Structured-Cut: A Max-Margin Feature Selection Framework for Video Segmentation

Nikhil S. Naikal*
Berkeley EECS

Abstract

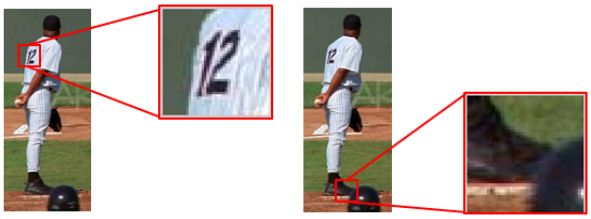
Segmenting a user-specified foreground object in video sequences has received considerable attention over the past decade. State-of-the-art methods propose the use of multiple cues other than color in order to discriminate foreground from background. These multiple features are combined within a graph-cut optimization framework and segmentation is predominantly performed on a frame by frame basis. An important problem that arises is the relative weighting of each cue before optimizing the energy function. In this paper, I address the problem of determining the weights of each feature for a given video sequence. More specifically, the implicitly validated segmentation at each frame is used to learn the feature weights that reproduce that segmentation using structured learning. These weights are propagated to the subsequent frame and used to obtain its segmentation. This process is iterated over the entire video sequence. The effectiveness of Structured-Cut is qualitatively demonstrated on sample images and video sequences.

Keywords: Segmentation, matting, feature weighting.

1 Introduction

Segmenting foreground objects has become an essential component in many video applications. It is necessary for a number of tasks including video editing and after effects for object removal, object deletion, layered compositions, etc. It is also useful for computer vision applications such as object recognition, 3D reconstruction from video, and compression. In the past, industry heavily relied on manual rotoscoping, and to this date there still is a need for an effective, easy-to-use video segmentation tool. This need remains due to the surprising difficulty of the problem. Video segmentation shares the difficulties of image segmentation, such as overlapping color distributions, weak edges, complex textures, and compression artifacts. While user-strokes based image segmentation has been well understood, the process of propagating user scribble specifications to successive video frames is a challenging problem.

These challenges arise because natural video generally contains several erratic changes that are hard to model and compute. For instance, large camera movement, motion blur, and occlusions can cause a lack of object overlap between successive frames. Illumination changes and shadows can alter the color distributions making the foreground indistinguishable from the background. Further, non-rigid motion of objects in 3D space can lead to confusion in precisely tracking the contour of the object in the 2D image projections. A given video sequence can easily exhibit many of these challenges. While a single cue might be insufficient, systematically combining multiple cues might be more efficient at separating foreground objects from background in video.



(a) (b)

Figure 1: *Pitcher's shirt can be separated from background wall (a) using color model, but separating his black shoe from a background player's helmet (b) requires other cues like motion, texture and blur.*

Many different kinds of features are generally observed in successive video frames to aid object selection. Such features include color, adjacent color relationships, texture, blur, shape, spatiotemporal coherence, etc. The relative importance of the features differs depending on the particular video sequence, the frame, and even the location within the frame. For example, in Fig 1.a. a simple color model can be used to distinguish the baseball player from the background wall, but in Fig 1. b, a different feature such as texture or blur needs to be used to discriminate the pitcher's shoe from another player's helmet. An algorithm that intelligently applies all of these cues based on specific circumstances will perform better than one relying only on a subset of these cues or on a static combination of all of them.

2 Related Work

Many approaches have been taken in interactive video segmentation. Some approaches focus on either boundary or region information only. Agarwala et al. [1] performs boundary tracking using splines that follow object boundaries between keyframes using both boundary color and shape-preserving terms. Bai and Sapiro [3] use region color to compute a geodesic distance to each pixel to form a selection. These approaches perform well when a single type of cue is sufficient for selecting the desired object. Many current techniques use graph cut to segment the video as a spatiotemporal volume. Graph cut, as formulated in [4], solves for a segmentation by minimizing an energy function over a combination of both region and boundary terms. It has been shown to be effective in the segmentation of images [5, 6] and volumes [2].

Boykov and Jolly [4] introduced a basic approach to segmenting video as a spatiotemporal volume. Their graph connects pixels in a volume, which implicitly includes spatiotemporal coherence information. Graph cut is applied using a region term based on a color model of the pixels under the user strokes and a boundary term based on gradient. Wang et al. [8] builds on this approach

*e-mail: nnaikal@eecs.berkeley.edu

by allowing users to segment video by drawing strokes on arbitrary slices of the spatiotemporal volume. While this permits a user to mark several frames at once, it requires a steep learning curve to know how to carve the volume so that the right pixels are visible along the slice. The method uses a global color model based on the user strokes as well as a local color model for static backgrounds in addition to gradient values.

In Li et al. [7], users segment every tenth frame, and graph cut computes the selection between the frames using global color models from the key-frames, gradient, and coherence as its primary cues. The user may also manually indicate areas to which local color models are applied. While this method performs well, it requires the manual segmentation of many frames in addition to corrections. In methods where the video is treated as a spatiotemporal volume [2, 3, 4, 8], the only information known for certain about the object and background are in the user-marked pixels. This provides very limited knowledge about the object interior and no knowledge about the boundary. While [7] is an exception to this, it requires the user to manually segment many frames.

The approach that closest resembles my method is *Video SnapCut* proposed by Bai et. al. [9]. They propose that multiple cues should be used for extracting the foreground, such as color, texture, shape and motion. Among these, shape plays an important role in their method. Further, they evaluate multiple cues both locally and globally, rather than just globally to maximize discriminant powers. Inspired by these principals, they propose a video segmentation model of overlapping localized classifiers which contains features that include color, shape and motion. However, their adaptive integration of these features is based on some naive assumptions that generally break down in complicated video sequences. For instance, they highly weight the shape feature which can cause an overfit and deteriorate the tracking performance when there are large topological changes in the object's shape.

2.1 Overview

Similar to Video SnapCut, I present a foreground object segmentation approach based on overlapping localized classifiers. It consists of a group of overlapping windows around the foreground object boundary, each associated with a local classifier which only segments a local piece of the foreground boundary. Assuming that the foreground object does not undergo any significant motion, the spatial locations of these classifiers is preserved across the subsequent frame. The segmentation of this new frame is achieved by aggregating local classification results together. Furthermore, each local classifier carries local image features that includes two color models, three texture models and one blur model. The weights of these features are adaptively learned using structured prediction, with the positive learning example provided by the implicitly validated current segmentation. This process of segmenting and then learning weights is then iterated over the entire video sequence. In Section 3, I present the framework for segmenting with localized classifiers. Each classifier includes multiple models for separating foreground from background, details of which are presented in Section 4. Section 5 presents the structured learning method for feature weighting. The method is tested on multiple images and a video sequence as presented in section 6. I conclude and discuss the approach in section 7.

3 Video Segmentation Framework

Given an input video sequence, the segmentation process starts by having the user provide a relatively accurate mask for the desired object on the first (key) frame, using image-based object selection approaches. I have implemented a simple GUI that is similar to

Figure 2: *GrabCut based GUI for selecting foreground mask. User draws a box around the foreground object to obtain initial segmentation. Refinement is done via user-scribbles with red representing foreground and blue representing background.*

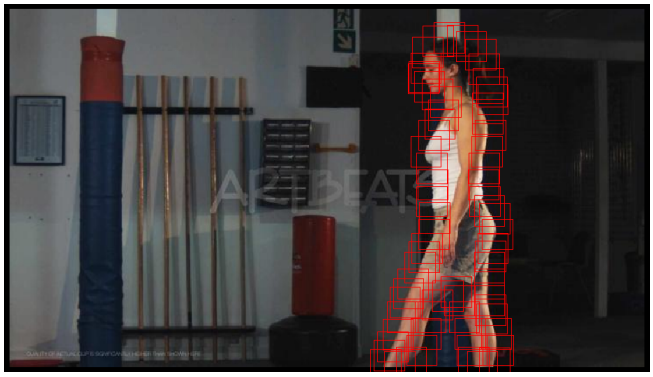


Figure 3: *The red boxes represent the overlapping local classifiers along the foreground boundary.*

Grab Cut to achieve this task, as shown in Fig.2. It only differs from GrabCut in that I use color histogram models instead of Gaussian Mixture Models (GMMs) as histogram computation is faster than Expectation Maximization (EM). Once the initial mask is created, a group of local classifiers are constructed around the foreground boundary, which are then propagated onto successive frames to segment the object. In this section I describe how the classifiers are initialized and propagated to the next frame for segmentation.

3.1 Local Classifiers for Segmentation

As shown in Fig. 3, given the initial mask L^j for the j'th keyframe I^j, I uniformly sample a set of overlapping windows ${W_1}^j \ldots {W_n}^j$ along its contour. The method is general enough to handle multiple contours but for now we assume a single contour exists around the foreground object. The size of the windows can vary according to the size of the object, and it is usually 15x15 to 40x40 pixels. Each window defines the application range of a local classifier, and the classifier will assign to every pixel inside the window a foreground (object) probability, based on the local statistics it gathers. Neighboring windows overlap for about 1/3rd of the window size to allow for topological changes in the object's contour in subsequent frames. Each classifier inside the window ${W_k}^j$ consists of two local color models M^{c_1}, M^{c_2}, three texture models $M^{t_1}, M^{t_2}, M^{t_3}$, and a blur model M^b each of which are explained in detail in Section 4.

It is well known that such segmentation problems can be formulated using a Markov Random Field (MRF) framework. This framework is typically used with a single unary and a single pairwise term as

shown below.

$$\mathbf{E}(\mathbf{s}) = \sum_i \Psi_u(x_i) + \lambda \sum_{i,j \in N} \Psi_p(x_i, x_j), \qquad (1)$$

where Ψ_u, Ψ_p are the unary and pairwise potentials, x_i is the i'th image pixel, and $\mathbf{s}$ is the segmentation. Given separable models and sufficient weighting λ for the pairwise potential, minimizing the energy function $\mathbf{E}(\mathbf{s})$ will produce the desired foreground/background segmentation. Thus, foreground and background sub-models need to be constructed for each local window.

Since each classifier is centered on a boundary pixel, the local window will contain both foreground and background pixels. These are used to construct foreground and background sub-models for each feature type mentioned above. For instance, the foreground sub-model for the first color model is represented by $M^{c_1}(\mathcal{F})$, and the corresponding background model is given by $M^{c_1}(\mathcal{B})$. For a pixel x in the window, its foreground probability generated from the first color model is computed as:

$$p^{c_1}(x) = p^{c_1}(x|\mathcal{F}) / (p^{c_1}(x|\mathcal{F}) + p^{c_1}(x|\mathcal{B})), \qquad (2)$$

where $p^{c_1}(x|\mathcal{F})$ and $p^{c_1}(x|\mathcal{B})$ are the corresponding probabilities computed from the first foreground and background color sub-models. Similarly, foreground and background pixel probabilities are computed for all the other feature models. These probabilities from the generative models are used to construct unary potentials in the MRF framework (1). The pairwise potentials for each feature type are constructed using a weighted smoothness function that is presented in section 4.

By dropping the pixel variable x and with slight abuse of notation, the segmentation energy function (1) can be augmented with these multiple unary and pairwise potentials.

$$\mathbf{E}(\mathbf{s}) = \sum_i \mathbf{\Psi}_u + \sum_{i,j \in N} \mathbf{\Psi}_p = \mathbf{w}^{\mathbf{T}} \mathbf{\Theta}(\mathbf{s}), \qquad (3)$$

where $\mathbf{w}$ is the relative feature wighting for the composite unary and pairwise potentials, $\mathbf{\Psi}_u$ and $\mathbf{\Psi}_p$ respectively which are grouped into $\mathbf{\Theta}(\mathbf{s})$. These terms are expanded in what follows:

$$\begin{array}{rcl} \mathbf{\Psi}_u & = & \lambda^{c_1} {\Psi_u}^{c_1}(x_i) + \lambda^{c_2} {\Psi_u}^{c_2}(x_i) + \ldots + \lambda^{b} {\Psi_u}^{b}(x_i), \\ \mathbf{\Psi}_p & = & \mu^{c_1} {\Psi_p}^{c_1}(x_i, x_j) + \mu^{c_2} {\Psi_p}^{c_2}(x_i, x_j) + \\ & \ldots & + \mu^{b} {\Psi_p}^{b}(x_i, x_j), \\ \mathbf{w} & = & [\lambda^{c_1}, \lambda^{c_2}, \ldots \lambda^{b}, \mu^{c_1}, \mu^{c_2}, \ldots \mu^{b}]^{\mathbf{T}}. \end{array}$$

Since the energy function (3) is still sub-modular and linear in the combination of multiple unary and pairwise potentials, it can be minimized using standard graph-cuts. Since window's overlap, a few pixels are segmented multiple times. The final labeling decision on whether such a pixel belong to foreground or background is taken by counting the number of times the pixel was assigned the associated label. If this foreground label count is higher than the background label count, then the pixel is assigned a foreground label, and vice-versa.

4 Multiple Features

The use of multiple features can help in discriminating foreground pixels from the background more accurately. The segmentation framework presented in the previous section allows for multiple features, and is not restrictive in the number of features used. I propose the use of six features to construct unary and pairwise potentials. These features are explained in what follows:

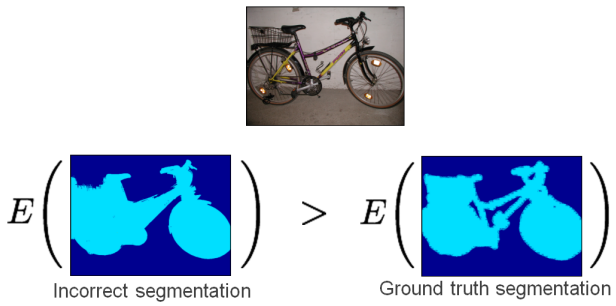


Figure 4: *Energy of ground truth segmentation is lower than energy of all incorrect segmentations.*

4.1 Color

I have used two color models. The first model is mixture of Gaussians (GMM) for foreground and background respectively. I have used only 3 Gaussians for each window as the local windows have a small number of pixels and this number was empirically found to be sufficient. The associated pairwise term for the color is given by

$${\Psi_p}^{c_1}(x_i, x_j) = \mathbf{exp}(-\beta^{c_1} \sum_r \|x_i(r) - x_j(r)\|), \qquad (4)$$

where β is empirically set and r represents the number of color channels.

The second model is a color histogram model. In *Lab* space I oversegment the local window and construct histogram models for foreground and background. For any given pixel in the window, the probability of belonging to a foreground or background is based on the χ^2 distance of the pixel's local color histogram from the corresponding histogram models. The pairwise histogram potential is also constructed using an equation similar to (4).

4.2 Texture

I construct three texture models for each window. The texture response of the local image window is found using three texton filter banks namely: LM, RFS and S filter banks. These texture responses are then used to construct associated GMM's for generating the texture unary potentials. The three pairwise texture potentials are derived again for neighboring pixels using an equation similar to (4).

4.3 Blur

To generate the blur unary and pairwise potentials I use the defocus map based approach presented in [ref:defocus magnification].

5 Feature Weighting via Structured Learning

Structured learning[10] has become very popular in cases where there isn't a single class label for each training instance, but instead a set of labels. If such labels are independent, then a simple multi-class SVM can be used for each label, but a more complex case occurs when the elements of the output vector are dependent. The binary segmentation problem I have described so far falls into this category.

As presented in Section 3.1, for any local window W_k, the corresponding user specified segmentation mask $\mathbf{s}_k$ acts as the single training instance. Since the associated energy for the k'th window is minimized with the correct segmentation, the minimizer

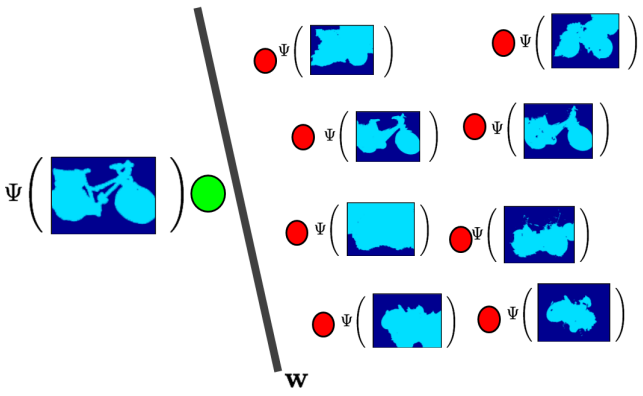


Figure 5: *The max-margin framework can be used to learn the feature weights the separate the ground truth segmentation from multiple incorrect instances.*

$\mathbf{s}^* = \arg\max_s \mathbf{E}(\mathbf{s})$ should be equal to the user specified segmentation, i.e, $\mathbf{s}^* = \mathbf{s}_k$. Thus, given this ground-truth segmentation, the constraint on all incorrect segmentations can be given by,

$$\mathbf{E}(\mathbf{s}_k) < \mathbf{E}(\mathbf{s}_{incorrect}) \Rightarrow \mathbf{w}^\mathbf{T}\mathbf{\Theta}(\mathbf{s}_k) < \mathbf{w}^\mathbf{T}\mathbf{\Theta}(\mathbf{s}_{incorrect}), \quad (5)$$

as seen in Fig. 4.

Thus, we need to learn weights $\mathbf{w}$ that generate at least as low an energy as that generated by the label configuration in the training example, $\mathbf{s}_k$. However, the inequalities in (5) may have multiple or no solutions. This is resolved by finding the parameters that satisfy the inequality with the largest possible energy margin γ, so that the ground truth labeling has the lowest energy relative to other labelings. This max-margin concept serves to regularize the problem and provide generalization to unseen test data. The margin may be negative if the original inequality has no solutions. Thus the solution to the optimization problem,

$$\begin{aligned} &\max_{w} \quad \gamma \\ s.t. \quad &\mathbf{w}^\mathbf{T}(\mathbf{\Theta}(\mathbf{s}_k) \quad - \quad \mathbf{\Theta}(\mathbf{s})) \geq \gamma \quad \forall \mathbf{s}, \end{aligned} \quad (6)$$

is the necessary weighting needed to separate the ground truth segmentation from all incorrect segmentations as seen in Fig. 5.

6 Experiments

In order to validate the feature learning scheme, I began by testing the method on the two image windows presented in Fig. 1. For the window in Fig 6. a., since the color of the background pixels was uniquely different from the foreground color, it was reasonable to expect the learning framework to give high weights to the color models. I reduced the number of features in this case to one GMM based color model, the LM texture model and the blur model. With these unary and pairwise terms, the weights learned for the window are presented in Fig 6. b.. As can be seen in the figure, high weighting is given to the color unary term as expected.

Now, for the window in Fig 7.a., the foreground shoe of the pitcher is very hard to discriminate from the background helmet of a different player. Thus, color by itself was not sufficient for segmentation. The weights learned by the algorithm are presented in Fig 7. b. As

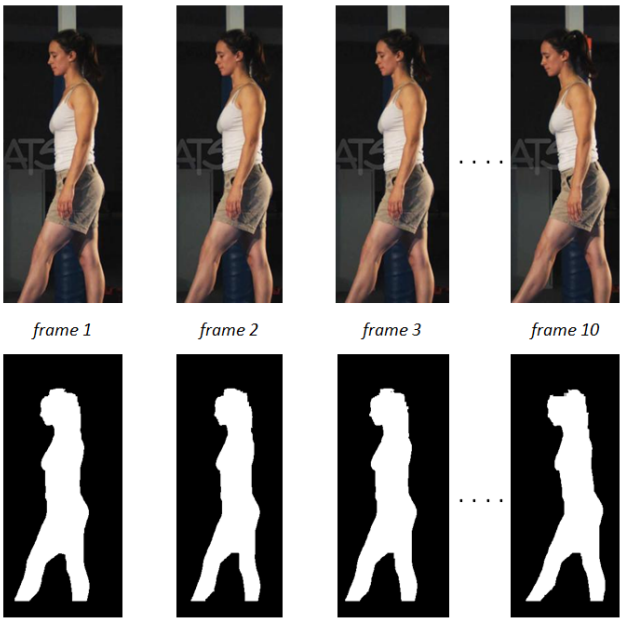


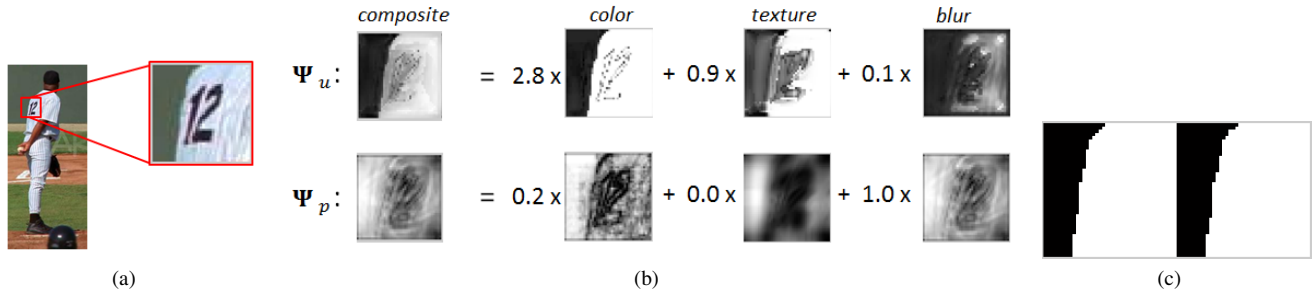
Figure 8: *Segmentation of the first 10 frames of the gymnast sequence with the first frame segmented by user using my GrabCut implementation.*

can be seen in this figure, a negative weight was learned for the unary blur potential as the model was very bad at discrimination. This is contrary to any existing feature weighting scheme that can not provide appropriate negative weighting for bad models. The combined unary and pairwise potentials for both cases were minimized using graph-cuts. I have used the max-flow implementation by Boykov [4]. The ground truth segmentation and the segmentation on the composite unary and pairwise potential are juxtaposed in Figs. 6. c. and 7. c. It is clear that the two segmentations are qualitatively very close.

Extending to Video: Before explaining the extension to video, I start by presenting the assumptions. (1) In any given frame, the size of the local windows is set to be large enough that they fully encompass the object in the next frame, (2) The foreground and background models learned in one frame can be propagated to the next frame because the image statistics do not change drastically in successive frames, and (3) Once the boundary pixels are determined from segmentation, the pixels within the contour are simply filled and considered to belong to foreground. Although they seem restrictive, these 3 assumptions hold in a large number of natural videos. This method of learning the weights in the current frame and propagating to the next frame is iterated over the first 10 frames of the gymnast sequence as presented in Fig. 8. It can be seen that the mask quite accurately covers the gymnast even in hard to distinguish regions near her hair and shorts.

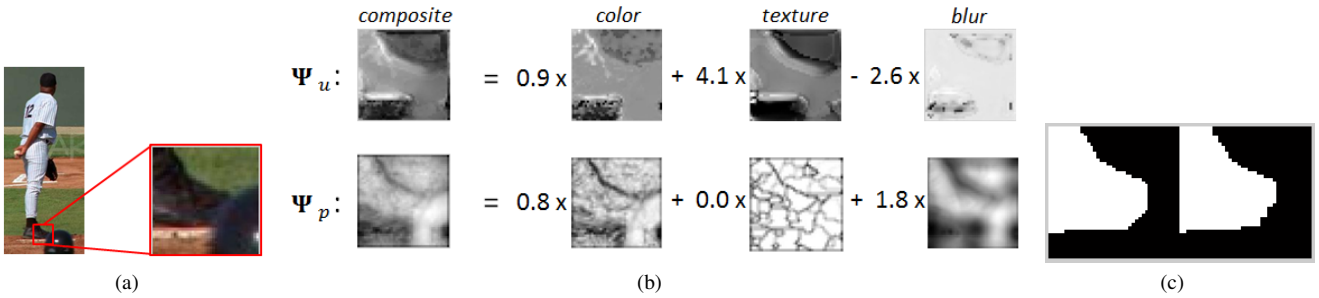
7 Conclusion and Future Work

In this paper I have presented a scheme for capturing user specifications for foreground object segmentation in generic video sequences. A user specifies the foreground mask in the first frame of the video using an interactive tool similar to GrabCut. The algorithm then determines the right combination of feature weights needed to reproduce the same segmentation using multiple features other than color. Since foreground/background models do not drastically change across successive frames, the weights learned in one frame can be used to infer the segmentation in the next frame. This



(a) (b) (c)

Figure 6: *(b)Feature weights and composite unary and pairwise potentials for window in (a). (c)* left: *Ground truth segmentation;* right: *Segmentation obtained by minimizing composite unary and pairwise potentials with graph-cuts.*



(a) (b) (c)

Figure 7: *(b)Feature weights and composite unary and pairwise potentials for window in (a). (c)* left: *Ground truth segmentation;* right: *Segmentation obtained by minimizing composite unary and pairwise potentials with graph-cuts.*

process of segment-then-learn is iterated over the entire video sequence and has shown promise to be the basic algorithm to improve foreground object segmentation.

In the future, I plan to improve the quality of the segmentation by incorporating a contour tracker, and a shape feature that I suspect will drastically improve the results. Currently the algorithm runs at an average rate of 50 seconds per frame with a combined Matlab/C++ interface. I plan to speed this up by porting local window computations to parallel processors of a GPU. Finally, I plan to incorporate a matting algorithm such as the Baye's matting approach to cutout foreground objects from challenging videos and composite them on other backgrounds. This would fully demonstrate the capabilities of the multiple feature selection scheme for accurate foreground object segmentation in video sequences.

References

[1] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. SIGGRAPH, 23(3):584591, 2004.

[2] C. Armstrong, B. Price, and W. Barrett. Interactive segmentation of image volumes with live surface. Computers and Graphics, 31(2), April 2007.

[3] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. ICCV, pages 18, 2007.

[4] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In IEEE ICCV, pages 105112, 2001.

[5] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In ACM SIGGRAPH 2004, pages 303308, 2004.

[6] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. In ACM SIGGRAPH 2004, pages 309314, 2004.

[7] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. ACM Trans. Graph., 24(3):595600, 2005.

[8] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. ACM Trans. Graph., 24(3):585594, 2005.

[9] Xue Bai, Jue Wang, David Simons and Guillermo Sapiro. Video SnapCut: robust video object cutout using localized classifiers. In ACM SIGGRAPH 2009.

[10] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. In Proceedings international conference on Machine learning, 2004.