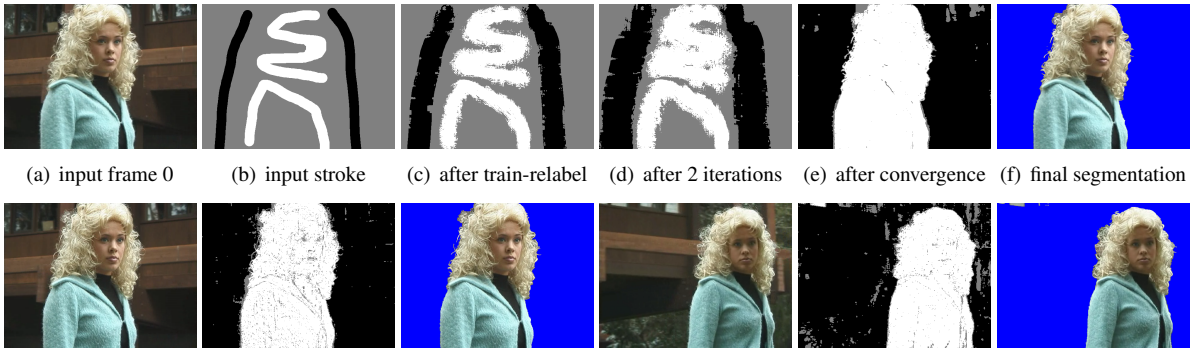
Foreground Segmentation of Live Videos using Locally Competing 1SVMs

Minglun Gong
Memorial University, Canada
http://www.cs.mun.ca/~gong

Li Cheng
BII, A*STAR, Singapore
http://web.bii.a-star.edu.sg/~chengli



(a) input frame 0 (b) input stroke (c) after train-relabel (d) after 2 iterations (e) after convergence (f) final segmentation

(g) input frame 1 (h) initial label (i) segmentation (j) input frame 99 (k) initial label (l) segmentation

Figure 1. Foreground segmentation of the "walk" sequence [8], 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). Repeating training and relabeling leads to convergence (d-e), even though ambiguous (grey) areas still exist. Final segmentation is obtained using graph cuts (f). When the new frames (g & j) arrive, they are first labeled (h & k) using the classifiers trained by previous frames, before the same train-relabel procedure is used to produce the segmentation results (i & l). Note that the proposed algorithm is able to extract the details of the hair without resorting to matting techniques.

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 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 (1SVMs) at each pixel location, which model local color distributions for foreground and background, respectively. We advocate the usage of two competing local classifiers, as it provides higher discriminative power and allows better handling of ambiguities. As a result, our algorithm can deal with a variety of videos with complex backgrounds and freely moving cameras with minimum user interactions. In addition, by introducing novel acceleration techniques and by exploiting the parallel structure of the algorithm, real-time processing speed is achieved for VGA-sized videos.

1. Introduction

Foreground 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 techniques have been proposed in both computer vision and graphics communities. However, some of them are limited to sequences captured by stationary cameras, whereas others require large training datasets or cumbersome user interactions. Furthermore, most existing algorithms are rather complicated and computationally demanding. As a result, there still lacks an efficient and powerful algorithm capable of processing challenging live video scenes with minimum user interactions.

This paper presents a novel foreground segmentation approach that fills this niche. 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 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 fore/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.

1.1. Related work

There is a vast body of existing work for foreground segmentation. Here we have to focus on the most related ones, which are categorized into unsupervised [18, 15, 14, 5, 10, 13] and supervised [11, 9, 17, 12, 16, 1, 2] approaches.

Unsupervised ones 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 [18, 15] or non-parametric [14, 5] methods. Some of these techniques can handle repetitive background motion, such as rippling water and waving trees, but none can deal with 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 [10]. Recently there is also some new development to deal with freely moving cameras by means of tracking the trajectories of salient features across the whole video [13], where the trajectories are used for estimating the background trajectory space, based on which foreground feature points can be detected accordingly. While this method automatically detects moving objects, it tends to classify background with repetitive motion as foreground, as well as to confuse large rigidly moving foreground objects with background.

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 important methods [11, 9, 17] 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 conferencing applications, these algorithms require a large set of fully annotated images and considerable amount of offline training, which bring up many issues when applying to different scene setups.

As part of the video matting process, foreground segmentation techniques are also considered and developed in graphics literature. Most of these algorithms [8, 12, 16, 1] model a video sequence as a 3D volume of voxels. Users are required to label fore/background on multiple frames or directly on the 3D volume. To enforce temporal coherence, these algorithms usually segment over the entire volume at one time by solving a global optimization problem, which restricts their capacity toward live video processing.

The most relevant method to ours might be Video SnapCut [2], where, starting from a segmentation of the first frame, both global and local classifiers are trained using color and shape cues, then labeling information is propagated to the rest of the video frame by frame. Although the workflows are somewhat similar, the two approaches have notably different objectives. Video SnapCut aims toward high quality video segmentation and produces very convincing results. However, 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, and processes new frames 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 fore/background motion. 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.

1.2. Motivation of the presented algorithm

The key insight of our approach is to maintain two Competing 1-class Support Vector Machines (C-1SVMs) at every pixel location. The two 1SVMs capture the local fore/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 6 suggest that comparing to using a global classifier, the idea of maintaining local classifiers could endow the resulting algorithm with higher discriminative power. Moreover, as will be explained in the next section, the usage of 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 Figure 1 and 7).

In summary, the proposed algorithm bears the following characteristics:

Ability to deal with challenging scenarios: As shown in Figure 1, 7 and 10, 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 fore/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.

Easy to implement: The same train-relabel 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 initial segmentation 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.

2. Key building blocks

2.1. Binary SVMs vs C-1SVMs

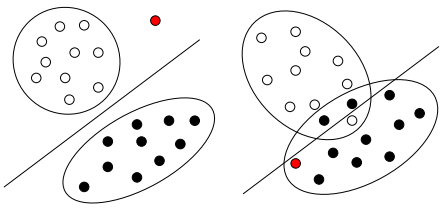
Intuitively, being a binary classification problem, foreground segmentation seems best solved by binary SVMs. However, we hypothesize that better performance can be achieved using two C-1SVMs, which learns fore/background distributions separately. The reasons are:

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. Furthermore, trying to train local binary SVMs at each pixel location is problematic as well since in most cases merely one of the two (fore/background) types of observations is locally available.

Moreover, even in areas that both fore/background examples are available, modeling the two sets separately using the C-1SVMs gives us 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 6.

2.2. Batch vs. online learning with reweighting

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.



(a) a separable case (b) an inseparable case

Figure 2. Comparison between a binary SVM and C-1SVMs. White circles and black dots represent the foreground and background training instances, respectively, while red dot denotes an unseen example. In scenario (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 the second case (b), binary SVM cannot confidently classify the test example since the margin is too small, whereas C-1SVMs is able to correctly label it as background.

Previous studies [3] have shown that a similar or even better generalization performance can be achieved using online learning with a *much less* computational cost, by showing all examples repetitively to an online learner, when comparing to that of batch learning. A less noticed but distinct advantage of online learning is that it produces a partially trained model immediately, which is then gradually refined toward the final solution. As will be discussed later, this turns out to be very helpful in this paper.

The online learner we use follows the one proposed by [7, 6]. Let $f_t(\cdot)$ be a score function of examples at time t and $k(\cdot,\cdot)$ be a kernel function, and denote α_t a non-negative weight of example of time t. 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_t arrives, the score function becomes $f_t(x_t) = \sum_{i=1}^{t-1} \alpha_i k(x_i, x_t)$, the update rule for weights is:

$$\begin{aligned} \alpha_t &= \text{clamp}\left(\frac{\gamma - (1-\tau) f_t(x_t)}{k(x_t, x_t)}, 0, (1-\tau)C\right), \\ \alpha_i &\leftarrow (1-\tau)\alpha_i \qquad \forall i = 1, \ldots, t-1, \end{aligned} \tag{1}$$

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

In what follows, we make a *notable* change to this online learning algorithm. Since it does not consider the situation where a given example is used repetitively during training, directly applying Eq.(1) adds multiple support vectors to the model, all come from the same sample but have different weights. In addition, once a support vector (x_t, α_t) is added to the 1SVM, over the time its weight α_t is only affected by the decay rate $(1-\tau)$. This leads to an explicitly *reweighting* scheme executed in our approach: If a training example x_t arrives and it turns out identical to an existing support vector (x_t, α_t) inside the model, this support vector is first taken out when computing the score function, it is then in-

cluded with its newly obtained weight, α'_t, to substitute the original one α_t.

$$f_t(x_t) = \sum_{i=1}^{t-1} \alpha_i \chi(x_i \neq x_t) k(x_i, x_t), \tag{2}$$

$$\alpha_t \leftarrow \alpha'_t = \text{clamp}\left(\frac{\gamma - f_t(x_t)}{k(x_t, x_t)}, 0, C\right), \tag{3}$$

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

Empirical simulations such as the ones reported in Figure 5 confirm that the performance of the learned model using this online learning with reweighting scheme is as competitive as that of batch learning.

2.3. 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 Ψ into N non-intersecting groups ψ_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. That is:

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

where $f^{\psi_i}(\cdot)$ is the score function trained using examples in subgroup ψ_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 5).

3. Our approach

The core of our approach is a train-relabel procedure: Two competing 1SVMs, $\mathcal{F}_p$ for foreground and $\mathcal{B}_p$ for background, are trained locally for each pixel p using pixels with known labels within the local window Ω_p. Once trained, $\mathcal{F}_p$ and $\mathcal{B}_p$ are used to jointly label p as either foreground, background, or unknown. Since the knowledge learned from neighboring pixels in Ω_p is used for labeling p, the above procedure effectively propagates known fore/background information to its neighborhood. As a result, based on only a few initial strokes, the algorithm can segment the whole image accordingly (see Figure 1(a-f)).

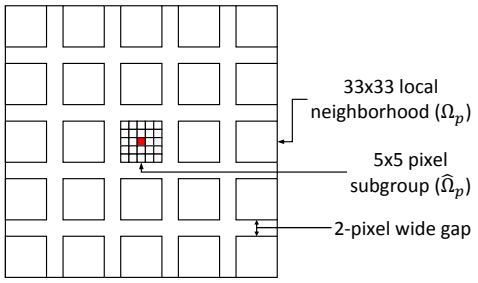


Figure 3. The neighborhood system used for 1SVM training. For the center pixel p shown in red, the pixels within the local 33×33 window is divided into 25 subgroups, with each subgroup having 25 pixels and a 2-pixel wide gap between adjacent subgroups.

The same train-relabel procedure is used 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 used to conduct the train-relabel process. Since $\mathcal{F}_p$ and $\mathcal{B}_p$ are trained using all pixels within Ω_p of frame t, if any of these pixels moves to p, $\mathcal{F}_p$ and $\mathcal{B}_p$ can properly classify it. Consequently, the algorithm can cope with arbitrary foreground and background movement without *a prior* motion information, as long as the amount of movement is less than the radius of Ω.

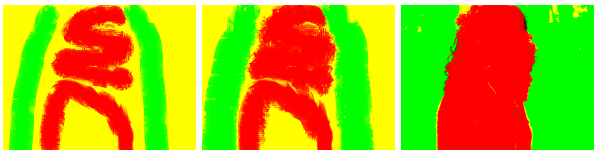
Under ideal situations, where the appearance distributions of fore/background pixels are locally separable, the above baseline procedure is sufficient. However, the two distributions may intersect due to fuzzy object boundary, motion blur, or low color contrast. To address these cases, global optimization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

3.1. Train local C-1SVMs at each pixel location

When training the two competing classifiers, $\mathcal{F}_p$ and $\mathcal{B}_p$, at each pixel p, the size of the local window Ω_p is an important parameter. It needs to be sufficiently small so that the local fore/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 Ω_p to 33×33 pixels, large enough to deal with motions of up to 16 pixels between adjacent frames.

Using a 33×33 window means 1089 examples are used for training each 1SVM. Considering training is performed for 1SVMs at all pixel locations, this is not affordable for real-time processing. To reduce the training cost, the techniques proposed in Sections 2.2 and 2.3 are applied here.

First, based on the max-pooling scheme of Section 2.3, the 1089 examples inside Ω_p are divided into 25 subgroups. To further take advantage of the spatial coherence among



(a) after 1 iteration (b) after 2 iterations (c) after convergence

Figure 4. The incurred losses calculated for frame 0 of the "walk" sequence 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.

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 endorses us to cut the computational costs of training from using 1089 examples to just 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 Ω_p, and we 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 2.2, online learning is employed to provide an on-demand feedback during the train-relabel process. As shown in Algorithm 1, instead of repetitively showing examples in $\hat{\Omega}_p$ to $\hat{\mathcal{F}}_p$ and $\hat{\mathcal{B}}_p$ and waiting for the training to converge, we only show these examples once in each train step. The partially trained 1SVMs are then immediately used to perform relabeling. This enable a faster propagation of labeling information through neighborhoods, hence empirically it allows a faster converge of the train-relabel process. For example, starting from the input user stokes shown in Figure 1(b), it takes about 40 iterations to propagates 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.

3.2. 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 fore/background only if the two competing classifiers output consistent predictions. Otherwise it is labeled as unknown.

As shown in Algorithm 2, 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 2.3, 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 have 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 for labeling p as either foreground or background (see Figure 4), respectively. p is thus 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.

Algorithm 1 Train C-1SVMs using frame I^t and label L^t

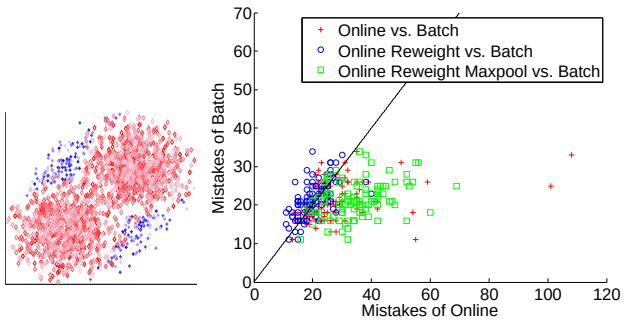
for each pixel p **do**
  for each pixel q in $\hat{\Omega}_p$ **do**
    if $L^t(q)$ equals foreground **then**
      Use $I^t(q)$ to train $\hat{\mathcal{F}}_p$ based on Equation (2) & (3);
    else if $L^t(q)$ equals background **then**
      Use $I^t(q)$ to train $\hat{\mathcal{B}}_p$ based on Equation (2) & (3);
    end if
  end for
end for

Algorithm 2 Relabel input frame I^t 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 $f_{\mathcal{F}}(p)$ and $f_{\mathcal{B}}(p)$ to 0;
  for each subgroup $\hat{\Omega}_q$ in Ω_p **do**
    Set $w = (1 - \tau_{spatial})^{\|p-q\|}$;
    Set $f_{\mathcal{F}}(p) = \max\left(f_{\mathcal{F}}(p), w\, f_{\hat{\mathcal{F}}}^{\hat{\Omega}_q}\left(I^t(p)\right)\right)$;
    Set $f_{\mathcal{B}}(p) = \max\left(f_{\mathcal{B}}(p), w\, f_{\hat{\mathcal{B}}}^{\hat{\Omega}_q}\left(I^t(p)\right)\right)$;
  end for
  Set foreground loss $l_{\mathcal{F}}(p) = \max\left(0, \gamma - f_{\mathcal{F}}(p)\right)$;
  Set background loss $l_{\mathcal{B}}(p) = \max\left(0, \gamma - f_{\mathcal{B}}(p)\right)$;
  if $\left(l_{\mathcal{F}}(p) < T_{\mathcal{F}}^{low}\right)$ && $\left(l_{\mathcal{B}}(p) > T_{\mathcal{B}}^{high}\right)$ **then**
    Set $L^t(p)$ to foreground;
  else if $\left(l_{\mathcal{B}}(p) < T_{\mathcal{B}}^{high}\right)$ && $\left(l_{\mathcal{F}}(p) > T_{\mathcal{F}}^{low}\right)$ **then**
    Set $L^t(p)$ to background;
  else
    Set $L^t(p)$ to unknown;
  end if
end for

3.3. Apply global optimization

While explicitly labeling ambiguous pixels as unknown helps to prevent error propagation during the train-relabel process, decisions still need to be made for these pixels in the final segmentation. In addition, the pixel-wise prediction result based on previous sections often tends to be noisy (see Figure 1(e)). To obtain a clean binary segmentation, our final step is to compute the global optimal solution G^t for a Markov random field based energy function, which consists of a data term and a contrast term. The



(a) synthetic data (b) mistakes over 100 trials

Figure 5. 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).

data term, i.e, the costs of assigning pixel p to foreground or background, is set using $l_{\mathcal{F}}(p)$ or $l_{\mathcal{B}}(p)$ directly. The contrast term is the same as the ones used in previous approaches [9, 15], which adaptively penalizes segmentation boundaries based on local color differences.

It is well known that G^t can be calculated efficiently using graph cuts. We implement a version of the push-relabel algorithm to compute the min cuts. In practice we limit the number of push-relabel steps to 20, which is found sufficient throghout our experiments.

3.4. Deal with incoming frames

When a new frame $t+1$ arrives, the following preparation procedures are performed before the train-relabel procedure: First, G^t is used to train the C-1SVMs 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. Nevertheless, pixels along the fore/background boundary often have mixed colors. Using these pixels for training may introduce bad examples to $\mathcal{F}$ and $\mathcal{B}$, which in turn may lead to incorrect labeling. This problem is circumvented in practice by applying a 2-pixel morphological erosion to both the foreground and the background regions in G^t to remove ambiguities. Next, To facilitate the adaptation of C-1SVMs to temporal changes in the fore/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 $(1-\tau_{temporal})$.

4. Experiments

To exploit the inherit parallel structure of the proposed algorithm, we implement it on GPU using DirectCompute,

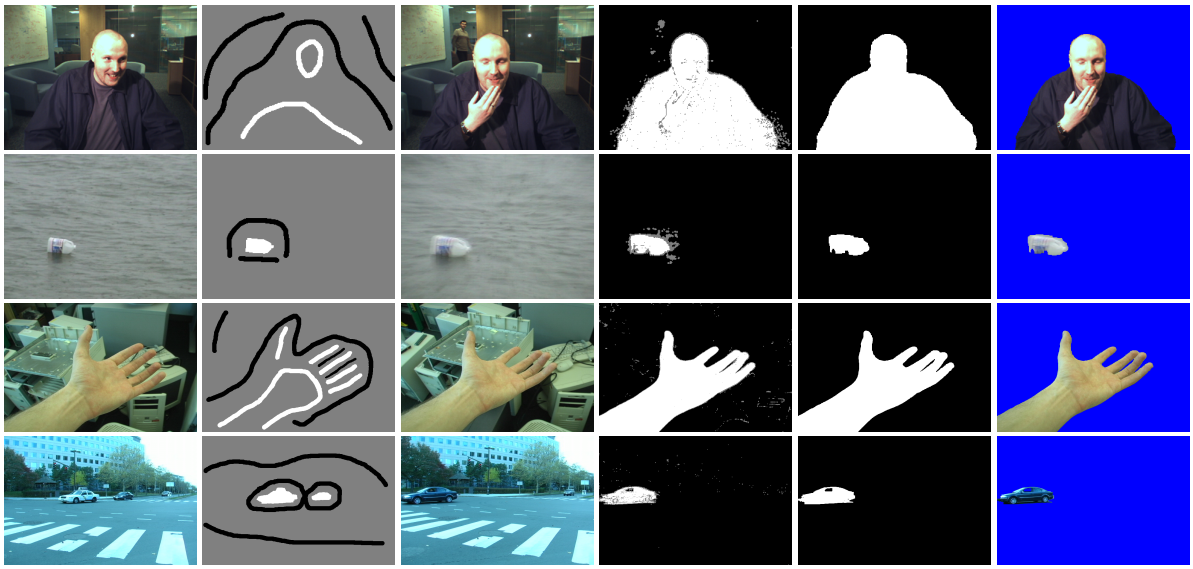
(a) train w/ stroke (b) labeled trimap (c) train w/ trimap

Figure 6. 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 manfully labeled trimap (b) only marginally improves the result (c).

which is proposed by Microsoft as an alternative to CUDA and is included in the Direct3D11 API. For VGA-sized videos, our current implementation runs at 14 FPS on a Lenovo ThinkStation S20 with nVidia GeForce GTX 480 GPU. Except for the cases presented in Figure 10 that we will explain later, the same set of parameter values are used throughout the experiments: $C = 0.5$, $\tau_{temporal} = 0.25$, $\tau_{spatial} = 0.05$, $T_{\mathcal{F}}^{low} = 0.1$, $T_{\mathcal{F}}^{high} = 0.3$, $T_{\mathcal{B}}^{low} = T_{\mathcal{B}}^{high} = 0.4$. Notice that we set the background labeling requirements ($l_{\mathcal{B}} < T_{\mathcal{B}}^{low}$ 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.

Comparison of batch and 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 5(a), 2000 two-dimensional instances are sampled under a predefined 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 5(b) shows that the online learning method [7] using Eq.(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 Eq.(3) is shown to perform as well as, or slightly better than, the batch learner. The performance drops when both reweight-



(a) first frame (b) input user strokes (c) test frame (d) converged label (e) final segmentation (f) foreground mask

Figure 7. Results on testbed sequences referred to as (from top to bottom) "talk", "jug", "hand", and "car". The results clearly demonstrate the capacity of our algorithm to deal with different challenges, such as background changes (in "talk"), repetitive background motion (in "jug"), camera motion (in "hand" & "car"), strong motion blur (caused by camera zooming in "jug"), non-rigid foreground deformations (in "talk" & "hand"), topology changes (holes in "hand" & "car"), and low fore/background color contrast (in "talk" & "car").

ing and max-pooling is applied, but is still comparable to the conventional online learning approach.

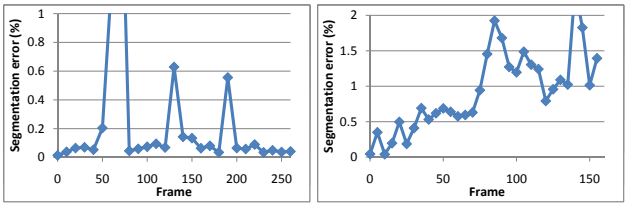
Comparison with binary SVM: To verify our previous claim that a global binary SVM does not work well for challenging sequences such as "walk" due to overlapping fore/background distributions, a comparison is carried out and shown in Figure 6. The results empirically confirm that global binary SVM cannot properly segment the image even when having access to much richer labeling information during training.

Results on testbed videos: As displayed in Figure 1 and 7, our algorithm is tested over a variety of video scenarios used by previous papers [18, 5, 13, 11, 9, 17, 1]. The segmentation results are visually satisfactory and are comparable to the state-of-the-art approaches that are designed specifically for video conferencing [11, 9, 17], background subtraction [18, 5], or for handling freely moving camera [13]. Due to the space limit, 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 [11, 9, 17], 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 8) show that the median segmentation errors are 0.07% and 0.88% for the two sequences, respectively. In comparison, [17] reports higher median errors of 0.27% and 2.56% for the same sequences.

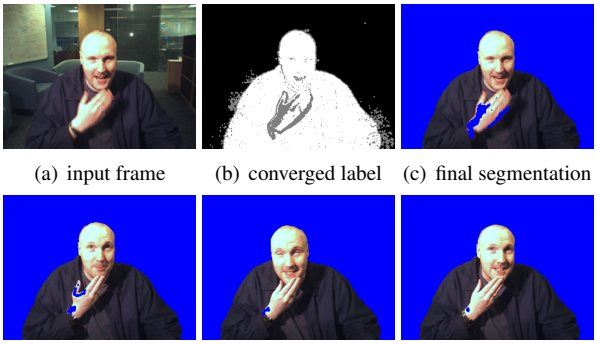
Ability to recover from incorrect predictions: While the aforementioned tendency of accepting unseen examples as background helps to correctly handle background changes, such as the background person in "talk" and new background scene in "walk", it occasionally introduces errors if locally unseen foreground appearances are presented (see *e.g.* Figure 9). Nevertheless, thank to the redundancy of using two competing 1SVMs, the incorrect labels do not affect the training of foreground 1SVMs, which gradually recognize the novel foreground colors. As a result, the mistakes are corrected after a few consecutive frames without any user intervention.

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. 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 10, 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 [18, 5].



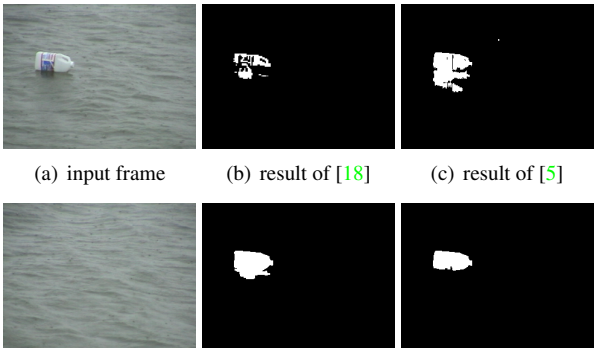
(a) sequence "JM" used in [9, 17] (b) sequence "IU" used in [11, 17]

Figure 8. Segmentation accuracy for two sequences, where a ground truth segmentation is available for every 5 or 10 frames. The errors of tree-based approach in (a) are based on our readings from the Figure 12(a) in [17].



(a) input frame (b) converged label (c) final segmentation

Figure 9. A failure case. When the hand first shows up in (a), both local foreground and background 1SVMs classify pixels with unseen colors as outliers, resulting unknown (grey) labels (b). The final segmentation (c) labels these pixels incorrectly due to the bias toward background. However, the algorithm corrects the mistakes in 10 frames without any user intervention (2nd row).



(a) input frame (b) result of [18] (c) result of [5]

(d) pure background (e) ours w/o user label (f) ours w/ user label

Figure 10. Comparisons to background subtraction algorithms on the "jug" sequence. The results of existing algorithms (b & c) are reported in their respective papers. Our algorithm labels the reflection as foreground (e) when initialized using a single background image (d), and labels the reflection as background (f) when initialized using user strokes shown in Figure 7 (b).

5. 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 object boundaries. Experiments on standard testbed videos demonstrate that our algorithm possesses comparable or superior performance comparing to the state-of-the-art approaches designed for specific applications [11, 9, 17, 18, 5, 13].

Possible future research directions include incorporating other visual cues such as texture and shape into the proposed algorithm, as well as integrating the algorithm with real-time video matting approaches.

Acknowledgement

The authors would like to thank financial support from NSERC.

References

[1] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 82(2):113–132, 2009. 2, 7

[2] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *SIGGRAPH*, 2009. 2

[3] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2008. 3

[4] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. 6

[5] L. Cheng and M. Gong. Realtime background subtraction from dynamic scenes. In *ICCV*, 2009. 2, 7, 8

[6] L. Cheng, M. Gong, D. Schuurmans, and T. Caelli. Real-time discriminative background subtraction. In *TIP*, 2011. 3

[7] L. Cheng, S. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. implicit online learning with kernels. In *NIPS*, 2007. 3, 6

[8] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. In *Siggraph*, pages 243–248, 2002. 1, 2

[9] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *CVPR*, 2006. 2, 6, 7, 8

[10] E. Hayman and J. Eklundh. Statistical background subtraction for a mobile observer. In *ICCV*, 2003. 2

[11] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bilayer segmentation of binocular stereo video. In *CVPR*, 2005. 2, 7, 8

[12] Y. Li, J. Sun, and H. Shum. Video object cut and paste. *SIGGRAPH*, 2005. 2

[13] Y. Sheikh, O. Javed, and T. Kanade. Background subtraction for freely moving cameras. In *ICCV*, 2009. 2, 7, 8

[14] Y. Sheikh and M. Shah. Bayesian object detection in dynamic scenes. In *CVPR*, 2005. 2

[15] J. Sun, W. Zhang, X. Tang, and H. Shum. Background cut. In *ECCV*, 2006. 2, 6

[16] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. *SIGGRAPH*, 2005. 2

[17] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *CVPR*, 2007. 2, 7, 8

[18] J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust Kalman filter. In *ICCV*, 2003. 2, 7, 8