

# Integrated Foreground Segmentation and Boundary Matting for Live Videos

Minglun Gong, *Member, IEEE*, Yiming Qian, and Li Cheng, *Senior Member, IEEE*

**Abstract**—The objective of foreground segmentation is to extract the desired foreground object from input videos. Over the years, there have been significant amount of efforts on this topic. Nevertheless, there still lacks a simple yet effective algorithm that can process live videos of objects with fuzzy boundaries (e.g., hair) captured by freely moving cameras. This paper presents an algorithm toward this goal. The key idea is to train and maintain two competing one-class support vector machines at each pixel location, which model local color distributions for both foreground and background, respectively. The usage of two competing local classifiers, as we have advocated, provides higher discriminative power while allowing better handling of ambiguities. By exploiting this proposed machine learning technique, and by addressing both foreground segmentation and boundary matting problems in an integrated manner, our algorithm is shown to be particularly competent at processing a wide range of videos with complex backgrounds from freely moving cameras. This is usually achieved with minimum user interactions. Furthermore, by introducing novel acceleration techniques and by exploiting the parallel structure of the algorithm, near real-time processing speed (14 frames/s without matting and 8 frames/s with matting on a midrange PC & GPU) is achieved for VGA-sized videos.

**Index Terms**—Foreground segmentation, video matting, support vector machine (SVM), one-class SVM (1SVM).

## I. INTRODUCTION

**F**OREGROUND segmentation, a.k.a. video cutout, studies how to extract objects of interest from input videos. It is a fundamental problem in computer vision and often serves as a pre-processing step for other video analysis tasks such as surveillance, teleconferencing, action recognition and retrieval. Over the years a significant amount of related techniques have been proposed in both computer vision and graphics communities. However, some of them are limited to sequences captured by stationary cameras, while others require significant amount of training examples or cumbersome user interactions.

Manuscript received April 27, 2014; revised September 4, 2014 and November 25, 2014; accepted February 2, 2015. Date of publication February 6, 2015; date of current version March 3, 2015. The work of M. Gong was supported by the Natural Sciences and Engineering Research Council of Canada under Grant 293127. The work of L. Cheng was supported by the Agency for Science, Technology and Research through the Joint Council Office. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Pierre-Marc Jodoin.

M. Gong and Y. Qian are with the Department of Computer Science, Memorial University, St. John's, NL A1B 3X9, Canada (e-mail: gong@cs.mun.ca; yq4048@mun.ca).

L. Cheng is with the Bioinformatics Institute, Agency for Science, Technology and Research, Singapore 138632, and also with the School of Computing, National University of Singapore, Singapore 119077 (e-mail: chengli@bii.a-star.edu.sg).

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

Digital Object Identifier 10.1109/TIP.2015.2401516

Furthermore, most existing algorithms are rather complicated and computationally too demanding to be operated in real-time. As a result, there still lacks an efficient and powerful algorithm capable of processing challenging live video scenes with minimum user interactions.

Motivated by the above finding, we here present a novel integrated foreground segmentation and boundary matting approach, which is an extension to our preliminary work on foreground segmentation [19]. As shown in Figure 1, with only a few strokes from user on the first frame of the video, the algorithm is able to propagate labeling information to neighboring pixels through a simple train-relabel-matting procedure, resulting in a proper segmentation of the frame. This same procedure is used to further propagate labeling information across adjacent frames, regardless of the foreground or background motions. Several techniques are also proposed in order to reduce computational costs. Furthermore, by exploiting the parallel structure of the proposed algorithm, real-time processing speed of 14 frames per second (FPS) is achieved for VGA-sized videos when matting is not applied, with the frame rate dropping to 8 FPS when matting over large fuzzy areas is needed.

## A. Overview of the Presented Algorithm

The key insight of our approach is to maintain two Competing one-class Support Vector Machines (C-1SVMs) at every pixel location. The two one-class Support Vector Machines (1SVMs) capture the local foreground and background color densities separately, but determine a proper label for the pixel jointly. By iterating between training local C-1SVMs and applying them to label the pixels, the algorithm effectively propagates initial user labeling to the entire image, as well as to consecutive frames.

Empirical studies such as Figures 4 and 7 suggest that, compared to using a global classifier, maintaining local classifiers endows the resulting algorithm with higher discriminative power. Moreover, as will be explained in section III-A, using C-1SVMs instead of binary SVMs allows better handling of ambiguous situations. As a result, the algorithm can deal with a variety of challenging scenarios studied by the state-of-the-art methods (see Figures 1 and 9). Finally, using two 1SVMs to model foreground and background color distributions separately facilitates the matting calculation along object boundaries, making it possible to solve foreground segmentation and boundary matting problems in an integrated manner (see Figures 1 and 13).

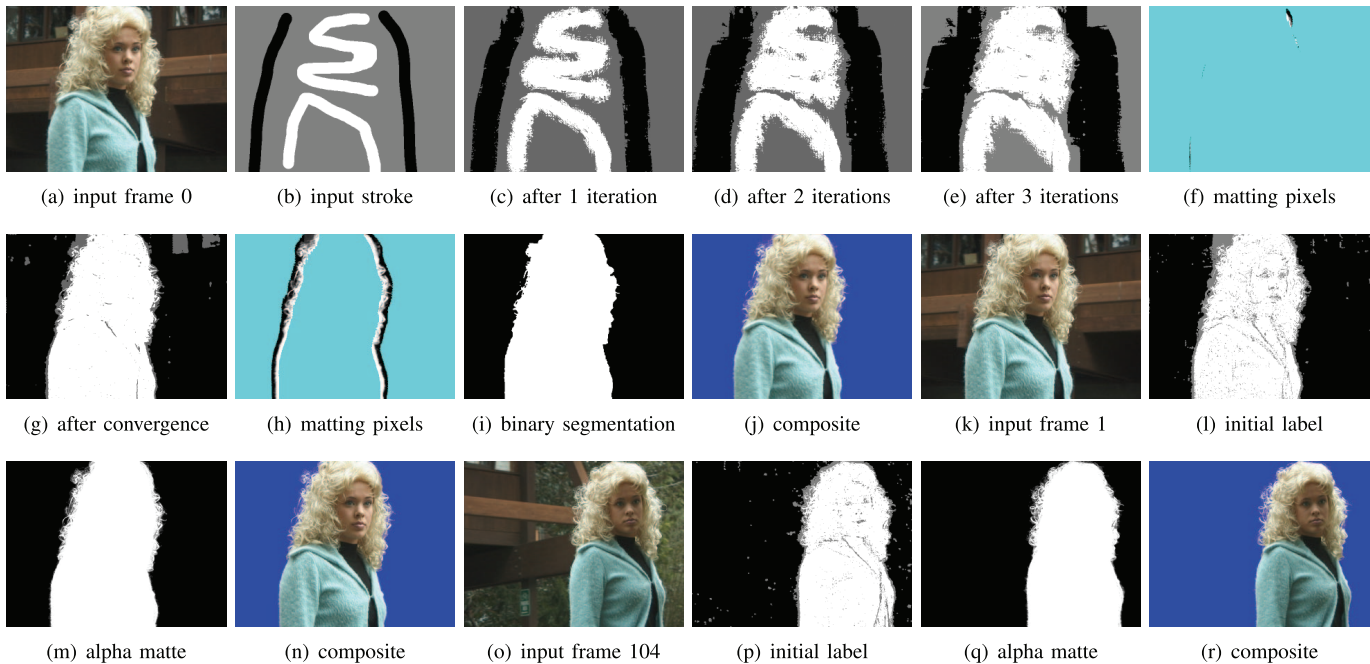


Fig. 1. Handling the “kim” sequence [9], which is challenging due to fuzzy object boundaries and camera motions. The user is only required to label the first frame (a) using strokes (b). Local classifiers are trained at each pixel location and then used to relabel the center pixel (c). Iterative training and relabeling leads to convergence (d, e, & g), even though ambiguous (grey) areas still exist. At each iteration, pixels along fore/background boundaries (non-cyan pixels in f & h) are detected, for which matting is performed. The final binary segmentation is computed through graph-cut optimization (i). Combining binary segmentation (i) with boundary matte (h) produces the full alpha matte, which is used to generate a blue screen composite (j). When new frames (k & o) arrive, they are initially labeled (l & p) using the classifiers trained by previous frames, before the same train-relabel-matting procedure takes place to produce the alpha mattes (m & q) and composites (n & r).

In summary, the proposed algorithm bears the following characteristics:

*Ability to Deal With Challenging Scenarios:* As shown in Figure 1 and 9, the algorithm performs competitively under a variety of challenge scenarios such as fuzzy object boundaries, camera motion, topology changes, and low fore/background color contrast.

*Minimal User Interaction:* Users are only asked to annotate foreground and background of the first frame with few key strokes. Alternatively, the algorithm can also be configured to train with only pure background images, allowing fully automatic segmentation (see Figure 11).

*Unified Framework for Segmentation and Matting:* The ability of C-1SVMs to train separate classifiers for foreground and background colors not only allows more robust labeling of the pixels, but also facilitates the matting procedure along object boundaries. This leads to an integrated solution for both foreground segmentation and boundary matting problems.

*Easy to Implement:* The same train-relabel-matting procedure is used to segment foreground objects from input user strokes, as well as to take care of fore/background motions in the video. No additional procedure is required for obtaining trimaps or estimating scene motions.

*Low Computational Cost:* The classifiers are trained using online learning, which allows much more efficient learning of a dynamic set of examples than batch learning. Two novel techniques are also proposed in this paper to further reduce computational cost of training.

*Parallel Computing:* The algorithm is designed for parallel execution at individual pixel locations. Our current

implementation processes VGA-sized videos in real-time using a mid-range graphics card.

The rest of the paper is organized as follows. Related works are reviewed in Section II. Section III discusses how to train classifiers and proposes novel techniques for reducing training costs. The details of the segmentation algorithm are presented in Section IV. Experimental results and comparisons to existing techniques are provided in Section V. Finally, Section VI concludes the paper and suggests future research directions.

## II. LITERATURE REVIEW

There is a vast body of existing work in the areas of foreground segmentation and video matting. Here the most related ones are discussed in the subsections below.

### A. Foreground Segmentation

Existing work on foreground segmentation can be categorized into unsupervised [5], [14], [22], [32], [33], [35], [43] and supervised [1], [2], [12], [24], [28], [36], [41], [42] approaches.

Unsupervised approaches try to generate background models automatically and detect outliers of the models as foreground. Most of them, referred as background subtraction approaches, assume that the input video is captured by a stationary camera and model background colors at each pixel location using either generative [35], [43] or non-parametric [5], [18], [33] methods. Some of these techniques [5], [18], [43] can handle repetitive background motion,

such as rippling water and waving trees, but none can deal with large camera motion.

Considering scenarios where camera motion does not change the viewing position, such as PTZ security cameras, the background motion can be described by a homography, which can be used to align different frames before applying the conventional background subtraction methods [22]. Recently there is also some new development to deal with freely moving cameras by means of tracking the trajectories of salient features across the frames [14], [32], where the trajectories are used for classifying the feature points into foreground or background based on their motion characteristics. While these methods automatically detect moving objects, they tend to classify background motion (e.g., waving trees or passing-by persons) as foreground. In addition, successful separation of foreground and background trajectories demands the detection of sufficient number of fore/background features and they having different motions, making it hard to handle scenes with weakly textured background or with large rigid foreground objects.

On the other hand, supervised methods allow users to provide training examples for both foreground and background, then use them to learn classifiers. A number of methods [12], [24], [41] along this line have been developed with impressive results, where multiple visual cues such as color, contrast, motion, and stereo are utilized. These cues are integrated with the help of structured prediction methods such as conditional random fields. Although working very well for video conference applications, these algorithms require a large set of fully annotated images and considerable amount of offline training.

Instead of resort to additional visual cues for higher discriminative power, approach has also been proposed to model location dependent color distributions using spatial-color Gaussian mixture models (SCGMM) [42]. We also model location dependent color distributions, but instead of learning and tracking SCGMMs, we train and update local C-1SVM based classifiers at every pixel location. This leads to a simple solution that segments foreground objects from input user strokes and tracks fore/background motions under the same procedure.

## B. Video Matting

Video matting techniques aim for extracting fuzzy boundaries along foreground objects [39]. They can be grouped into batch processing [1], [8], [9], [27], [28], [36] and online processing [2], [15]–[17], [20], [23], [29], [40] approaches.

A pre-captured video sequence is often perceived by the batch approaches as a 3D volume of voxels. Users are then required to label foreground and background on multiple frames [1], [8], [9], [28] or directly on the 3D volume [36]. Dense and precise labeling in the form of trimaps are required by many [9], [28] as necessary input, while some more recent approaches accept causally drawn strokes [1], [8], [27], [36]. To enforce temporal coherence, these algorithms usually segment over the entire volume altogether by solving a global optimization problem, which unfortunately restricts their capacity toward live video processing.

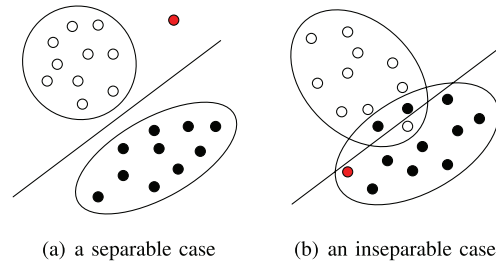


Fig. 2. Comparison between a binary SVM and C-1SVMs under two situations. White circles and black dots represent the foreground and background training instances, respectively, while red dot denotes an unseen example. The straight line indicates the decision boundary of the binary SVM, whereas the ellipsoids show the boundaries of the two C-1SVMs. In (a), binary SVM classifies the test example as foreground, whereas the C-1SVMs labels it as unknown, since neither of the 1SVMs accepts it as inlier. In (b), binary SVM cannot confidently classify the test example since it is too close to the decision boundary, whereas C-1SVMs is able to label it as background with confidence since only background 1SVM accepts it as inlier.

The frame-by-frame processing characteristic of the online approaches, on the other hand, make them more natural to work with live videos. Some of these approaches assume that foreground and background are properly and densely labeled in each frame by an existing algorithm [17], [20]; others try to solve foreground segmentation and matting problems in an integrated manner by either propagating user provided labeling information forward to future frames [2], automatically generating scribbles through salient point tracking and discriminative fore/background color models [15], [16], or by utilizing additional cues, such as focus [23], [29] or depth [40].

The workflow of our approach is similar to the Video SnapCut [2]. Starting from a segmentation of the first frame, Video SnapCut trains both global and local classifiers using color and shape cues, then propagates labeling information to the rest of the video frame by frame. However, the two approaches possess notably different objectives. Video SnapCut aims toward high quality video segmentation and produces very convincing results, but it expects users to provide a fine annotation of the entire first frame which can be challenging for fuzzy objects, and runs at about 1 FPS for VGA-sized videos (excluding the time for matting). Meanwhile our approach is designed to process live videos, which starts by inquiring only a few strokes from users, takes 1-2 seconds to initialize and segment the first frame, then new frames are processed in real-time. Another key difference is how the local classifiers are used. Video SnapCut utilizes local classifiers *only* along the object boundaries, whereas in our approach they are trained at each of the pixel locations. Appearing as being redundant, this in fact facilitates the propagation of label information to future frames without explicitly tracking foreground and background motions. In contrast, SIFT features are firstly employed in Video SnapCut to estimate rigid motion, which is followed by optical flow to compute per-pixel motion. This nevertheless leads to a much more complex and computational demanding algorithm.

Finally, compared to our preliminary work that focuses on foreground segmentation [19], the algorithm discussed here incorporates an additional matting step (Section IV-C) into the original train-relabel procedure, allowing both foreground segmentation and boundary matting problems to be

**Algorithm 1** Foreground Segmentation From User Strokes

---

```

for each input frame  $I^t$  do
  if  $t == 0$  then
    Initialize the label map  $L^0$  based on input stroke;
    Initialize the matting pixel set  $M$  to empty;
  else
    Train  $\mathcal{F}$  &  $\mathcal{B}$  using  $I^{t-1}$  &  $G^{t-1}$  for  $p \notin M$  (Alg. 2);
    Label  $I^t$  using  $\mathcal{F}$  &  $\mathcal{B}$  to obtain  $L^t$  (Alg. 3 w/o spatial decay);
    Apply temporal decay to all support vectors in  $\mathcal{F}$  &  $\mathcal{B}$ ;
    Reset the matting pixel set  $M$  to empty;
  end if
  repeat
    Train  $\mathcal{F}$  &  $\mathcal{B}$  using  $I^t$  &  $L^t$  for  $p \notin M$  and using  $\alpha^t$ ,  $F^t$ ,
    &  $B^t$  for  $p \in M$  (Alg. 2);
    Relabel  $I^t$  using  $\mathcal{F}$  &  $\mathcal{B}$  and update  $L^t$  (Alg. 3);
    if matting is needed then
      Update the matting pixel set  $M$ ;
      Estimate  $\alpha^t$ ,  $F^t$ , &  $B^t$  for  $p \in M$ ;
    end if
  until there are no more changes in  $L^t$ 
  Find optimal binary segmentation  $G^t$  using graph cuts;
end for

```

---

solved in an integrated manner. To properly utilize the information extracted from matting calculation, the training process (Section IV-A) has been revised. The benefit of this integrated approach is demonstrated in Figure 12. In addition, more detailed explanation on the algorithm (e.g., Algorithm 1), new experimental results (e.g., Figures 13-12), analysis on limitations (Section V-D), and more precise report on processing time (Section V-E) are added throughout the paper.

### III. KEY BUILDING BLOCKS

#### A. Binary SVMs vs C-1SVMs

Intuitively, being a binary classification problem, foreground segmentation seems best solved by binary SVMs. However, we hypothesize that better performance can be achieved using two C-1SVMs, which learns foreground and background distributions separately. Here are the reasons:

First, foreground and background may not be well separable in the color feature space. For example, the black sweater and the dark background shown in Figure 1(a) share a similar appearance. As a result, it is not proper to deal with this scenario by means of training a global binary SVM and use it to classify the entire image. Moreover, trying to train local binary SVMs at each pixel location is problematic as well since in most cases merely one of the two (foreground or background) types of observations is locally available. In fact, even in areas that both foreground and background examples are available, modeling the two sets separately using the C-1SVMs produces two hyperplanes that enclose the training examples more tightly. As illustrated in Figure 2, this helps toward better detecting and handling of ambiguous cases. This hypothesis is also supported by empirical evidence such as the experiments conducted in Figure 7.

#### B. Batch vs. Online Learning

Training a SVM using a large set of examples is a classical batch learning problem, the solution of which can be found

through minimizing a quadratic objective function. Previous studies [3] have shown that a similar performance can be achieved using online learning by showing all examples repetitively to an online learner. A distinct advantage of online learning is that it produces a partially trained model immediately, which is then gradually refined toward the final solution. In our application, 1SVM training is part of an iterative process: the learned 1SVMs are used to classify the pixels, which in turn are used to update the 1SVMs. Therefore, faster convergence can be achieved using online learning since we can classify the pixels using partially learned 1SVMs, providing quicker feedback in the loop.

The online learner we use follows the one proposed by [6] and [7]. Let  $f_j(\cdot)$  be a score function of example  $j$  in the online learning sequence and  $k(\cdot, \cdot)$  be a kernel function, and denote  $w_j$  a non-negative weight of example  $j$ . We further denote  $\text{clamp}(\cdot, A, B)$  an identical function of the first argument bounded from both sides by  $A$  and  $B$ . When a new example  $x_j$  arrives, the score function becomes  $f_j(x_j) = \sum_{i=1}^{j-1} w_i k(x_i, x_j)$ , and the update rule for weights is:

$$w_j = \text{clamp}\left(\frac{\gamma - (1 - \tau)f_j(x_j)}{k(x_j, x_j)}, 0, (1 - \tau)C\right),$$

$$w_i \leftarrow (1 - \tau)w_i \quad \forall i = 1, \dots, j - 1, \quad (1)$$

where  $\gamma := 1$  is the margin,  $\tau \in (0, 1)$  the decay parameter, and  $C > 0$  the cut-off value.

#### C. Reweighting Scheme

The above online learning algorithm does not consider the situation where a given example is used repetitively during training. Hence, directly applying Equation (1) adds multiple support vectors to the model, all come from the same example but have different weights. To address this, here we apply an explicitly *reweighting* scheme: If a training example  $x_j$  arrives and it turns out identical to an existing support vector  $(x_j, w_j)$  inside the model, this support vector is first taken out when computing the score function, it is then included with its newly obtained weight,  $w'_j$ , to substitute the original one  $w_j$ . That is:

$$f_j(x_j) = \sum_{i=1}^{j-1} w_i \chi(x_i \neq x_j) k(x_i, x_j), \quad (2)$$

$$w_j \leftarrow w'_j = \text{clamp}\left(\frac{\gamma - f_j(x_j)}{k(x_j, x_j)}, 0, C\right), \quad (3)$$

where  $\chi(\cdot)$  is an indicator function:  $\chi(\text{true}) = 1$  and  $\chi(\text{false}) = 0$ .

Intuitively, our modified online learning method resets the weight component of a particular support vector  $(x_j, w_j)$ , based on how well the separating hyperplane defined by the remaining support vectors is able to classify example  $x_j$ . This reweighting process can either increase or decrease  $w_j$  and hence decay is no longer necessary. With fewer operations, this leads to a method with shorter training time. Empirical simulations such as the ones reported in Figure 8 confirm that the performance of the learned model using this online

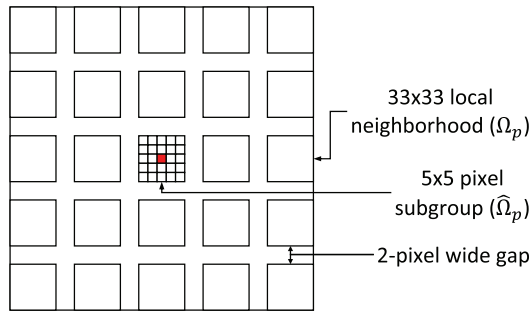


Fig. 3. The neighborhood system used for 1SVM training. For the center pixel  $p$  shown in red, the pixels within the local  $33 \times 33$  window is divided into 25 subgroups, with each subgroup having 25 pixels and a 2-pixel wide gap between adjacent subgroups. This setup reduces the number of examples used for training at each pixel location from 1089 to 25; see text for details.

learning with reweighting scheme is as competitive as that of batch learning.

#### D. Max-Pooling of Subgroups

Training 1SVMs with large scale examples is known to be computationally expensive, which becomes a serious issue in our real-time processing scenario. In addition to online learning, we propose a novel idea to alleviate this by what we term as max-pooling of subgroups: We divide the whole example set  $\Psi$  into  $N$  non-intersecting groups  $\psi_i$  ( $0 \leq i < N$ ) and train a 1SVM on each group. Then the original 1SVM score function is approximated by the maximum operation of these 1SVM score functions from subgroups:

$$f(x) \approx \max_{0 \leq i < N} f^{\psi_i}(x), \quad (4)$$

where  $f^{\psi_i}(\cdot)$  is the score function trained using examples in subgroup  $\psi_i$ .

As will become more clear in later sections, when dividing examples into subgroups, our approach exploits the spatial coherence of images so that the 1SVM trained on each subgroup models local appearance density. Nevertheless, Empirical simulations show that the error introduced by the above approximation is acceptable even when the subgroups are randomly generated (see Figure 8).

## IV. OUR APPROACH

As shown in Algorithm 1, the core of our approach is a train-relabel-matting procedure: Two competing 1SVMs,  $\mathcal{F}_p$  for foreground and  $\mathcal{B}_p$  for background, are trained locally for each pixel  $p$  using known foreground and background colors within the local window  $\Omega_p$ . Once trained,  $\mathcal{F}_p$  and  $\mathcal{B}_p$  are used to jointly label  $p$  as either foreground, background, or unknown. Pixels along the boundary between foreground and background regions are then detected and form a *matting pixel* set  $M$ , on which matting operation is performed.

Since the knowledge learned from neighboring pixels in  $\Omega_p$  is considered in labeling  $p$ , the above procedure effectively propagates known foreground and background information to its neighborhood. As a result, based on only a few initial strokes, the algorithm can segment the whole image and perform matting along object boundaries (see Figure 1(a-i)).

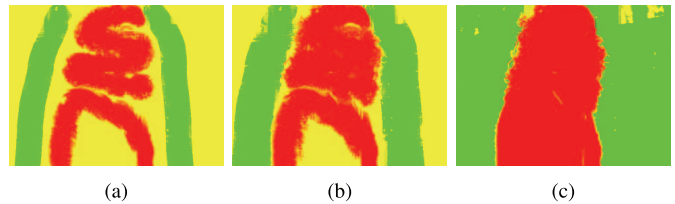


Fig. 4. The incurred losses calculated after different numbers of iterations. These loss values are used to produce the label maps of Figure 1(c-e). Red channel encodes the losses from background 1SVMs and green encodes those from foreground 1SVMs. Yellow and black colors indicate that the corresponding areas are ambiguous in cases where the foreground and background losses are both high or both low. (a) After 1 iteration. (b) After 2 iterations. (c) After convergence.

The same train-relabel-matting procedure is employed for handling temporal changes as well. When a new frame  $t + 1$  arrives, the label  $L^{t+1}(p)$  is initialized automatically using the existing  $\mathcal{F}_p$  and  $\mathcal{B}_p$ . The initial labels, together with newly observed colors, are then utilized to conduct the train-relabel-matting process. Since  $\mathcal{F}_p$  and  $\mathcal{B}_p$  are trained using all pixels within  $\Omega_p$  of frame  $t$ , if any of these pixels moves to  $p$ ,  $\mathcal{F}_p$  and  $\mathcal{B}_p$  are able to classify it properly. Consequently, the algorithm can cope with arbitrary foreground and background movement without *a priori* motion information, as long as the amount of movement is less than the radius of  $\Omega$ .

Under ideal situations, where the appearance distributions of foreground and background pixels are locally separable, the above baseline procedure is sufficient. However, the two distributions may intersect due to fuzzy object boundary, motion blur, or low color contrast. To address these cases, global optimization is employed in favor of a globally consistent and smooth solution. In addition, when moving to a new frame, decaying is applied to existing support vectors for better adapting to temporal changes. Details of the above steps are further discussed in each of the following subsections.

#### A. Train Local C-1SVMs at Each Pixel Location

*Examples Used for Training:* The two competing classifiers at each pixel  $p$ ,  $\mathcal{F}_p$  and  $\mathcal{B}_p$ , are trained using known foreground and background colors within the local window  $\Omega_p$ . In our previous approach [19], where matting is not performed, the observed colors of neighboring pixels with known labels are used as training examples. For foreground objects with highly fuzzy boundaries, some pixels along the boundaries may be labeled as foreground (or background) even though their observed colors contain trace of background (or foreground) colors. In these cases, using observed colors as training example may cause the fore/background 1SVM models being contaminated.

To address this problem, the following strategy is used in our integrated segmentation and matting approach. If a neighboring pixel  $q$  is a non-matting pixel and is labeled as foreground (or background), we use its observed color  $I(q)$  as a training example to update  $\mathcal{F}_p$  (or  $\mathcal{B}_p$ ) based on Equations (2) and (3). If  $q$  is a matting pixel, regardless whether  $q$  has been labeled or not, the estimated foreground and background colors (i.e.  $F(q)$  and  $B(q)$ ) are both used as training examples. To properly use these estimated colors, two

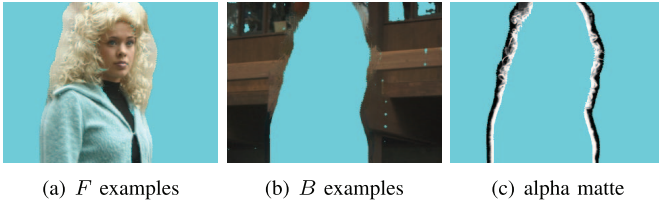


Fig. 5. Alpha matte estimation. Cyan color in (a) and (b) indicates no foreground or background training examples are available for the corresponding pixels. Cyan in (c) indicates non-matting pixels, where the number of foreground or background examples within the local neighborhoods is insufficient.

modifications are made to the aforementioned online learning method:

Firstly, for a given pixel  $q$ , its estimated  $F(q)$  and  $B(q)$  colors may vary during the iterative process. Standard online learning method may insert different variations of the estimated colors into the SVMs, resulting in unnecessary bias in the trained models. To address this problem, we adjust the reweighting scheme so that not only identical examples but also different examples originated from the same pixel are excluded from score calculation. That is:

$$f_j(x_j) = \sum_{i=1}^{j-1} w_i \chi(x_i \neq x_j) \chi(s_i \neq s_j) k(x_i, x_j), \quad (5)$$

where  $s_i$  is a vector keeping the pixel coordinates and frame ID for example  $i$ .

Secondly, the foreground and background colors estimated through matting may not be accurate. The accuracy of an estimated color is related to the corresponding alpha value. For example, if a pixel  $p$  is mostly covered by the foreground ( $\alpha(p) \approx 1$ ), its observed color is close to its true foreground color ( $F(p) \approx I(p)$ ), the accuracy of estimated  $F(p)$  is likely high whereas the one for  $B(p)$  is likely low. Hence, we only use  $F(p)$  for training if  $\alpha(p)$  is larger than a threshold  $\eta$  and use  $B(p)$  if  $\alpha(p) < 1 - \eta$ . Furthermore, to give lower weights to inaccurate training examples, we revise the weight update function to:

$$w_j \leftarrow w'_j = \rho \text{clamp} \left( \frac{\gamma - f_j(x_j)}{k(x_j, x_j)}, 0, C \right), \quad (6)$$

where we set parameter  $\rho = \alpha(p)$  when training foreground 1SVM and  $\rho = 1 - \alpha(p)$  when training background 1SVM.

*Methods for Reducing Training Costs:* Besides the examples used for training, the size of the local window  $\Omega_p$  is also an important parameter. It needs to be sufficiently small so that the local foreground and background appearance distributions are separable, while remain large enough for effective propagation of label information and for covering fore/background motions. Throughout the experiments of this paper, we set  $\Omega_p$  to  $33 \times 33$  pixels, large enough to deal with motions of up to 16 pixels between adjacent frames. While the discussion below focuses on how to reduce training costs under this setting, the idea can be easily applied for other window sizes.

Using a  $33 \times 33$  window means 1089 examples are used for training each 1SVM. Considering training is performed

---

**Algorithm 2** Train C-1SVMs Using Frame  $I$ , Label  $L$ , Alpha Matte  $\alpha$ , Estimated Fore/Background Colors  $F$  and  $B$

---

**for** each pixel  $p$  **do**

**for** each pixel  $q$  in  $\hat{\Omega}_p$  **do**

**if**  $q \notin M$  (i.e., non-matting pixel) **then**

**if**  $L(q) = 1$  (i.e., foreground) **then**

        Use  $I(q)$  to train  $\hat{\mathcal{F}}_p$  based on Eqs. (2) & (3);

**else if**  $L(q) = 0$  (i.e., background) **then**

        Use  $I(q)$  to train  $\hat{\mathcal{B}}_p$  based on Eqs. (2) & (3);

**end if**

**else**

**if**  $\alpha(q) > \eta$  **then**

        Use  $F(q)$  to train  $\hat{\mathcal{F}}_p$  based on Eqs. (5) & (6);

**end if**

**if**  $\alpha(q) < 1 - \eta$  **then**

        Use  $B(q)$  to train  $\hat{\mathcal{B}}_p$  based on Eqs. (5) & (6);

**end if**

**end if**

**end for**

**end for**

---

for 1SVMs at all pixel locations, this is not affordable for real-time processing. To reduce the training cost, the techniques proposed in Sections III-C and III-D are applied.

First, based on the max-pooling scheme of Section III-D, the 1089 examples inside  $\Omega_p$  are divided into 25 subgroups. To further cut computational costs by taking advantage of the spatial coherence among neighboring pixels, we leave a 2-pixel wide gap between adjacent subgroups (see Figure 3). Pixels inside the gap are not used to train  $\mathcal{F}_p$  and  $\mathcal{B}_p$ , but are used for their immediate neighbors. Since these local 1SVMs are trained at all pixel locations simultaneously, after splitting the examples into subgroups, we only need to train the center subgroup at each pixel location. The training for the remaining 24 subgroups will occur at their corresponding center pixel locations. This strategy enables us to reduce the computational costs of training from 1089 examples to merely 25 examples. For the sake of clarity, we reserve symbols  $\mathcal{F}_p$  and  $\mathcal{B}_p$  for the two C-1SVMs obtained through training with all examples in  $\Omega_p$ , and use  $\hat{\mathcal{F}}_p$  and  $\hat{\mathcal{B}}_p$  to denote the C-1SVMs trained using pixels in the center subgroup  $\hat{\Omega}_p$ .

Moreover, as suggested in Section III-C, online learning is employed to provide an on-demand feedback during the train-relabel-matting process. As shown in Algorithm 2, instead of repetitively processing examples in  $\hat{\Omega}_p$  to  $\hat{\mathcal{F}}_p$  and  $\hat{\mathcal{B}}_p$  and waiting for the training to converge, we only process these examples once in an online learning fashion during each train step. The partially trained 1SVMs are thus directly used to perform relabeling. This enables a faster propagation of labeling information through neighborhoods, and empirically it exhibits a faster convergence of the train-relabel-matting process. For example, starting from the input user stokes shown in Figure 1(b), it takes about 40 iterations to propagate labeling information to the whole image and generate segmentation for the first frame. Afterward, it only takes 2-3 iterations to update the 1SVMs and segment a new frame when matting is not required, or up to 5-6 iterations when alpha mattes need to be extracted for a large fuzzy area.

---

**Algorithm 3** Relabel Input Frame  $I$  Using C-1SVMs
 

---

**Require:** Threshold parameters:  $T_{\mathcal{F}}^{low}$ ,  $T_{\mathcal{F}}^{high}$ ,  $T_{\mathcal{B}}^{low}$ ,  $T_{\mathcal{B}}^{high}$ ;  
**for** each pixel  $p$  **do**  
 Initialize approximate scores  $\tilde{f}_{\mathcal{F}}(p)$  and  $\tilde{f}_{\mathcal{B}}(p)$  to 0;  
**for** each subgroup  $\Omega_q$  in  $\Omega_p$  **do**  
 Set  $\xi = (1 - \tau_{spatial})^{\|p-q\|}$ ;  
 Set  $\tilde{f}_{\mathcal{F}}(p) \leftarrow \max(\tilde{f}_{\mathcal{F}}(p), \xi f_{\mathcal{F}_q}(I(p)))$ ;  
 Set  $\tilde{f}_{\mathcal{B}}(p) \leftarrow \max(\tilde{f}_{\mathcal{B}}(p), \xi f_{\mathcal{B}_q}(I(p)))$ ;  
**end for**  
 Set foreground loss  $l_{\mathcal{F}}(p) = \max(0, \gamma - \tilde{f}_{\mathcal{F}}(p))$ ;  
 Set background loss  $l_{\mathcal{B}}(p) = \max(0, \gamma - \tilde{f}_{\mathcal{B}}(p))$ ;  
**if** ( $l_{\mathcal{F}}(p) < T_{\mathcal{F}}^{low}$ ) && ( $l_{\mathcal{B}}(p) > T_{\mathcal{B}}^{high}$ ) **then**  
 Set  $L(p)$  to foreground;  
**else if** ( $l_{\mathcal{B}}(p) < T_{\mathcal{B}}^{high}$ ) && ( $l_{\mathcal{F}}(p) > T_{\mathcal{F}}^{low}$ ) **then**  
 Set  $L(p)$  to background;  
**else**  
 Set  $L(p)$  to unknown;  
**end if**  
**end for**

---

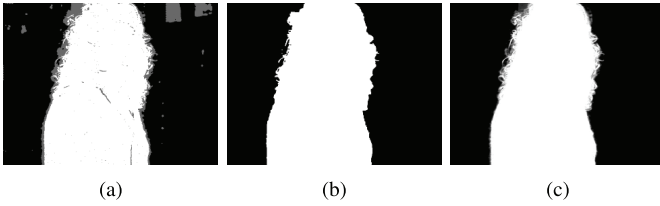


Fig. 6. Global optimization step. The label map (a) computed through applying dual thresholding over the converged incurred losses (Figure 4(c)) contains unlabeled pixels. Performing graph cuts over the whole image based on the incurred losses yields a global optimal binary segmentation (b). The final alpha matte (c) is generated by merging segmentation labels of non-matting pixels in (b) with alpha values of matting pixels in Figure 5(c).

### B. Relabel Each Pixel Using Learned C-1SVMs

Once  $\mathcal{F}_p$  and  $\mathcal{B}_p$  are trained, they are used jointly to classify pixel  $p$ . That is,  $p$  is labeled as foreground or background only if the two competing classifiers give consistent predictions. Otherwise it is labeled as unknown.

As shown in Algorithm 3, the relabeling module starts by computing the scores of observation  $I(p)$  based on Equation (2) with both  $\mathcal{F}_p$  and  $\mathcal{B}_p$ . As depicted in Section III-D, these two quantities are approximated by the scores of  $\hat{\mathcal{F}}$  (or  $\hat{\mathcal{B}}$ ) from a set of nearby subgroups and by taking the maximum. To allow examples closer to  $p$  having higher influence than those further away, we additionally incorporate a spatial decay parameter  $\tau_{spatial}$ .

Both scores are then used to compute the incurred losses of labeling  $p$  as either foreground or background (see Figure 4), respectively.  $p$  is subsequently classified as foreground (or background), iff. the loss of  $\mathcal{F}_p$  (or  $\mathcal{B}_p$ ) is low and the loss of  $\mathcal{B}_p$  (or  $\mathcal{F}_p$ ) is high. This dual thresholding strategy facilitates the detection of ambiguities, which is crucial to prevent incorrect labeling information from being propagated.

### C. Perform Matting Along Foreground Boundary

Both motion blur and fuzzy foreground objects such as hair strands may cause pixels near foreground boundary having a mixture of foreground and background colors. In our previous

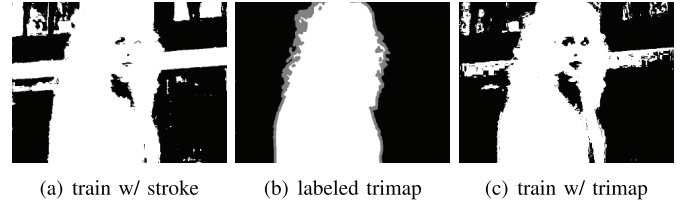


Fig. 7. Classification using the LibSVM implementation of C-SVM. The SVM trained using user strokes shown in Figure 1(b) cannot properly segment the image (a). Allowing the binary SVM to access manually labeled trimap (b) only marginally improves the result (c).

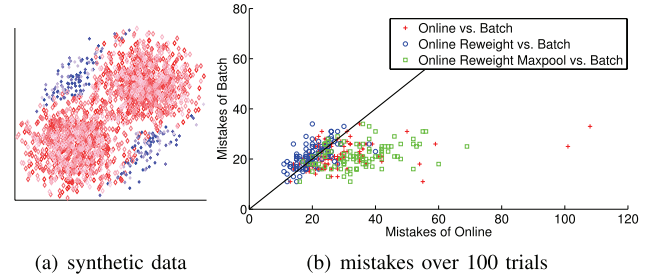


Fig. 8. Performance of batch learning vs. online learning with and without reweighting/max-pooling over 100 trials. (a) Data used for one trial, where foreground and background examples are colored in blue and red, respectively. (b) The mistake made by batch learning for each trial is compared against the conventional online learning (shown in red plus), the online learning with reweighting (blue circle), and the online learning with both reweighting and max-pooling (green square).

approach [19], these pixels are initially labeled as unknown by the aforementioned dual thresholding process and afterwards classified as either foreground or background through graph-cut based global optimization. Here we decompose the observed colors for these pixels into fore/background values and the alpha mattes, directly producing a soft segmentation for the foreground.

It is well-known that matting is an ill-posed problem with possibly multiple solutions. At each pixel, the unknowns on the right side of the following image compositing equation need to be estimated using the known observed color  $I$ :

$$I = \alpha F + (1 - \alpha)B, \quad (7)$$

where  $\alpha \in [0, 1]$  is the alpha matte;  $F$  and  $B$  are the real foreground and background colors for the pixel, respectively.

To solve the problem, we add a further constrain that  $F$  and  $B$  should fit the local foreground and background 1SVMs ( $\mathcal{F}_p$  and  $\mathcal{B}_p$ ) as much as possible. That is, the scores  $f_{\mathcal{F}_p}(F)$  and  $f_{\mathcal{B}_p}(B)$  should be high. Consequently, we optimize the following energy function at each pixel:

$$\arg \max_{F, B, \alpha} e^{-(\alpha F + (1-\alpha)B - I)^2 / 2\sigma_c^2} + f_{\mathcal{F}_p}(F) + f_{\mathcal{B}_p}(B), \quad (8)$$

where parameter  $\sigma_c$  controls the support of the Gaussian. Since  $\mathcal{F}_p$  and  $\mathcal{B}_p$  are not explicitly trained in our approach, we approximate functions  $f_{\mathcal{F}_p}(\cdot)$  and  $f_{\mathcal{B}_p}(\cdot)$  using  $f_{\hat{\mathcal{F}}_u}(\cdot)$  and  $f_{\hat{\mathcal{B}}_v}(\cdot)$ , respectively, where  $u$  and  $v$  are computed by:

$$\begin{cases} u = \arg \max_{q, \hat{\Omega}_q \in \Omega_p} \left( \max(f_{\hat{\mathcal{F}}_q}(F), f_{\hat{\mathcal{F}}_q}(I)) \right) \\ v = \arg \max_{q, \hat{\Omega}_q \in \Omega_p} \left( \max(f_{\hat{\mathcal{B}}_q}(B), f_{\hat{\mathcal{B}}_q}(I)) \right), \end{cases} \quad (9)$$

where  $F'$  and  $B'$  are the foreground and background colors estimated for the pixel in the previous train-relabel-matting iteration. Please note that for a given pixel  $p$ , both  $u$  and  $v$  are constant with respect to the unknowns. As a result, when differentiable kernels such as Gaussian kernel are used for ISVMs, both Equation (8) and its approximation are differentiable. This allows us to use gradient-based approaches to search for the optimal  $F$ ,  $B$  and  $\alpha$  values.

In practice, the matting module starts with determining the matting pixel set  $M$ . We treat a pixel  $p$  as a matting pixel *iff.* sufficient number of examples are used to train both the foreground and the background ISVMs,  $\mathcal{F}_p$  and  $\mathcal{B}_p$ . Here we require the numbers of both foreground and background examples within the  $33 \times 33$  local window to be greater than 50. Please note that, as explained in Subsection IV-A, the training examples can be either the observed colors of labeled non-matting pixels or the previously estimated fore/background colors of matting pixels; see Figure 5(a & b).

Once the matting pixels are determined, a nonlinear conjugate gradient technique is applied to these pixels in parallel. The initial  $F$  and  $B$  values are set to the mean colors of foreground and background training examples within the local  $33 \times 33$  window, respectively. The initial  $\alpha$  is set to the local mean of the label map  $L$  for the first train-relabel-matting pass and to the mean of previously estimated alpha matte for the following passes. During the optimization, we explicitly enforce  $F$ ,  $B$  to remain in range  $[0, 255]$  and  $\alpha$  to remain in range  $[0, 1]$ .

It is worth noting that, even though the matting is performed for different pixels in parallel without explicitly enforcing the smoothness, the estimated alpha matte is still smooth due to the coherence among neighboring ISVMs that models foreground and background colors. In a similar manner, the temporal coherence among the alpha mattes of adjacent frames is also enforced implicitly.

#### D. Apply Global Optimization

Since users provide foreground and background examples using only a few strokes, there may be pixels in the frame with colors that are not recognized by either foreground or background ISVMs. Similar situation also occurs when pixels with new foreground or background colors show up in the frame due to motions. These pixels are labeled as unknown at the end of the train-relabel-matting process. Our next task is to label these unknown pixels, as well as pixels along fore/background boundaries, so that a clean binary segmentation  $G$  can be generated. It is worth noting that  $G$  is needed even when the desired output is an alpha matte  $\alpha$ , in which case  $\alpha$  is obtained by combining the estimated alpha values for matting pixels and labels from  $G$  for non-matting pixels; see Figure 6. Here, we compute  $G$  through optimizing a Markov random field (MRF) based energy function defined over a lattice graph:

$$E(G) = \sum_p U(G(p)) + \sum_{(p,q)} V(G(p), G(q)). \quad (10)$$

where the data term  $\sum U(\cdot)$  sums together the energy (cost) of assigning each pixel to its label, which can be constructed

directly from the local losses:

$$U(G(p)) = \begin{cases} l_{\mathcal{F}}(p) & \text{if } G(p) = 1, \\ l_{\mathcal{B}}(p) & \text{if } G(p) = 0, \end{cases} \quad (11)$$

and the contrast term  $\sum V(\cdot, \cdot)$  encourages segmentation boundary to be aligned with image edges. Here we adopt a widely used contrast term to penalize labeling changes adaptively based on appearance distance between neighboring pixels [12], [35]:

$$V(G(p), G(q)) = \begin{cases} 0 & \text{if } G(p) = G(q), \\ \lambda e^{-\mu \|I(p) - I(q)\|} & \text{else,} \end{cases} \quad (12)$$

where  $\|I(p) - I(q)\|$  computes the Euclidian appearance distance between pixels  $p$  and  $q$ .  $\lambda$  is a constant controlling the strength of the smoothness constraint.  $\mu = (2\langle \|I(p) - I(q)\|^2 \rangle)^{-1}$ , where  $\langle \cdot \rangle$  denotes expectation over all pairs of neighbors in an image sample.

It is well known that graph cuts can usually infer  $G$  efficiently, and it is able to pick up the optimal  $G$  for binary MRFs. As shown in Figure 6(b), unknown pixels are now properly labeled. In practice, we use a GPU version of the push-relabel algorithm to compute the min cuts [18], and limit the number of push-relabel steps to 20, which is found sufficient throughout our experiments.

#### E. Deal With Incoming Frames

When a new frame  $t + 1$  arrives, the following preparations are performed before the train-relabel-matting procedure:

First, the non-matting pixels in  $G^t$  are used to train the C-ISVMs one more time. Since the ambiguous areas are labeled in  $G^t$  using smoothness constraints, using it to train  $\mathcal{F}$  and  $\mathcal{B}$  helps to resolve ambiguities in future frames.

Next, the updated  $\mathcal{F}$  and  $\mathcal{B}$  are used to label the new frame  $t + 1$ . It is worth noting that when labeling the new frame using the C-ISVMs trained from the previous frames, the spatial decay weighting term  $\xi$  is removed. The reason is that a pixel  $p$  may move to any place in its local window  $\Omega_p$  due to the observed motion. The bias toward the center of the window is no longer justified.

Finally, to facilitate the adaptation of C-ISVMs to temporal changes in the foreground and background appearance distributions, a temporal decay is applied: after  $L^{t+1}$  is predicted, the weights of existing support vectors in  $\mathcal{F}$  and  $\mathcal{B}$  are down-weighted by a factor  $(1 - \tau_{temporal})$ .

## V. EXPERIMENTS

To exploit the inherit parallel structure of the proposed algorithm, we implement it on GPU using DirectCompute, which is proposed by Microsoft as an alternative to CUDA and is included in the Direct3D11 API. Unless for the cases (e.g., Figure 11) specified below, the same set of parameter values are used throughout the experiments:  $C = 0.5$ ,  $\tau_{temporal} = 0.25$ ,  $\tau_{spatial} = 0.05$ ,  $T_{\mathcal{F}}^{low} = 0.1$ ,  $T_{\mathcal{F}}^{high} = 0.3$ ,  $T_{\mathcal{B}}^{low} = T_{\mathcal{B}}^{high} = 0.4$ ,  $\sigma_c = 5$ ,  $\eta = 0.2$ . Notice that we set the background labeling requirements



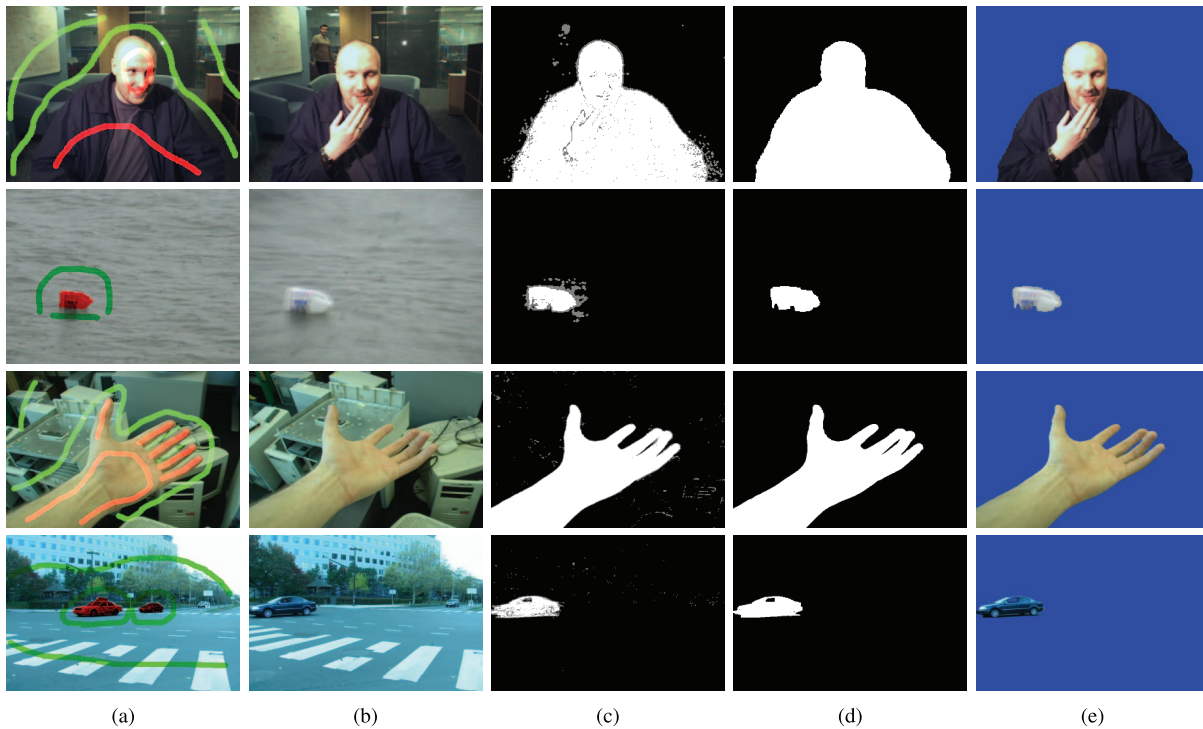


Fig. 9. Results on testbed sequences referred to as (from top to bottom) “JM” [12], “jug” [43], “hand”, and “car” [32]. The results clearly demonstrate the capacity of our algorithm to deal with different challenges, such as background changes (in “JM”), repetitive background motion (in “jug”), camera motion (in “hand” & “car”), strong motion blur (caused by camera zooming in “jug”), non-rigid foreground deformations (in “JM” & “hand”), topology changes (holes in “hand” & “car”), and low fore/background color contrast (in “JM” & “car”). (a) First frame w/strokes. (b) Test frame. (c) Converged label. (d) Binary segmentation. (e) Foreground mask.

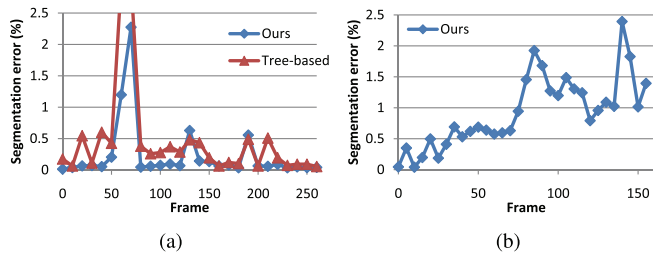


Fig. 10. Segmentation accuracy for two sequences, where ground truth segmentations are available for every 5 or 10 frames. The errors of tree-based approach [41] shown in (a) are based on our readings from Figure 12(a) of their paper. (a) Dataset “JM” used in [12] and [41]. (b) Dataset “IU” used in [24] and [41].

( $l_{\mathcal{B}} < T_{\mathcal{B}}^{low}$  and  $l_{\mathcal{F}} > T_{\mathcal{F}}^{high}$ ) to be looser than those for foreground objects. Leveraged with a relatively high temporal decay  $\tau_{temporal}$ , this introduces a tendency of accepting unseen examples as background, which allows proper management of background changes. Besides, the kernel function  $k(\cdot, \cdot)$  is computed as a Gaussian kernel with  $\sigma = 10$ . The batch learners used for comparison is from LibSVM [4], where the Gaussian kernels are used and the parameters are tuned through cross-validation.

#### A. Evaluation on C-1SVM

We start with comparing the performances of the proposed C-1SVM with the standard binary SVM, as well as evaluating the proposed reweighting and max-pooling strategies.

*C-1SVM v.s. Binary SVM:* Figure 7 shows the segmentation result generated by a global binary SVM for the

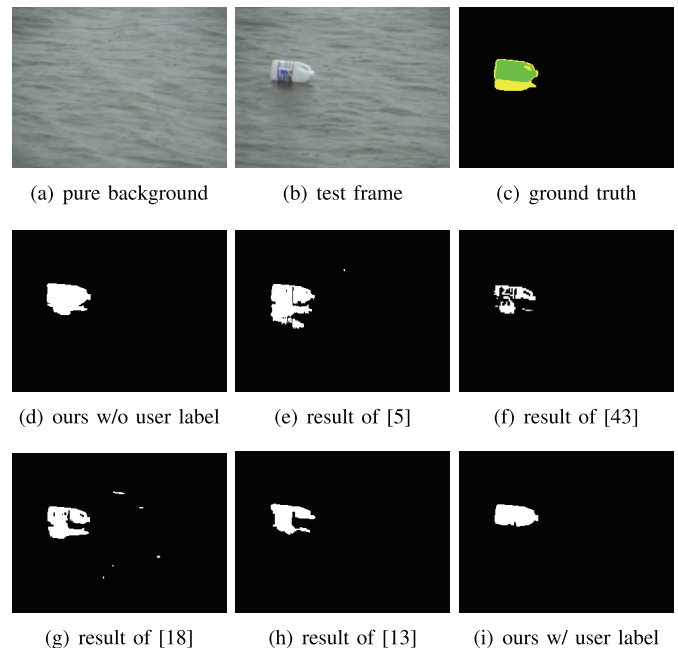


Fig. 11. Comparisons to background subtraction algorithms. When only the background 1SVM is trained using the pure background image (a), the foreground detected by our algorithm for test frame (b) is shown in (d). The results of existing algorithms (e-h) are reported in their respective papers. Alternatively, training both 1SVMs using Figure 9(a) allows our algorithm to label the reflection as background (i).

same image shown in Figure 1(a). The results empirically confirm our previous claim that a global binary SVM does not work well for challenging sequences due to overlapping

fore/background distributions. Having access to much richer labeling information during training does not solve the problem either.

*Batch v.s. Online Learning:* To evaluate the performance of our dedicated online learning method using reweighting and max-pooling strategies, we also compare it to standard batch learning method on synthesized dataset. As shown in Figure 8(a), 2000 2D instances are sampled under a pre-defined Bernoulli prior from two partially overlapped stationary distributions, each forms a mixture of two Gaussians. The Bernoulli prior is biased so more instances are drawn from the distribution representing the background than that from the foreground one. The task is to train a background 1SVM model with 1000 of the instances, which is then used to classify the remaining instances into one of the two classes. When applying max-pooling, the 1000 training instances are randomly split into 25 non-overlapping subgroups, each is used to train a local 1SVM. The sequence of training instances are presented to the learner in 20 repeats for all online learning methods.

Figure 8(b) shows that the online learning method [7] using Equation (1) performs inferior to the batch learning counterpart, which is to be expected and well aligned with previous work [7]. The online learner with reweighting using Equation (3) is shown to perform as well as, or slightly better than, the batch learner. The performance drops when both reweighting and max-pooling is applied, but is still comparable to the conventional online learning approach.

### B. Evaluation on Binary Segmentations

To assess the performance of our algorithm for foreground segmentation tasks, here we conduct both visual and quantitative evaluations on the binary segmentations  $G^t$  generated.

*Results on Testbed Videos:* As displayed in Figure 9, our algorithm is tested over a variety of video scenarios used by previous foreground segmentation papers [5], [12], [24], [32], [41], [43]. The segmentation results are visually satisfactory and are comparable to the state-of-the-art approaches that are designed specifically for video conferencing [12], [24], [41], background subtraction [5], [43], or for handling freely moving camera [32]. Here we refer readers to these papers for their results on the same sequences.

*Quantitative Evaluation Using Ground Truth:* We further evaluate the segmentation quality quantitatively on two sequences. For a fair comparison with previous approaches [12], [24], [41], which uses multiple annotated images for training, here the C-1SVMs are trained using both the first frame and one more selected frame where the initially occluded foreground portion is visible. The quantitative evaluations (see Figure 10) show that the median segmentation errors are 0.07% and 0.88% for the two sequences, respectively. In comparison, [41] reports higher median errors of 0.27% and 2.56% for the same sequences.

*Ability to Work in Background Subtraction Scenario:* With a different set of threshold settings ( $T_{\mathcal{F}}^{low} = \infty$  and  $T_{\mathcal{B}}^{low} = 0.2$ ), the algorithm can also be initialized by one or a few pure background image(s) instead of any stroke. Under this setup, only the local background 1SVMs are trained initially.

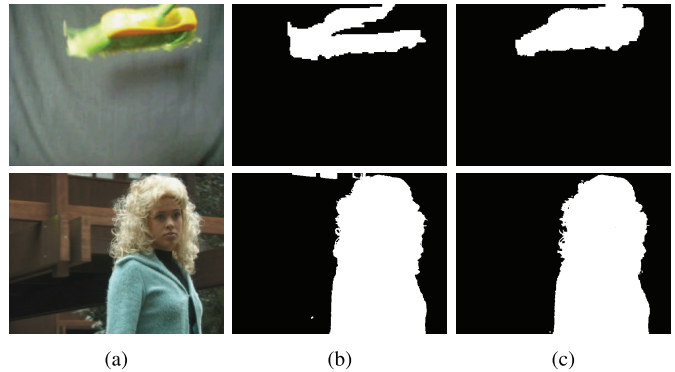


Fig. 12. Comparison on the binary segmentation results generated with and without using the estimated colors to train the classifiers. Top row shows the frame 58 of the “broom” sequence and the bottom row is for the frame 89 of the “kim” sequence. The final alpha mattes generated for these two frames can be found in Figures 13(c) and 16(b), respectively. (a) Input frame. (b) w/o estimated colors. (c) With estimated colors.

As new frames are processed, outliers to the background 1SVMs are classified as foreground, which are then utilized to initiate the training of local foreground 1SVMs. Meanwhile inliers are used to update the background 1SVMs, allowing the algorithm adapt to dynamic changes and background motion. As displayed in Figure 11, the additional foreground 1SVMs help our algorithm remembering the detected foreground appearance, and as a result, lead to better performance than previous work using only local background models such as [5], [18], and [43].

*Benefit of Integrated Matting Operation:* Instead of performing segmentation and matting as two separate steps, our approach uses an integrated train-relabel-matting operation. As a result, not only the segmentation result can guide the matting through providing fore/background color models, but also the matting operation can benefit the segmentation through estimating fore/background colors in fuzzy areas for training the corresponding classifiers. To confirm whether the matting step can indeed improve binary segmentation, Figure 12 compares the results generated with and without using the estimated colors. It shows that when the estimated colors are used to train the fore/background 1SVMs, the binary segmentation results are more accurate.

### C. Evaluation on Matting Results

Next, we evaluate the performance of our algorithm on extracting objects with fuzzy boundaries and compare it with existing approaches.

*Comparison With Existing Approaches:* Figures 1 and 9 show the results of our algorithm for several sequences used by previous video matting papers [1], [8], [9], [17], [20]. Note that for the “broom” and “class” datasets, we adjusted the parameter  $T_{\mathcal{F}}^{low} = 0.3$  so that the portions of occluded-then-reappeared foreground can be properly segmented.

Visual inspection suggests that the alpha mattes estimated are smooth and rich in details. Available results generated by authors of existing approaches are also shown in Figures 9 and 14 for comparison. They suggest that the performance of our algorithm is on par with existing approaches,

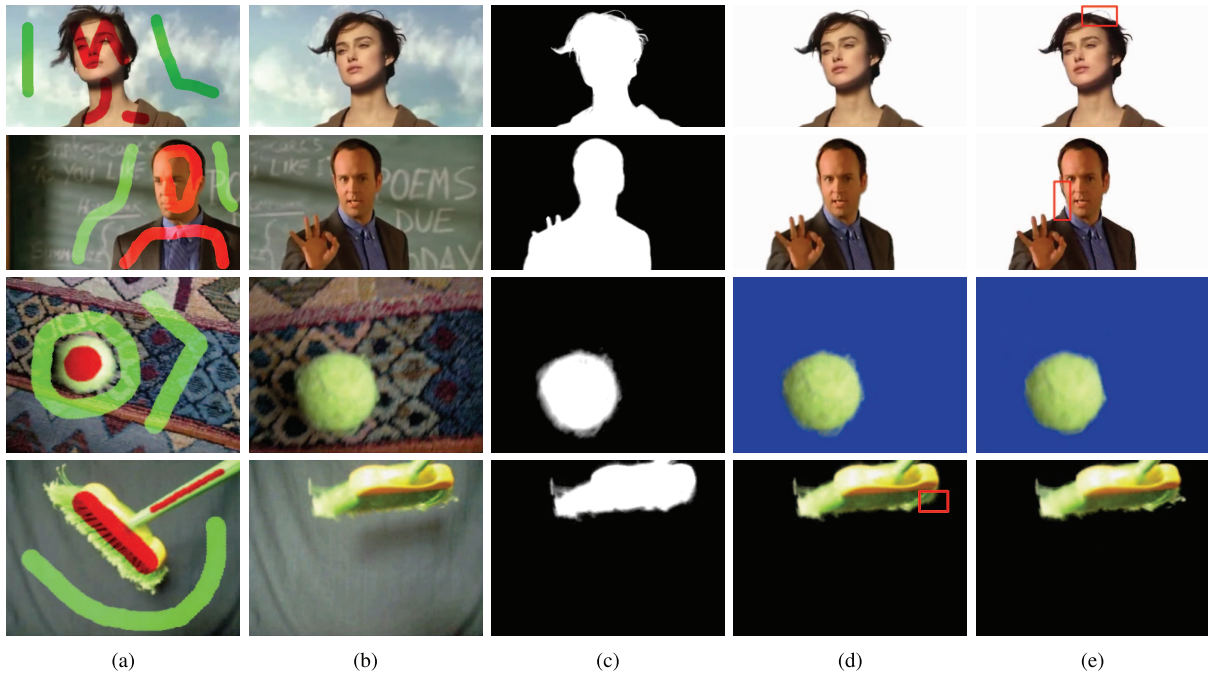


Fig. 13. Results on testbed sequences referred to as (from top to bottom) “wind”, “class” [1], “ball”, and “broom” [17]. The results show that our algorithm can properly handle background motions (in “class” & “ball”) and strong motion blur (in “ball” & “broom”). The results of geodesic matting [1] (for “wind” & “class”) and shared matting [17] (for “ball” & “broom”) are generated by the authors (e). For better comparison, matching background colors are used in our composites (d). Areas with suboptimal matte results are highlighted with red boxes. (a) First frame w/strokes. (b) Test frame. (c) Alpha matte. (d) Composite. (e) Other approaches.

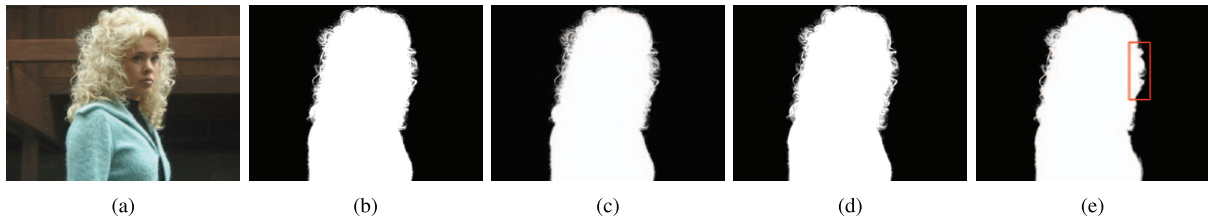


Fig. 14. Comparison on alpha mattes generated by different approaches. Areas with suboptimal matte results are highlighted with red boxes. (a) Test frame. (b) Our result. (c) Bayesian matting [9]. (d) Multi-ch. Poisson [20]. (e) Nonlocal matting [8].

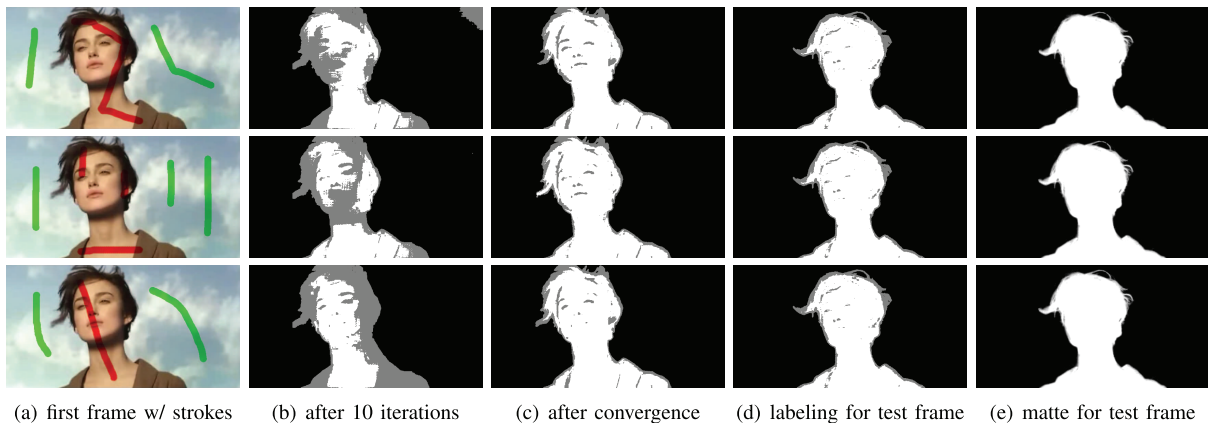


Fig. 15. Matting results obtained under different stroke inputs than the one shown in Figure 13(a). Under different stroke inputs (a), labeling information is propagated differently across the image (b), but the final per-pixel labeling results (c) are similar. The impact of the stroke variations on the first frame is even less noticeable in the per-pixel labeling results for the test frame (d). As a result, the alpha mattes generated for the test frame (e) are nearly identical to the one shown in Figure 13(c).

which often require additional steps to compute dense trimaps [17], [20] or background colors [9] for each frame. Some of these approaches also require long computational time [8], [9] or process all frames in a batch manner [1], making them hard to be applied to real-time video processing.

*Robustness w.r.t. Input Stroke Variations:* To evaluate the impact of the users’ input stroke variations on the matting results, here we also test our algorithm under different input settings. As shown in Figure 15, changing input stroke locations affects how the labeling information is propagated across

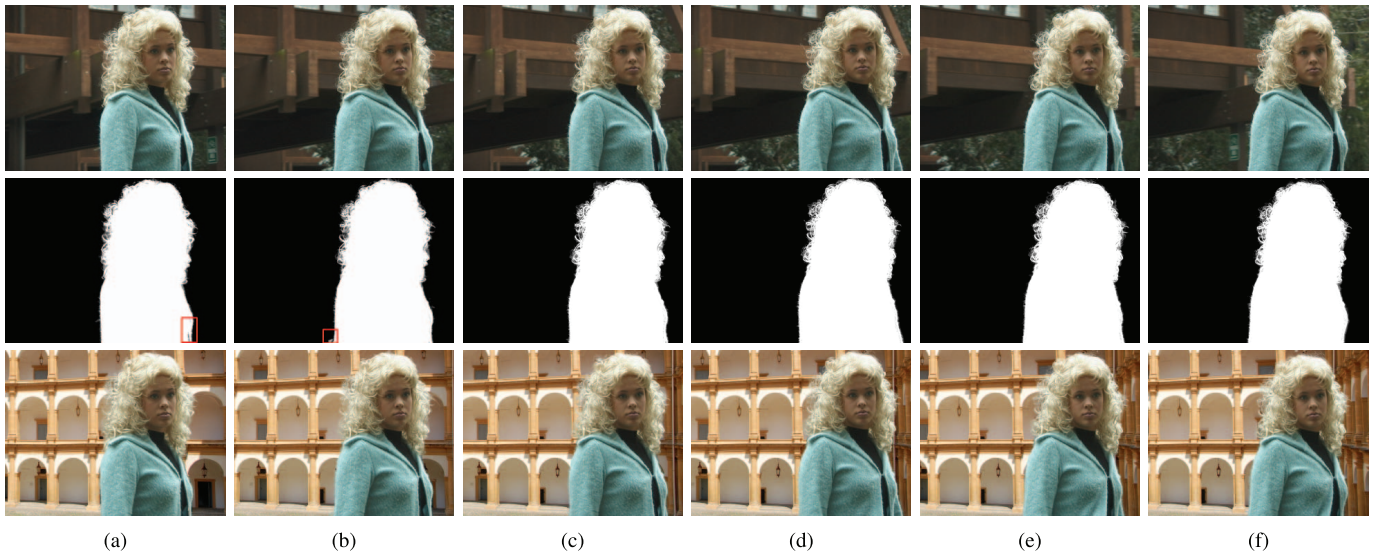


Fig. 16. Alpha mattes (middle row) and composites (bottom row) generated for adjacent frames. Despite that the background behind the fuzzy hair changes from bright window to brown building then to green leaves, our approach extracts temporal coherent alpha mattes. Please note that although artifacts show up in the estimated alpha mattes (highlighted with red boxes), they are hardly noticeable in the final composites. (a) Frame 86. (b) Frame 89. (c) Frame 92. (d) Frame 95. (e) Frame 98. (f) Frame 101.

the image, but has very little impact on the final alpha matte generated. This suggests that our program is robust against users' input variations.

*Evaluation on Temporal Coherence:* Unlike some existing video matting approaches [2], [8], [28], our approach does not explicitly enforce the temporal coherence among adjacent frames. Instead, it relies on the local fore/background classifiers trained at each pixel location,  $\mathcal{F}$  and  $\mathcal{B}$ , being stable across multiple frames. As a result, for pixels from different frames but with similar observed colors, the optimal alpha values found by maximizing Equation (8) will be similar. To evaluate the effectiveness of this strategy, Figure 16 shows the matting results for six nearby frames of the "kim" dataset, which contains a large and detailed fuzzy area. The results demonstrate that our approach is able to generate coherent alpha mattes for the foreground object even in front of a changing background.

To quantitatively evaluate temporal coherence, we used the measure proposed by Lee, et al. [25], which quantifies the amount of temporal flicker by computing the weighted difference in alpha values between successive frames. Figure 17 plots this weighted alpha difference measure for our approach and three existing techniques. It shows that ours produces more coherent video matting results than the multi-frame nonlocal matting algorithm, which is one of the state-of-the-art techniques [8].

*Quantitative Evaluation on Image Matting:* Due to the lack of video sequences with ground truth alpha mattes, we cannot perform quantitative evaluation on videos. Hence, here we test our approach on an image matting testbed [38] by treating each test image as the first frame of a video and the corresponding trimap as input user strokes. We choose this testbed over the newer one [30] because: 1) it includes trimaps with ten different levels of accuracy, allows us to evaluate the performances under different user input settings; and 2) the newer one contains highly transparent

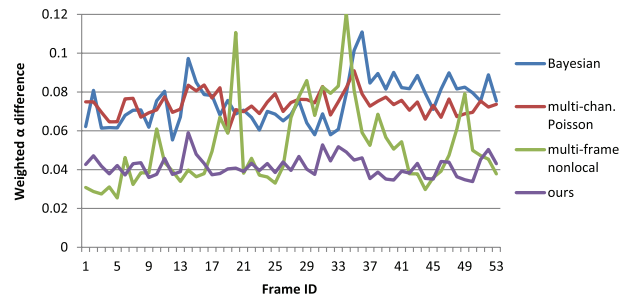


Fig. 17. Quantitative temporal coherence comparison among four approaches on the "kim" dataset: the Bayesian matting [9], the multi-channel Poisson matting [20], the multi-frame nonlocal matting [8], and our approach.

























objects, which boundary matting algorithms, such as ours, cannot handle.

The performances of our approach under the best and the worst trimaps, in comparison with several classical matting algorithms, are shown in Table I. The results indicate that overall our approach performs better than Poisson matting [34], random walk [21], and Bayesian matting [10] approaches, where the last one is used for processing individual frames in Bayesian video matting [9]. However, it does not do as well as closed-form matting [26], robust matting [38], and matting with comprehensive sampling sets [31] since it is designed to efficiently extract foreground objects from videos in real-time and apply matting only along object boundaries. It trains local classifiers using fore/background examples from a relatively small neighborhood and expects examples from previous frames to contribute to the training. When dealing with single images and when matting pixels do not have sufficient foreground or background examples nearby, such as the thin and isolated hair strands in T7, our approach does not perform well.

It is worth noting that the relative performance of our approach improves as the trimap gets less accurate. On tests with the largest unknown areas, i.e., T8, T1, and T2 under their worst trimaps, our approach ranks 3, 4, and 4, respectively.

TABLE I

QUANTITATIVE COMPARISON WITH IMAGE MATTING TECHNIQUES ON A GROUND TRUTH DATASETS WITH EIGHT TEST IMAGES (T1~T8) [38]. EACH CELL SHOWS THE MINIMUM AND MAXIMUM ERRORS UNDER TRIMAPS WITH DIFFERENT LEVELS OF ACCURACY. THE RESULTS OF THE FIRST SEVEN TECHNIQUES ARE REPORTED IN [38], WHEREAS THOSE OF THE NEXT TWO ARE GENERATED USING THE IMPLEMENTATIONS THAT THE AUTHORS MADE AVAILABLE. ON AVERAGE OUR APPROACH RANKS 6.3 IN MINIMUM ERROR AND 5.1 IN MAXIMUM ERROR AMONG THE TEN APPROACHES

	T1	T2	T3	T4	T5	T6	T7	T8
Test image								
Best trimap								
Worst trimap								
Poisson [34]	717:1959	477:1577	843:1736	340:1330	451:2891	879:3174	359:1830	832:2442
Rand. Walk [21]	285:343	482:748	218:278	198:307	151:393	279:638	274:401	1732:1795
Knockout2 [11]	155:321	326:678	113:180	154:596	33:336	338:1387	150:516	435:888
Bayesian [10]	453:1198	415:2006	137:1005	82:724	28:687	194:938	69:406	120:4994
Iterative BP [37]	61:182	118:337	130:205	69:356	27:227	207:903	78:362	214:553
Closed-form [26]	79:97	105:234	61:80	59:137	23:356	157:237	77:143	503:582
Robust [38]	46:85	44:101	38:67	41:95	10:155	69:381	31:165	114:394
KNN [27]	335:341	338:388	135:141	194:276	52:92	196:455	100:188	474:600
CSS [31]	82:116	170:256	28:84	54:90	8:44	72:303	113:146	192:307
Ours (rank)	124(5):167(4)	191(5):301(4)	209(8):228(7)	99(6):177(4)	50(7):273(5)	204(6):693(6)	201(8):450(8)	237(5):430(3)

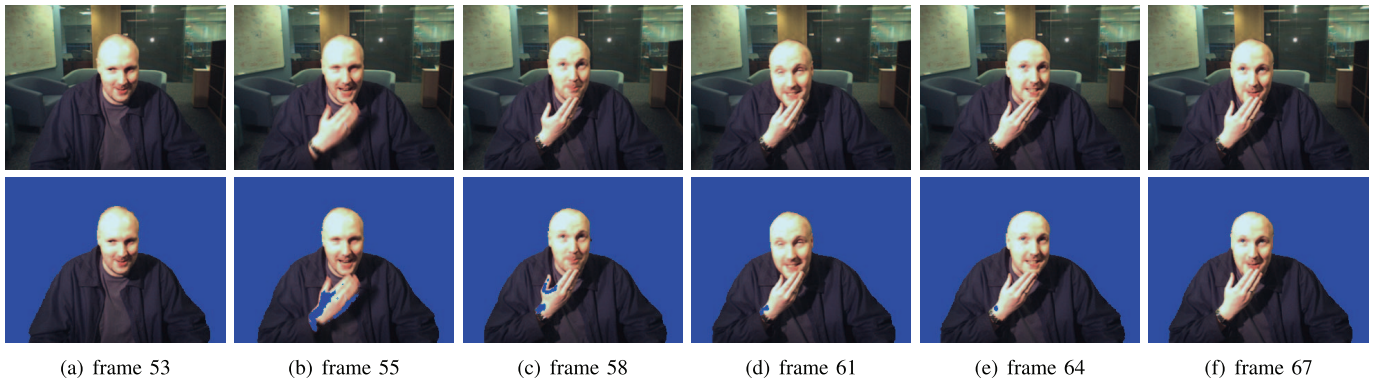


Fig. 18. A failure case: When the hand suddenly shows up in (b), both local foreground and background ISVMs classify pixels with unseen colors as outliers. The final segmentation labels these pixels incorrectly due to the bias toward background. The algorithm corrects the mistakes in 10 frames without any user intervention (c-f).

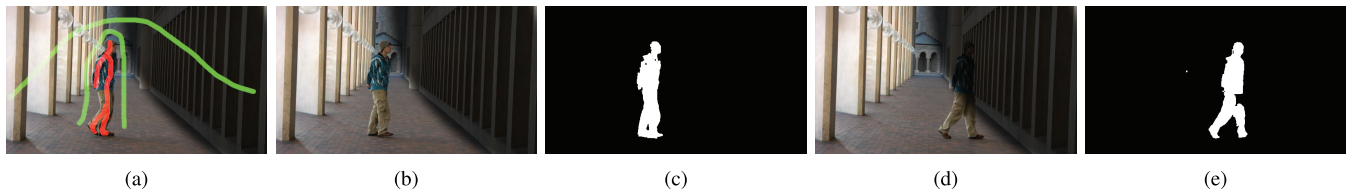


Fig. 19. Another failure case for the “walkman” [2] sequence: Given the training frame and user labels (a), our approach leaves holes on the foreground coat (c) since the details are undersampled. It also fails to label the foreground pants when the person walks into the shadow (e) because its color becomes closer to the background wall than its original color. (a) First frame w/strokes. (b) Frame 5. (c) Segmentation for (b). (d) Frame 41. (e) Segmentation for (d).

This suggests that our approach is robust with respect to labeling variation, a finding that agrees with Figure 15. It also hints that our relative performance can be even better when sparse user strokes rather than trimaps are used as input, which our approach is designed for.

#### D. Limitations

*Handling (Locally) Novel Foreground Colors:* To achieve real-time performance through parallel processing, our approach labels a pixel based on local classifiers only.

When pixels with unseen colors show up in the local neighborhood, both foreground and background classifiers return low scores, resulting the pixels being labeled as unknowns. When computing the final binary segmentation through global optimization, these unknown pixels tend to be labeled as background due to the aforementioned bias toward accepting unseen examples as background. While this strategy helps to correctly handle background changes, such as the background person in “JM” and new background scene in “kim”, it occasionally introduces errors if locally unseen foreground appearances are presented (see e.g. Figure 18). Nevertheless, thanks to the redundancy of using two competing ISVMs, the incorrect labels do not affect the training of foreground ISVMs, which gradually recognize the novel foreground colors. As a result, the mistakes are corrected after a few consecutive frames without any user intervention.

*Handling Low Resolution Images:* To obtain proper labeling, the proposed C-ISVMs need sufficient training examples. When the input video is of low resolution or there are areas rich in details but undersampled, a given fore/background color may be represented by only a couple of pixels and the observed fore/background color may change across the frames due to aliasing. In these cases, the C-ISVMs may fail to recognize the colors and improperly label the pixels; see Figure 19(b-c).

*Handling Illumination Changes:* The proposed algorithm separates foreground and background using color cues only. When sudden illumination changes cause the color of foreground object to vary rapidly and match to the one for background, or vice versa, incorrect labels will be given; see Figure 19(d-e). To address this limitation, additional user interactions would be needed [2].

*Handling Large Semi-Transparent Objects:* Our approach is mainly designed for extracting foreground objects from dynamic backgrounds with minimal user interactions. While the added video matting functionality allows the algorithm to properly extract alpha mattes along the boundaries of fuzzy objects, it cannot be used to handle large semi-transparent regions since the matting relies on locally trained foreground and background classifiers. For highly transparent objects, such as the ones used for image matting evaluation [30], a more global classifier trained using non-local examples would be needed.

### E. Processing Time

Finally, to measure the processing time of the proposed approach, we run our implementation on a Lenovo ThinkStation S20 with nVidia GeForce GTX 480 GPU, which is a system released four years ago. For the VGA-sized “kim” sequence with large fuzzy area, our approach runs at 14 FPS without matting and 8 FPS with matting. Please note that this does not include the processing time for the first frame, which highly depends on the size of the unknown area. When the input is sparse user strokes as shown in Figure 1(b), the algorithm takes 1.5 seconds to propagate known labels to the rest of the frame. On the other hand, if a detailed trimap is provided, the algorithm can process the first frame almost as fast as the remaining frames.

TABLE II  
THE PROCESSING TIME FOR THE FIVE MATTING  
SEQUENCES SHOWN IN FIGURES 1 AND 13

	sequences		time (sec/frame)	
	resolution	# of frames	1st frame	remaining frames
kim	640 × 480	105	1.587	0.130
class	624 × 360	70	1.303	0.093
wind	688 × 358	77	1.384	0.110
ball	316 × 236	72	0.448	0.032
broom	316 × 236	114	0.496	0.036

The precise processing times needed for the different frames on all five matting sequences are reported in Table II. They show that the frame rates for the four sequences shown in Figure 13 range from 9 FPS to 31 FPS, depending mostly on the resolution of the video.

## VI. OUTLOOK AND DISCUSSION

A novel foreground segmentation algorithm is proposed in this paper that is able to efficiently and effectively deal with live videos. The algorithm is easy to implement, simple to use, and capable of handling a variety of difficult scenarios, such as dynamic background, camera motion, topology changes, and fuzzy objects. For fuzzy objects, the integrated boundary matting step can effectively pull the matte, allowing seamless composites over new backgrounds. Experiments on standard testbed videos demonstrate that our algorithm possesses comparable or superior performance comparing to the state-of-the-art approaches designed for specifically for background subtraction [5], [18], [43], foreground segmentation [32], [41], and video matting [9], [20].

Possible future research directions include incorporating other visual cues such as texture and shape into the proposed algorithm. We also plan to embed the proposed algorithm into an real-time interactive video matting application, where users can see the matting results as soon as they draw the strokes. This would allow them to easily control the matting and to correct errors in future frames by providing additional strokes.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the constructive comments that they received from anonymous reviewers. They would also like to thank Dr. Yung-Yu Chuang, Dr. Antonio Criminisi, Dr. Stan Sclaroff, Dr. Yaser A. Sheikh, Dr. Seth Teller, Dr. Xue Bai, Dr. Jue Wang, Dr. Yu-Wing Tai, and Mr. Eduardo S. L. Gastal for sharing their datasets or experiment results with us or online.

## REFERENCES

- [1] X. Bai and G. Sapiro, “Geodesic matting: A framework for fast interactive image and video segmentation and matting,” *Int. J. Comput. Vis.*, vol. 82, no. 2, pp. 113–132, 2009.
- [2] X. Bai, J. Wang, D. Simons, and G. Sapiro, “Video SnapCut: Robust video object cutout using localized classifiers,” *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. ID 70.
- [3] L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” in *Advances in Neural Information Processing Systems 20*. Red Hook, NY, USA: Curran Associates, 2008, pp. 161–168.

- [4] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001. [Online]. Available: <http://www.csie.ntu.edu.tw/~jlin/libsvm/>
- [5] L. Cheng and M. Gong, "Realtime background subtraction from dynamic scenes," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 2066–2073.
- [6] L. Cheng, M. Gong, D. Schuurmans, and T. Caelli, "Real-time discriminative background subtraction," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1401–1414, May 2011.
- [7] L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli, "Implicit online learning with kernels," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA, USA: MIT Press, 2007, pp. 249–256.
- [8] I. Choi, M. Lee, and Y.-W. Tai, "Video matting using multi-frame nonlocal matting Laplacian," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 540–553.
- [9] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," in *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, 2002, pp. 243–248.
- [10] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, "A Bayesian approach to digital matting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Dec. 2001, pp. II-264–II-271.
- [11] *Knockout User Guide*, Powerworld Corp., Champaign, IL, USA, 2002.
- [12] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 53–60.
- [13] G. Dalley, J. Migdal, and W. E. L. Grimson, "Background subtraction for temporally irregular dynamic textures," in *Proc. IEEE Workshop Appl. Comput. Vis.*, Jan. 2008, pp. 1–7.
- [14] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 228–241.
- [15] J. Fan, X. Shen, and Y. Wu, "Closed-loop adaptation for robust tracking," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 411–424.
- [16] J. Fan, X. Shen, and Y. Wu, "Scribble tracker: A matting-based approach for robust tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1633–1644, Aug. 2012.
- [17] E. S. L. Gastal and M. M. Oliveira, "Shared sampling for real-time alpha matting," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 575–584, May 2010.
- [18] M. Gong and L. Cheng, "Real-time foreground segmentation on GPUs using local online learning and global graph cut optimization," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
- [19] M. Gong and L. Cheng, "Foreground segmentation of live videos using locally competing ISVMs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2105–2112.
- [20] M. Gong, L. Wang, R. Yang, and Y.-H. Yang, "Real-time video matting using multichannel Poisson equations," in *Proc. Graph. Interf.*, 2010, pp. 89–96.
- [21] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," in *Proc. Vis., Imag., Image Process.*, 2005, pp. 423–429.
- [22] E. Hayman and J.-O. Eklundh, "Statistical background subtraction for a mobile observer," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Oct. 2003, pp. 67–74.
- [23] N. Joshi, W. Matusik, and S. Avidan, "Natural video matting using camera arrays," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 779–786, Jul. 2006.
- [24] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 407–414.
- [25] S.-Y. Lee, J.-C. Yoon, and I.-K. Lee, "Temporally coherent video matting," *Graph. Models*, vol. 72, no. 3, pp. 25–33, 2010.
- [26] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 228–242, Feb. 2008.
- [27] D. Li, Q. Chen, and C.-K. Tang, "Motion-aware KNN Laplacian for video matting," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3599–3606.
- [28] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 595–600, 2005.
- [29] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand, "Defocus video matting," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 567–576, 2005.
- [30] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1826–1833.
- [31] E. Shahrian, D. Rajan, B. Price, and S. Cohen, "Improving image matting using comprehensive sampling sets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 636–643.
- [32] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 1219–1225.
- [33] Y. Sheikh and M. Shah, "Bayesian object detection in dynamic scenes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 74–79.
- [34] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson matting," in *Proc. ACM SIGGRAPH*, 2004, pp. 315–321.
- [35] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum, "Background cut," in *Proc. 9th Eur. Conf. Comput. Vis.*, 2006, pp. 628–641.
- [36] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen, "Interactive video cutout," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 585–594, 2005.
- [37] J. Wang and M. F. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, Oct. 2005, pp. 936–943.
- [38] J. Wang and M. F. Cohen, "Optimized color sampling for robust matting," in *Proc. IEEE Conf. CVPR*, Jun. 2007, pp. 1–8.
- [39] J. Wang and M. F. Cohen, "Image and video matting: A survey," *Found. Trends Comput. Graph. Vis.*, vol. 3, no. 2, pp. 97–175, Jan. 2007.
- [40] L. Wang, M. Gong, C. Zhang, R. Yang, C. Zhang, and Y.-H. Yang, "Automatic real-time video matting using time-of-flight camera and multichannel Poisson equations," *Int. J. Comput. Vis.*, vol. 97, no. 1, pp. 104–121, 2012.
- [41] P. Yin, A. Criminisi, J. Winn, and I. Essa, "Tree-based classifiers for bilayer video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
- [42] T. Yu, C. Zhang, M. Cohen, Y. Rui, and Y. Wu, "Monocular video foreground/background segmentation by tracking spatial-color Gaussian mixture models," in *Proc. IEEE Workshop Motion Video Comput.*, Feb. 2007, p. 5.
- [43] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *Proc. 9th Int. Conf. Comput. Vis.*, Oct. 2003, pp. 44–50.



**Minglun Gong** received the Ph.D. degree from the University of Alberta, in 2003, the M.Sc. degree from Tsinghua University, in 1997, and the B.Eng. degree from Harbin Engineering University, in 1994. He is currently an Associate Professor with the Memorial University of Newfoundland. His research interests cover various topics in the broad area of visual computing, including computer vision, graphics, visualization, image processing, and pattern recognition.



**Yiming Qian** received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2012, and the M.Sc. degree from the Memorial University of Newfoundland, St. John's, NL, Canada, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada. His research interests include machine learning and computer vision.



**Li Cheng** received the Ph.D. degree in computer science from the University of Alberta, Edmonton, AB, Canada. Prior to joining the Bioinformatics Institute (BII) in 2010, he was with the Statistical Machine Learning Group, NICTA, Australia, TTI-Chicago, USA, and the University of Alberta. He is currently a Research Scientist with BII. His research expertise is mainly on machine learning and computer vision.