Real-Time Patch-Based Tracking with Occlusion Handling

Jian Tian and Yue Zhou

Institute of Image Processing and Pattern Recognition
Shanghai Jiao Tong University
Shanghai, China

Abstract. A new method for real-time occlusion-robust tracking is proposed. By analyzing the process of occlusion occurrence, we present a fast and effective occlusion detection algorithm based on the spatio-temporal context information. As a result, we can always obtain correct target location using adaptive template matching with patch-based structure description, regardless of the occlusion situation. Our extensive experiments on many sequences verify the good performance of our algorithm. In addition, based on the framework of our algorithm and properties we find, more effective occlusion-robust tracking algorithms can be developed.

Keywords: Real-time tracking, occlusion detection, patch based model, structure description, context.

1 Introduction

In computer vision, visual object tracking is an important subject for a wide range of applications. Abrupt object motion, changing appearance patterns of both the object and the scene, non-rigid object structures, occlusion and camera motion challenge the tracking algorithms [1]. Occlusion is a common problem in tracking. It is different from other interference such as deformation and varying illumination. The tracker should update the appearance model to accommodate the object appearance, when the object orientation or illumination changes. However, the occluder should be regarded as another object. Therefore, occlusion in tracking is a particular interference and needs more attention to solve. Based on patch-based structure description and context information, we present a simple and fast tracking and a novel occlusion detection algorithm.

Our algorithm is a patch-based structure description method. It describes the spatial structure of tracking object and has good robustness for geometric variation. Previous work has presented many similar algorithms [2, 3, 4]. Junseok Kwon et al. [3] presented a non-rigid object tracking algorithm based on patch-based model and adaptive basin hopping monte carlo sampling (BHMC). It is robust for deformation effectively. However, the patch-based model in [3] can not resist scale variance and occlusion well.

Many tracking algorithms rely on learning to resist occlusion [5,6]. Kaihua Zhang et al. [6] presented a spatio-temporal context learning algorithm (STC). Spatio-temporal context makes tracking stable and accurate, context learning contributes to occlusion-robust. However, learning algorithms detect occlusion based on similarity between image and object model, it is difficult to resist occlusion and other interference at the same time. Yi Deng et al. [7] proposed a patch-based occlusion detection algorithm. It focuses on patches obtained by segment at the boundary of target. Occlusion at the patches means target will be sheltered. Similar to it, our algorithm focuses on the boundary of target, however, we detect occlusion by analyzing the process of occlusion occurrence. It considers temporal context which is an essential information.

Our main contributions are: (a) Based on affine invariant patch-based model, we present a simple and fast tracking algorithm. (b) From a new perspective, we find a property to detect occlusion in tracking. (c) Based on the property, we present a patch-based occlusion detection algorithm, it is easy to compute and transplant.

2 Patch-Based Model and Real-Time Tracking

We propose to represent the object by many patches which can be moved, deleted or newly added. Each patch helps to locate the possible positions and scales of the object in the current frame. The patches are not based on an object model. To make such a representation more robust, a patch should be deleted if it belongs to the background, and a structure description is introduced to find it. We use template matching to move the patches because of its relative simplicity and low computational cost.

2.1 Patch Initialization

In order to get effective patches, first we use FAST [8] to find key points. It is an efficient key point detection algorithm. A patch is more likely to contain effective information if its position is close to target center. We set a key point as one of the patch's vertices which is farthest to the target center (Fig.1 (a)). Then we compare each patch with screen around target based on template matching. If a patch is similar to screen, it should be abandoned (Fig.1 (b)).

2.2 Examining the Patches by Structure Description

To update the patches, we use normalized correlation template matching. Other matching algorithms are available, such as LK matching [9], MSER [10]. Tracking procedure mainly includes two parts: (a) Rough localization. We use the whole target as template to match. It provides a stable target center to calculate proper search region and structure description similarity which will be introduced later. (b) Accurate adjustment. Based on patch similarity and structure description similarity, we remove inaccurate patches.

We use the ratio of each patch distance to average distance as structure description (Fig.1 (c)):

$$D_i = R_i / (\sum_{j=1}^{n} R_j / n), \mathrm{i} = 1, 2, \ldots, n \tag{1}$$

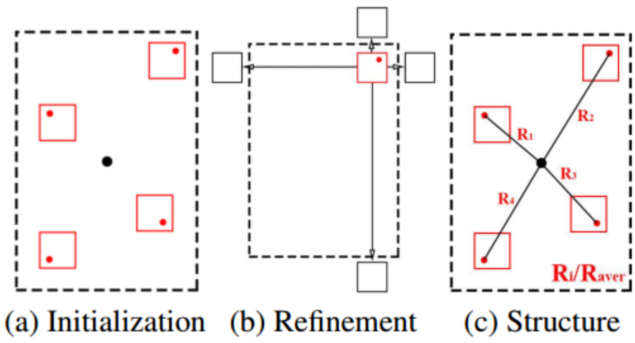

Fig. 1. Patch-based model. (Black point is target center, red points are FAST points, red boxes are patches.).

where R_i is the distance between the center of patch i and target center. In Patch initialization, it is impossible to avoid generating new patches which do not belong to the object. Likewise, a patch may be updated to the background. These situations will reduce the accuracy of the appearance of the model. Wrong patches cannot be avoided, but it will exhibit motion characteristics different to other patches when object moves. We can use the center calculated by all patches except the current patch i to get R_i. Based on (1), if D_i is larger than it is in the last frame, we can consider it is a wrong patch and remove it.

2.3 Updating the Appearance Model

After updating patches, we can find those patches which are correctly tracked and locate the target in new frame.

At every frame, we can find a smallest rectangle which can include all well tracked patches, we call that rectangle the bounding rectangle:

$$\mathrm{BoundRect} = \mathrm{argmin}_r \mathrm{r.area}() \quad \forall i \; r \;\&\; pathc[i] == patch[i] \tag{2}$$

, where r means an arbitrary rectangle in the frame on the condition that all N patches is in the rectangle r.

Estimation of the target displacement is the same as the BoundRect displacement and the width and height of the target can changed in proportion as the BoundRect. So the shape of the object can be updated adaptively.

In addition, performance of structure description depends on the number of patches, we should generate more patches when they are not enough. Scale of target often changes, we can properly adjust patch size based on scale of target.

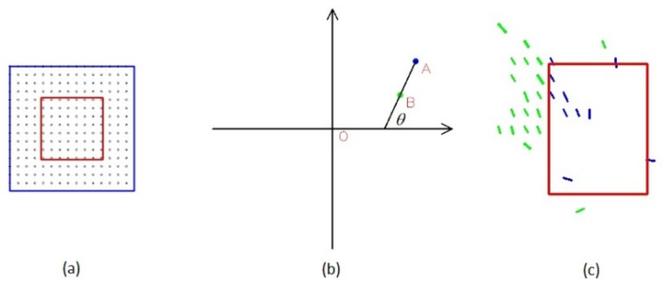

Fig. 2. Occlusion detection model

3 Occlusion Detection

As mentioned in Sec.1, most tracking algorithms merely rely on learning to resist occlusion. However, it is difficult to ensure the robustness for occlusion and other interference at the same time.

In this paper, we exploit the spatio-temporal context information to analyze the occlusion situation. Occlusion has two properties that the other interference do not have.

Property 1: Occlusion has the process that occluder relatively moves to object from screen. However, variation caused by other interference comes from object itself. Therefore, we can detect the process of occlusion from screen to tracking object.

Property 2: At the place occluder is moving from screen to target, local regions of screen and target move towards the target inner at the same time.Because property 2 considers the relation between frames and the relation between screen and target, it is spatio-temporal context information.

Based on the 2 properties, we present a simple model to detect occlusion in tracking. Fig.2 shows our model (it can be adjusted according to different tracking problems).

The tracker accepts a pair of images I_t , I_{t+1} , and a bounding box β_t , β_{t+1} (Sec. 2).Red box is the target region (Fig.2 (a)),Blue box has the same center as the red one, and its width and height is 2 times longer than the red one. A set of points is initialized on the blue rectangular grid. These points are then tracked by the affine lucas kanade feature tracker in [11] which generates a sparse motion flow between I_t and I_t , $\overrightarrow{AB}$ is an example of the motion flow of a point(Fig.2 (b)),Point O is the center of target which is moved by our tracker in Sec. 2. We select the motion flow D:

$$D = \{d_i = \overrightarrow{A_iB_i} \mid |\overrightarrow{OA_i}| - |\overrightarrow{OB_i}| > 0\} \tag{3}$$

And calculate the angle θ_i between $\overrightarrow{A_iB_i}$ and the horizontal axis. D is divided into two groups D_{in} and D_{out} (Fig.2 (c):

$$D_{in} = \{d_i = \overrightarrow{A_iB_i} \mid \overrightarrow{A_iB_i} \in D, B_i \in \beta_{t+1}\} \tag{4}$$

$$D_{out} = \{d_i = \overrightarrow{A_iB_i} | \overrightarrow{A_iB_i} \notin D, B_i \in \beta_{t+1}\} \tag{5}$$

The distance between d_i and d_j is defined as:

$$Dist(d_i, dj) = min\{|\theta_i - \theta_j|, 360 - |\theta_i - \theta_j|\} \tag{6}$$

$$D_{valid} = \{d_i \in D_{in}, \exists d_j \in D_{out}, Dist(d_i, d_j) < T_D\} \tag{7}$$

D_{valid} includes points that may be from the occluder. If the size of D_{valid} is larger than a threshold T_N , the target may be occluded and we should stop adding new patches to avoid adding patches belong to the occluder. If any patch intersects with an element in D_{valid}, we remove it. If the number of patches is less than a threshold T_C, the target is covered. Before the target is covered, we can still track the target through the remaining patches. Fig.3 shows the flowchart of our occlusion detection algorithm.

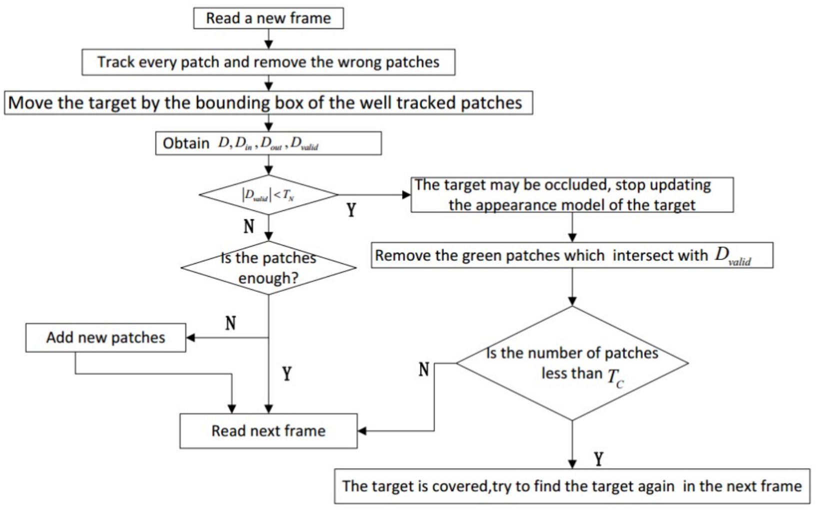
Read a new frame
Track every patch and remove the wrong patches
Move the target by the bounding box of the well tracked patches
Obtain $D, D_{in}, D_{out}, D_{valid}$
$|D_{valid}| < T_N$
Y
The target may be occluded, stop updating the appearance model of the target
N
Is the patches enough?
Remove the green patches which intersect with D_{valid}
N
Add new patches
Y
N
Is the number of patches less than T_C
Read next frame
Y
The target is covered,try to find the target again in the next frame

Fig. 3. The flowchart of our occlusion detection algorithm

4 Experimental Results

We conduct some experiments to verify the performance of our algorithm. Development environment of our algorithm is Visual Studio 2010 and Intel OpenCV library on Intel Core i3-2120 CPU and 4.00GB RAM. Experimental videos can be downloaded[1].

The main advantage of the proposed occlusion detection algorithm is that it can be applied to a range of trackers and is easy to implement. Kaihua Zhang et al. [6] presented a spatio-temporal context learning algorithm (STC). Spatio-temporal context makes tracking stable and accurate, context learning contributes to occlusion-robust.However, long-term occlusion will lead to tracking failure. With our occlusion detection, STC can promote the robustness of tracking under occlusions. We just stop learning Spatio-Temporal context model when the occlusion happens.

Fig.4 shows experimental results by STC, STC-OD and our patch-based tracker with occlusion detection (PBT-OD). The first row shows STC tracker can't handle heavy occlusion. The second row shows that with our occlusion detection algorithm STC-OD can recover the target .The third row shows the result by PBT-OD, which can also track the face well.

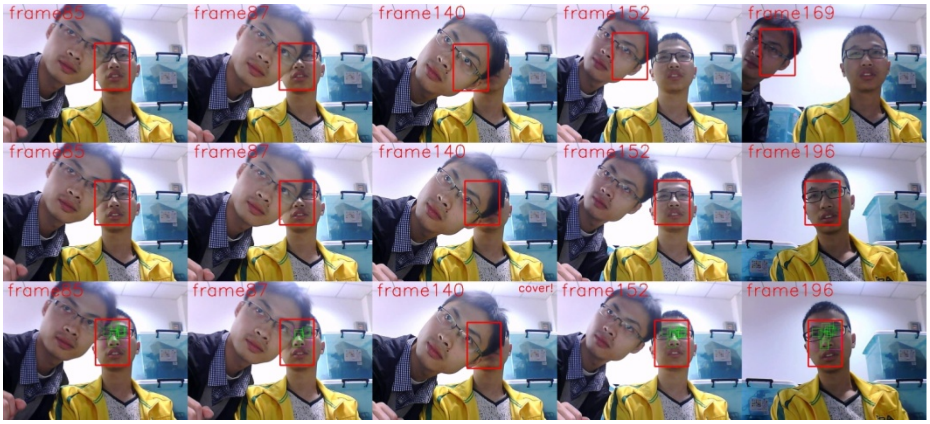

Fig. 4. Experimental results by STC, STC-OD and PBT-OD

We perform experiments on a wide range of video sequences downloaded from http://www.votchallenge.net/vot2013/evaluation_kit.html. This website provides many well-known short videos labeled with ground truth. We use Success plot in [12] to evaluate our algorithm. Given the tracked bounding box RT and the ground truth bounding box RA. The overlap score is defined as

$$S = \frac{RT \cap RA}{RT \cup RA} \tag{8}$$

The numerator and denominator in (8) represent the number of pixels in the intersection and union of two regions. To measure the performance on a sequence of frames, we count the number of successful frames whose overlap S is larger than the given threshold. The success plot shows the ratios of successful frames at the thresholds varied

[1] Download link:
`ftp://tianjian2013:public@public.sjtu.edu.cn/iconip2014/`

from 0 to 1. Using one success rate value at a specific threshold for tracker evaluation may not be fair or representative. Instead we use the area under curve (AUC) of each success plot to evaluate our algorithm. Fig.5 is the tracking sequences for evaluation and Fig. 6 shows the performance score of our tracker. The performance of the state-of-the-art trackers can be found in [12].

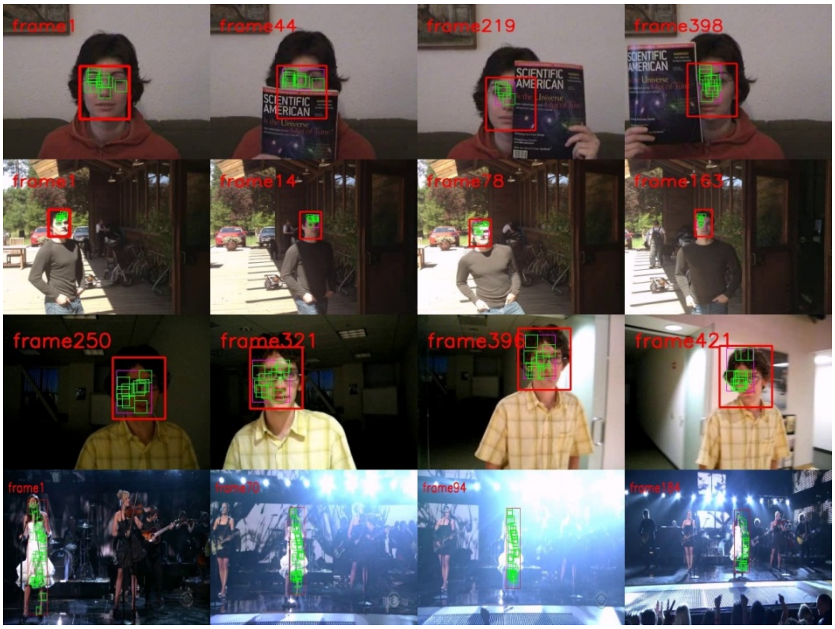

Fig. 5. Sample tracking results on four public challenging image sequences

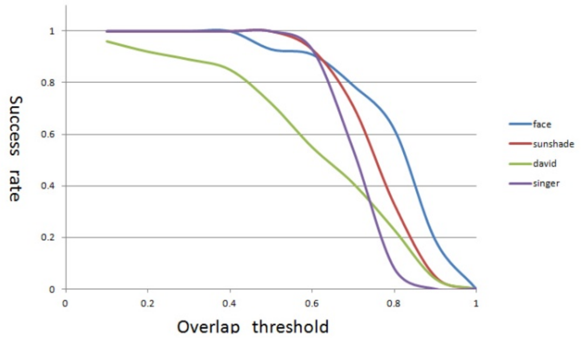

Fig. 6. Success rate plot

5 Conclusion

In this paper, we mainly focus on fast and occlusion-robust tracking problem. We present an affine invariant patch-based structure description and a fast tracking algorithm based on it. By analyzing the process of occlusion occurrence, we pre-sent two properties of occlusion detection and a portable algorithm based on patch-based context information. Comparative experiments verify the good per-formance of our algorithm. Our algorithm is simple to implement, and can run at real-time speed.

References

1. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys (CSUR) 38(4), 13 (2006)
2. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1. IEEE (2006)
3. Kwon, J., Lee, K.M.: Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009. IEEE (2009)
4. Dihl, L., Jung, C.R., Bins, J.: Robust adaptive patch-based object tracking using weighted vector median filters. In: 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images (Sibgrapi). IEEE (2011)
5. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(7), 1409–1422 (2012)
6. Zhang, K., et al.: Fast Tracking via Spatio-Temporal Context Learning. arXiv preprint arXiv:1311.1939 (2013)
7. Deng, Y., et al.: A symmetric patch-based correspondence model for occlusion handling. In: Tenth IEEE International Conference on Computer Vision, ICCV 2005, vol. 2. IEEE (2005)
8. Rosten, E., Drummond, T.W.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
9. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. IJCAI 81 (1981)
10. Donoser, M., Bischof, H.: Efficient maximally stable extremal region (MSER) tracking. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1. IEEE (2006)
11. Bouguet, J.-Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation 2, 3 (2001)
12. Wu, Y., Lim, J., Yang, M.-H.: Online object tracking: A benchmark. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2013)