TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.2, February 2014, pp. 1005 ~ 1011
DOI: http://dx.doi.org/10.11591/telkomnika.v12i2.4214

Robust Tracking based on Failure Recovery

Daode Zhang*, Cheng Xu, Yuanzhong Li
School of Mechanical Engineering, Hubei University of Technology, Wuhan, China
*Corresponding author, e-mail: hgzdd@126.com

Abstract

Object tracking is a issue in the domain of computer visual, most of current state-of-art approaches for visual tracking adapt tracking-by-detection, using detection to address tracking problem. While suitable for cases when the object is always in the sense and these algorithms always results in failures and can't track back after failure. This paper we propose a tracking method based on failure recovery. After we choose an object to tracking in the first frame, the object is tracked by improved optical flow method forward and backward in time then compute the distance between these two trajectories. While the distance larger then threshold tracking likely to fail, but the latest object model return by detector will re-initialize the tracker. Tracking an object on camera video approve our approach can work at 20fps with long-time robustness.

Keywords: *robustness, long-time tracking, tracking-by-detection, failure recovery*

1. Introduction

Visual tracking is the process of locating an object in a video sequence which has wide application in human-machine interaction surveillance and robot navigation. This thesis investigates visual tracking of a specified single object, which significantly changes its appearance and moves in and out of the camera view. Although visual tracking has been studied by a large number of studies for a long time, but still exist several challenges need to be settle. The most difficult challenge is occlusion and disappears. To settle this challenge we propose a algorithm to address the occlusion and disappear by failure detection and failure recovery. And this algorithm will dramatically improve visual tracking robustness so that to realize long-term tracking.

Long-term tracking has been widely research by many researchers. When occlusion occurs, object lost most of the feature information, traditional tracking algorithm will missed the object. Center weighted approach settle the partially occluded problem by set a large weight for center pixel. Typical center weighted tracking algorithm is mean-shift. Part matching approach settles occlusion by divide the object into several areas. Trajectory prediction method use motion information such as velocity and acceleration to predict the object's location on the next frame. In view of non-linear, non-Gauss assumption problem, Kalman filter is no longer suitable. Bayesian theory treats tracking as a problem to maximize Bayesian posterior probability distribution.

The most important point for long-time tracking which impact its performance is failure recovery. Long-time tracking unavoidable to challenge by appearance changes, illumination changes, scale changes, occlusion and disappearance of the object. Traditional tracking algorithm use detector to detect object and then tracker work solely after the object has been detected. But only suitable for those cases when the object does not disappear from the frame and occlusions. For this reason, the increasing availability of high robustness long-time visual tracking algorithm use detects method to address tracking problem [1, 2]. Tracking-by-detection methods estimate the object state only in current frame. This principle overcomes occlusion and disappearance and remedies the effect of error accumulation [3]. However the detector must be trained beforehand, which means a large scale of samples are need to train detector. And samples are very significant to such algorithms and good samples will results a well performance to tracker.

Neither tracking nor detection can settle long-time tracking alone. Therefore we combine tracker and detector, and add real-time learning. Algorithmically (Figure 1 shows

Received July 25, 2013; Revised September 4, 2013; Accepted October 2, 2013

algorithm structure), the tracking is initialized by manually selecting the interested object, then tracker and detector work mutual noninterference. Tracker estimate the object's motion between continues frames [4]. Detector treats every frame as independent and scans full frame to localize the state of the object. As any other tracker, the tracker is tending to fail and never recover if the object moves out of the scene. For this problem we propose a novel tracking algorithm that enables to detect tracking failures by itself, the detector will reinitialize the tracker once failures be detected.

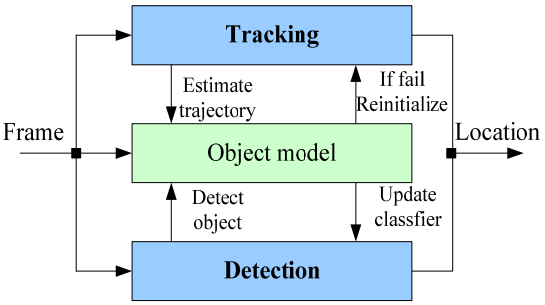

Figure 1. Object Selected by User in First Frame is tracked for a Long Time based on Failure Detection and Recovery

The rest of the paper is organized as follow: Section 2 explains how tracker self-evaluates its reliability and compare with traditional Optical Flow. Section 3 proposes failure recovery which can guarantee long-time tracking against tracking failures. In Section 4 we test our algorithm on camera video with challenging background and finishes with conclusions.

2. Failure Detection based on Forward-Backward Consistency

2.1. Object Representation

Tracking by detection is tracking by learning. To this type of algorithm, object model is very important. The object state is defined by a bounding box and a single instance object's appearance is represented by an image patch P. Object model is a dynamic data structure which represents the object and its surrounding observed so far. The object model in our algorithm defines as:

$$M = \left\{ P_1^+, P_2^+, ..., P_m^+, P_1^-, P_2^-, ..., P_n^- \right\} \quad (1)$$

Where P^+ and P^- represent the object and background patches. The relative between an arbitrary patch p and object model is define as:

$$S^r = \frac{S^+}{S^+ + S^-} \quad (2)$$

Where S^+ and S^- is similarity to nearest positive neighbor and nearest negative neighbor.

2.2. Tracking Failure Detection Algorithm

Automatic detection of tracking failures is an important feature. In this section we describe how tracker detects the tracking failures. We study the long-term tracking problem from the perspective of frame-to-frame tracking. And frame-to-frame tracking is not a good model as it inevitable failures. Rather than trying to avoid these failures, we proposed a novel measure that indicates the reliability of a tracker. The failure detection is based on forward-backward consistency assumption that correct tacking should be independent of time-flow. Tracker

estimates frame-to-frame object motion and handle appearance and illumination changes. In order to increase the robustness of the tracker, we use the Euclidean distance between the original point on the forward trajectory and the point represented the same object on the backward trajectory to judge whether the tracker failed, just like Figure 2 illustration. Let (t,t+1,…,t+k) be an image sequence and x_i stand for the location of the tracked object in time i(i=t,t+1,…,t+k). We track point x_t forward for k steps, resulting a forward trajectory:

$$\vec{T}_k = (x_t, x_{t+1}, ..., x_{t+k}) \tag{3}$$

Then point x_{t+k} ($\hat{x}_{t+k} = x_{t+k}$) is tracked backward to frame t and obtain a backward trajectory $\overleftarrow{T}_k = (\hat{x}_t, \hat{x}_{t+1}, ..., \hat{x}_{t+k})$. The Euclidean distance between these two trajectory is define as the distance between initial point x_t and end point $\hat{x}_t$:

$$dis(\vec{T}_k, \overleftarrow{T}_k) = \left\| x_t - \hat{x}_t \right\| \tag{4}$$

The distance proposed to measure tracking failures can be applied to many trackers and is easy to implement. In our implementation, the distance is used in conjunction with the similarity of the patch P_1 surrounding point x_t and P_2 surrounding point $\hat{x}_t$. The similarity of these two patches is compared using the Normalized Correlation Coefficient, defining as:

$$NCC(P_1, P_2) = \frac{1}{n-1} \sum_{x=1}^{n} \frac{(P_1(x) - \mu_1)(P_2(x) - \mu_2)}{\sigma_1 \sigma_2} \tag{5}$$

Where μ_1 and σ_1 are mean and standard deviation of P_1.

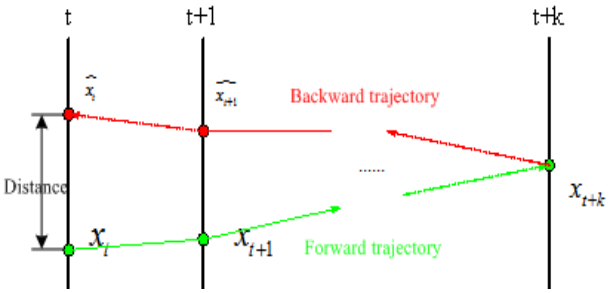

Figure 2. Illustration of the Distance between Forward Trajectory and Backward Trajectory

The block diagram of the proposed tracker is shown in Figure 3. In frame t, bounding box b_t was initialized on a regular grid of size10x10. Then track the points from frame t to frame t+1 using Lucas-Kanade tracker forward and backward. Following the distance compute approach, we calculate the distance of each point between forward and backward trajectory so that we can get the median distance med_{dis} of 100 points. The first filter condition is reject those points which distance between forward and backward trajectory larger than med_{dis}. On the other hand, use sub-pixel precision to extract point-centered patches sized 10x10 from frame t-1 and t. Match the corresponding patches result in the similarity of every points and calculate the median similarity med_{NCC} by sorting. Points exhibiting a similarity measure less

than med_{NCC} is filtered out. The tracker only keeps less than 50% points to estimate the location of the bounding box in the frame t+1.

The med_{dis} larger than 10 pixels or the bounding box move out of current frame is a failure criterion of our tracker. Under this condition, tracker does not feedback the location of the bounding box on frame t+1. If the med_{dis} less than 10 pixel then calculate the relative similarity between patches and object model. Relative similarity ranges from 0 to 1 and higher values means more confident that the patches depicts the object. Our tracker regard that the relative similarity S^r less than 0.65 as tracking failure. Our tracker re-initializes all points in every frame and the points can always keep on a rectangular grid. This allows our tracker to estimate scale of the object. Moreover, the failure detection can restrict the failures effectively so that to against long-time tracking.

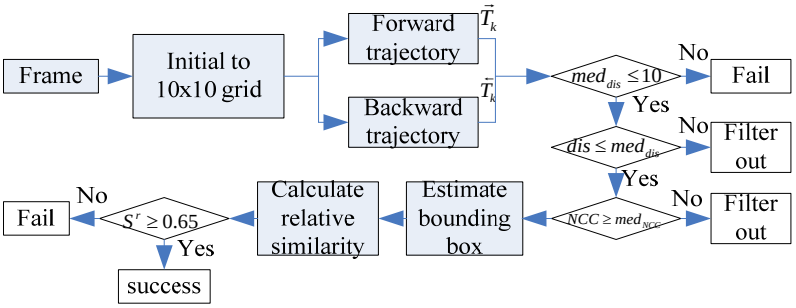

Figure 3. The Block Diagram of the Tracker

3. Object Detection based on Cascade Classifier

3.1. Random Ferns

For a large number of tracking problems, the class of the interested object is known. It is therefore possible to train an object detector and integrate it into the tracking process. Random ferns can effectively solve image classification problem. Set the class label is $c_i, i = 1,...,H$ and image feature define as $f_j, j = 1,...,H$. So classifier problem can be expressed as :

$$\hat{c}_i = \arg\max_{c_i} P(C = c_i \mid f_1, f_2, ..., f_N) \tag{6}$$

Where C is a random variance indicates class of image. According to Bayesian theory, we get the equation:

$$P(C = c_i \mid f_1, f_2, ..., f_N) = \frac{P(f_1, f_2, ..., f_N \mid C = c_i) P(C = c_i)}{P(f_1, f_2, ..., f_N)} \tag{7}$$

Supposition prior probability P is the uniform distribution, classifier problem can be depict as:

$$\hat{c}_i = \arg\max_{c_i} P(f_1, f_2, ..., f_N \mid C = c_i) \tag{8}$$

The value of binary feature f_i is depend on compare of two pixel grayscale value.

$$f_i = \begin{cases} 1, & I_1 < I_2 \\ 0, & I_1 \geq I_2 \end{cases} \tag{9}$$

Due to the simple form of f_i, a sufficient number of characteristics is needed to ensure that the classification is correct. Distribution $P(f_1, f_2, ..., f_N \mid C = c_i)$ needs to solve 2^N parameters. In order to reduce the number of parameters and ensure dependencies between features, our classifier divides features into M groups. And every group contains $S = N/M$ features. The features is related in a group and not related in different groups, so this group defines as ferns. In this case, only need to calculate each group's joint probability distribution. So conditional probability can be approximated to it as follows:

$$P(f_1, f_2, ..., f_N \mid C = c_i) = \prod_{k=1}^{M} P(F_k \mid C = c_i) \tag{10}$$

Where $F_k = \left\{ f_{\sigma(k,1)}, f_{\sigma(k,2)}, ..., f_{\sigma(k,s)} \right\}$ indicate the k'th fern and σ indicate random function. So the class of input patch is :

$$\hat{c}_i = \arg\max_{c_i} \prod_{m=1}^{M} P(F_m \mid C = c_i) \tag{11}$$

This conditional probability only needs to calculate $M \times 2^S$ parameters, which coordinated the computational complexity and accuracy of classification.

3.2. Cascade Classifier

Traditional tracking algorithm [5-6] is focus on speed, robustness and adaptability. This type of approach is trying to avoid failure, but can not solve tracking failure problem essentially. In this paper failure recovery is based on object detection. This section we discuss long-time tracking from object detection. The tracker can only estimates the failures but unable to recover from failure by itself. Object detection enables to re-initialize the failed tracker by the latest object model. The object detector localized the appearances of the object model is based on sliding window. In sliding-window-based approaches for object detection, sub-images of an input image are tested whether they contain the object of interest [7-8]. As the number of bounding boxes to be evaluated is large, but the detector has to be efficient. To speed up the process, the detector we designed is a cascaded classifier which consists of three parts: variance filter, ensemble classifier and nearest neighbor classifier. Only if a sub-window is accepted by one stage, the next stage is evaluated. Variance filter is the first stage of our detector. Filter keeps only those patches which gray-value variance is small than 50% of the variance of the initial patch. The ensemble classifier based on random ferns [9-10] is the second stage of our detector and it is consist of 10 base classifiers. In this stage we calculate the posterior of each base classifier, and we classified a patch as positive by if the average posterior is larger than 50%. In the third stage we employ a Nearest Neighbor Classifier. A patch is classified as the object if $S^r > 0.6$.

As we know in our approach both the tracker and the detector return a bounding box to locate the object in the current frame respectively. Once the object is tracked and detected, we compare the conservative similarity S^r and output a bounding box with the maximally confident. When the tracker failed, detector re-detects the object and returns a bounding box to initial the recursive tracker. What's more, even if the tracker does not failed but output a bounding box with a low confident than the detector, it will be initialize by the detector.

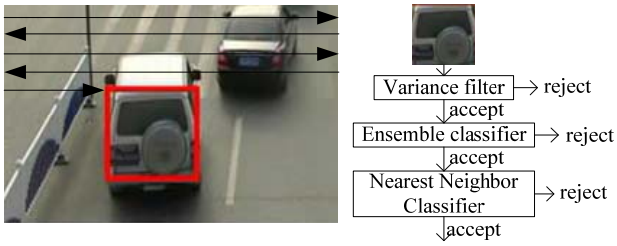

Figure 4. The Cascaded Classifier Slide the Frame to Test whether it Contains Object

4. Conclusion

In this section, our approach is evaluated on challenging camera video and shows a very well performance. The tracker is implemented in c++, which runs at 20 frames per second.

Figure 2. Test Result

We evaluate our algorithm on 5 challenging sequences. Since the tracker involve randomness, we run 10 times and report the average result for each video. And the result depict on Table 1.

Table 1. The Average Result

Video	Frames	fps	Precision	Recall	F_1-measure
①	681/759	21.23	0.73	1.00	0.84
②	274/312	20.84	1.00	0.90	0.94
③	147/181	48.34	1.00	1.00	1.00
④	851/943	44.78	0.98	1.00	0.99
⑤	2275/8573	14.03	0.69	0.32	0.43

According to the test result, we can draw a conclusion that our tracker can detect tracking failure and recovery from failure. Because of this, so the tracker can resolve occlusion problem and be able to achieve the long-term tracking. So be concrete, this algorithm can against many challenges such as:

(1) Scale changes. Median-flow estimates scale change even when the object is partially out of frame. And detectors localize the object in multiple scales. In sequence 5, the object changes scale from 20*20 to 100*100 pixel.

(2) Illumination changes. Median-Flow adapts the tracked templates and the detector is based on illumination invariant pixel comparisons and NCC. In sequence 1, a face is tracked from a dark room to full illumination.

(3) Appearance changes. Median-Flow handles changes of appearance caused by pose change or articulations. The detector localizes all appearances observed in the past. Partial occlusions. Our algorithm deals with partial occlusions. Median-Flow tracker estimates reliable points within the bounding box and filters out parts of the object that are occluded.

(4) Full occlusions and disappearances. The main power of our algorithm is the ability to re-detect the target after full occlusion or disappearance of the object from the scene.

Acknowledgement

This work is supported by Natural Science Foundation of Hubei Province (No. 2012FFB00604) and National Natural Science Foundation of China (No. 51174084).

References

[1] S Avidan. Ensemble Tracking. *IEEE Trans on Pattern Analysis and Machine*. 2007; 29(2): 261-271.
[2] S Avidan. Support Vector Tracking. *IEEE Trans on Pattern Analysis and Machine.* 2004; 1064-1072.
[3] G Nebehay. Robust Object Tracking Based on Tracking-Learning-Detection. Vienna: Faculty of Informatics, Vienna University of Technology. 2012.
[4] Z Kalal, K Mikolajczyk, J Matas. Tracking-Learning-Detection. *IEEE Trans on Pattern Analysis and Machine.* 2012; 99.
[5] Wang Lixia, Jiang Dalin. *The Application of Mathematical Morphology in Vehicle Identification Technology*. In proceedings of the 2011 International Conference on Multimedia Technology. 2011; 801-805.
[6] Kim Changwan, Son Hyojoo, Kim Changmin. Automated color model-based concrete detection in construction-site images by using machine learning algorithms. *Journal of Computing in Civil Engineering*. 2012; 26(3):421-433.
[7] CH Lampert, MB Blaschko, T Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. CVPR. 2008.
[8] Gao Chi, Gao Hhi. Direction Identification System of Garlic Clove Based on Machine Vision. *TELKOMNIKA Indonesian Journal of Electrial Engineering*. 2013; 11(5): 2323-2329.
[9] M Ozuysal, M Calonder, V Lepetit, P Fua. Fast Keypoint Recognition using Random Ferns. *IEEE Trans on Pattern Analysis and Machine.* 2010; 448-461.
[10] Anny Yuniarti. Classification and Numbering of Dental Radiographs for an Automated Human Identification System. *TELKOMNIKA Indonesian Journal of Electrial Engineering*. 2012; 10(1): 137-146.