Transductive Segmentation of Live Video with Non-stationary Background

Fan Zhong[1] Xueying Qin[2] Qunsheng Peng[1*]
[1]State Key Lab. of CAD&CG, Zhejiang University
[2]Department of Computer Science, Shandong University
{zhongfan, xyqin, peng}@cad.zju.edu.cn

Abstract

Online foreground extraction is very difficult due to the complexity of real scenes. Almost all the previous methods assume that the background is stationary, which not only incur unreliable result due to background activities like dynamic shadow, moving background objects etc., but also makes them hard to be extended to the case of non-stationary background.

In this paper we assume that the background is continuous instead of stationary, and present a transductive video segmentation method that can handle dynamic scenes captured by a hand-held moving camera. The segmentation is propagated based on local color models and temporal prior, as well as a dynamic global color model (DGKDE) in the case of occlusion. A novel local color modeling method, FLKDE, is proposed to model both local color distribution and temporal prior at each pixel. FLKDE can be learned additively to reach real-time speed. Finally, a very fast geodesic-based method is adopted to solve for the segmentation. Experiments show that our method can generate good quality segmentation for wide variety of scenes, and can reach 15∼25 fps for 640×480 size of input image sequences.

1. Introduction

High-quality online video segmentation is a crucial step for many vision-based systems, e.g. network meeting and augmented reality systems, etc. In these systems the foreground object must be extracted accurately at frame-rate to enable real-time background replacement or special effect synthesis.

In [9], V. Kolmogorov et al. proposed an effective binocular video segmentation method, which makes use of stereo matching to improve segmentation accuracy. Almost all succeeding works, however, deal with monocular video segmentation [12, 5, 16, 17]. Background, appearance, motion as well as image contrast have been proposed as the cues. Different cues are typically fused into an energy equation that can be solved efficiently with max-flow/min-cut [4], which can reach $10 \sim 15$fps for 320×240 size of input images.

Almost all previous methods, either explicitly or implicitly, assume that the scene keeps certain "stationary" properties. Typically, the background is assumed to be stationary; the appearance and motion models are also considered to be "stationary" so that they can be pre-trained offline and then be used without online updating. These *stationarity assumptions* greatly simplify the problem and can bring us with both accuracy and efficiency. However, in practice we often find that real scenes are not stationary, dynamic shadows, moving background objects as well as camera shaking are common in real scenes. Even in some well-controlled laboratorial environment, shadow of the moving foreground object still makes trouble.

There have been some efforts to deal with dynamic background. A. Monnet [10] et al. proposed a background modeling method that can handle background activities like waving trees. J. Sun et al. [12] suggested some useful strategies to handle sudden illumination change, camera shaking etc. P. Yin et al. [16] proposed to use pre-trained motion classifier to separate foreground object from moving background object. However, these methods are still based on the stationarity assumptions, and thus are hard to be extended to deal with more general cases. In [8], Z. Dong et al. proposed a method to segment video captured by a rotating camera, but their method requests a pre-constructed background panoramic image for online registration and background subtraction, hence hard to be extended to the case of moving camera as panorama construction and registration are both very difficult in this case.

Therefore, stationarity is not a proper assumption of real scenes, even if the camera is fixed. A better assumption is to reckon the input video as temporal continuous, which means that for any registered pair of pixels between two adjacent frames, large displacement would not occur in both image and color spaces. We call this assumption as *con-*

*This author is corresponding author

978-1-4244-6985-7/10/$26.00 ©2010 IEEE

tinuity assumption. Unlike the stationarity assumptions, which are often violated in real scenes, continuity assumption holds in most cases if only there is no fast camera movement and sudden change of lighting condition.

Under the continuity assumption, adjacent frames must be very similar, so a natural question is whether the segmentation of previous frame can be propagated to the current frame. Unfortunately, although the temporal continuity of video has been noted for a long time, there is yet no effective way to make use of this property in online video segmentation. Interactive video segmentation methods make use of continuity by optimization over the 3D video cube [15], but this is obvious infeasible for online segmentation. The temporal prior proposed in [5] is the first attempt to make use of the temporal continuity in online segmentation. However, its temporal prior is represented by only four statistical transition probabilities (i.e. BB, BF, FB, FF) learned from labeled video, which is very inaccurate in dynamic scenes.

In this paper we propose a transductive online video segmentation method based *only* on the continuity assumption. Our contribution includes: 1) a real-time algorithm to propagate segmentation by combining local color distribution and temporal prior, which is insensitive to continuous changes in both foreground and background, hence sufficient accuracy when dealing with dynamic scenes; 2) a dynamic global appearance model which allows fast online updating and dynamic adjustment of weights of frames, and can be used to handle occlusion; 3) a geodesic-based method to solve for the final segmentation by incorporating smooth prior and image contrasts, which is much faster than min-cut, hence capable of dealing with larger size of input image sequences in real-time.

2. Overview

2.1. Background

A popular way of image and video segmentation is to formulate the problem as the following energy minimization problem:

$$E(\boldsymbol{\alpha}) = \sum_i E_1(\alpha_i) + \lambda \sum_{(i,j)\in\mathcal{E}} E_2(\alpha_i, \alpha_j) \qquad (1)$$

where $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_i, \cdots, \alpha_N)$ is the segmentation we want to get, and α_i is the state of the i-th pixel. $\mathcal{E}$ denotes the set of all neighboring pixels. λ is a free parameter. E_1 is the data term, which measures the cost under the assumption that the state of the i-th pixel is α_i. E_2 is the smooth term, which encodes our prior knowledge about the segmentation, e.g. smooth and contrast-sensitive [3].

Each data term relates to only one pixel, and can be computed as the negative log of the probability of a pixel belonging to foreground and background, so data terms are uniquely determined by the probability map. The probability map encodes our knowledge about different layers, e.g. color distribution, motion model, etc. and can be regarded as the observed value of the true segmentation. Smooth terms can be used to remove minor errors, and is helpful to attain more accurate boundary if contrast information is incorporated. The way of computing the smooth terms takes no large difference in previous methods, specifically, almost all are based on the contrast-sensitive Ising prior [3].

Equation (1) can be minimized efficiently with min-cut [4]. Although min-cut is fast, for online video segmentation it is still not fast enough. Min-cut costs about $60 \sim 90$ms for an input image of size 320×240, and thus cannot be used for larger size of input image sequence if real-time speed should be guaranteed.

2.2. Transductive video segmentation

Our basic idea to handle scenes with dynamic background is to segment input frames transductively, adopting the segmentation of the previous frame as a non-parametric model to segment the current frame. In this way we avoid to maintain the segmentation model, but the segmentation model is always updated. This is different from previous methods, in which the segmentation model should be maintained explicitly, and any change of the scene may introduce inconsistence.

Given the segmentation of previous frame, how can we propagate it to current frame in a reliable way? In dynamic scenes both background and motion information are hard to be extracted, the most straightforward way is then to propagate the segmentation based on the global color distribution (GCD). Previous methods usually use a pair of Gaussian Mixture Models (GMMs) to model the GCD of foreground and background. GMM is a parametric model that needs to be trained through Expectation Maximization (EM), which is too slow to reach real-time speed. More importantly, GCD cannot provide us with sufficient accuracy by itself, in previous methods it is always used with other cues like background and motion.

Under the continuity assumption, not only the global but also the local color distribution (LCD) at each pixel can be considered consistent between adjacent frames. LCD is much more accurate than GCD because much less irrelevant pixels are included in the sample set. However, by using LCD we need to train the color model for every pixel, which would definitely introduce large computational cost if no effective acceleration method is available.

To overcome the above difficulties, we propose the Fast Local Kernel Density Estimation (FLKDE). FLKDE not only can model LCD in real-time, but also combines the temporal prior in an elegant way. FLKDE works if the continuity assumption holds, and performs very well for dynamic scenes.

(a) input (b) FLKDE (c) DGKDE (d) combined

Figure 1. Computing the probability map: (a) input frame, note that in region A, the background has nearly the same color as the head of the people, and in region B, the small background region is occluded in the previous frame (not showed here); (b) probability map produced by FLKDE, region A is correctly classified; (c) probability map produced by DGKDE, region A is misclassified, but in region B it performs much better than FLKDE; (d) the combined probability map.

Occlusion may cause propagation fail because a region appeared in the current frame may be occluded in the previous frame (e.g. region B in Fig.1). To handle this case we must resort to information from other frames. A dynamic global appearance model, namely, the Dynamic Global KDE (DGKDE), is proposed for this purpose. Unlike previous global color modeling methods, DGKDE not only supports fast online updating, but also can dynamically adjust the weights of sampled frames.

Fig.1 illustrates our procedure to compute the probability map. The results of FLKDE and DGKDE are combined to get the final probability map, after that we can solve for the segmentation by incorporating smooth prior and image contrasts. To overcome the bottleneck of efficiency, we introduce a novel geodesic-based segmentation method which is much faster than min-cut.

The idea to segment images transductively is not new. In [7], J. Cui et al. proposed a transductive object cut method to segment a group of similar images. After the first image is manually segmented, other images can be segmented automatically. Recent popular video object cut methods also work transductively [2, 11], and it has been proved that local classifiers perform much better than their global counterparts [2]. However, so far there is no local classifier can reach real-time speed, and the FLKDE to be introduced in this paper is the first local classifier that can be used for frame-rate segmentation.

3. Algorithm

3.1. Propagation by FLKDE

As we have mentioned above, GMM is not fast enough to be updated online. In [5], A. Criminisi et al. suggested to use 3D color histogram instead of GMM, which allows fast online updating. Kernel Density Estimation (KDE) is a nonparametric probability estimation method that is very similar to histogram. The advantage of KDE over histogram is that it can result in smooth pdf and is independent on the end points of bins (the point where the pdf is to be estimated is always the center of the bin) . However, accurate estimation of kernel density involves large amount of computation, and thus is not suitable for our problem. In the following we propose an approximate KDE method specific for the $256 \times 256 \times 256$ color space, and then introduce the Local KDE as well as its acceleration algorithm.

Approximate KDE: Given the training color sample set S, a color histogram of $8 \times 8 \times 8$ size of bins is first constructed. Denoted by B_i the i-th bin and $|B_i|$ the number of color samples falling into B_i, the frequency of samples appearing at the position of x (in color space) can be approximately estimated as:

$$f(x|S) = \sum_{B_j \in \mathcal{E}(B_{[x]})} \omega_j |B_j| \tag{2}$$

where $B_{[x]}$ is the bin containing x, and $\mathcal{E}(B_{[x]})$ denotes the set of neighbors of $B_{[x]}$, including the bin itself, so there are total 27 bins in $\mathcal{E}(B_{[x]})$. The weighting function ω_j can be used to smooth the resulted pdf, and is defined as an exponential function of the distance of B_j to the center bin $B_{[x]}$. Note that ω_j is independent of x, so it can be precomputed.

The probability density function then should be:

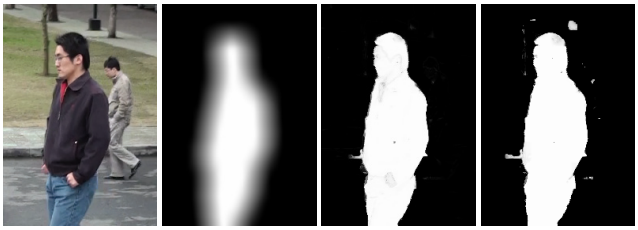
$$\hat{p}(x|S) = \frac{1}{h^3|S|} f(x|S) \tag{3}$$

where $|S|$ is the number of samples contained in S, h is the bandwidth, which should be 24 (=8×3) here.

The above method is in fact an approximation of Parzen windows of size $24 \times 24 \times 24$. Compared with the color histogram used in [5], our method can produce smoother pdf and at the same time, would never put the estimated point to the boundary of the window (although it is still not exactly the center). The sacrifice is a little more computational cost, which is bearable for our problem. Therefore, approximate KDE is a better trade-off between accuracy and efficiency.

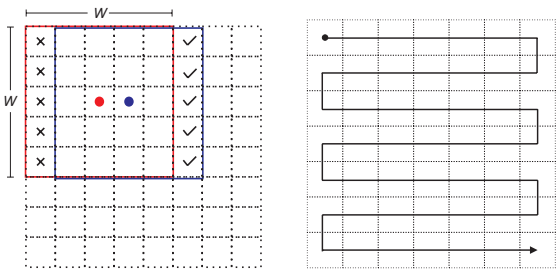
Local KDE: The "local" here means that the KDE model (or the estimated pdf) is specific for each pixel, and not shared by all pixels. Given the segmentation of the previous frame and a pixel location x (note that we use x to denote a pixel as well as its color), the sample set can be obtained from the $W \times W$ window centered at x. Denoted by $S^{\mathcal{F}}$ and $S^{\mathcal{B}}$ the foreground and background sample set, respectively. $S^* = S^{\mathcal{F}} \cup S^B$ is the set of all samples. The foreground pdf *with temporal prior* then can be expressed as:

$$\tilde{p}^{\mathcal{F}}(x) = \hat{p}(x|S^{\mathcal{F}}) p(\mathcal{F}|S^*) \tag{4}$$



(a) (b) (c) (d)

Figure 2. The temporal prior and its effect: (a) input frame; (b) the temporal prior; (c)(d) probability map produced by FLKDE with (c) and without (d) temporal prior.



(a) (b)

Figure 3. Illustration of Fast Local KDE. (a) deleted (marked by $\times$) and new added (marked by $\surd$) pixels when making horizontal movement; (b) scan order of pixels.

where the first term is the pdf estimated from foreground samples, which models the local color distribution of foreground at the location of x; the second term is the temporal prior, which should be measured according to the segmentation of previous frame so that the temporal continuity is preserved, here we define it as:

$$p(\mathcal{F}|S^*) = \frac{|S^{\mathcal{F}}|}{|S^*|} \tag{5}$$

one can easily interpret this by considering the special case when the whole sample window is located in background region ($|S^{\mathcal{F}}| = 0$), in which case the pixel x must be very far from the foreground in previous frame, hence little chance to be foreground in current frame and $p(\mathcal{F}|S^*)$ should be 0.

Expanding equation (4) we have:

$$\tilde{p}^{\mathcal{F}}(x) = \frac{1}{h^3|S^*|} f(x|S^{\mathcal{F}}) \tag{6}$$

the above equation looks very similar to equation (3), and can be interpreted as the foreground pdf with respect to the sample set S^*. The background pdf $\tilde{p}^{\mathcal{B}}(x)$ can also be derived in a similar way.

Now the probability of a pixel belonging to foreground can be computed as:

$$p^{\text{lkde}}(x) = \frac{\tilde{p}^{\mathcal{F}}(x)}{\tilde{p}^{\mathcal{F}}(x) + \tilde{p}^{\mathcal{B}}(x)} = \frac{f^{\mathcal{F}}(x)}{f^{\mathcal{F}}(x) + f^{\mathcal{B}}(x)} \tag{7}$$

where $f^{\mathcal{F}}(x) = f(x|S^{\mathcal{F}})$ and $f^{\mathcal{B}}(x) = f(x|S^{\mathcal{B}})$. The above equation tells us that by using frequency instead of probability density, the resulted probability is of the temporal prior as defined in (4). Interestingly, equation (7) has even simpler form than the version without temporal prior, which needs to divide the frequency by the number of samples. Fig.2 demonstrates the temporal prior as well as its effect.

Fast Local KDE: With the above approximate KDE method, to train a KDE model is equivalent to construct a color histogram. Although it is very fast, we still cannot afford the time cost to re-construct the color histogram at every pixel. Fortunately, the local color histograms can be constructed additively using sliding window, which can reduce its time cost greatly without loss of precision.

As illustrated in Fig.3(a), considering two successive pixels x_0 (red) and x_1 (blue) on the same line, with the color histogram of x_0 already constructed. Let W_0 and W_1 be the sample window of x_0 and x_1. Obviously, W_0 and W_1 are overlapped except the first column of W_0 and the last column of W_1, to get the histogram of x_1, we need only slightly modify the histogram of x_0, i.e. delete the first column of W_0, and add the last column of W_1 into it. Since histogram is in fact a counting table, so deleting and adding samples are both very easy.

Fig.3(a) illustrates the case of horizontal forward movement, horizontal backward and vertical movement can be performed additively in a similar way. To apply local KDE for all pixels, we first apply it for the left-top pixel (boundary should be filled properly first), and then adopt the additive approach to construct the KDE model of other pixels in the order illustrated in Fig.3(b). In this way the complexity is reduced from $O(W^2N)$ to $O(WN)$, where N is the number of pixels. In addition, our additive approach avoids re-initializing the large histogram table at every pixel, which also saves a lot of time.

For 640×480 size of videos and $W = 31$, fast FLKDE takes about 80ms for each frame, while the naive implementation of FLKDE takes more than 6000ms. Although this has been a great acceleration, it is still not fast enough, in section 4 we will introduce some techniques that can further reduce its time cost.

3.2. Handle Occlusion by DGKDE

Dynamic occlusion can introduce newly appeared regions that are occluded in previous frame, in these regions FLKDE would definitely fail due to the missing of training data. To handle this case we must resort to information from other frames.

A global appearance model is used to record the color distribution of foreground and background in past frames. The technique we adopted is the approximate KDE method proposed in section 3.1, which enables fast online updating. After each frame is segmented, we first get the bounding box of foreground, and then extend it to include some more background samples. Note that it is unnecessary to use all background pixels as samples because under the continuity assumption, background pixels far from the foreground object are irrelevant to the background color distribution in a short future. To update the KDE model, we first scale the histogram bins by a factor of $s, 0 < s < 1$, and then add the new samples into it. The purpose of pre-scaling the bins is to attenuate the influence of frames far from the current frame. One can easily figure out that the average weight of the t-th frame's samples should be:

$$\pi_t = \frac{s^{t'-t}}{\sum_{p=0}^{t'} s^p} \tag{8}$$

where t' is current time (zero based). In our experiment s is chosen to be 0.95.

The above global appearance model is named as Dynamic Global KDE (DGKDE). Unlike the case in FLKDE, in DGKDE the temporal prior cannot be represented as equation (4), so the probability of a pixel belonging to foreground should be:

$$p^{\text{gkde}}(x) = \frac{\hat{p}^{\mathcal{F}}(x)}{\hat{p}^{\mathcal{F}}(x) + \hat{p}^{\mathcal{B}}(x)} \tag{9}$$

where $\hat{p}^{\mathcal{F}}(x)$ and $\hat{p}^{\mathcal{B}}(x)$ are the foreground and background probability densities as defined in equation (3).

The advantage of DGKDE over previous appearance modeling methods is that, first, it allows fast online updating; second, it can dynamically adjust the weight of involved frames. These advantages make DGKDE suitable to be used in dynamic scenes. In addition, besides occlusion, DGKDE can also be used to handle large displacement in image space, which provides us a possible way to deal with fast camera movement.

3.3. Combine FLKDE and DGKDE

As we know, the key of combining multiple classifiers is to evaluate their individual reliability. Obviously, FLKDE would be unreliable if the frequency of foreground and background are both very high or very low. The former case is caused by ambiguous colors, while the latter implies the case of under-sampling, which is usually caused by small regions of colors or the violation of the continuity assumption. Therefore, *FLKDE is considered to be reliable at the location of pixel x iff $min\{f^{\mathcal{F}}(x), f^{\mathcal{B}}(x)\} < f_0$ and $max\{f^{\mathcal{F}}(x), f^{\mathcal{B}}(x)\} > f_1$, with $0 < f_0 < f_1$ two given thresholds.*

(a) prob. map (b) trimap (c) segmentation

Figure 4. Solve for the segmentation from probability map. The gray pixels in (b) are unknown pixels need to be solved.

Our method to combine FLKDE and DGKDE is simple: for each pixel x, FLKDE is first applied, if it is reliable, we set the output probability $p(x)$ to be $p^{\text{lkde}}(x)$ and then move to the next pixel; otherwise DGKDE is applied and $p(x) = p^{\text{gkde}}(x)$.

Note that we did not make choice between FLKDE and DGKDE by comparing their individual reliability, this is because FLKDE is much more accurate than DGKDE for most pixels. Our method prefers to trust FLKDE, DGKDE is applied only for a small quantity of pixels which may be occluded in the previous frame.

3.4. Solve for the Segmentation

Once the probability map is obtained, many methods can be used to solve for the segmentation. As we have mentioned in section 2.2, min-cut is not fast enough for large images. Fortunately, recent works on image segmentation show that geodesic-based methods can be much faster than min-cut [1, 6]. Here we introduce a simple yet effective geodesic-based method that fits to our problem very well.

Given the probability map, morphological opening and closing are first performed to remove small isolated regions, in this way we can emulate the effect of smooth constraint [6]. Then boundary pixels are marked out by using fast max-/min-filters, the result is then a trimap as demonstrated in Fig.4. The trimap divides image into three types of regions: foreground (Ω^F), background (Ω^B) and unknown (Ω^U). The next step is to determine the states of unknown pixels by incorporating contrast information. Defining the distance of two pixels as the square of their colors' difference: $d_{ij} = \| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2$, then the geodesic distance of unknown pixels to foreground and background can be computed efficiently by fast geodesic distance transform [13], after that we can get the alpha map as follows:

$$\alpha(x) = \begin{cases} \frac{g^B(x)}{g^F(x)+g^B(x)} & \text{if} \quad x \in \Omega^U \\ p(x) & \text{if} \quad x \in \Omega^F \cup \Omega^B \end{cases} \tag{10}$$

where $g^F(x)$ and $g^B(x)$ are geodesic distance from x to foreground and background, respectively. The states of pix-

els in Ω^F and Ω^B are copied from the probability map, so only unknown pixels need to be solved. The larger the distance of unknown pixels to background, the greater their alpha values. The final segmentation is obtained by thresholding the alpha map in order to remove false semi-transparent regions, then the boundary is smoothed to suppress flickering.

The above method looks very similar to the method in [1], the main difference is that in [1], the distance field is defined over the probability map but not over the original image, which makes it sensitive to error of the estimated probability; on the contrary, our method makes use of only image contrasts. Note that due to occlusion and color blending, the estimated probability is error-prone near the foreground/background boundary. By using image contrasts instead, our method can result in more accurate boundary. Another advantage of our method is that it enables us to compute the probability map in lower resolution and then to solve for the segmentation in full resolution for acceleration. We will discuss this problem further in section 4.

Note that with our method, image contrasts not on the foreground/background boundary may cause undesirable variation of alpha map. Fortunately, in practice the unknown region is usually very narrow, in which case the foreground/background boundary can dominate the variation of alpha map and the effect of other contrasts would be invisible.

4. Acceleration

The method proposed in previous section yet cannot achieve real-time speed, in this section we will introduce two techniques that can further boost it performance.

The first technique is to exclude some background pixels by tracking the position of foreground. Under the continuity assumption, the position of foreground would not vary much between two adjacent frames, so we do not need to process the whole image, instead, we extend the bounding box of foreground in the previous frame so that it can contain the foreground in the current frame, then take the resulted rectangle region as the ROI (region of interest) to be processed. In this way we can save a lot of computational cost if foreground object is small.

Second, since the probability map is just used to construct the trimap, and takes no effect to the segmentation of unknown region (see section 3.4). In other words, any error of the probability map in unknown region is ignorable. Therefore, it is unnecessary to calculate the probability map in full resolution, and a probability map upsampled from half of the original size should be accurate enough. In addition, in lower resolution we can use smaller sample windows for FLKDE, which makes the effect of acceleration double-fold.

	1	2	3	4	5	$\cdots$	average
GMM	1.70	1.63	3.77	6.08	2.69	$\cdots$	2.91
AKDE	1.31	1.67	2.87	6.11	2.42	$\cdots$	2.64

Table 1. Comparison of the error rates (%) of GMM and approximate KDE.

5. Experiments

We implemented the proposed method in C++, and tested it on a PC with 2.2GHz CPU and 4G RAM. The tested image sequences were captured by a DV camera of frame rate 25fps[1].

Implementation Details: In previous sections we have not mentioned how to initialize our algorithm, that is, how to get the segmentation of the first frame. Automatic initialization is very hard in dynamic scenes, so in experiments we initialized our program by segmenting the first frame manually. Real systems can initialize itself in any way according to the available information at the initialization phase, for example, it can be initialized as in [12] if the background image of the first frame is known, or as in [5] and [16] if the camera can keep stationary for a while at the beginning.

The sample window size W of FLKDE is chosen to be one third of the minimum dimension size of foreground object in previous frame, bounded by a minimum value of 15 and maximum value of 51. Larger sample window can capture larger displacement in image space; however, it also involves more computations and can affect the accuracy of FLKDE.

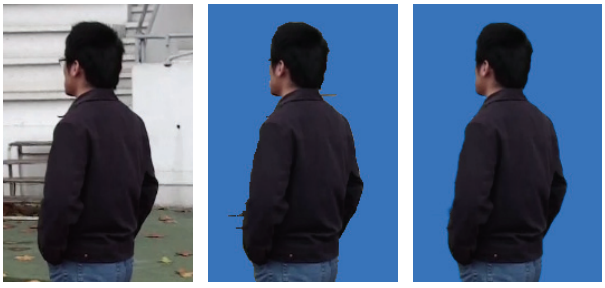
Accuracy of approximate KDE: Approximate KDE (AKDE) is the foundation of FLKDE and DGKDE. In order to verify its accuracy we compare it with GMM, which has been widely used in previous methods. For convenience we tested with image instead of video. The test image set contains 20 images that have been manually segmented. For each image, AKDE and GMM are first trained according to the segmentation and then be used to classify pixels. Tab.1 lists the error rates of AKDE and GMM. Surprisingly, AKDE performs even better than GMM. Note that the accuracy of GMM is heavily dependent on the number of Gaussian components K, the reported error rate is the lowest one for $K \in \{5, 6, \cdots, 20\}$.

Local vs. Global: As demonstrated in Fig.1, FLKDE can produce very accurate probability map (except in occluded regions), this is why our method can work under the continuity assumption without using background image and motion information, which are key to the accuracy for previous

[1]The test image sequences and initialization data are included in the supplement material.

	1	2	3	4	5	···	average
min-cut	0.54	0.43	0.72	5.61	1.76	···	1.12
our	0.57	0.37	0.69	5.84	1.77	···	1.17

Table 2. Comparison of the error rates (%) of min-cut and our segmentation method.



(a) (b) (c)

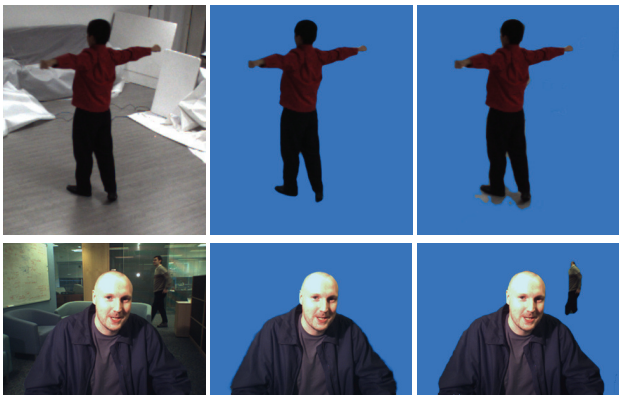
Figure 5. Background fine structures may be mistaken as foreground (b), and can be removed by smoothing the boundary (c).

methods. In experiments we tried to use only DGKDE to propagate segmentation (with FLKDE disabled), and found it would fail in most cases. Examples about this can be found in supplement material.

Accuracy of segmentation: To evaluate the accuracy of the segmentation method proposed in section 3.4, we compared it with min-cut using the 20 test images above. The probability map was first learned through approximate KDE, then min-cut and our method were used to get the segmentation, respectively. The error rates are listed in table 2, which shows that the accuracy of our method is comparable with min-cut.

As is well known, min-cut may suffer from the problem of "shrink bias" due to the incorporation of smooth prior, which may cause long thin object being cut off [14]. While in our method, the smooth prior is imposed only implicitly to remove small fragments by morphological operations, and not used when solving the unknown region. Therefore, our method can better keep fine structures of foreground. However, if the background contains fine structures connected with foreground, these structures may be mistaken as foreground (Fig.5(b)). Fortunately, most of these errors can be removed easily by smoothing the boundary (Fig.5(c)).

Robustness to shadow, etc.: As we have mentioned in the introduction, dynamic shadow and moving background object, etc. may cause problems in previous methods based on the stationarity assumption. However, in our method they would no longer cause any problem because the continuity assumption still hold when these events occur. In order for verification we compared our method with Background Cut [12] by using some videos captured by fixed camera. The results are demonstrated in Fig.6. Comparison of the whole sequence can be found in supplement material.



(a) input (b) our (c) Bkg. Cut

Figure 6. Our method is insensitive to dynamic shadow (top) and moving background object (bottom), column (c) is the result of Background Cut [12] for comparison.

Computational efficiency: Recall that for better efficiency, our method processes only the ROI that is large enough to contain the foreground object. This strategy makes the speed of our method dependent on the size of foreground object. In the worst case when the whole image needs to be processed, our method still can reach about 10fps for 640×480 size of inputs, in normal case it is able to reach $15 \sim 25$fps, without any code-level optimization.

Failure cases: Our method would fail when neighbor regions of foreground and background have similar colors, in which case neither of FLKDE and DGKDE can produce reliable predictions. In fact, ambiguous colors may be the greatest trouble of video segmentation, including previous methods adopting background and motion cues (background model can be regarded as a per-pixel color model [12, 17]; motion is in fact represented by the color difference of adjacent frames [5, 16]).

Fast camera movement and sudden illumination change may violate the continuity assumption and cause the transduction to fail. Fortunately, in practice these two cases occur rarely.

6. Conclusion

We have presented a method to segment live video captured by a hand-held moving camera. Since the conventional stationarity assumption does not hold for many cases, we propose to adopt the continuity assumption instead, which tolerates continuous changes for both foreground and background, thus introduces a natural way to handle non-stationary background. Our method provides the first

Figure 7. Segmentation results of six dynamic scenes.

transductive framework of real-time video segmentation. FLKDE can propagate segmentation reliably by combining local color distribution and temporal prior; DGKDE is a dynamic appearance model capable of handling occlusion; the geodesic-based segmentation method offers a very fast way to solve for the segmentation. Fig.7 shows more segmentation results extracted from accompany videos.

The limitation of our method is that it makes use of only color information, future work should account for other cues like shape. A method to evaluate the segmentation quality is also necessary if we want to detect and recover from failure cases.

Acknowledgments: This paper is supported by 973 program of china (No. 2009CB320802), Natural Science Fund for Distinguished Young Scholars of Shandong Province (No. JQ200920) and NSF of China (No. 60870003).

References

[1] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, 2007.

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

[3] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *ECCV*, 2004.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:359–374, 2004.

[5] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *Proceedings of CVPR*, pages 53–60, 2006.

[6] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In *ECCV*, 2008.

[7] J. Cui, Q. Yang, F. Wen, Q. Wu, C. Zhang, L. V. Gool, and X. Tang. Transductive object cutout. In *CVPR*, 2008.

[8] Z. Dong, L. Jiang, G. Zhang, Q. Wang, and H. Bao. Live video montage with a rotating camera. In *Pacific Graphics*, 2009.

[9] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *Proceedings of CVPR*, pages 407–414, 2005.

[10] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In *ICCV*, pages 1305–1312, 2003.

[11] B. Price, S. Cohen, and B. Morse. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009.

[12] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum. Background cut. In *Proceedings of ECCV*, pages 628–641, 2006.

[13] P. J. Toivanen. New geodesic distance transforms for gray-scale images. *Pattern Recognition Letters*, pages 437–450, 1996.

[14] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008.

[15] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *ACM SIGGRAPH*, 2005.

[16] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *Proceedings of CVPR*, 2007.

[17] F. Zhong, X. Qin, J. Chen, W. Hua, and Q. Peng. Confidence-based color modeling for online video segmentation. In *Asia Conference on Computer Vision*, 2009.