PAPER

Video Segmentation with Motion Smoothness

Chung-Lin WEN[†a)], ***Nonmember***, **Bing-Yu CHEN**[†b)], ***and*** **Yoichi SATO**[††c)], ***Members***

SUMMARY In this paper, we present an interactive and intuitive graph-cut-based video segmentation system while taking both color and motion information into consideration with a stroke-based user interface. Recently, graph-cut-based methods become prevalent for image and video segmentation. However, most of them deal with color information only and usually failed under circumstances where there are some regions in both foreground and background with similar colors. Unfortunately, it is usually hard to avoid, especially when the objects are filmed under a natural environment. To make such methods more practical to use, we propose a graph-cut-based video segmentation method based on both color and motion information, since the foreground objects and the background usually have different motion patterns. Moreover, to make the refinement mechanism easy to use, the strokes drawn by the user are propagated to the temporal-spatial video volume according to the motion information for visualization, so that the user can draw some additional strokes to refine the segmentation result in the video volume. The experiment results show that by combining both color and motion information, our system can resolve the wrong labeling due to the color similarity, even the foreground moving object is behind an occlusion object.

key words: *video segmentation, stroke propagation, graph cuts*

1. Introduction

As visual effects become a crucial part of current film and commercial television production, video segmentation, a critical step for many visual effects, arouses much interest in the film industry and research community. The so-called video segmentation is done to extract the foreground moving objects from the background. It could be used in multiple ways, e.g. to replace the actors to a different scene, or reversely replace the actors with CGI (computer-generated imagery) components. However, to extract the foreground moving object from a complex natural background is not only tedious but also extremely time-consuming.

Recently, graph-cut-based methods are prevalent for image and video segmentation, but most of them use only color similarity along with some smoothness constraints as the segmentation criteria, which may fail when some regions in both foreground and background have similar colors. For instance, Fig. 1 (b) shows a result of an ordinary video segmentation method with smaller smoothness weightings, and hence some background regions (the tree trunk) are wrongly labeled, since the color of the tree trunk is similar to that of the walking bear. In contrast, if a greater smoothness weighting is being used, some background regions (around the foot of the bear) near the foreground will be merged into the foreground regions as shown in Fig. 1 (c).

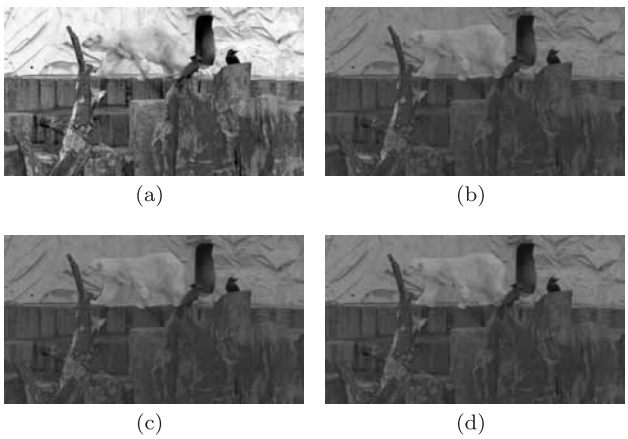


Fig. 1 The comparison of the segmentation results of a walking bear video. (a) One frame of the ordinal video. (b) Using smaller color smoothness weighting, a part of the tree trunk is wrong labeled. (c) Using greater color smoothness weighting, there are some labeling errors around the foot of the bear. (d) Our result.

Hence, in this paper, we propose a new approach to this problem. Through our observation, in most of the cases in video segmentation, the motion pattern of the foreground moving object is usually quite different from that of the background. Thus, our video segmentation method uses both color and motion information to improve the usability. Through our method, the wrong labeling due to the color similarity could be resolved, even under the condition that the moving foreground object is behind an occlusion object.

Moreover, the motion information is also used for improving the stroke-based interface. Although some previous methods allow the user to draw the strokes to indicate the foreground and background in the temporal-spatial video volume, it is not intuitive enough for the user. To let the user understand how the strokes effect the video segmentation result and where to draw some additional strokes, we utilize the motion information again to propagate the strokes drawn in some certain frames to other neighboring frames. Hence, in our system, the user can first draw the foreground and background strokes on the first or a certain frame as using previous stroke-based image or video segmentation

Manuscript received August 19, 2009.
Manuscript revised December 13, 2009.

[†]The authors are with National Taiwan University, Taipei, 10617 Taiwan.
[††]The author is with The University of Tokyo, Tokyo, 153–8505 Japan.
a) E-mail: jonathan.clwen@gmail.com
b) E-mail: robin@ntu.edu.tw
c) E-mail: ysato@iis.u-tokyo.ac.jp

DOI: 10.1587/transinf.E93.D.873

tools, then refine the video segmentation result by drawing some additional strokes on other frames while referring to the initially drawn strokes. Furthermore, since the additional strokes are used to refine the original result, the weighting of them is set to be different from the original strokes in our video segmentation algorithm.

In the rest of this paper, a brief survey of related work is introduced in Sect. 2. The details of our video segmentation algorithm and the user interface issues are described in Sect. 3 and Sect. 4, respectively. Finally, the results are shown in Sect. 5 and the conclusion and future work are listed in Sect. 6.

2. Related Work

To extract a foreground object from a natural background, there are some approaches to manually fit the boundary of the object by editable curves, such as B-spline, which is often called rotoscoping. The process is usually operated in a fully manually fashion, with a minor help of image snapping tools [1]–[3] that adhere to the high contrast region or interpolation tools [4] that interpolate the keyframe curves to the potential contours in the other frames, mainly according to the gradient.

Recently, graph-cut-based methods are prevalent for image and video segmentation, since the image and video segmentation could be treated as a multi-labeling energy minimization problem with extremely high complexity. Thus, it is natural to derive an approximate solution with guaranteed qualities. Boykov *et al.* [5] pioneered in the domain by developing the graph cuts algorithm to solve the multi-labeling energy minimization problem. Kolmogorov and Zabin [6] featured the energy function that could be minimized: the necessary conditions that could be applied in the graph cuts framework. Then, Boykov and Kolmogorov [7] presented some observations with experimental results while comparing with other alternative modern approaches.

Based on the graph cuts algorithm, Li *et al.* presented Video Object Cut and Paste [8] by extending their Lazy Snapping [9] from a graph-cut-based image segmentation method to a video volume with similar stroke-based user interface. In addition to the data term and color smoothness term used in Lazy Snapping, to eliminate the temporal artifacts, they introduced temporal coherence cost into the energy function to be minimized. Based on Lazy Snapping, Progressive Cut [10] proposed by Wang *et al.* introduces some high level observations about the user intentions into the original energy function of the graph-cut-based image segmentation method. Besides the stroke-based image segmentation methods, there are also rectangle-based ones. For instance, in GrabCut [11], the user can indicate the foreground location by using a rectangle. However, it is considered that the stroke-based system can better utilize the geometry nature of the input image.

Similar to Video Object Cut and Paste, Interactive Video Cutout [12] proposed by Wang *et al.* also uses the strokes to collect color information and conducted the video segmentation in temporal-spatial space by graph cuts algorithm. In the system, the strokes are allowed to be drawn in the temporal-spatial space directly instead of a specific frame, although to draw a stroke in the video volume is not so intuitive for ordinary users. In the above methods, since the color information is the only consideration, if the foreground objects and the background have similar colors, there would be some errors in the segmentation result. Hence, in this paper, we also take the motion information into account to handle the cases that are hard to segment solely by only color information.

There are also some video segmentation methods deal with the occlusion condition. Xiao *et al.* [13] proposed a system to conduct motion layer extraction in the presence of occlusion condition. The system mainly has two stages, the first one is to segment the seed regions using motion similarity. Then, by employing some heuristics that the occlusion area will always increase with time, the system conducts the final segmentation by graph cuts algorithm to refine the segmentation result. Although the system also uses motion information for video segmentation, our method is more general since we do not make any assumption on the type of occlusion.

Bai *et al.* [14] also presented a video segmentation method to solve the same problem by incorporating additional user input. Whenever the occlusion occurs, the user can draw additional strokes in the other side of the video volume relative to the initial strokes. However, since our video segmentation method takes the motion information into consideration, in many cases, the user is not requested to draw the additional strokes to solve the occlusion problem. Moreover, through our intuitive stroke-based user interface with propagated strokes, it is much easier to draw the additional strokes to refine the segmentation result if necessary.

3. Video Segmentation with Color & Motion

Before explaining the details of our method, we first briefly summarize a basic form of the energy function used in the graph-cut-based video segmentation methods. In most of the graph-cut-based video segmentation techniques, a 3D graph is constructed based on the temporal-spatial video volume and suitable cuts are obtained by minimizing the following energy function:

$$E(l_p) = E_d(p) + \alpha E_s(p) + \beta E_t(p), \tag{1}$$

where $l_p \in \{\mathcal{F}, \mathcal{B}\}$ is a possible labeling of pixel p, and $\mathcal{F}$ and $\mathcal{B}$ denote the foreground and background labels, respectively, and α and β are the weightings to adjust the importance of the spatial and temporal color smoothness terms $E_s(p)$ and $E_t(p)$, respectively. $E_d(p)$ is the data term defined in the same way as most of the previous work [8]–[10], which measures the color similarity between the pixel p and pre-constructed foreground and background color models. The foreground and background color models are constructed by collecting the pixels under the foreground and

background strokes drawn by the user at certain frames. $E_s(p)$ and $E_t(p)$ encode color smoothness constraints for the pixel p in the same frame and neighboring frames, respectively.

The data term $E_d(p)$ is defined as:

$$E_d(p) = \begin{cases} \dfrac{P(c_p|G_{\mathcal{F}})}{P(c_p|G_{\mathcal{F}}) + P(c_p|G_{\mathcal{B}})}, & \text{if } l_p = \mathcal{F} \\ \dfrac{P(c_p|G_{\mathcal{B}})}{P(c_p|G_{\mathcal{F}}) + P(c_p|G_{\mathcal{B}})}, & \text{if } l_p = \mathcal{B} \end{cases}$$

where c_p is the color value of the pixels p, $P(c_p|G_x)$ is the probability that a specific pixel value c_p belongs to the color model G_x ($x \in \{\mathcal{F}, \mathcal{B}\}$). We use 3D GMMs (Gaussian Mixture Models) to describe the color distribution by collecting the color values of the pixels under the foreground or background strokes.

The spatial smoothness term $E_s(p)$ is defined by imposing the color smoothness penalty in the intra-frame neighboring pixels. The smoothness penalty is reverse proportional to the color difference, so that the energy minimization procedure would prefer to assign the same label to the pixels that have smaller color difference, which is defined as:

$$E_s(p) = \sum_{q \in N_p^s} |l_p - l_q| \cdot g(\|c_p - c_q\|^2),$$

where N_p^s denotes the spatial neighboring ($|N_p^s| = 8$) pixels of the pixel p in the same frame, and $g(x) = 1/x + 1$ is used to implement inverse relation.

In addition, video segmentation is different from image segmentation in that the former one should also consider the problem of temporal coherence. Otherwise, it will introduce serious temporal artifacts, which people are more sensitive to. Hence, besides the intra-frame color coherence, we should also encode the temporal coherence by adding the inter-frame arcs and imposing the penalty for temporal incoherence. The temporal coherence penalty also follows the principle that pixels have similar color should be labeled as the same value, thus is also set to the reverse proportion to the color difference as:

$$E_t(p) = \sum_{q \in N_p^t} |l_p - l_q| \cdot g(\|c_p - c_q\|^2),$$

where N_p^t denotes the temporal neighboring ($|N_p^t| = 2$) pixels of the pixel p in the neighboring frames. After encoding the temporal coherence, the result is free from most of the temporal artifacts.

3.1 The Motion Smoothness Term

Although the above energy function works well in certain cases, there are still several failed cases when some of the foreground and background regions have similar colors. The failed cases may be solved by adjusting the weightings α and β carefully, but it is a tedious task and suitable weightings are usually difficult to find. To make the system more easy to use, we must enlarge the suitable range for parameter tuning. Hence, we find another criteria that is different in foreground and background. It can be observed that the motion patterns in the video are usually different for foreground and background objects, thus we propose to use the motion information as the other segmentation criteria and encode the motion smoothness term $E_m(p)$ in Eq. 1, so Eq. 1 can be rewritten as:

$$E(l_p) = E_d(p) + \alpha E_s(p) + \beta E_t(p) + \gamma E_m(p). \quad (2)$$

Similar to color smoothness terms $E_s(p)$ and $E_t(p)$, we prefer to label the pixels that have similar motion patterns with the same value, thus we impose motion smoothness penalty for the pixels that are different in motion direction. Hence, the motion smoothness term $E_m(p)$ is defined as:

$$E_m(p) = \sum_{q \in N_p^s} |l_p - l_q| \cdot g(v_p \cdot v_q),$$

where $v_p \cdot v_q$ is the inner product of the motion vectors v_p and v_q of the pixels p and q.

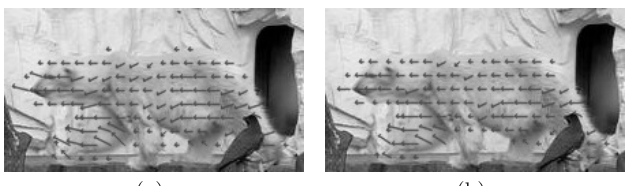
To make the motion vectors v_p and v_q as stable as possible, a revised optical flow algorithm proposed by Brox *et al.* [15] is used, since it can produce relatively denser optical flow information, which is a critical condition for the calculation of the motion smoothness cost. However, since the raw optical flow still has some noises, we further conduct color-guided weighting average in the fashion similar to [16]:

$$v_p = \frac{\sum_{q \in \{N_p^s + N_p^t + p\}} v_q \cdot w(p,q)}{\sum_{q \in \{N_p^s + N_p^t + p\}} w(p,q)}, \quad (3)$$

where the weighting $w(p,q)$ is defined in an inverse proportion to the color difference as:

$$w(p,q) = 1/(\|c_p - c_q\|^2 + \epsilon),$$

where ϵ is a small value for avoiding the division by zero, and the L^2-norm color difference is calculated in RGB color space. Figure 2 shows the motion information before and after the color-guided refinement. Even if there is a pixel p has no motion information, Eq. 3 can also be used to obtain the motion information from its neighboring pixels, which improve the robustness of our system.



(a) (b)

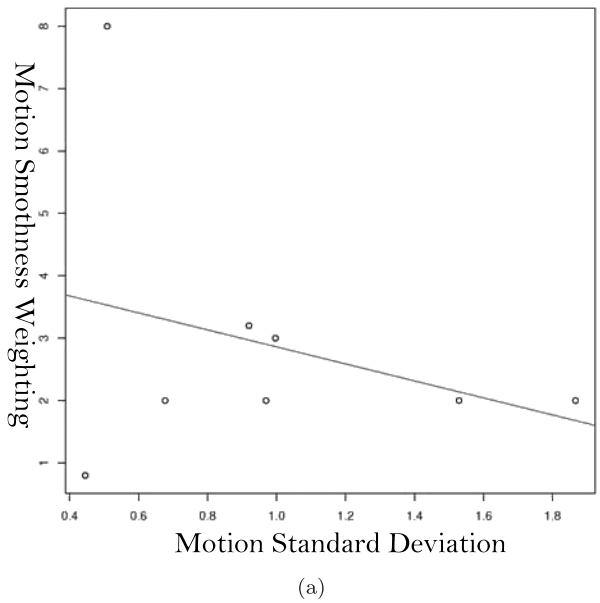
Fig. 2 The comparison of the optical flow before and after the color-guided refinement. (a) Before the refinement, there are some noises in the boundary region. (b) After the refinement, some noises are removed.

Table 1 The statistics of standard deviation and weightings.

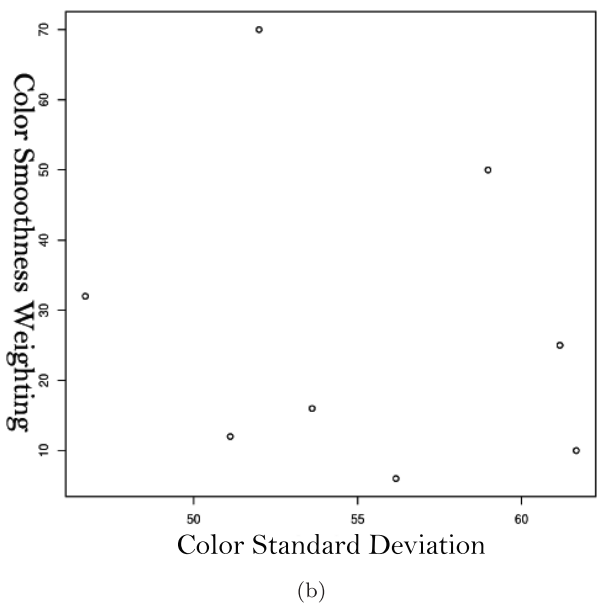
filename	image sd	motion sd	color weighting	motion weighting
walking bear	61.170	0.445	25.0	0.8
moving keyboard	51.114	0.677	12.0	12.0
running car	56.168	1.867	6.0	2.0
sitting toy	56.717	0.924	32.0	2.0
swimming fish	61.669	0.509	10.0	8.0

3.2 The Weightings

About the weightings of the smoothness terms α, β, and γ, although the user can leverage his or her high level knowledge to control whether the data term or each smoothness criteria should be prioritized, to make the system easier to use, it is better to provide some guides for users to control the process of parameter tuning rather than adjusting them in a try-and-error fashion. In our experience, the temporal coherence (β) does not cause too much difference, instead, it is more critical to keep a good balance between the color (α) and motion (γ) information.



(a)



(b)

Fig. 3 (a) The plot of the standard deviations of motion with their corresponding weightings (i.e. γ). (b) The plot of the standard deviations of color with their corresponding weightings (i.e. α).

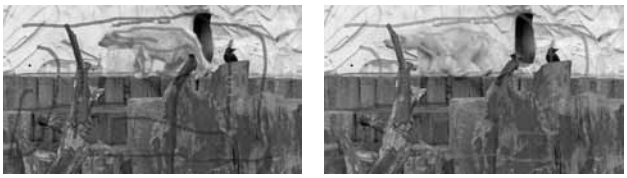
To check the relationship of the weightings with the color and motion information, we provide statistics-based guess for the weightings and plot the standard deviation of the color and motion with their corresponding weightings α and γ in Fig. 3 (b) and Fig. 3 (a), respectively. Besides, values of parameters and standard variations are summarized in Table 1. For motion (as shown in Fig. 3 (a)), the standard deviation is roughly reversely proportional to the weighting γ, while there is no such a clear relation for color (as shown in Fig. 3 (b)). The ratio of color and motion is also roughly proportional between their standard deviations and weightings. However, although we have the above mentioned observation, more test data and their corresponding parameters is needed to derive a qualitative rules for a statistical guess, which may be one of our future work.

4. User Interface

4.1 Propagated Strokes

To make the system easy to use, we provide a stroke-based user interface like other similar systems. However, to ask the user to draw the strokes in the video volume like [8] is not very intuitive. On the other hand, to let the user only draw the strokes on certain frames makes him or her hardly to draw some additional strokes on other frames if he or she wants to modify the current segmentation result. Hence, rather than asking the user to draw the strokes in the video volume, we propagate the strokes drawn by the user to indicate the foreground and background regions on a certain frame to other neighboring frames according to the motion information with light colors to make him or her to understand how the strokes effect the segmentation result and where to draw the additional strokes to improve the result as shown in Fig. 4.

As a fail-safe feature, the weighting of the propagated strokes is in reverse proportion to the color difference between the pixel value under the original location and that of the propagated location. The mechanism can prevent the interference of inaccurate propagation. In addition, as



(a) (b)

Fig. 4 The strokes drawn by the user are automatically propagated by motion information. (a) The strokes drawn on a certain frame. (b) The strokes automatically propagated to the neighboring frames by motion information for visualization with light colors.

(a) (b)

Fig. 5 The walking bear is occluded by the tree trunk. (a) Without motion smoothness, it is hard to deal with the occlusion. (b) Combined the motion smoothness with the color smoothness, the result improves much.

Fig. 4 (b) shows, to make the impact of each stroke can be easily understood by the user, our system shows the propagated strokes with different alpha values, to indicate the different weightings they possess.

4.2 Local Refinement by Additional Strokes

Due to the local color nature, it is usually the case that there may be some regions that are hard to segment right in the first time. Hence, we provide the facility to the user to refine the result like [10]. In mathematical form, we encode the constraint that the probability of the label change decreases with the distance by adding a user term $E_u(p)$ to Eq. 2 as:

$$E(l_p) = E_d(p) + \alpha E_s(p) + \beta E_t(p) + \gamma E_m(p) + \kappa E_u(p),$$

where $E_u(p)$ is defined as:

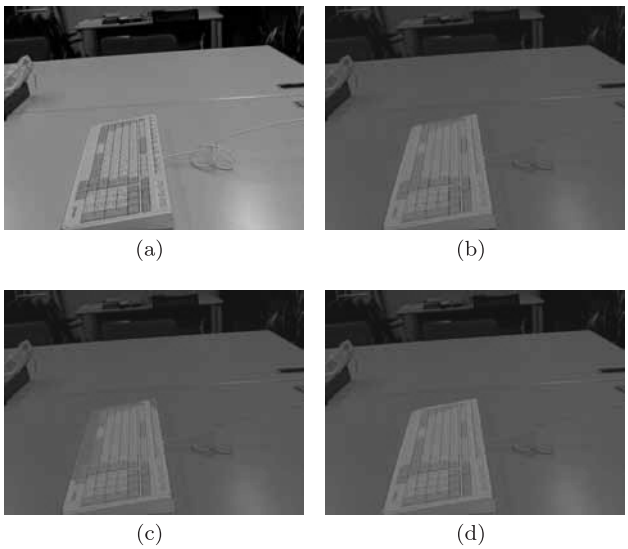
$$E_u(p) = |l_p - l'_p| \frac{\text{argmin}_s \|x_p - x_s\|}{r},$$

where $|l_p - l'_p|$ is an indicator of label change from the original label l'_p to the new label l_p of the pixel p, $\|x_p - x_s\|$ is the distance between the current pixel p to the pixel s under the strokes drawn by the user, x_p and x_s denote the positions of the pixels p and s, and r is a user-defined parameter to control the range of the user attention. By encoding the constraint in the observations, the system successfully prevents the over-expansion and over-shrinkage problems at once.

5. Results

In this section, we show some experimental results to compare with previous methods. To visualize the results, red color is used to blend with the foreground and blue color is used to blend with the background. Note that in the results presented in this section, strokes are drawn on the first frame only, which is a quite challenging task for previous methods.

Figure 1 shows a comparison of the segmentation results of a walking bear video. Figure 1 (b) shows the result with smaller color smoothness weightings α and β and Fig. 1 (c) shows that with larger ones by using Eq. 1 of [8] and [12]. In contrast, with the motion smoothness term shown in Eq. 2 we proposed, the result improves much as shown in Fig. 1 (d). Due to the benefits of the motion smoothness term, even the walking bear is occluded by the tree trunk as shown in Fig. 5, we can still obtain an acceptable result as shown in Fig. 5 (b).



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

Fig. 6 The comparison of the segmentation results of a moving keyboard video. In this video, the white keyboard is dragged on a white table. (a) One frame of the original video. (b) Using smaller color smoothness weighting, there are some errors between the keyboard buttons. (c) Using greater color smoothness weighting, there are some foreground regions merged into the background. (d) Our result.

Figures 6∼9 show other comparisons of our method and previous ones by some different videos. In Fig. 6, Fig. 6 (b) shows a sequence that has some wrongly labeled background between the keystrokes, because the dark color between the keystrokes is similar to some of the background colors. Of course, one may raise the weighting of the color smoothness term to solve the problem, but since there is a strong data term penalty, the weighting of the color smoothness term should be raised to a rather huge amount to completely fix the problem. Before that, the overly strong color smoothness term introduces other artifacts. As shown in Fig. 6 (c), some portions of the foreground, i.e. keyboard has been wrongly labeled to the background, since the colors of the table and the keyboard are similar. By only adjusting the color smoothness term, they have the tendency to be labeled as the same value. In contrast, if the motion information could be utilized, it would be relatively easy to segment the keyboard with the background as shown in Fig. 6 (d).

Similar to the above case, Fig. 7 (b) shows another ex-

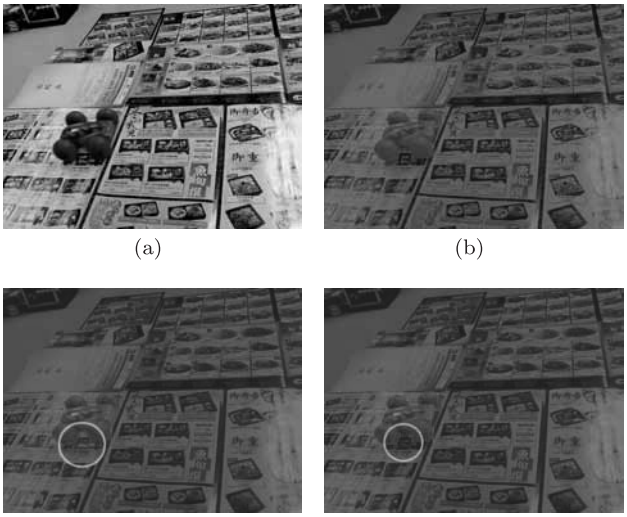


Fig. 7 The comparison of the segmentation results of a running car video. In this video, the car is running on several colorful leaflets. Noted that yellow circles are imposed to show the area that has major difference. (a) One frame of the original video. (b) Using smaller color smoothness weighting, there are some errors on the leaflets. (c) Using greater color smoothness weighting, a part of the leaflets is labeled as the foreground. (d) Our result.

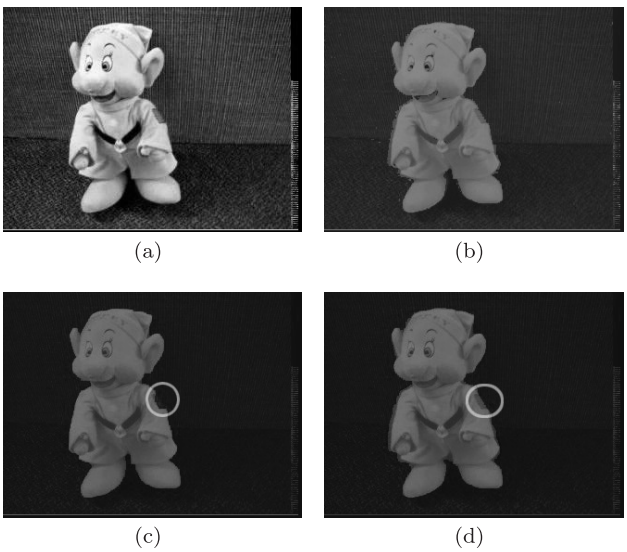


Fig. 8 The comparison of the segmentation results of a sitting toy video. In this video, the toy is static but the camera moves. Noted that yellow circles are imposed to show the area that has major difference. (a) One frame of the original video. (b) Using smaller color smoothness weighting; some parts of the chair are labeled as the foreground. (c) Using greater color smoothness weighting, a part of the toy is labeled as the background. (d) Our result.

ample that the segmentation could not be done with color information alone, since the background color is too complex and there are similar color between the car and the background colorful leaflets. As the background has some regions similar to the foreground, it is usually the case that the regions would be wrongly labeled as the foreground. If

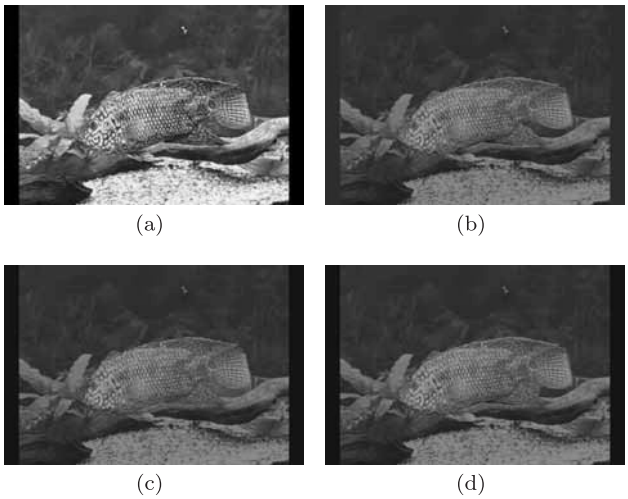


Fig. 9 The comparison of the segmentation results of a swimming fish video. (a) One frame of the original video. (b) Using smaller color smoothness weighting, there are some errors in both the foreground and background. (c) Using greater color smoothness weighting, a part of the fish is merged into the background. (d) Our result.

the weighting of the color smoothness term is raised to solve the problem, it introduces other artifacts. As the foreground, i.e. the toy car moves to the right, the background regions that have similar color to the toy car would be "merged" as the foreground, by using the preference with greater weighting of the color smoothness term, as shown in Fig. 7 (c). In this video, during the car running, the background colorful leaflets also move to several different directions since the leaflets are not fixed on the table. Since the motion patterns are quite different for the foreground and background, even some foreground and background regions have similar colors and the background is also unstable, we can still obtain an acceptable result as shown in Fig. 7 (d).

In Fig. 8, the toy is sitting on a wicker chair, the environment is static but the camera is moving. Although the difference of the motion patterns between the foreground and background is not very much, the result is still improved much by integrating the motion smoothness term. Figure 8 (b) shows that with small color smoothness weighting, the regions with similar dark color as the background, such as eye, belt and mouth, would be wrongly labeled as the background. In contrast, the region on the left hand (in the right side of the image) in some frames is "merged" into the background as shown in Fig. 8 (c). Figure 9 (b) shows that with small color smoothness weighting, in the condition that the black side of the video has not been drawn with the background stroke, it would be labeled as the foreground, since its color is similar with the foreground color model. However, if the system uses large color smoothness weighting, some artifacts may be occurred, such as the result shown in Fig. 9 (c), where some of the regions on the head of the fish have been "merged" into the background. In contrast, our results are shown in Fig. 8 (d) and Fig. 9 (d).

The ratios of correctly segmented pixels for each result are summarized in Tabel 2 by comparing the results of our

Table 2 The ratio of correctly segmented pixel.

filename	proposed method	previous method
walking bear	0.971	0.963
moving keyboard	0.984	0.972
running car	0.987	0.974
sitting toy	0.946	0.922
swimming fish	0.965	0.907

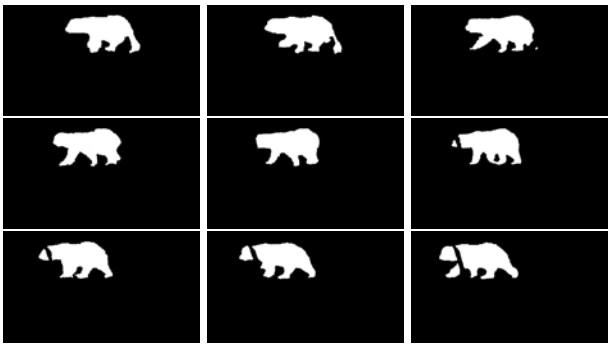


Fig. 10 Several frames of the manually labeled ground truth of walking bear.

Table 3 The statistics of running time and dimensions.

video	dimension	running time (sec.)
walking bear	480×270	77.190
moving keyboard	480×360	84.831
running car	480×360	77.033
sitting toy	324×244	86.005
swimming fish	504×336	104.728

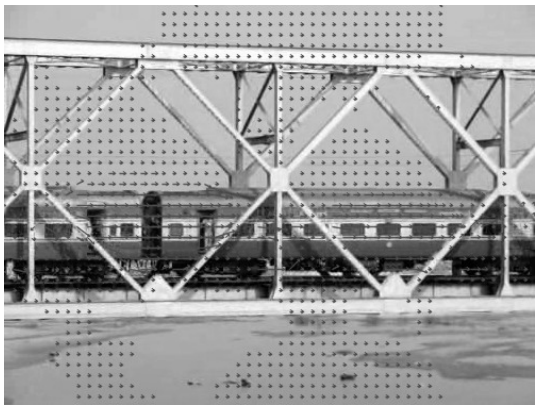


Fig. 11 An instance that hard to use motion information for segmentation: shot with unstable or irregular camera motion.

Fig. 12 Another instance that hard to use motion information for segmentation: complex motion inside the foreground object.

and previous methods to the manually labeled ground truth as shown in Fig. 10. Although the differences are moderate, by observing the figures, we can infer that the differences are appeared in the regions that attract more user attention, such as the border between foreground and background.

Note that the system is not aim at solving partial alpha. However, as long as a relatively accurate segmentation could be achieved, it is straight forward to dilate the region in the boundary binary segmentation to produce the trimap, which is usually required by most matting processes. Hence, by employing the state-of-the-art matting techniques, such as Coherent Matting [17], we could also have the result of video matting.

The system is implemented in C++, compiled under GNU G++ 4.3 with maximal optimize level, and tested under a Linux environment (Ubuntu 8.04) with a laptop PC that equipped with a 3.2 GHz CPU and 3 GB RAM. The computation costs about 1 or 2 minutes for a video with 20 frames, which does not include the calculation of the optical flow and its refinement. Since the content of the video will not be changed during the video segmentation process, the optical flow calculation and its refinement are performed as an offline pre-processing. Table 3 lists the details of the performance and dimension information for each test video. For these cases, we should note that the user time is considerably small, since there are only few strokes drawn on the first frame only.

Although the motion information works considerably well in the clips that shot with a fixed camera, for the clips that shot with a non-fixed camera, it becomes inaccurate to use even with the above mentioned weighted-averaging.

Figure 11 shows an example that is shot with an unstable camera. The irregular of the camera produces noisy optical flow information, which is difficult to be used since it has no clear separation between the foreground and background. While actually there does have such separation in the motion pattern from the train to the background.

It is also hard for complex motion. Figure 12 shows such a case. As the fish move upward as a non-rigid body, there has quite complex motion patterns between each part of its body, which not only produces some noise during the optical flow calculation process, but also encodes motion smoothness cost to prevent the body to be labeled as a whole.

Besides, whenever the disocclusion occurs, in the first few frames, the foreground may be wrongly labeled as the background, since the region of the disocclusion regions is rather small. In this situation, for a comparably high smoothness term, which maybe necessary in global, may cause the smoothness cost, rather than the data cost to dominate the energy function minimization as shown in Fig. 13.

Fig. 13 The system may fail in the first few frames after the disocclusion occurs.

6. Conclusion and Future Work

A new video segmentation method is proposed in this paper. The video segmentation problem is encoded into a 3D temporal-spatial graph that can be solved by the graph cuts algorithm. Besides the traditional spatial color smoothness and temporal coherence, we also encode the motion smoothness into the formulation. In many cases, the system with motion smoothness outperforms the traditional one which takes only color information into account, even if the foreground moving object is occluded or the difference of the foreground and background movement is considerably small. Besides this, we also provide a propagated-stroke-based user interface, which allows the user to modify the video segmentation result interactively and intuitively.

Although the motion information works well in the video clips shot with a fixed camera or the moving camera with stable movement, for the video clips that shot with an unstable camera or by the camera with irregular camera motion, the motion information becomes inaccurate to use, since the irregular motion of the camera produces noisy optical flow information, which is difficult to use for separating the foreground and background. To stabilize the video clips by using methods such as [16], [18] before segmentation may be a possible solution to the problem. It is also difficult if the foreground object has complex or irregular motion patterns, such like a rapid swimming goldfish. We could try to derive more sophisticated optical flow average algorithm to improve the quality of the motion information.

By integrating the motion information into the video segmentation, we can deal with the occlusion situation such as the case shown in Fig. 5. However, first few frames of the disocclusion might fail, since the comparably small disoccluded region may be smoothed by the color smoothness term, while the motion information is not accurate enough to produce a correct preference for motion segmentation. Finding a more elaborated mechanism for temporal-spatial-adapted parameter to deal with such a complicated case would be part of our future work.

We are also considering to investigate the user behavior for the user to draw the strokes by machine learning approaches. If we could mimic the pattern ordinary user draw the strokes. A much accurate fully automatic image/video segmentation framework may be proposed. Finally, it would be interesting to find features other than color and motion to separate the foreground and background.

Acknowledgement

This paper was partially supported by National Science Council of Taiwan under NSC97-2622-E-002-010, and also by the Excellent Research Projects of the National Taiwan University under NTU97R0062-04.

References

[1] A. Blake and M. Isard, Active Contours, Springer, 1998.
[2] M. Gleicher, “Image snapping,” Proc. ACM SIGGRAPH 1995 Conference, pp.183–190, 1995.
[3] E.N. Mortensen and W.A. Barrett, “Intelligent scissors for image composition,” Proc. ACM SIGGRAPH 1995 Conference, pp.191–198, 1995.
[4] T. Mitsunaga, T. Yokoyama, and T. Totsuka, “Autokey: Human assisted key extraction,” Proc. ACM SIGGRAPH 1995 Conference, pp.265–272, 1995.
[5] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” IEEE Trans. Pattern Anal. Mach. Intell., vol.23, no.11, pp.1222–1239, 2001.
[6] V. Kolmogorov and R. Zabin, “What energy functions can be minimized via graph cuts?,” IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.2, pp.147–159, 2004.
[7] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.9, pp.1124–1137, 2004.
[8] Y. Li, J. Sun, and H.Y. Shum, “Video object cut and paste,” ACM Trans. Graphics, vol.24, no.3, pp.595–600, 2005 (Proc. SIGGRAPH 2005 Conference).
[9] Y. Li, J. Sun, C.K. Tang, and H.Y. Shum, “Lazy snapping,” ACM Trans. Graphics, vol.23, no.3, pp.303–308, 2004 (Proc. SIGGRAPH 2004 Conference).
[10] C. Wang, Q. Yang, M. Chen, X. Tang, and Z. Ye, “Progressive cut,” Proc. ACM Multimedia 2006 Conference, pp.251–260, 2006.
[11] C. Rother, V. Kolmogorov, and A. Blake, “‘grabcut’: Interactive foreground extraction using iterated graph cuts,” ACM Trans. Graphics, vol.23, no.3, pp.309–314, 2004 (Proc. SIGGRAPH 2004 Conference).
[12] J. Wang, P. Bhat, R.A. Colburn, M. Agrawala, and M.F. Cohen, “Interactive video cutout,” ACM Trans. Graphics, vol.24, no.3, pp.585–594, 2005 (Proc. SIGGRAPH 2004 Conference).
[13] J. Xiao and M. Shah, “Motion layer extraction in the presence of occlusion using graph cuts,” IEEE Trans. Pattern Anal. Mach. Intell., vol.27, no.10, pp.1644–1659, 2005.
[14] X. Bai and G. Sapiro, “A geodesic framework for fast interactive image and video segmentation and matting,” Proc. 2007 IEEE International Conference on Computer Vision, pp.1–8, 2007.
[15] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” Proc. 2004 European Conference on Computer Vision, pp.25–36, 2004.
[16] B.Y. Chen, K.Y. Lee, W.T. Huang, and J.S. Lin, “Capturing intention-based full-frame video stabilization,” Comput. Graph. Forum, vol.27, no.7, pp.1805–1814, 2008 (Proc. Pacific Graphics 2008 Conference).
[17] H.Y. Shum, J. Sun, Y. Li, and C.K. Tang, “Pop-up light field: An interactive image-based modeling and rendering system,” ACM Trans. Graphics, vol.23, no.2, pp.143–162, 2004.
[18] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.Y. Shum, “Full-frame video stabilization with motion inpainting,” IEEE Trans. Pat-

tern Anal. Mach. Intell., vol.28, no.7, pp.1150–1163, 2006.

Chung-Lin Wen received the B.S. degree in Information Management and the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, in 2006 and 2009, respectively. He was a visiting graduate student to The University of Tokyo from 2008 to 2009. His research interests include computer graphics, computer vision, computational photography and human-computer interaction.

Bing-Yu Chen received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and received the Ph.D. degree in Information Science from The University of Tokyo, Japan, in 2003. He is currently an associate professor jointly affiliated with the Department of Information Management, Department of Computer Science and Information Engineering, and Graduate Institute of Networking and Multimedia, of the National Taiwan University, and is a visiting associate professor in the Department of Computer Science of The University of Tokyo. His research interests are mainly for computer graphics, geometric modeling, image and video processing, and human-computer interaction. He is a member of ACM, ACM SIGGRAPH, Eurographics, IEEE, and IICM.

Yoichi Sato is an associate professor jointly affiliated with the Graduate School of Interdisciplinary Information Studies, and the Institute of Industrial Science, at the University of Tokyo, Japan. He received the BSE degree from the University of Tokyo in 1990, and the M.S. and Ph.D. degrees in robotics from the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1993 and 1997 respectively. His research interests include physics-based vision, reflectance analysis, image-based modeling and rendering, tracking and gesture analysis, and computer vision for HCI.