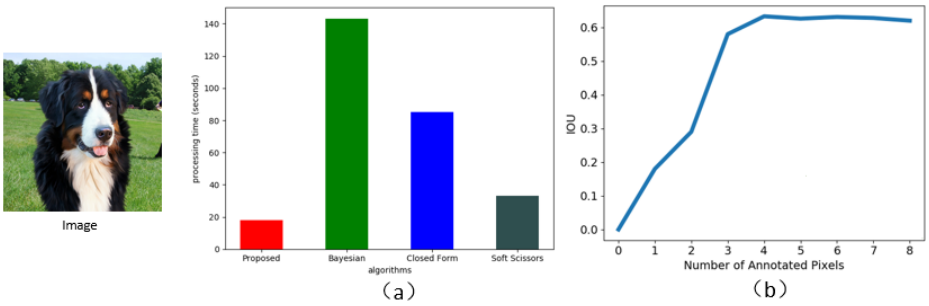

Figure 10. (**a**) The processing time of different methods. (**b**) The amount of interactive annotations in the image.

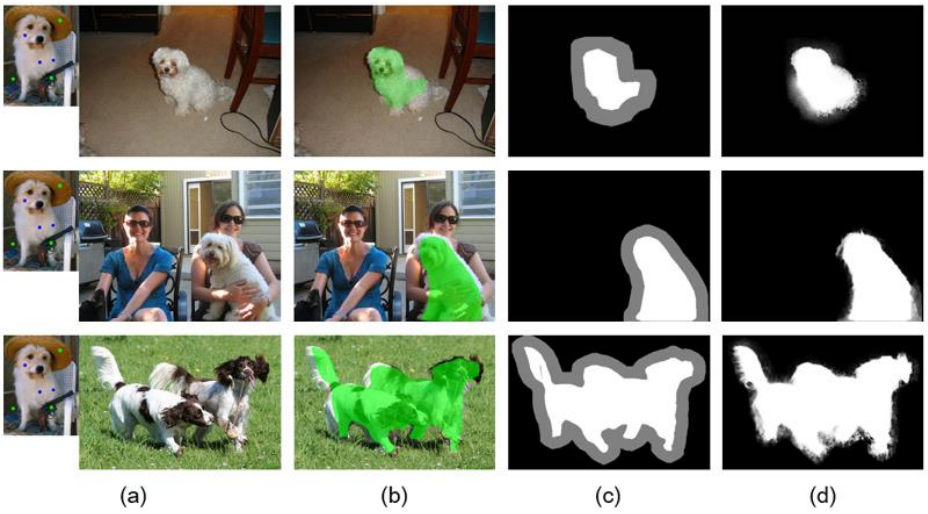

Figure 11. Test guidance ability of the model. (**a**) The support set on the left with annotations and the query set on the right; (**b**) one-shot semantic segmentation result, the query image is overlaid with our predicted result in green; (**c**) generated trimaps, and (**d**) our predicted alpha matte using (**c**) by [13]. First row, different spatial position. Second row, different foreground objects. Last row, multiple foreground objects.

One question arose: how many pixel-level clicks are needed? The experiment proved that our method is insensitive to the amount of annotations, as shown in Figure 11b. The IOU accuracy does not increase after 3 pixels. This is an ill-posed problem for unseen class generalization from one shot. One-shot learning cannot cover the complete visual information in an unseen class, for example, matting a black, long-haired dog, as in Figure 7(16) cannot be achieved given a brown, short-haired dog like in Figure 7(18) as guidance. The guidance struggles to make judgments when the color, texture, and pose are very different. The solution is method improving using shot, one-shot to few-shot.

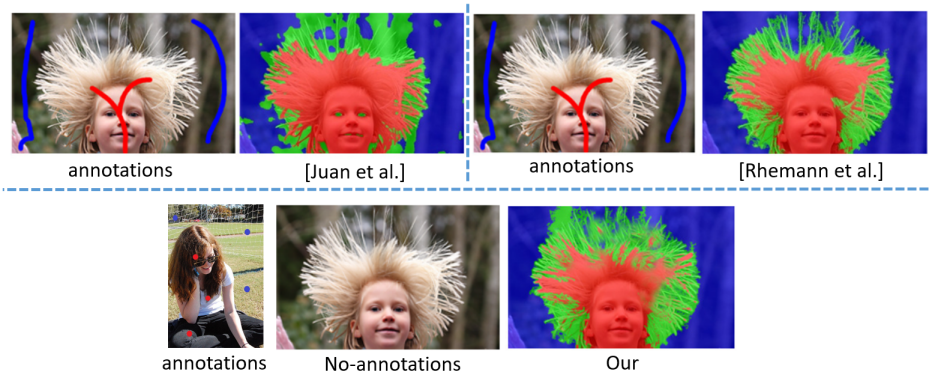

Figure 12. This figure shows trimap segmentation results. We compare our method with Interactive Trimap segmentation methods [14,17], the experimental results showed that our model deliver semantic representation between query and support images, and the generated trimaps are accurate enough.

6. Conclusions

In this paper, to reduce the user interaction workload, we adopt the one-shot learning-based segmentation algorithm to generate trimaps for image matting. Our model only needs a few clicks from the user to generate a high-precision trimap. Compared with scribble-based image matting, our method has better robustness. Compared with trimap-based algorithms, our method reduces the required user interaction time. Simultaneously, our model turns user-provided clicks into guidance, which implements interactive information sharing between the same semantic classes and minimizes user interaction workload. However, the proposed method still has limitations when the foreground and background have similar features in the query image.

Author Contributions: Conceptualization, Y.Z.; methodology, Z.C.; software, J.L.; validation, X.L. and Y.Z.; formal analysis, W.J. and Y.Z.; investigation, R.L.; resources, W.J. and C.L.; data curation, Z.C.; writing—original draft preparation, W.J.; writing—review and editing, Y.Z.; visualization, C.L.; supervision, X.L.; project administration, Y.Z.; funding acquisition, Y.Z. and W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Nature Science Foundation of China (No.: 21978139, 81871508); Focus on Research and Development Plan in Shandong Province (No.: 2019GNC106115); Shandong Province Higher Educational Science and Technology Program (No.: J17KA008, J18KA308); China Postdoctoral Science Foundation (No.: 2018M630797); Taishan Scholar Program of Shandong Province of China (No.: TSHW201502038).

Acknowledgments: Thanks to all the editors and anonymous reviewers who have worked for this article, due to your help, the level of the paper has been significantly improved. Special thanks to Susie Li for her works and patience.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chuang, Y.Y.; Curless, B.; Salesin, D.H.; Szeliski, R. A bayesian approach to digital matting. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, USA, 8–14 December 2001; Volume 2.
2. Wang, J.; Cohen, M.F. An iterative optimization approach for unified image segmentation and matting. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China, 17–21 October 2005; Volume 2, pp. 936–943.
3. Levin, A.; Rav-Acha, A.; Lischinski, D. Spectral matting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 1699–1712. [CrossRef]
4. Gupta, V.; Raman, S. Automatic trimap generation for image matting. In Proceedings of the 2016 International Conference on Signal and Information Processing (IConSIP), Nanded, India, 6–8 October 2016; pp. 1–5.
5. Hsieh, C.L.; Lee, M.S. Automatic trimap generation for digital image matting. In Proceedings of the 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, Kaohsiung, Taiwan, 29 October–1 November 2013; pp. 1–5.

6. Zheng, Y.; Kambhamettu, C. Learning based digital matting. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 889–896.
7. Levin, A.; Lischinski, D.; Weiss, Y. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 228–242. [CrossRef]
8. Rother, C.; Kolmogorov, V.; Blake, A. "GrabCut" interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. TOG* **2004**, *23*, 309–314. [CrossRef]
9. Li, Y.; Sun, J.; Tang, C.K.; Shum, H.Y. Lazy snapping. *ACM Trans. Graph. ToG* **2004**, *23*, 303–308. [CrossRef]
10. Wang, J.; Agrawala, M.; Cohen, M.F. Soft scissors: An interactive tool for realtime high quality matting. *ACM Trans. Graph.* **2007**, *26*, 9-es. [CrossRef]
11. Cho, D.; Tai, Y.W.; Kweon, I. Natural image matting using deep convolutional neural networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 626–643.
12. Chen, Q.; Li, D.; Tang, C.K. KNN matting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2175–2188. [CrossRef] [PubMed]
13. Xu, N.; Price, B.; Cohen, S.; Huang, T. Deep image matting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2970–2979.
14. Rhemann, C.; Rother, C.; Rav-Acha, A.; Sharp, T. High resolution matting via interactive trimap segmentation. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
15. Shahrian, E.; Rajan, D.; Price, B.; Cohen, S. Improving image matting using comprehensive sampling sets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 636–643.
16. Cho, D.; Kim, S.; Tai, Y.W.; Kweon, I.S. Automatic Trimap Generation and Consistent Matting for Light-Field Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1504–1517. [CrossRef]
17. Juan, O.; Keriven, R. Trimap segmentation for fast and user-friendly alpha matting. In Proceedings of the International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision, Beijing, China, 16 October 2005; pp. 186–197.
18. Shen, X.; Tao, X.; Gao, H.; Zhou, C.; Jia, J. Deep automatic portrait matting. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 92–107.
19. Gastal, E.S.; Oliveira, M.M. Shared sampling for real-time alpha matting. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2010; Volume 29, pp. 575–584.
20. Feng, X.; Liang, X.; Zhang, Z. A cluster sampling method for image matting via sparse coding. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 204–219.
21. Karacan, L.; Erdem, A.; Erdem, E. Image matting with KL-divergence based sparse sampling. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 424–432.
22. Aksoy, Y.; Ozan Aydin, T.; Pollefeys, M. Designing effective inter-pixel information flow for natural image matting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 29–37.
23. Zhu, L.; Huang, Z.; Li, Z.; Xie, L.; Shen, H.T. Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5264–5276. [CrossRef] [PubMed]
24. Fang, Y.; Zhang, H.; Ren, Y. Graph regularised sparse NMF factorisation for imagery de-noising. *IET Comput. Vis.* **2018**, *12*, 466–475. [CrossRef]
25. Zong, J.; Meng, L.; Zhang, H.; Wan, W. JND-based Multiple Description Image Coding. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 3935–3949.
26. Jiang, Y.; Zheng, Y.; Hou, S.; Chang, Y.; Gee, J. Multimodal image alignment via linear mapping between feature modalities. *J. Healthc. Eng.* **2017**, *2017*. [CrossRef]
27. Zhao, M.; Zhang, H.; Meng, L. An angle structure descriptor for image retrieval. *China Commun.* **2016**, *13*, 222–230. [CrossRef]
28. Zhao, M.; Zhang, H.; Sun, J. A novel image retrieval method based on multi-trend structure descriptor. *J. Vis. Commun. Image Represent.* **2016**, *38*, 73–81. [CrossRef]
29. Liu, H. Generative 3d images in a visual evolutionary computing system. *Comput. Sci. Inf. Syst.* **2010**, *7*, 111–126. [CrossRef]

30. Wang, C.X.; Xu, M.; Wan, W.; Wang, J.; Meng, L.; Li, J.; Sun, J. Robust Image Watermarking via Perceptual Structural Regularity-based JND Model. *TIIS* **2019**, *13*, 1080–1099.
31. Deng, X.; Zheng, Y.; Xu, Y.; Xi, X.; Li, N.; Yin, Y. Graph cut based automatic aorta segmentation with an adaptive smoothness constraint in 3D abdominal CT images. *Neurocomputing* **2018**, *310*, 46–58. [CrossRef]
32. Li, J.B.; Chu, S.C.; Pan, J.S. *Kernel Learning Algorithms for Face Recognition*; Springer: Berlin, Germany, 2014.
33. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
34. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
35. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
36. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848. [CrossRef]
37. Khoreva, A.; Benenson, R.; Hosang, J.; Hein, M.; Schiele, B. Simple does it: Weakly supervised instance and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 876–885.
38. Bearman, A.; Russakovsky, O.; Ferrari, V.; Fei-Fei, L. What's the point: Semantic segmentation with point supervision. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 549–565.
39. Shaban, A.; Bansal, S.; Liu, Z.; Essa, I.; Boots, B. One-shot learning for semantic segmentation. *arXiv* **2017**, arXiv:1709.03410.
40. Rakelly, K.; Shelhamer, E.; Darrell, T.; Efros, A.A.; Levine, S. Few-shot segmentation propagation with guided networks. *arXiv* **2018**, arXiv:1806.07373.
41. Xu, N.; Price, B.; Cohen, S.; Yang, J.; Huang, T.S. Deep interactive object selection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 373–381.
42. Everingham, M.; Van Gool, L.; Williams, C.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
43. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
44. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
45. Rhemann, C.; Rother, C.; Wang, J.; Gelautz, M.; Kohli, P.; Rott, P. A perceptually motivated online benchmark for image matting. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 1826–1833.