Discontinuity-Aware Video Object Cutout

Fan Zhong[†] Xueying Qin[†*] Qunsheng Peng[‡] Xiangxu Meng[†]

[†]*School of Computer Science and Technology, Shandong University*
[†]*Shandong Provincial Key Laboratory of Software Engineering*
[‡]*State Key Lab. of CAD&CG, Zhejiang University*

Abstract

Existing video object cutout systems can only deal with limited cases. They usually require detailed user interactions to segment real-life videos, which often suffer from both inseparable statistics (similar appearance between foreground and background) and temporal discontinuities (e.g. large movements, newly-exposed regions following disocclusion or topology change).

In this paper, we present an efficient video cutout system to meet this challenge. A novel directional classifier is proposed to handle temporal discontinuities robustly, and then multiple classifiers are incorporated to cover a variety of cases. The outputs of these classifiers are integrated via another classifier, which is learnt from real examples. The foreground matte is solved by a coherent matting procedure, and remaining errors can be removed easily by additive spatio-temporal local editing. Experiments demonstrate that our system performs more robustly and more intelligently than existing systems in dealing with various input types, thus saving a lot of user labor and time.

CR Categories: I.4.6 [Computer Graphics]: Image Processing and Computer Vision—Segmentation - Pixel classification;

Keywords: video segmentation, object cutout, pixel classification

Links: DL PDF WEB VIDEO

1 Introduction

Extraction of dynamic video object is normally a labor-intensive and time-consuming task. Although great successes have been achieved over the past decade [Chuang et al. 2002; Agarwala et al. 2004; Li et al. 2005; Bai et al. 2009], existing methods for video object cutout are not yet efficient enough for real-life videos, and numerous user interactions are required to remove the errors caused by inseparable statistics and temporal discontinuities. Inseparable statistics are notoriously problematic in the fields of image and video segmentation. Temporal discontinuities can be caused by sudden occlusion/disocclusion, topology changes and fast motion, as demonstrated in Fig. 1. Discontinuous regions are difficult to segment correctly due to the lack of reliable temporal constraints.

Early video cutout methods based on global classifiers [Li et al. 2005; Wang et al. 2005] have proved to be sensitive to inseparable statistics. To overcome this problem, Bai et al. [2009] proposes local classifiers. The resulting system is known as Video Snapcut, and has been successfully incorporated into Adobe After Effect CS5 as the Rotobrush tool. However, as shown in Fig. 1, when compared with global classifiers, local classifiers are more sensitive to temporal discontinuities because of their limited coverage. In fact, the problem is not restricted to the local classifier, all methods relying on local temporal continuity suffer from the same problem, including the 3D graph-cut [Li et al. 2005; Tong et al. 2011] and the 3D extension [Tang et al. 2011] of the color line model [Levin et al. 2008] in video cube.

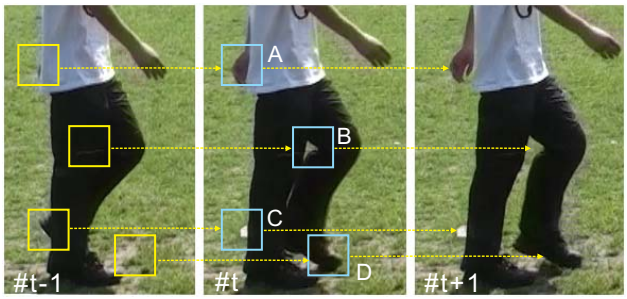


Figure 1: *Temporal discontinuities in video. Local classifiers cannot correctly segment the newly-exposed regions in frame t by sampling only in the corresponding windows in frame $t-1$, and the errors then would be magnified in frame $t+1$.*

Some efforts have been made to address the problem of temporal discontinuity. Bai et al. [2010] proposes a multi-size classifier with a size which is adaptively adjusted based on the local registration error. However, in newly-exposed regions such as regions A and B in Fig. 1, the proper window size is obviously independent of local registration errors, and is thus difficult to determine. In [Zhong et al. 2010], local and global classifiers are combined in order to deal with occlusion. However, because global classifier is sensitive to inseparable statistics, this method cannot achieve robust segmentation for videos of complex scenes.

Our Approach: We present a video cutout system which is capable of handling all common cases well; our system produces much fewer errors than existing systems, and therefore can greatly reduce the number of user interactions. We achieve this mainly by a combination of three approaches:

• *A medium-scale classifier to robustly handle temporal discontinuity.* We propose the unbiased directional classifiers (UDCs). Unlike local classifiers in previous methods, whose support windows are always square or circular [Bai et al. 2009; Zhong et al. 2010; Bai et al. 2010], UDCs reside in a set of long-narrow and directional support windows, which enables them to explore long-distance region similarities while maintaining relatively small coverage, making them robust against both inseparable statistics and temporal discontinuities. By integrating the classifiers in different directions, large image movements in any direction can be captured, and spatial context information can also be explored more effectively.

*Email: qxy@sdu.edu.cn

ACM Reference Format
Zhong, F., Qin, X., Peng, Q., Meng, X. 2012. Discontinuity-Aware Video Object Cutout. *ACM Trans. Graph. 31* 6, Article 175 (November 2012), 10 pages. DOI = 10.1145/2366145.2366194 http://doi.acm.org/10.1145/2366145.2366194.

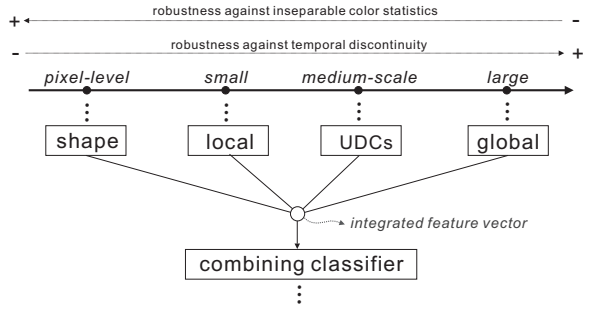


Figure 2: *Classifiers used in our system.*

• *A learning-based approach to combine multiple classifiers for pixel classification*, which can robustly deal with complicated cases. As illustrated in Fig. 2, our system involves four types of base classifier: the shape prior [Bai et al. 2009], a set of local classifiers and UDCs, and a global classifier. From left to right, the classifiers become increasingly discontinuity-aware, but also increasingly sensitive to inseparable statistics. Each type of classifier is suitable for handling specific cases (see Section 1.1 for details), and by integrating them together, our system covers all common cases.

The idea of using multiple classifiers is not new. However, previous methods usually adopted a weighted-sum approach to combine the classifiers [Bai et al. 2009; Price et al. 2009; Yin et al. 2007], which is difficult for complex cases, and often requires hard-to-tune parameters. To address this problem, we propose a novel learning-based approach to combine the classifiers. As illustrated in Fig. 2, the outputs of the base classifiers are integrated into a new feature vector, which is then used as the input for a combining classifier to produce final predictions. The combining classifier is pre-trained from a set of segmented training videos, so solutions to complex cases can be learnt from real examples without manual setting.

• *An efficient approach for progressive refinement of results.* We propose *joint bidirectional propagation* (JBP) and *additive propagation* (AP). JBP can adaptively integrate the results of forward and backward propagations to remove segmentation errors. It is very effective because the errors produced by forward and backward propagations often appear in different regions due to the opposite direction of the image motion (leading to different newly-exposed regions). AP allows the user to progressively refine the result in multiple passes without worrying about undesirable overwriting. This is very important for attaining high-quality results.

A comprehensive review of the related literature can be found in [Bai et al. 2009]. The rest of this paper is organized as follows. After we provide an overview of the proposed system, we introduce UDCs in Section 2. Other classifiers and the learning-based combination approach are described in Section 3. The matte solving method is then presented in Section 3.4. The progressive refinement methods are discussed in Section 4, and are followed by the experimental results and conclusions.

1.1 System Overview

Our system adopts the propagation workflow proposed in [Bai et al. 2009]. The user first manually segments a frame with an efficient image cutout tool; the system then propagates the result through the whole video, frame-by-frame, under the user's supervision. Both forward and backward propagations are supported, so that the user can start to edit spatio-temporal regions from any frame.

In this paper, unless stated otherwise, we consider only forward propagation, backward propagation can be implemented in the same way. The task of forward propagation is to segment frame t (the target frame) based on the segmentation result of frame $t-1$ (the source frame). Let I_{t-1}, I_t denote the two frame images, and M_{t-1}, M_t denote the foreground mattes. To compensate for large image motion, we first register I_{t-1} to I_t using optical flow, and M_{t-1} is also warped in the same way. The resulting image and matte are denoted by I'_{t-1} and M'_{t-1}, respectively.

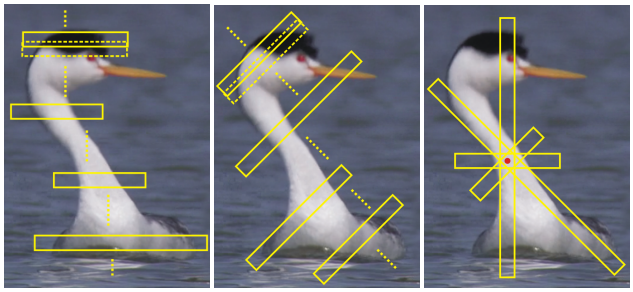


Figure 3: *Illustration of UDCs. The left and the center images show UDCs in the directions of* 0° *and* 45°*, respectively. The right image shows the 4-directional UDCs at a single pixel* $(0^\circ, 45^\circ, 90^\circ, 135^\circ)$*.*

Given these inputs, the classification process illustrated in Fig. 2 is applied to I_t, assigning a probability and a confidence value to each pixel. The roles of the classifiers in our system are distinct from each other, and can be summarized as follows:

- ◇ **local classifier**: to handle continuous regions; is insensitive to inseparable statistics and small registration error.
- ◇ **shape prior**: to handle continuous regions of local inseparable statistics; is sensitive to registration error.
- ◇ **UDCs**: to handle discontinuous regions; is insensitive to inseparable statistics and registration errors.
- ◇ **global classifier**: to emphasize the user intention, and can be customized by the user interactively (see Section 3.2).

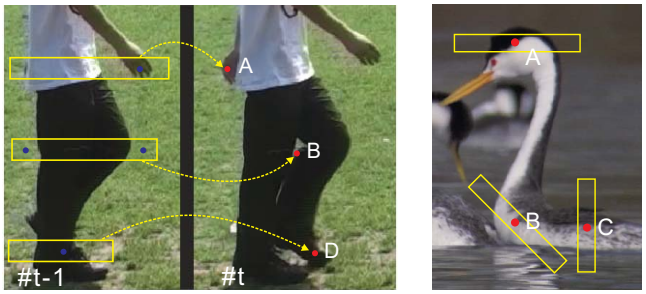
These classifiers are integrated for pixel classification, based on which the foreground matte is solved, and the resulting errors can be removed by spatio-temporal local editing.

2 Unbiased Directional Classifiers

In this section we will introduce Unbiased Directional Classifiers (UDCs), other classifiers as well as the combining approach will be presented in the next section.

2.1 Long-Narrow Directional Support Windows

A global classifier accounts for the whole image or video, while a local classifier accounts for only a small local window. Unlike either of them, UDCs reside in a set of medium-scale support windows, as illustrated in Fig. 3, which have the following characteristics: 1) *Long-Narrow shape*, each window covers parts of both the foreground and the background so that the sampling process would not be biased towards either of them; for this purpose, the window size is fixed in one dimension but varies in the other dimension, resulting in long narrow windows across the foreground object. 2) *Directionality*, the windows can be set in different directions, and windows in the same direction are parallel. In our system, we adopt the 4-directional UDCs illustrated in Fig. 3(c). Using more directions would be helpful in improving the accuracy, but in practice, we find that this is not necessary, especially when considering the



(a) (b)

Figure 4: *The classification ability of UDCs. (a) Handling temporal discontinuities by exploring long-distance region similarities. (b) Handling inseparable color statistics, where A, B, and C can be best separated from the background by UDCs in the directions of* $0^\circ, 135^\circ, 90^\circ$*, respectively.*

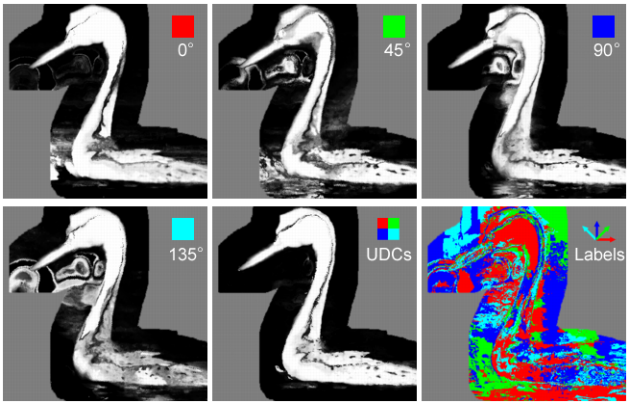


Figure 5: *The results of UDCs, where the input image is the same as in Fig. 4(b). The grayed regions are the pixels that are not covered by the classifiers (background). The combined result (bottom-center) contains much fewer errors than that in each direction. The bottom-right panel shows the selected direction at each pixel.*

computational cost. 3) *Overlapping*, adjacent windows in the same direction are overlapped for better spatial coherence; specifically, one third of each window overlaps with its neighbor at each side. 4) *Full coverage*, the windows in each direction can fully cover the foreground object, and each pixel can be covered by multiple windows in different directions.

In each window, a classifier is established and applied to the pixels inside. Because each pixel is covered by multiple windows and thus by multiple classifiers, we select the one with the highest confidence to produce the final predictions (Section 2.2). In comparison with local and global classifiers, UDCs provide a better compromise between robustness against inseparable statistics and temporal discontinuities:

• **Handling temporal discontinuities:** As demonstrated in Fig. 4(a), each UDC can explore regional similarities in a long narrow image window, and therefore can cope with discontinuous cases, as in regions A, B, and D in Fig. 1. The unbiased and fully-covered properties give every pixel an equal chance to change its state, which thus allows arbitrary foreground topology change. On the other hand, because each UDC is insensitive to object motion in its direction, and each pixel is covered by multiple UDCs in different directions, image motion in any direction can thus be captured by the UDCs in at least one direction.

• **Handling inseparable statistics:** UDCs are also insensitive to inseparable statistics. The scale of UDCs is much smaller than that of the global classifier, and meanwhile, is larger than that of the local classifier in only one dimension. In addition, by combining the classifiers in different directions, UDCs can make better use of spatial context information, which also helps to solve the ambiguities of color statistics, as shown in Fig. 4(b).

The long narrow shape of UDCs is essential for dealing with cases of discontinuity while maintaining relatively small coverage. Previous local classifiers were either square [Bai et al. 2009; Zhong et al. 2010] or circular [Bai et al. 2010], so large window is necessary in order to discover large-scale correspondence of image regions, this would definitely ruin the locality of classifiers and make them sensitive to inseparable statistics.

2.2 Classification of Pixels using UDCs

Like most of the previous methods, we adopted Gaussian mixture models (GMMs) to model the color distribution in each UDC window. GMMs can be trained efficiently and have been shown to be insensitive to noise and slight lighting changes. Each UDC consists of a pair of GMMs: one for the foreground, and the other for the background. Let $\hat{p}(x \mid F)$ and $\hat{p}(x \mid B)$ denote the probability densities output by the foreground and the background GMMs, respectively. Here, we omit the index of UDCs for conciseness. The probability of a pixel belonging to the foreground should then be:

$$\hat{p}(x) = \hat{p}(x \mid F)/(\hat{p}(x \mid F) + \hat{p}(x \mid B)) \tag{1}$$

Because each pixel is covered by multiple classifiers, the key to the integration of these classifiers is the evaluation of their confidence. In [Bai et al. 2009], the confidence of each local classifier is uniform for all pixels, while in practice, the accuracy of a GMM varies considerably over different clusters of colors. It would therefore be better to assign an individual confidence to each pixel. Here, we propose to use the following confidence function:

$$\hat{f}(x) = \frac{|\hat{p}(x \mid F) - \hat{p}(x \mid B)|}{\hat{p}(x \mid F) + \hat{p}(x \mid B) + \epsilon} \cdot q(x) \tag{2}$$

in which the first term can penalize both inseparable color statistics and newly-exposed regions. It would be small in two cases: 1) when both $\hat{p}(x \mid F)$ and $\hat{p}(x \mid B)$ are not small (in which case their difference would be small compared with their sum), which means that x has high density in both the foreground and the background distributions, and is thus inseparable; 2) when both $\hat{p}(x \mid F)$ and $\hat{p}(x \mid B)$ are very small in comparison with ϵ (corresponding to the case of newly-exposed regions). Because GMM attenuates rapidly when the distance to the centers becomes large, ϵ is easy to determine (1e-3 in our implementation).

The second term $q(x)$ encodes the priors with regard to the accuracy of the GMMs, which can be learnt from the training process. The GMMs can be expressed as $\hat{p}(x \mid l) = \sum_{k=1}^{K_l} \beta_k \, G(x \mid \Theta_k, l)$, in which $l \in \{F, B\}$ is the label of the foreground and the background, K_l is the number of Gaussian components, and Θ_k is the parameter set of the k-th Gaussian component. The contribution of each sample s to the k-th component should be:

$$w^k(s \mid l) = \frac{\beta_k G(s \mid \Theta_k, l)}{\hat{p}(s \mid l)} \tag{3}$$

which is also the weight function used in the training process [Bilmes 1998]. Now, we can evaluate the accuracy of each

component using the labeled training data sets S_F and S_B:

$$\begin{aligned} e_l^k &= \textstyle\sum_{s\in S_l} w^k(s\,|\,l)|L(s)-\hat{p}(s)| \\ c_l^k &= \textstyle\sum_{s\in S_l} w^k(s\,|\,l)(1-|L(s)-\hat{p}(s)|) \end{aligned} \tag{4}$$

where $L(s)$ is the label of s (1 if foreground, 0 if background), so e_l^k and c_l^k actually measure the quantities of the pixels that are misclassified and correctly classified, respectively. The accuracy of the k-th component should then be $q_l^k = c_l^k/(c_l^k + e_l^k)$, and the prior confidence of the GMM with respect to an unknown color x is:

$$q(x\,|\,l) = \sum_{k=1}^{K_l} w^k(x\,|\,l)\, q_l^k \tag{5}$$

subsequently:

$$q(x) = \hat{p}(x)q(x\,|\,F) + (1-\hat{p}(x))q(x\,|\,B) \tag{6}$$

which distributes $q(x)$ to the foreground and the background GMM components based on the correlation of x to them.

The above method can reliably estimate the confidence of the classifiers at each pixel. As a result, the multiple classifiers covering each pixel can be integrated simply by selecting the classifier with maximum confidence. We use $f_u(x)$ and $p_u(x)$ to denote the outputs of the selected classifier for a pixel x. Fig. 5 shows how our approach correctly incorporates the results of UDCs in different directions.

Newly-exposed regions such as region C in Fig. 1 cannot be correctly segmented in the above way, because they are completely occluded in the source frame. Discriminative classification of these regions requires to reconstruct the background image [Bai et al. 2010; Sarim et al. 2009], which is difficult for complex scenes with a dynamic background. In fact, because most regions of this type are caused by foreground-background occlusion, and the foreground color distribution is usually stable, we can deal with these cases in a generative way without knowing the background appearance model exactly, as in many object tracking methods [Ross et al. 2008]. In our system, this is performed automatically by the combining classifier based on a prior model learnt from training examples (see Section 3.3 for details).

3 Segmentation Using Multiple Classifiers

As shown in Fig. 2, apart from UDCs, our system uses three other types of base classifier: local classifier, global classifier and shape prior. The final predictions are obtained by integrating the results of all base classifiers via the combining classifier.

3.1 Local Classifier

Several methods were proposed for implementing local classifiers, including the overlapped GMM windows used in [Bai et al. 2009] and [Bai et al. 2010]. In [Zhong et al. 2010], local kernel density estimation (LKDE) is proposed for local propagation. LKDE can achieve real-time performance, but we found that in comparison with the overlapped GMM windows, LKDE is more sensitive to object appearance change caused by shading variations and illumination changes. A common disadvantage of both methods is that they are both based on the foreground and background color densities, and therefore may overlook small regions, which always leads to low densities in the color space.

We adopt local k-NN (LKNN) as classifiers for better segmentation of small regions. For each pixel x to classify, we search for its k nearest neighbors (in the color space) from the local window in I'_{t-1}. The local window is centered at x and has a size of W. Let $s_1, s_2, \cdots, s_k$ denote the k samples, and then the prediction is computed as:

$$\begin{aligned} p_l(x) &= \frac{\sum_i^k w_i l_i}{\sum_i^k w_i} \quad \text{with} \quad w_i = \exp(-\frac{\parallel s_i - x \parallel^2}{\sigma_l^2}) \\ f_l(x) &= \frac{\sum_i^k w_i}{k}\,(1 - 4\,\mathrm{var}(l_i)) \end{aligned} \tag{7}$$

where l_i is the label of s_i (1 for foreground, 0 for background). $\mathrm{var}(l_i)$ is the variance of the labels, which penalizes the case when equal amounts of foreground and background samples are found (inseparable statistics). It is easy to verify that $0 \le 1 - 4\,\mathrm{var}(l_i) \le 1$. The first term of $f_l(x)$ penalizes the case of very few close neighbors, which is usually due to temporal discontinuities.

The above method is simple, but effective, and is insensitive to the setting of parameters (in our implementation $k = 9$, $W = 21$, $\sigma_l = 20$). Because only a few samples are necessary to identify a region, LKNN would not suffer from the same problems as GMM and LKDE. Also, when compared with LKDE, it is more robust against temporal changes of object appearance.

3.2 Global Classifier and Shape Prior

Global classifier is mainly used to emphasize user intention in our system. If a region is found to be error-prone, the user can add its statistics to the global classifier to prevent it from being missegmented again. For this purpose, our global classifier consists of two sets of samples: one contains the pixels of a segmented frame, which can be re-specified at any time by the user with a simple command; the other consists of pixels manually marked using scribble brushes, which can be specifically selected pixels, or seed pixels added to remove segmentation errors (an option is provided for this purpose). The manually selected samples are stored with their positions on the time axis, and for each frame, only the samples of the nearest 3 frames that contain selected samples are used. The probability and the confidence are evaluated using the method proposed in Section 2.2, and are denoted as $p_g(x)$ and $f_g(x)$, respectively, in the following.

As in [Bai et al. 2009], we also use the warped matte of the source frame (M'_{t-1}) to represent the shape prior $p_s(x)$. The shape confidence is computed as:

$$f_s(x) = 1 - \exp\left(-d_s^2(x)/\sigma_s^2\right) \tag{8}$$

in which $d_s(x)$ is the distance to the object boundary computed using distance transform; σ_s is set to be 5 in our implementation. The shape prior computed in this way can be regarded as a pixel-level classifier based on dense optical flow. Because optical flow is computed using only low-level image features, the shape prior is different from other classifiers in that it does not use color statistics. This is why it is more robust to inseparable color statistics, but meanwhile is more sensitive to registration errors.

3.3 Learning-Based Combination of Classifiers

To combine multiple classifiers, previous methods usually adopted a weighted-sum approach to produce the final prediction, with weights that are adaptive to the confidences of the classifiers. The problem with this approach is that, the confidences of different classifiers are not strictly comparable because they are measured in different ways without an uniform criterion, and as a result, less-confident classifiers may take greater effect.

Figure 6: *Training video data set used in our experiments. In the parentheses is the total number of frames and videos in each subset.*

Inspired by the idea of meta-learning [Vilalta and Drissi 2002], we propose a learning-based approach to integrate the base classifiers involved. As illustrated in Fig. 2, the integration is regarded as another classification problem, $v(x) \rightarrow \{0, 1\}$, with the feature vector $v(x)$ to be:

$$v(x) = (p_u, f_u, p_l, f_l, p_g, f_g, p_s, f_s, e)(x) \tag{9}$$

which is a 9-dimensional vector containing the outputs of the base classifiers and the normalized registration error $e(x)$. $e(x)$ is introduced to better distinguish the usages of base classifiers, it is computed as: $e(x) = 1 - \exp(-\dot{e}^2(x)/\sigma_e^2)$, in which $\dot{e}(x)$ is the average registration error in the 7×7 window centered at each pixel, and where σ_e is set to be 20.

To train the combining classifier, we first carefully segment a set of videos, which are then used to generate the training data set $\mathcal{D} = \{(v_i, l_i)\}$, where v_i is an instance of $v(x)$ and $l_i \in \{0, 1\}$ is the label. This needs only perform the segmentation process between adjacent frames, and then use the outputs of the base classifiers for each pixel as v_i, with the known label of each pixel as l_i. To reduce the redundancy of $\mathcal{D}$, only the pixels around the object boundary (± 30 pixels) are used for training. Fig. 6 shows some examples of the training videos. Currently our data set contains 21 videos, including videos of humans, animals, cartoons, and static objects. CARTOON and STATIC are introduced mainly for evaluation (see Section 5.1). STATIC contains very few temporal discontinuities, and thus can serve as a bad training data set.

We select *kernel density estimation* as the combining classifier, so the task of learning is to get foreground and background probability densities over the space of $v(x)$. Because the data set $\mathcal{D}$ is very large, without an effective method, neither the training nor the testing can be performed efficiently. Rather than learning the complete density function, we learn it only at certain lattice points in the feature space, for which purpose we first reduce the feature vector to lower dimensions. Because only binary classification is necessary, the probability and the confidence values can be merged as:

$$r_*(x) = 0.5 + f_*(x)(p_*(x) - 0.5) \tag{10}$$

in which $* \in \{u, l, g, s\}$ denotes the different classifiers. $r_*(x)$ is the merged output, with *sign*$(r_*(x) - 0.5)$ representing the label and $|r_*(x) - 0.5|$ encoding the confidence. The feature vector then becomes:

$$\hat{v}(x) = (r_u, r_l, r_g, r_s, e)(x) \tag{11}$$

with each component within a range of $[0, 1]$. The space of $\hat{v}(x)$ is then uniformly subdivided into 20^5 grids, and each grid is represented by a lattice centered at it, resulting in a look-up table with 3.2M elements.

Given the training data set $\mathcal{D}$, the foreground density d_k^F and the background density d_k^B at the k-th lattice should be:

$$d_k^l = \sum_{i=1}^{|\mathcal{D}|} [l = l_i] \exp(-\frac{\| c_k - \hat{v}_i \|^2}{\sigma_d^2}) \qquad l \in \{F, B\} \tag{12}$$

in which c_k is the coordinate of the k-th lattice; σ_d is set to be 0.1 in our implementation. It is not necessary to scan the data set $\mathcal{D}$ for each lattice; alternatively, we add each $\hat{v}_i$ to the lattices near it (far lattices with very small weights are ignored). In this way, $\mathcal{D}$ needs to be scanned only once, and thus is not necessary to be stored. The training process therefore can be accomplished simultaneously with segmentation propagations. Note that the quantity of $\hat{v}_i$ is proportional to the number of pixels. This is why we did not adopt more sophisticated classifiers such as a support vector machine (SVM), which may need a long time and large storage capacity to be trained.

Given the look-up table, it is very easy to combine the base classifiers. For each pixel x, $\hat{v}(x)$ is first generated based on the outputs of the base classifiers, and then the foreground density $d^F(x)$ and the background density $d^B(x)$ are retrieved from the nearest lattice. Finally the ensemble prediction can be computed as:

$$p(x) = \frac{d^F(x)}{d^F(x) + d^B(x)} \tag{13}$$

$$f(x) = \frac{|d^F(x) - d^B(x)|}{d^F(x) + d^B(x) + \tilde{\epsilon}} \tag{14}$$

The principle to evaluate the confidence is the same as that for the first term of Equation (2), which penalizes the cases where the foreground and background densities are both very high or both very low. $\tilde{\epsilon}$ is chosen to be 5 in our implementation.

The above method does not directly account for the confidences of the base classifiers, scaling and shifting the confidence functions has little effect on the ensemble predictions. $v(x)$ is different for different input cases; for example, in newly-exposed regions, e(x) should be large, while in well-registered regions, f_l(x) should be high. By using $v(x)$ as the feature, each case is assigned a unique classification function learnt from real examples. In this way, complicated cases can be handled easily. For example, for the case of region C in Fig. 1, because most such regions are backgrounds in the training data, our system tends to classify such regions as background. Informally, a pixel with a large temporal discontinuity ($e(x)$ is large), and that is not recognized by both UDCs and local classifiers ($f_u(x)$ and $f_l(x)$ are small), and at the same time is not predicted as foreground by global classifier ($p_g(x) < 0.5$ or $f_g(x)$ is small), would be likely to be classified as background. We can see how difficult it would be to discover and use such a vague rule with the weighted-sum approach.

3.4 Segmentation by Coherent Matting

To solve the matte, previous methods usually first do binary segmentation, and then refine the boundary by border matting, which can be accomplished based on image matting methods, such as Bayesian matting [Chuang et al. 2002; Apostoloff and Fitzgibbon 2004], Possion matting [Gong et al. 2010] and closed-form matting [Bai et al. 2009]. In fact, for propagation-based video segmentation, because the probability map resulted by local propagation is very close to the binary segmentation, it should be better to solve the matte in one step. In our system, the final matte is solved by minimizing the following energy equation:

$$\begin{aligned} \alpha = \arg\min_{\alpha} \sum_{x \in U} [\, \lambda_x^T (\alpha_x - \alpha_x^T)^2 + \lambda_x^C (\alpha_x - \alpha_x^C)^2 \,] \\ + \lambda^S \alpha^T L_m \alpha \end{aligned} \tag{15}$$

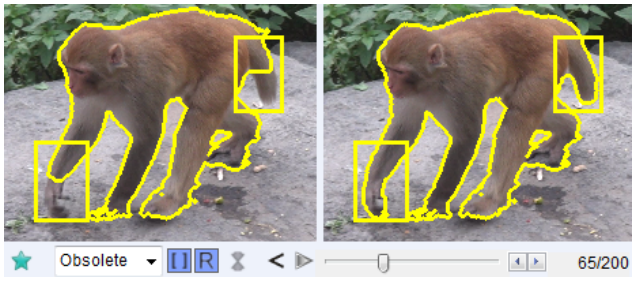


Figure 7: *The user interface of our system for local editing. The comparative view shows the result before (left) and after (right) the editing, and the rectangles highlight the regions being edited, so that the user can supervise the changes easily.*

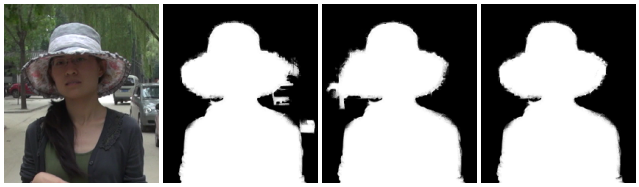


Figure 8: *Using JBP to remove errors. (a) The input image, where the object is moving left, and so for forward propagation, the right side is newly exposed, and for backward propagation, the left side is newly exposed; (b) the result of a single forward pass; (c) The result of a single backward pass; and (d) The result of a forward pass followed by an additive backward pass.*

which has the same form as the matting equation adopted in [Bai et al. 2009]. U is the unknown region. L_m is the matting Laplacian matrix [Levin et al. 2008], and λ^S is set to be 20. The two unary terms encode the temporal and the spatial constraints for α.

To determine U, we first compute a combined confidence map as:

$$C(x) = (1 - |p_{\max}(x) - p_{\min}(x)|)\frac{|p(x) - 0.5|}{0.5} f(x) \qquad (16)$$

where $p_{\max}(x)$ and $p_{\min}(x)$ are the results of applying 7×7 max-filters and min-filters to $p(x)$, respectively. The first term excludes the regions with non-uniform probability (the object boundary); the second term excludes the pixels whose probability approaches neither 0 nor 1; the third term excludes the pixels with low confidence. The unknown region is then determined by thresholding of $C(x)$.

α_x^T encodes the temporal likelihoods; in [Bai et al. 2009], it is computed by tracking the local windows, which is equivalent to conducting a second pass propagation. With the one-step approach, α_x^T can be computed directly from $p(x)$. We set λ_x^T to $f(x)$, and defined α_x^T as $\alpha_x^T = S(p_o(x))$, where $S(.)$ is a sigmoid function [Mortensen and Barrett 1999] used to impose a sparsity prior [Rhemann et al. 2008], which can greatly reduce the numbers of false semi-transparent pixels. α_x^C is used to impose boundary conditions, for foreground and background boundary pixels it is set to be 1 and 0, respectively. λ_x^C is set to be a large number for boundary pixels, and for interior pixels it is set to be 0, which eliminates the effect of spatial terms.

Equation (15) can be minimized by solving a sparse linear system. For an image containing 10^4 unknown pixels, our system takes about 0.35 s to solve the equation. For cases with more unknown pixels, we first solve the equation at a lower resolution, and then upsample the result to the full resolution using bilateral upsampling [Kopf et al. 2007], the boundary is further refined using the method proposed in [Yang et al. 2011]. The quality of the resulting matte is comparable with that solved in full resolution, but the time cost is greatly reduced. The total time for each frame does not exceed 0.8 s for any of the examples that we have tested, which contain unknown pixels numbering up to 10^5.

4 Progressive Refinement

For videos of complex scenes, it is very difficult to achieve good results with only a single pass propagation. Our system supports progressive refinement of the results. To edit a spatio-temporal region, the user can first remove its errors in any one frame, and then propagate the changes forward and backward. As shown in Fig. 7, a prompt window is presented for each region undergoing editing, and the effect of local editing can be observed through the comparative view, which shows the results before and after the last time of local propagation.

The key issue of local editing is to avoid undesirable overwriting, for which purpose it should take effect only in the spatio-temporal region to be edited. In [Bai et al. 2009], the updated region of local editing is determined in the source frame according to the distance to the brushed pixels, which may cause neighbor regions to be updated improperly. In addition, local editing is not always triggered by user brushes, for example, small regions may be removed by connected component analysis. An efficient local editing tool should be able to detect all unpropagated edits.

4.1 Additive Propagation

In our system, spatio-temporal local editing is mainly achieved by additive propagation, which requires two additional images, M_t^{left} and M_t^{right}, to be stored for each frame to save respectively the matte being propagated to the previous frame and the next frame at the last time. When performing additive propagation, only the differences of M_t and M_t^{right} (for forward propagation) or M_t^{left} (for backward propagation) are propagated. Denoted by D_s the set of pixels to be propagated, and D_s' the corresponding pixels of D_s in the target frame (determined using optical flow). For additive propagation, we first calculate the bounding box for D_s, and then propagate the matte in the rectangle to the target frame using the method proposed in Section 3. The resulting probability and confidence maps are then combined with the matte of the target frame as follows:

$$\begin{aligned} p'(x) &= \tau(x)p(x) + (1 - \tau(x))\alpha^0(x) \\ f'(x) &= \tau(x)f(x) + (1 - \tau(x)) \end{aligned} \qquad (17)$$

in which $\alpha^0(x)$ is the current alpha value of x, and the weight function $\tau(x) = \exp(-d_g(x)/\sigma_g)$, where $d_g(x)$ is the geodesic distance from x to D_s' [Bai and Sapiro 2007]. The final matte is solved based on $p'(x)$ and $f'(x)$ with the method in Section 3.4.

The above method is easy to be implemented, and M_t^{left} and M_t^{right} are necessary to be stored only when the matte is to be modified. Compared to the local editing method in [Bai et al. 2009], it is more accurate in identifying the affected regions, and thus can better avoid undesirable overwriting. Also, by using M_t^{left} and M_t^{right} to detect residual matte differences, any edit to the matte can be accumulated and saved. As a result, the user can work freely without worrying that the edits might not be propagated sufficiently.

(a) Source (b) Target (c) Local (d) UDC (e) Global (f) Combined (g) UDC-0° (h) UDC-90°

Figure 9: *Probability maps produced by different classifiers. See text for explanation.*

4.2 Joint Bidirectional Propagation

Based on the additive propagation, our system can perform bidirectional propagation more effectively than in previous methods. For each video, a forward pass is first applied; the user then edits the last frame to remove any errors, and this is followed by an additive backward pass. In this way, the backward pass would take effect only in the regions that had been mis-segmented in the forward pass. We call this process as *Joint Bidirectional Propagation (JBP).*

Using JBP can greatly reduce the number of user interactions required, as demonstrated in Fig. 8. The propagation algorithm is more likely to produce errors in newly-exposed regions, which are very different in forward and backward propagations because of the opposite directions of the image movements. Therefore, the errors produced by forward and backward propagations often appear in different regions. By adaptively integrating the results of the forward and backward propagations, JBP can remove a large number of errors with only a little user effort.

The most effective way to use our system is to first perform a round of JBP, and then use AP to remove the remaining errors. During the forward pass of JBP, the user can temporarily ignore the errors in newly-exposed regions, which is better to be removed in the backward pass. Please see the accompanying video for a demonstration.

5 Experiments

We implemented the proposed cutout system in C++ and tested it on a PC with four 3.3 GHz CPUs. UDCs in different directions along with other base classifiers are evaluated in parallel in multiple threads for acceleration. Note that UDCs and LKNN in different windows are independent, so they are highly suitable for implementation on GPU. However, we are currently mainly focused on improving the robustness of our system against complex cases. A GPU-based fast implementation will be considered in the future. For most of the used examples, our system takes about 1.5 s to perform a full propagation. AP can be much faster, depending on the area of the regions to be propagated. If not stated otherwise, the combining classifier is trained with the training data set described in Section 3.3. The testing examples are made to be different from the training examples in both foreground and background for a fair evaluation[1].

5.1 Evaluation of the Proposed Methods

Fig. 9 shows the probability map produced by the different classifiers for a typical example containing both newly-exposed regions and inseparable color statistics. Note that only UDCs correctly segment the newly-exposed background hole. Global classifier produces a lot of errors because of the inseparable color statistics. Local classifier mis-segments the background hole and produces many errors near the hands, both of which are caused by the large temporal discontinuity. The integrated result is more accurate than that of each individual classifier. Fig. 9(g) and Fig. 9(h) are produced by horizontal and vertical UDCs, respectively, from which we can see how the UDCs correctly deal with the complex case, where the horizontal UDCs contribute more to recognition of the background hole, while the vertical UDCs contribute more to separation of the color statistics.

Table 1: *The error rates (%) resulted by different combinations of the base classifiers. The examples are shown in Fig. 10 except the CAR, which is shown in the accompanying video. Φ denotes the set of the four base classifiers.*

	CJ	TMM	DX	CW	CAR
Φ	3.02	1.37	4.05	3.71	1.36
Φ - shape	3.09	1.34	4.02	3.82	1.40
Φ - local	3.32	1.41	4.18	4.20	1.53
Φ - UDCs	3.67	1.50	4.51	3.84	1.46
Φ - global	2.95	1.33	3.95	3.79	1.30
shape + local	3.54	1.78	4.64	4.34	1.22
UDCs + global	3.93	1.45	4.37	4.06	1.92

To evaluate the effects of each base classifier, we tested the four base classifiers in different combinations. The results are listed in Table 1. For each combination, the combining classifier is first retrained. Each test video is uniformly divided into a series of 10-frame subsequences[2], and for each subsequence, the result of its first frame is initialized with the ground-truth, and is then propagated forward automatically until the last frame, whose error rate is regarded as the error rate of the subsequence[3]. The error rate of each video is the mean error rates of its subsequences. From Table 1, we can find that removing either local classifiers or UDCs will significantly increase the error rates. Removing local classifiers will increase the errors near the object boundary, and introduce more false background holes (as in Fig. 13). UDCs contribute a lot if the video contains many discontinuities that cannot be correctly recognized by global classifier (e.g. BF). Both shape prior and global classifier are not very stable, and may degrade the accuracy at times, particularly the global classifier. Intuitively, the training process will assign global classifier lower priority, but it seems that this is still

[1]The training and testing examples can be found in the supplemental materials except the CARTOON set.

[2]The length of subsequences can be chosen freely, but should not be too short so that accumulated errors can be observed.

[3]In order to reflect visual effects, the error rates in our experiments are normalized by the number of foreground pixels.

Table 2: *Evaluation of different combination methods of the base classifiers. MAX-CONF selects the classifier with maximum confidence to produce the outputs, other rows show cross-validation of the proposed learning-based method. The data set ALL contains all training videos. The error rates (%) are computed as in Table 1.*

	HUMAN	ANIMAL	CARTOON	STATIC
MAX-CONF	5.38	4.84	4.89	1.78
ALL	2.91	2.20	2.11	1.05
HUMAN	2.88	2.12	2.05	1.09
ANIMAL	2.95	2.26	2.13	1.04
CARTOON	3.13	2.32	2.14	1.06
STATIC	3.54	3.03	2.68	1.12

Table 3: *Comparisons of user labor (average NoBs by each user) and time (in minutes) costs, and the resulting accuracy (mean absolute error). In order to reduce the users' work, only the first 100 frames were used except for the example CAR.*

		RotoBrush			Ours		
	Frames	NoBs	Time	Error (%)	NoBs	Time	Error (%)
CJ	100/386	392	22.4	1.41	213	14.6	1.33
TMM	100/390	197	12.9	0.79	114	8.7	0.68
DX	100/300	423	26.5	1.63	192	13.5	1.45
CW	100/300	484	25.1	1.08	143	11.2	0.91
CAR	252	83	11.2	0.94	56	9.3	0.92

not able to completely restrain its side effect. The example CAR is the video of a static car, and thus contains much fewer temporal discontinuities than other examples, in this case adding UDCs and global classifier would not help to improve the accuracy.

Table 2 shows cross-validation of the learning-based combination method, and compares it with the max-confidence strategy. The error rates are produced by the automatic 10-frame propagation process described above. We can find that the accuracy achieved by using HUMAN, ANIMAL, and CARTOON for training are comparable. The reason is that the combining classifier is not affected by shape and appearance of the training video objects, because the feature vector $v(x)$ contains only the outputs of the base classifiers and the local registration errors, and any raw image features such as colors and gradients are not involved. It seems that the only requirement of the training video set is that it should contain sufficient temporal discontinuities and inseparable statistics, so that the space of $\hat{v}(x)$ can be covered. To prove this we introduced the data set STATIC, which are captured in static scenes, and thus contains very few temporal discontinuities. From table 2 we can find that STATIC results in significant more errors than other training data sets, this is the case even when it itself is used for testing. The max-confidence strategy produces much more errors than the proposed learning-based method, this is expected because as we have analyzed in Section 3.3, the confidence values resulted by different classifiers are not strictly comparable. A careful tuning of the confidence functions may be helpful for improving the accuracy, but that would be difficult when multiple classifiers are involved.

5.2 Results and Comparisons

Fig. 10 shows some of the examples used in our experiments. These examples are challenging because of the complex backgrounds, the inseparable color statistics, the low-contrast object boundaries, and the complex varying topologies. Our system can deal with all of these cases efficiently. Note that it would be difficult to remove the errors via a scribble brush in small regions, such as the tail of the elephant, and the small background hole of TMM. Thanks to the power of UDCs, our system can correctly segment these regions with only a little user interactions.

Fig. 11 shows the results of our method for automatic segmentation of 10 frames of the CJ example, as well as some clips obtained using Rotobrush and [Zhong et al. 2010] for comparison. Rotobrush is based on Video Snapcut [Bai et al. 2009], which uses local classifiers for segmentation propagation. [Zhong et al. 2010] combines local and global classifiers to handle occlusions. As shown, Rotobrush produced significant errors in region A, B because of the fast motion and newly-exposed regions. [Zhong et al. 2010] can recognize only a part of the background hole in region B, and produced significant errors in region A and C because of the sensitivity of global classifier to complex color distributions. Our method correctly dealt with all challenging cases.

By allowing sufficient user interactions, an interactive segmentation system can always achieve high-quality results. The most important measure of the system is therefore not the resulting accuracy, but the amount of user work and time required to obtain an accurate result. Because most existing systems use scribble brushes to remove the segmentation errors, it is reasonable to take the *number of brushstrokes* (NoBs) required by a system as a measure of the user work. To evaluate our system, we asked 17 users to perform a study, where 15 of them were familiar with Photoshop, 3 of them had experience with Rotobrush, but none of them had used our system. The users were first trained with Rotobrush and our system, and were then asked to segment the testing videos with both systems. They were not allowed to finish until the referee considered the result to be good enough. Table 3 shows the results of the test, including both the labor and time costs and the resulting accuracy.[4] Compared with Rotobrush, our system requires less time, and far fewer interactions, while at the same time achieving better accuracy. Note that although UDCs does not do positive contribution for the example CAR (see Table 1), our system still outperforms RotoBrush because of the use of JBP. Even in static scenes, a lot of errors will be caused by discontinuities due to depth variations.

5.3 Failure Cases

As with most of the previous methods, our method mainly uses color statistics for pixel classification, which inevitably introduces some limitations. Because the local classifier and shape prior cannot take effect in discontinuous regions, it is more likely to suffer from inseparable statistics than in the continuous regions. Fig. 12 shows a typical failure case for our method, where the newly-exposed background regions are incorrectly segmented because of the similar appearance of the foreground. The errors in region A can be removed easily by a backward propagation, while the errors in region B require more brushstrokes for them to be removed.

Fig. 13 shows another typical failure case for our method that resulted in false background holes, which is a side-effect of non-local propagations. False background regions may appear only in newly-exposed foreground regions whose appearances are similar to some parts of the background, and such cases do not occur very often in practice (far less frequently than newly-exposed background regions). Also, most false background holes are very small, and can thus be removed easily by appropriate setting of the minimum size of the background regions. We should also note that it is much easier to remove a background hole than to produce such a hole interactively, because the latter requires more careful interactions.

[4]For Rotobrush, we used a keyboard-mouse hook program to record the number of brushes for each frame.

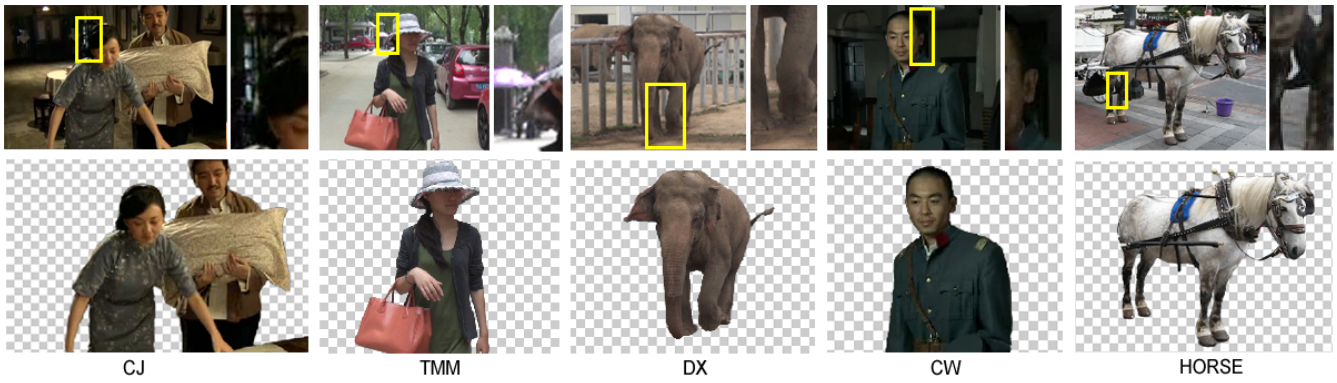


Figure 10: *Some of the examples used in our experiments. The regions in the rectangles demonstrate the challenges.*

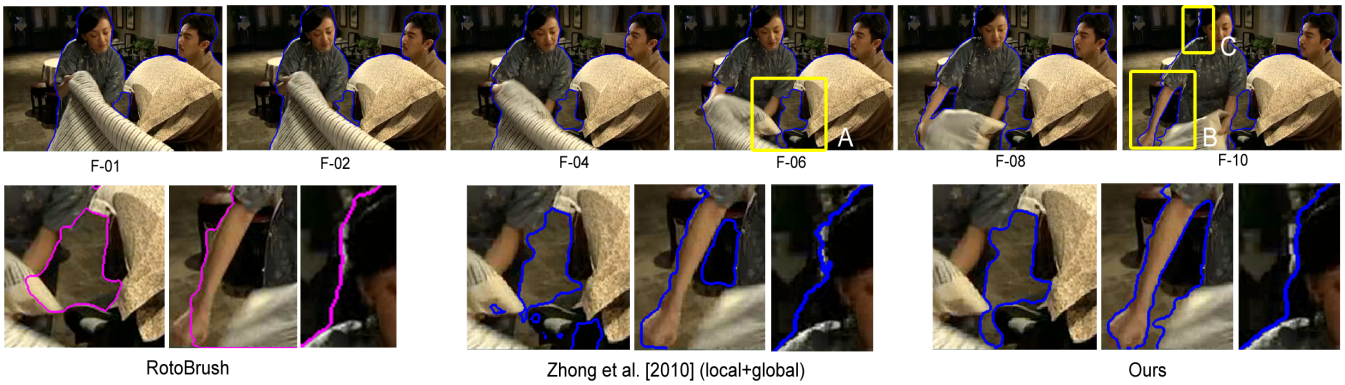


Figure 11: *Results and comparisons. Top: the results of our method for automatic segmentation of 10 frames of the example CJ; Bottom: comparisons of our method with the results of Rotobrush and Zhong et al. [2010] in the regions A, B, and C.*

Figure 12: *Failure case caused by inseparable statistics in discontinuous regions.*

Figure 13: *Failure case with false background holes, where some foreground colors are newly-exposed.*

As in previous systems, our method also suffers from motion blur and illumination changes [Bai et al. 2009]. Also, although it does not use binary segmentation to produce the results, our method still cannot be used for semi-transparent objects or hairs, because the segmentation results are propagated by binary classifiers.

5.4 Discussions

Video segmentation is still a difficult task even if accurate optical flow is available. Although temporal discontinuity can be detected using optical flow, it does no help to the classification of pixels in discontinuous regions. Temporal discontinuity may appear frequently, even for the case of a simple foreground topology (e.g. CW in Fig. 10, please see the accompanying video for a demonstration). For both Rotobrush and our system, most errors are produced in discontinuous regions (as in Fig. 12). In discontinuous regions, only UDCs and global classifier can take effect, so for further improvements, more advanced features are necessary. With the proposed learning-based combination approach, it should be easy to introduce new features. However, using more features would result in a higher dimensionally integrated feature vector, whose space would be difficult to cover by using a look-up table. We adopted the current approach just in order to handle the large quantity of training data. Considering the scalability requirements, a well-designed cascade classifier may be a better choice.

In our system, the propagation of segmentation results is always conducted between adjacent frames. This is different from Video Snapcut, which stabilizes the foreground to some keyframes in order to reduce the accumulation errors. Accumulation errors are not a great problem for our system, because first, our system produces far fewer errors than the previous propagation methods; second, the accumulated errors can be removed easily using JBP (see the accompanying video). Because adjacent frames are easier to be regis-

tered, temporal coherence can be better preserved using our system.

Compared with the multi-size classifier proposed in [Bai et al. 2010], our method is more practical. Although in theory the multi-size classifier covers all scales, from pixel-level to the global level, in practice, it is difficult to determine the proper window size. More importantly, when considering inseparable color statistics and computational efficiency, traditional classifiers with regular support windows are not suitable for larger scales, as discussed at the end of Section 2.1. The advantages of UDCs are mainly due to its long narrow shape; the directionality is important for handling different cases of object shape and motion, and for better use of the spatial context information. For the case where the foreground only moves horizontally, it is enough to achieve good results by using only UDC-0°.

6 Conclusions

We have presented a video cutout system that is capable of dealing with temporal discontinuities while also remaining robust to inseparable color statistics. Achieving these two goals simultaneously is not easy, because they tend to conflict. We introduced UDCs, a novel segmenting classifier to handle temporally discontinuous regions. Multiple classifiers in different scales are then combined via a learning-based approach. Compared to the traditional weighted-sum approach, the learning-based combination approach can significantly improve the segmentation accuracy. The foreground matte is finally solved using a coherent matting procedure, and remaining errors can be removed efficiently by JBP and AP.

Experiments show that when compared with existing systems, our system performs more intelligently in handling complex cases, and can save a great deal of user effort and time. Our approach can make full use of color statistics over different scales for pixel classification. The learning-based combination approach can also be improved by incorporating more features. Our future work will explore the inclusion of more advanced features for better robustness, and the power of GPU for better efficiency.

Acknowledgements

The authors would like to thank the anonymous reviewers for their great helps to improve this paper. Thank Yan Huang for dubbing the accompanying video. Thank Shandong Film&TV Production Center for the permission to use their footages. This paper is supported by 973 program of China (No. 2009CB320802), NSF of China(No. U1035004, No. 61202149, No. 61173070), Natural Science Fund for Distinguished Young Scholars of Shandong Province (No. JQ200920), Chinese Postdoctoral Science Foundation (No. 2012M511509).

References

AGARWALA, A., HERTZMANN, A., SALESIN, D. H., AND SEITZ, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graphics 23* (August), 584–591.

APOSTOLOFF, N., AND FITZGIBBON, A. 2004. Bayesian Video Matting Using Learnt Image Priors. In *CVPR*, 407–414.

BAI, X., AND SAPIRO, G. 2007. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, 1–8.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graphics 28* (July), 70:1–70:11.

BAI, X., WANG, J., AND SAPIRO, G. 2010. Dynamic color flow: a motion-adaptive color model for object segmentation in video. In *ECCV*, 617–630.

BILMES, J. 1998. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Tech. rep., International Computer Science Institute, Berkeley.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Trans. Graph. 21* (July), 243–248.

GONG, M., WANG, L., YANG, R., AND YANG, Y.-H. 2010. Real-Time Video matting using Multichannel Poisson Equations. In *Graphics Interface*, 89–96.

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graph. 26* (July).

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2008. A closedform solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 228–242.

LI, Y., SUN, J., AND SHUM, H.-Y. 2005. Video object cut and paste. *ACM Trans. Graphics 24* (July), 595–600.

MORTENSEN, E. N., AND BARRETT, W. A. 1999. Toboggan-Based Intelligent Scissors with a Four Parameter Edge Model. In *CVPR*.

PRICE, B., COHEN, S., AND MORSE, B. 2009. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 779–786.

RHEMANN, C., ROTHER, C., RAV-ACHA, A., AND SHARP, T. 2008. High resolution matting via interactive trimap segmentation. In *CVPR*, 1–8.

ROSS, D. A., LIM, J., LIN, R.-S., AND YANG, M.-H. 2008. Incremental learning for robust visual tracking. *International Journal of Computer Vision 77* (May), 125–141.

SARIM, M., HILTON, A., AND GUILLEMAUT, J.-Y. 2009. Non-Parametric patch based video matting. In *BMVC*, British Machine Vision Association, 98.1–98.11.

TANG, Z., MIAO, Z., WAN, Y., AND ZHANG, D. 2011. Video matting via opacity propagation. *The Visual Computer (19 April)*, 1–15.

TONG, R.-F., ZHANGY, Y., AND DING, M. 2011. Video brush: A novel interface for efficient video cutout. *Computer Graphics Forum 30*, 2049–2057.

VILALTA, R., AND DRISSI, Y. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review 18*, 77–95.

WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cutout. *ACM Trans. Graphics 24* (July), 585–594.

YANG, L., SANDER, P. V., LAWRENCE, J., AND HOPPE, H. 2011. Antialiasing recovery. *ACM Trans. Graph. 30* (May), 22:1–22:9.

YIN, P., CRIMINISI, A., WINN, J., AND ESSA, I. 2007. Tree-based classifiers for bilayer video segmentation. In *CVPR*, 1–8.

ZHONG, F., QIN, X., AND PENG, Q. 2010. Transductive segmentation of live video with non-stationary background. In *CVPR*, 2189–2196.