Automatic Fence Segmentation in Videos of Dynamic Scenes

Renjiao Yi[1,2], Jue Wang[3], and Ping Tan[1]

[1]Simon Fraser University
[2]National University of Defence Technology
[3]Adobe Research

Abstract

We present a fully automatic approach to detect and segment fence-like occluders from a video clip. Unlike previous approaches that usually assume either static scenes or cameras, our method is capable of handling both dynamic scenes and moving cameras. Under a bottom-up framework, it first clusters pixels into coherent groups using color and motion features. These pixel groups are then analyzed in a fully connected graph, and labeled as either fence or non-fence using graph-cut optimization. Finally, we solve a dense Conditional Random Filed (CRF) constructed from multiple frames to enhance both spatial accuracy and temporal coherence of the segmentation. Once segmented, one can use existing hole-filling methods to generate a fence-free output. Extensive evaluation suggests that our method outperforms previous automatic and interactive approaches on complex examples captured by mobile devices.

1. Introduction

It is a common case that one has to shoot an interesting scene through fences or wires. For instance, capturing a video of a walking tiger behind an enclosing fence in a zoo, or a building through wires or tree branches. Such videos are usually unpleasant to watch due to the strong distraction caused by the occluders. A common photography trick to alleviate this problem is to adjust the focus length and aperture of the camera to make the fence out-of-focus, thus less distracting when watching the video. However its effectiveness is limited and is only applicable to relatively advanced cameras, excluding most mobile phone cameras. Removing fence from videos at the postprocessing stage is thus highly desirable.

Despite a few recent attempts [22, 14, 7], removing fence from videos with unconstrained scene dynamics and camera movement is largely an open problem. In particular, it is hard to automatically detect and segment fence in videos. Fences contain very thin structures, which are difficult to segment even for interactive tools such as GrabCut[19] or Rotobrush[1]. Furthermore, there is usually no distinctive colors or strong textures on a fence, making it hard to track, Their repetitive structure patterns often lead to tracking and motion estimation errors. A recent work [21] successfully removes fence from videos, but only for static scenes. For videos capturing dynamic scenes, the commonly used two-layer motion model breaks out due to the existence of large dynamic objects, rendering methods that rely on static scene reconstruction insufficient, as we will show in the experimental section.

In this paper, we present a new method for automatic fence segmentation from casual videos capturing dynamic scenes or objects. By allowing dynamic scenes, our approach has a much wider application range than previous work that are constrained to static ones. Our approach can also deal with videos shot with a moving camera, which is quite common for novice users capturing with hand-held mobile devices. We show that, while introducing object and camera motion brings new challenges to the task, they in turn provide additional information that can facilitate fence detection and segmentation. Specifically, the camera motion gives the fence a rigid motion in the video that is usually quite distinctive from the object motion behind it, allowing better segmentation using local motion contrast.

Our method takes a bottom-up approach. It begins by computing optical flow between neighboring frames, and grouping pixels in each frame according to color and motion. In the first round, we treat each group as a super-pixel and consider labeling each one as either fence or non-fence. Each group's probability of being fence is evaluated according to its structural and appearance features. The compatibility between two neighboring groups are computed from their color, motion, and structural similarities. We then solve a graph-cut optimization to produce initial labeling. The initial labeling, done on a per-frame basis, suffers from imprecise fence localization and poor temporal coherence.

It is further refined by a spatio-temporal dense Conditional Random Field (CRF) optimization[9], which improves fence segmentation in both spatial accuracy and temporal coherence.

We evaluate the proposed approach on various videos, including mobile phone videos captured by ourselves, and Youtube video clips with completely unknown camera setting. Our segmentation results are quantitatively evaluated on a new dataset with manually labeled ground truth. The results show that our method achieves much better precision and recall than previous approaches. Finally, we demonstrate simple hole-filling with existing inpainting techniques [5] to remove detected fences.

2. Related work

Hays *et al*. [6, 13] detect fence structures from a single image by extracting near regular repetitive texture patterns. Park *et al*. [17] enhance the repetitive structure detection to deal with deformations due to perspective camera projection and non-planar underlying shapes. Online learning and classification are adopted to further enhance the detection [16]. Generally speaking, these methods rely on the success of the challenging task of repetitive structure detection, which is difficult to handle certain types of fence structures such as window blinds. Although our method uses image gradients to measure local fence structure compatibility, it does not explicitly assume any particular fence pattern.

Fence detection and removal can be easier when multiple input images or a video clip is available. Yamashita *et al*. [22] use flash and non-flash images together with multi-focus images to detect and remove fence. Khasare *et al*. [7] manually label fence pixels with existing interactive segmentation tools. Mu *et al*. [14] detect and remove fence using parallax cues from video clips under the assumption of a *static* scene. Xue *et al*. [21] separate fence from the background using motion cues through an optimization process. This approach achieves high quality results, but is limited to static scenes.

Image inpainting [3] [5] [2] techniques can fill-in small image regions given their masks. Video inpainting [15][20] can recover missing structures on the current frame by transferring pixels from neighboring frames. The success of these methods rely on accurate segmentation masks as input, which are hard to achieve for fences even with advanced interactive segmentation tools [10, 19, 1]. Our segmentation approach provides such masks automatically.

The video compass work[8] used histograms to describe orientations of lines, which is similar to our gradient-based term in initial fence segmentation described in later sections.

In the spirit of creating an enhanced image from a video clip, our work is relevant to TrackCam[12] and super-resolution [18], while we target on a completely different problem.

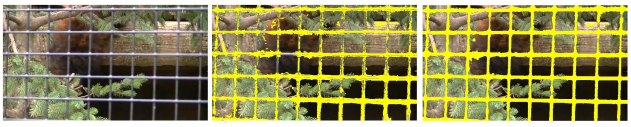


Figure 1. (a) One frame in input video; (b) initial fence segmentation by graph-cut; (c) final fence segmentation by dense CRF.

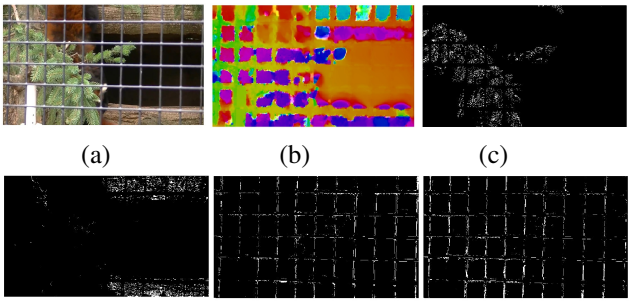
3. Fence segmentation

Our fence segmentation includes three major steps. Firstly, pixels in each frame are clustered into a fixed number of groups based on color and motion information. Secondly, each of these groups is labeled as fence or non-fence by a graph-cut optimization applied to each video frame individually. Finally, a dense condition random field (CRF) is optimized over all frames simultaneously to label each pixel as fence or non-fence to improve the temporal coherence and spatial accuracy of fence segmentation. As an example, the fence segmentation results after per-frame graph-cut and multi-frame CRF is shown in Figure 1 (b) and (c) respectively, where the input frame is in Figure 1 (a).

3.1. Pixel Grouping

Fences have distinctive structural features, *e.g*. they typically (but not necessarily) have two sets of thin wires pointing at two nearly perpendicular directions. This inspires us to form pixel groups to exploit spatial structural features for fence detection. We apply K-means clustering to pixels at each frame according to color and motion information. This clustering is based on the observation that fences pixels often have similar colors, and distinctive motion from the background due to their short distances to the camera. Even in dynamic scenes, the moving objects in background tend to have quite different motion from the fence.

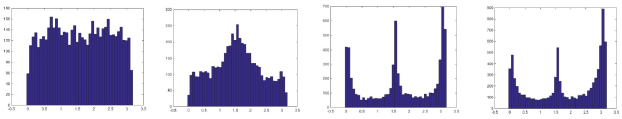
We apply the optical flow algorithm in [11] to compute local motion between neighboring frames. One example of computed flow field is showed in Figure 2 (b). The flow vectors in each frame are normalized by subtracting the minimum value and then divided by their value range (*i.e*. the difference between the maximum and minimum values). For each pixel, we concatenate its RGB color (in $[0, 1]$) and the normalized flow vector to form a 5D feature. K-means is applied to generate 50 groups for each frame: examples are shown in Figure 2 (c) – (f). Typically, fence pixels and background pixels are separated into different groups due to their difference in either color or motion. In the following, we seek to identify fence pixel groups according to fence structural features.



(a) (b) (c)

(d) (e) (f)

Figure 2. (a) input frame; (b) optical flow filed; (c)–(f) some representative pixel groups. Note that fence and background pixels are largely separated into different groups due to color and/or motion difference.



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

Figure 3. (a)–(b) gradient orientation histograms of two background clusters (see Figure 2 (c) and (d)); (c)–(d) gradient orientation histograms of two fence clusters (see Figure 2 (e) and (f)).

3.2. Initial Fence Segmentation

On each frame, we form a fully-connected graph where each pixel group is a vertex. We optimize a fence or non-fence label at each vertex by graph-cut, which minimizes the following objective function:

$$E = \sum_i D(c_i, l_i) + \sum_{(i,j)} S(c_i, l_i; c_j, l_j). \quad (1)$$

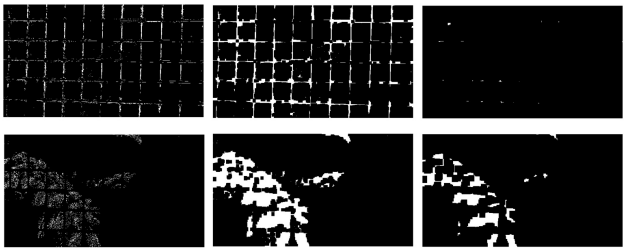
Here, c_i, c_j indicates the i-th and j-th pixel group, l_i, l_j are the binary fence labels on c_i, c_j respectively. The data term $D(\cdot)$ measures the probability of a pixel group being fence, define as:

$$D(c_i, l_i) = l_i \cdot (1 - P(c_i)) + (1 - l_i) \cdot P(c_i), \quad (2)$$

where $P(c_i)$ is the probability that c_i being fence. It includes a gradient-based term and a geometry-based term:

$$P(c_i) = (1 - D_1(c_i)) \cdot (1 - D_2(c_i)). \quad (3)$$

The gradient-based term $D_1(\cdot)$ exploits the fact that fences typically contain two sets of nearly perpendicular wires. We build a gradient orientation histogram for all pixels in a group. The histogram of a fence group should have two dominant peaks in two nearly perpendicular orientations. In contrast, a non-fence group tends to have a flat histogram. Some example are showed in Figure 3, where Figure 3 (a), (b) and (c), (d) are histograms of non-fence and fence groups, respectively. Their corresponding pixel groups are shown in Figure 2 (c), (d) and (e), (f), respectively. To exploit this observation, for each histogram, we firstly search the global highest peak c, and then search another local peak in an interval centered at $c+\pi/2$ with width $\pi/5$. We then take the histogram value at the middle point of these two peaks. For fence groups, this middle point is often associated with a low histogram value, *e.g.* in the valley between two histogram peaks in Figure 3 (c)–(d). D_1 is computed as the ratio of the histogram value at the middle point over that at the two peaks. Sometimes, the occluder contains multiple wires of similar orientations, which leads to a single dominant peak in the gradient orientation histogram. Our definition of $D_1(\cdot)$ can deal with such cases.



(a) (b) (c)

Figure 4. A fence and non-fence pixel group after (a) initial K-means grouping, (b) 'close operator', and (c) erosion.

The geometry-based term D_2 exploits the fact that fences are usually thin structures. A morphological erosion should remove most of pixels in a fence group. In contrast, a non-fence group usually has many more remaining pixels after this operation. To be robust to noisy grouping results, we first apply a 'close operator' to connect nearby isolated pixels. Figure 4 (a), (b), and (c) show results for a fence and non-fence group by initial K-means grouping, 'close operator', and erosion respectively, where morphological masks are 10×10. D_2 is computed as the percentage of pixels remained after the erosion. Both D_1 and D_2 are then linearly normalized to $[0, 1]$.

The smoothness term $S(\cdot;\cdot)$ in Equation (1) measures similarities between pixel groups based on their color, gradients orientation histogram, and dominant gradient orientations (the two histogram peaks selected when evaluating D_1). It is defined as:

$$S(c_i, l_i; c_j, l_j) = \mu(l_i, l_j) \cdot (1 - S_1(c_i, c_j)) \cdot (1 - S_2(c_i, c_j)) \cdot (1 - S_3(c_i, c_j)). \quad (4)$$

Here, $\mu(l_i, l_j)$ is the Pott model: 1 when $l_i \neq l_j$, and 0 otherwise. S_1 is the L_1 color histogram distance of two groups, computed in ab channels only in Lab space in order to be roust to illumination variations. S_2 is the L_1 distance between two gradient orientation histograms. S_3 is the difference of dominant gradient orientations. Suppose

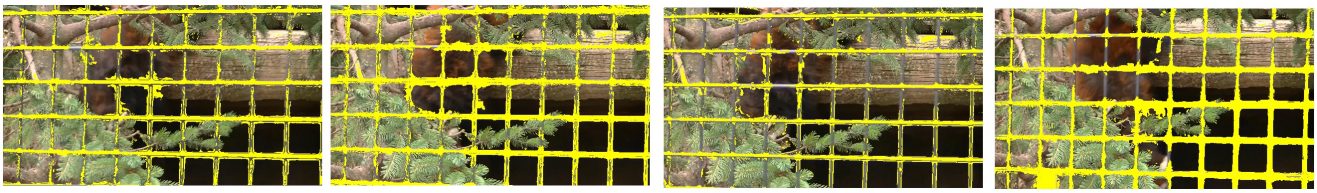


Figure 5. Initial segmentation by graph-cut optimization.

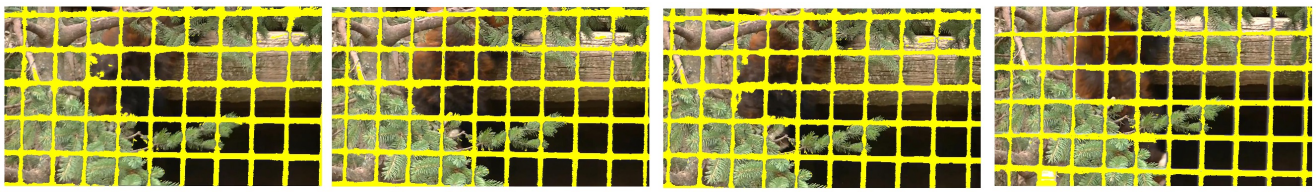


Figure 6. Fence segmentation by the multi-frame dense CRF optimization on same frames in Figure 5.

$g_1(\cdot), g_2(\cdot)$ are the two dominant gradient orientations of a pixel group, we measure S_3 as:

$$\min(dis_{g_1}, \pi - dis_{g_1}) + \min(dis_{g_2}, \pi - dis_{g_2}).$$

Here, $\min(dis_{g_1}, \pi - dis_{g_1})$ and $\min(dis_{g_2}, \pi - dis_{g_2})$ compute the closest peak in c_j to the first and second peaks in c_i respectively. Specifically, we compute them as the following:

$$dis_{g_1} = \min\left\{|g_1(c_i) - g_1(c_j)|, |g_1(c_i) - g_2(c_j)|\right\}, \quad (5)$$

$$dis_{g_2} = \min\left\{|g_2(c_i) - g_1(c_j)|, |g_2(c_i) - g_2(c_j)|\right\}. \quad (6)$$

S_1, S_2, and S_3 are all linearly normalized to be in $[0, 1]$.

We use graph-cut [4] to solve for a fence or non-fence label at each group. Some results are showed in Figure 5. Note that fence segmentation at this stage is roughly correct but inaccurate, *i.e.* fence boundaries do not align well with image edges. There are also occasional frames with poor segmentation results. This is because K-means clustering fails to produce correct low-level clustering results for frames with very little motion. Next, we build a dense CRF over all video frames to further improve the segmentation result.

3.3. Spatio-temporal Segmentation Refinement

In our dense CRF, each pixel on each frame is a vertex, and it connects to all other vertices. This spatio-temporal graph construction gives us a chance to enhance both temporal coherence and spatial accuracy of the segmentation. The total energy is defined in the same way as Equation (1) with data and smoothness terms defined differently. The data term is defined as:

$$\mathbb{D}(x, l_x) = l_x \cdot (1 - \mathbb{P}(x)) + (1 - l_x) \cdot \mathbb{P}(x).$$

where $\mathbb{P}(x)$ is evaluated as:

$$\mathbb{P}(x) = \mathbb{P}_1(x) \cdot \mathbb{P}_2(x). \quad (7)$$

Here, the term $\mathbb{P}_1(x)$ encourages the result from CRF optimization to be consistent with the initial graph-cut labeling result, defined as:

$$\mathbb{P}_1(x) = \begin{cases} 1 - \alpha, & L_0(x) = 0 \\ \alpha, & L_0(x) = 1 \end{cases} \quad (8)$$

where α is a parameter determining the confidence of initial graph-cut segmentation. In our system we simply use a constant probability at 0.8, although one could further make it adaptive according to the features of each pixel group. $L_0(x)$ is the initial label of pixel x, which is 1 or 0 for fence and non-fence pixels, respectively. The term $\mathbb{P}_2(x)$ is defined as

$$\mathbb{P}_2(x) = P(c_i), \qquad x \in c_i. \quad (9)$$

Here, $P(c_i)$ is the probability evaluated in Equation (3) in the previous step. $x \in c_i$ means that pixel x is in the i-th group.

The smoothness term $\mathbb{S}$ ensures similar pixels to have similar label. It is defined as:

$$\mathbb{S}(x, l_x; y, l_y) = \mu(l_x, l_y) \cdot k(x, y). \quad (10)$$

Here, μ is again the Pott model. Following [9], the similarity function $k(x, y)$ is defined as:

$$k(x, y) = w_1 exp\left(-\frac{|Dis(x,y)|}{2\theta_1^2} - \frac{|I_x - I_y|}{2\theta_2^2}\right) + w_2 exp\left(-\frac{|Dis(x,y)|}{2\theta_3^2}\right). \quad (11)$$

which describes the similarities between x and y in their spatial position and color. Here, I_x, I_y are the RGB colors at x, y. In all our experiments,we set $\theta_1 = 6, \theta_2 = 2, \theta_3 = 1.7$, the weights of two kernels are $w_1 = 10, w_2 = 3$.

The color difference between two pixels is computed as the L_1 distance between two color vectors. The spatial difference $Dis(x, y)$ for pixels in different frames requires some special handling. For a pixel x in the t_x-th frame and

Figure 7. Rotobrush[1] results on two examples. From left to right: manually-labeled keyframe; results after propagating 5 frames; 10 frames; 15 frames and 20 frames. The segmentation results deteriorate quickly in the temporal propagation process.

Figure 8. Alpha mattes (bottom) extracted by the method proposed in [21] on some examples (top) in our dataset.

a pixel y in the t_y-th frame, we use optical flow to track x to the frame t_y. The spatial distance is then evaluated as:

$$Dis(x, y) = \left| x + m_{t_x \to t_y}(x) - y \right|. \quad (12)$$

where $m_{t_x \to t_y}$ is the motion of pixel x from frame t_x to the frame t_y. This motion vector is obtained by concatenating optical flow vectors from adjacent frames.

Once the graph is constructed, we used the method proposed in [9] to minimize the total energy. Solving the multi-frame CRF enforces temporal coherence. If a pixel is temporally connected to pixels in other frames that have high fence probabilities, optimizing this CRF will help correct its label even its original fence probability is low. Figure 6 shows some fence segmentation results after dense CRF optimization. Comparing with the initial segmentation shown in Figure 5, the refined segmentation is more accurate on individual frames, and also maintains better temporal coherence.

4. Experiments

The dataset. We evaluate our method on a dataset of 18 video clips. Seven of them (the first seven data shown in Figure 9) were downloaded from Youtube. The following three (the eighth to tenth shown in Figure 9) are from [21]. The rest were captured by ourselves with a mobile phone. Nine of these videos contain moving objects of various sizes. The example "Blue Fence" captures a dynamic scene, and "Running Lion" captures a dynamic scene with large perspective distortion. All videos except "Jaguar" are captured with a moving camera. Our dataset also includes two examples that contain non-fence occluders: "Wire" and "Tree branch", to test the robustness and generalization of each method.

Figure 9 shows some representative frames and their final fence segmentation results. For each example, we show two sample frames, where the segmentation results are overlaid on the input frame. The initial and final segmentation results on the full video clips are provided in the supplementary material. The results show that our method generates accurate and temporally coherent segmentation for most examples.

Evaluation and Comparison. In order to quantitatively evaluate the segmentation result, for each video sequence, we manually label "ground truth" segmentation on evenly-distributed ten keyframes. We compare our method with the Rotobrush [1] video segmentation tool in Adobe After Effect, and the recent method proposed in [21] [1]. Rotobrush is an interactive segmentation tool that needs an manually segmented keyframe as additional input. We thus manually segment the first frame, and use Rotobrush to propagate this segmentation to the next 20 frames for comparison. We lim-

[1] The authors of [21] have kindly generated the alpha matte on one frame for each of our input video. The evaluation of their method is based on that given frame.

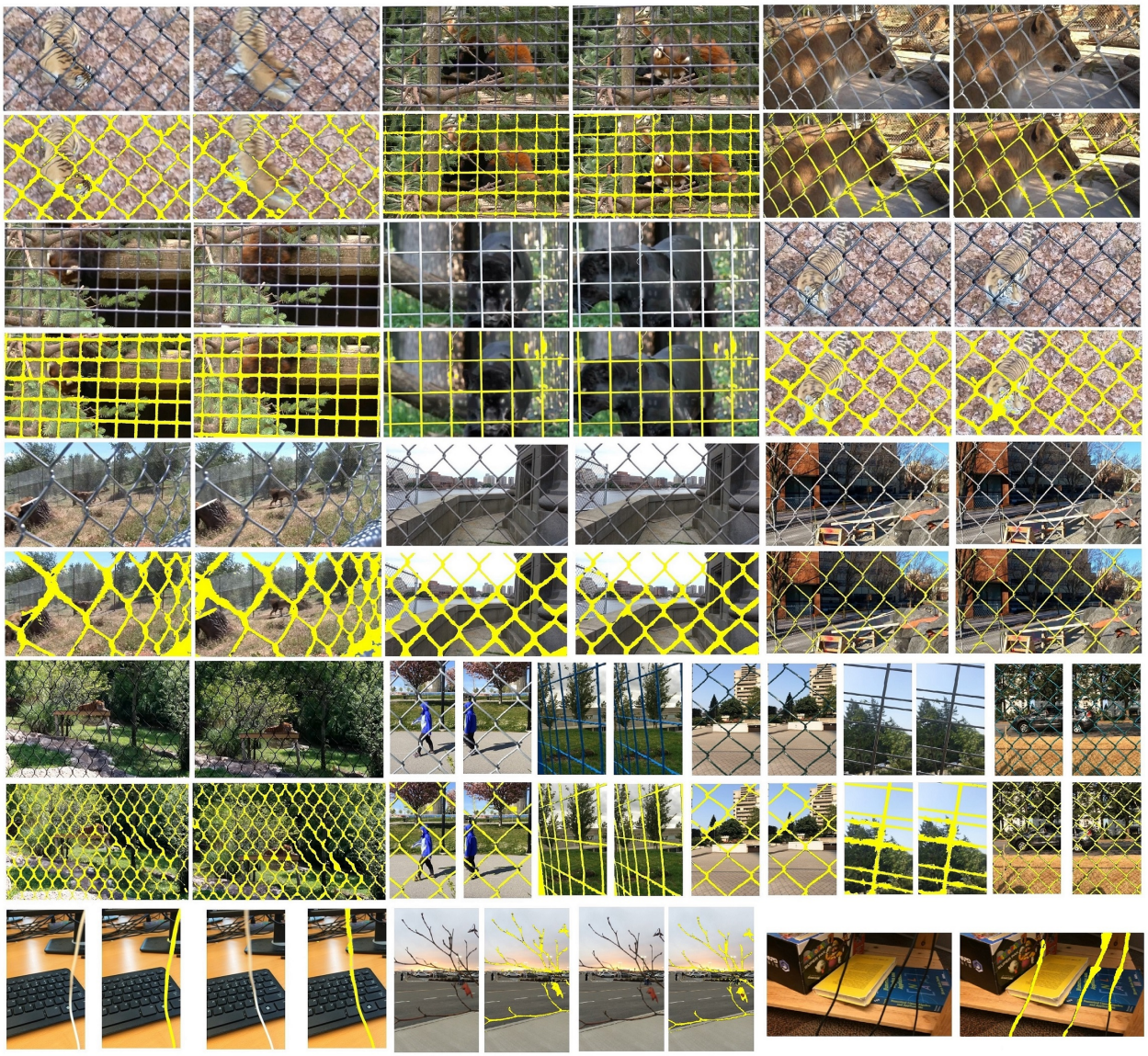


Figure 9. More fence segmentation results. For each example, we show two frames with the fence segmentation overlaid. Please refer to our supplementary video for results on the complete video clips.

it the propagation to 20 frames, because after that the results are severely deteriorated. Figure 7 shows two examples of the manually-segmented keyframes and the automatically segmented results of Rotobrush.

The precision and recall of three methods are shown in Table 1. To demonstrate the effectiveness of the CRF-based refinement, we also compare the initial segmentation computed by graph-cut optimization with the final result produced by the CRF refinement. The results show that our method in general outperforms previous approaches: the average precision and recall for our method are 80.92% and 82.31%, respectively, which are significantly higher than those of the other two methods. Fence segmentation is difficult even for interactive tools such as the Rotobrush. Its average precision and recall are 57.43% and 52.25%, much lower than ours. Looking at individual examples, for videos containing dynamic scenes, the best result is achieved on the "Jaguar" example, due to the fact that its fence color is most distinctive from the background. Our method also achieves reasonable results on the "wire" and "tree branch" example, demonstrating the generalization of our method to non-fence occludes. Furthermore, the dense CRF improves both precision and recall in all examples.

The method described in [21] produces an alpha matte of the fence for one frame of the input video: some are shown in Figure 8. Given that this method is designed for videos

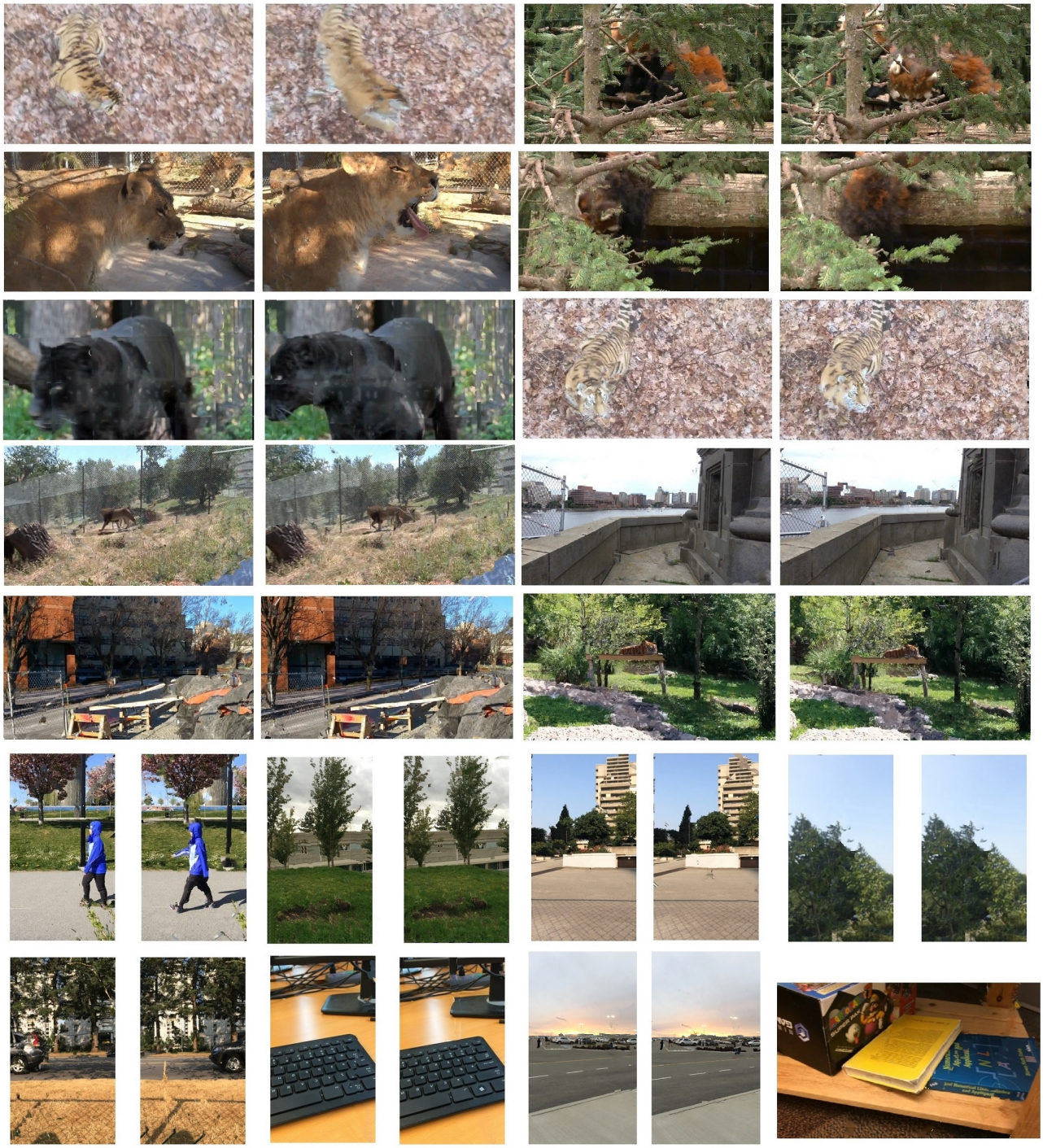
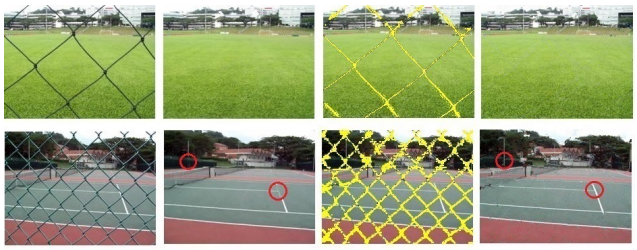


Figure 10. Fence removal results on some selected frames.

with static background, its results are poor on many examples with dynamic backgrounds (e.g. "Tiger"). It also produces poor results on examples captured with a static camera (e.g. "Jaguar"), which violate the underlying assumption of this method. To evaluate their precision and recall, we search through $[0, 1]$ for an optimal threshold that gives the largest value on (precision$\times$recall). The average precision and recall computed in this way are 46.34 and 69.61, respectively, which is significantly lower than ours. Moreover, our method considers both color and motion to form pixel groups. So it can largely tolerate optical flow errors. For example in Figure 2, though the flow is quite poor as in

Data	Rotobrush[1] Precision (%)	Precision (%) of method in [21]	Initial Precision (%)	Final Precision (%)	Rotobrush[1] Recall (%)	Recall (%) of method in [21]	Initial Recall (%)	Final Recall (%)
Tiger1	51.43	15.73	76.00	**78.97**	58.22	**95.61**	59.93	77.42
Little Panda1	51.30	19.63	78.13	**80.05**	**82.36**	64.56	75.04	78.91
Lion	78.08	53.61	67.49	**80.49**	49.78	69.89	61.11	**80.58**
Little Panda2	37.36	20.94	77.10	**80.19**	24.68	66.77	77.38	**78.92**
Jaguar	**96.47**	11.59	81.09	86.93	47.04	59.27	77.60	**90.54**
Tiger2	57.22	18.78	78.05	**84.09**	44.20	39.00	75.65	**82.70**
Running Lion	32.27	70.27	71.54	**79.75**	64.74	81.06	70.15	**86.72**
Gray Fence1	**85.84**	76.88	71.44	78.48	32.86	**88.25**	70.13	83.88
Gray Fence2	32.56	53.47	77.03	**80.76**	11.94	**85.56**	75.52	80.37
Zoo	31.22	59.01	73.57	**82.39**	2.16	69.57	72.23	**84.72**
Walking Person	70.57	51.73	74.19	**78.49**	**82.79**	67.90	67.50	79.95
Blue Fence	56.89	20.64	84.21	**89.41**	62.80	75.66	65.61	**91.51**
Building	61.77	84.89	77.29	**85.02**	74.82	66.96	80.36	**82.82**
Tree	76.14	49.24	77.82	**82.19**	69.97	41.95	78.73	**81.76**
Car	41.75	59.79	75.81	**81.51**	81.61	74.49	61.80	**83.93**
Wire and Keyboard	67.46	56.81	62.67	**90.58**	76.33	87.82	67.56	**89.25**
Tree Branch	67.68	\	50.80	**74.19**	39.62	\	58.80	**73.44**
Wires	34.74	**78.99**	74.27	63.01	34.56	49.13	62.41	**79.22**
Average Value	57.43	46.34	73.81	**80.92**	52.25	69.61	69.86	**82.31**

Table 1. Precision and recall of initial segmentation and final segmentation.



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

Figure 11. Comparison with [14] on their data. (a) selected frames from the original video; (b) de-fencing results from [14]; (c) our fence segmentation results; (d) our fence removal results.

Figure 2 (b), the pixel-groups in (c)(f) are quite reasonable.

Meanwhile, we also tested our methods on data from Video [14] to provide a direct comparison. Since there are no video fence segmentation results provided, we compared with it by fence removal results as shown in Figure 11 On the second example, our method produces superior results in the red circles which suggests better fence segmentation. Please note that [14] can only deal with static scenes.

Fence Removal: Once the fence is segmented, we can apply existing image and video inpainting techniques, such as [5], to remove the fence from video frames. Figure 10 shows some frames with fence removed using the method in [5]. We believe better fence removal can be achieved by exploiting multiple frame information such as in [20], which is our future work.

5. Conclusion

We present a fully-automatic method to detect and segment fence-like occluders from a video clip to generate a fence-free photo. The main advantage of our method over previous work is that it handles both dynamic scenes and moving cameras. Our method first groups pixels according to their motion and color similarity. It then exploits spatial structural features in a graph-cut optimization framework to produce initial segmentation. The initial segmentation is further refined by solving a dense CRF to achieve better spatial accuracy and temporal coherence. Fence removal is demonstrated with existing inpainting techniques, which shows that our method is a promising building block towards a fully automatic, high quality fence removal solution for general videos.

Acknowledgments

This work is supported by the NSERC Discovery Grant 31-611664, the NSERC Discovery Accelerator Supplement 31-611663, and an Adobe research gift. Renjiao Yi is supported by scholarship from the China Scholarship Council.

References

[1] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 28(3):70, 2009. 1, 2, 5, 8

[2] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. CVPR*, volume 1, pages I–355. IEEE, 2001. 2

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. ACM SIGGRAPH*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000. 2

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. PAMI*, 26(9):1124–1137, 2004. 4

[5] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13(9):1200–1212, 2004. 2, 8

[6] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *Proc. ECCV*, pages 522–535. Springer, 2006. 2

[7] V. S. Khasare, R. R. Sahay, and M. S. Kankanhalli. Seeing through the fence: Image de-fencing using a video sequence. In *Proc. ICIP*, pages 1351–1355, 2013. 1, 2

[8] J. Košecká and W. Zhang. Video compass. In *Proc. ECCV*, pages 476–490. Springer, 2002. 2

[9] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems*, 2011. 2, 4, 5

[10] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, volume 23, pages 303–308. ACM, 2004. 2

[11] C. Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, MIT, 2009. 2

[12] S. Liu, J. Wang, S. Cho21, and P. Tan. Trackcam: 3d-aware tracking shots from consumer video. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, 33(6):198, 2014. 2

[13] Y. Liu, T. Belkina, J. Hays, and R. Lublinerman. Image de-fencing. In *Proc. CVPR*. 2006. 2

[14] Y. Mu, W. Liu, and S. Yan. Video de-fencing. *arXiv preprint arXiv:1210.2388*, 2012. 1, 2, 8

[15] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez. Video inpainting of complex scenes. 2014. 2

[16] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Image de-fencing revisited. In *Proc. ACCV*, pages 422–434. Springer, 2011. 2

[17] M. Park, R. T. Collins, and Y. Liu. Deformed lattice discovery via efficient mean-shift belief propagation. In *Proc. ECCV*, pages 474–485. Springer, 2008. 2

[18] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003. 2

[19] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 23(3):309–314, 2004. 1, 2

[20] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Trans. PAMI*, 29(3):463–476, 2007. 2, 8

[21] T. Xue, M. Rubinstein, C. Liu, and W. T. Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)*, 34(4):79, 2015. 1, 2, 5, 6, 8

[22] A. Yamashita, A. Matsui, and T. Kaneko. Fence removal from multi-focus images. In *Proc. ICPR*, pages 4532–4535. IEEE, 2010. 1, 2