Context Tracker: Exploring Supporters and Distracters in Unconstrained Environments

Thang Ba Dinh[1], Nam Vo[2], and Gérard Medioni[1]
[1]Institute of Robotics and Intelligent Systems University of Southern California, Los Angeles, CA 90089
{thangdin,medioni@usc.edu}
[2]Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
{nam.vongoc@gmail.com}

Abstract

Visual tracking in unconstrained environments is very challenging due to the existence of several sources of varieties such as changes in appearance, varying lighting conditions, cluttered background, and frame-cuts. A major factor causing tracking failure is the emergence of regions having similar appearance as the target. It is even more challenging when the target leaves the field of view (FoV) leading the tracker to follow another similar object, and not reacquire the right target when it reappears. This paper presents a method to address this problem by exploiting the context on-the-fly in two terms: Distracters and Supporters. Both of them are automatically explored using a sequential randomized forest, an online template-based appearance model, and local features. Distracters are regions which have similar appearance as the target and consistently co-occur with high confidence score. The tracker must keep tracking these distracters to avoid drifting. Supporters, on the other hand, are local key-points around the target with consistent co-occurrence and motion correlation in a short time span. They play an important role in verifying the genuine target. Extensive experiments on challenging real-world video sequences show the tracking improvement when using this context information. Comparisons with several state-of-the-art approaches are also provided.

1. Introduction

1.1. Motivation

Long-term visual tracking in unconstrained environments is critical for many applications such as video surveillance, and human computer interaction. A major research axis has been focused on building a strong model to encode the variations of object appearance while distinguishing it from the background. By doing this, a fundamental dilemma occurs: the more complex the appearance model, the more expensive it is. At the extreme, the emergence of cluttered background and the occurrence of regions having similar appearance as the target makes appearance modeling is very challenging.

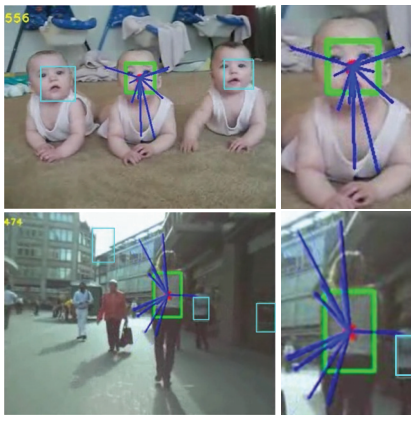


Figure 1. The context elements are automatically explored by our tracker. Distracters: the cyan bounding box. Supporters: the end points of blue lines.

In fact, there is additional information which can be exploited instead of using only the object region. Context information has been applied actively in object detection [9], object classification [20, 23], object recognition [25]. It has been employed recently in several tracking methods [27, 13]. One reason for context being overlooked is the fast run-time requirement. Also, visual tracking, especially single object tracking, is considered as a semi-supervised problem where the only known data is the object bounding box in the first frame (or in first few frames), which means learning such a context needs to be performed on-the-fly.

1.2. Our approach

Here, we propose to exploit the context information by expressing it in two different terms: 1) *Distracters* are regions that have similar appearance as the target, 2) *Sup-*

porters are local key-points around the object having motion correlation with our target in a short time span. Some examples of these elements automatically explored during tracking by our tracker are shown in Figure 1, where the distracters are drawn in cyan bounding boxes, and supporters are the end-points of blue lines. Distracters share the same type as the target, for example other faces in face tracking, or other pedestrians in pedestrian tracking. Supporters occur in regions belonging to the same object as the target, but are not included in the initial bounding box. In other words, the goal of our algorithm is to find all possible regions which look similar to the target to prevent drift, and to look for useful information around the target to have strong verification. The target and distracters are detected using shared sequential randomized ferns [24]. They are represented by individual evolving templates. The supporters, on the other hand, are represented as keypoints, and described using descriptors of the local region around them. Our experiments show that using these context elements helps the tracker avoid drifting to another object in a cluttered background and helps reacquire the right target after it leaves the FoV, or after total occlusion without confusion.

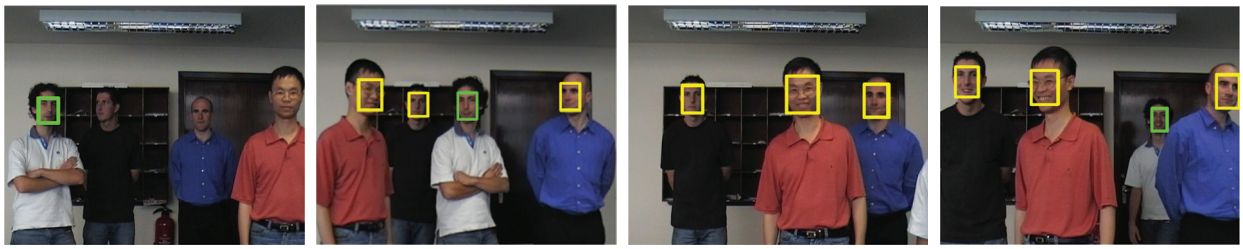
1.3. Related work

Recently, region-based tracking approaches are very popular in visual tracking. To track an object frame by frame, most algorithms try to search for the best match [3, 1, 12, 26, 28, 16]. Some methods simply assume a search area where the object is expected [1, 3, 12] while some others [26, 28, 22] use state prediction such as a particle filter [14]. Certainly, these methods face several challenges in practice such as abrupt motion, frame-cut, and object leaving FoV. To address this issue, Zdenek *et al*. [16, 15] proposed a fast detector based on a randomized forest to exhaustively scan through the whole image, selecting several candidates, followed by an evaluation step from the online template-based object model.

In region-based visual tracking, the appearance model plays an essential role. Some methods model the appearance of an object in a generative way such as using histogram [1] and linear subspace [26] while others build the appearance model as a classifier between the object and background [3, 12]. Also, some hybrid methods have been proposed to fuse both types of them [28, 19]. Since visual tracking is a semi-supervised problem where the incoming data is unknown, online appearance modeling [26, 3, 12, 28] is preferable to an offline one [1, 2]. To track the target successfully, these methods build an appearance model which is not only able to adapt to all changes well, but also robust against background. This leads to a trade-off: a tracker is fast but vulnerable to drift.

Moreover, the context information such as other moving objects, other regions interacting with the target have been overlooked by these approaches. Yang *et al*. [27] proposed to explore a set of auxiliary objects which have strong motion correlation and co-occurrence with the target in a short term. Also, they need to be tracked easily. The color over segmentation is employed to find such an object; then Meanshift [7] is applied to track them. However, Meanshift and color information are vulnerable in a cluttered background and the fusion of multiple components, *i.e*. the auxiliary objects in this method, is ad-hoc. To get support from other areas, Grabner *et al*. [13] introduced the definition of supporters which are useful features to predict the target location. These features need to have some temporal relation in motion with the current target. However, to detect and match all of local features in each frame is expensive. Also, in unconstrained environments, the motion of the object leaving the FoV and under total occlusion is not easily predicted, a wrong guess may ruin the result. Recently, Fan *et al*. [11] proposed to learn the attentional regions which have strong discriminative power in their discriminative domains, *i.e*. the regions which are distinctive from others in the defined region. But this method is still limited by the use of local and semi-local areas around the target, which is not efficient in the presence of abrupt motions and frame-cuts.

The most closely related work of our approach is the tracker built on P-N learning [16]. Basically it inherits the power of tracking-learning-detection (TLD) framework [15] while focusing on exploring the structure of unlabeled data, *i.e*. the positive and negative structures. Even though the method claims to explore the hard negative samples which contain other moving objects (*Distracters* in our terminology), we argue that when an object has similar appearance to the target, dominant positive training samples allow the tracker to detect it as positive. In contrast, training those hard negative samples makes the object model over-fit. Moreover, since the method uses a template-based appearance model, it requires the object region to fit into a bounding box without much background included. This fact limits the tracker from taking advantage of more useful information on other parts of the target with complex shape appearance such as a human body. Also, because the P-N Tracker purely relies on template matching to find the best match among several candidates, it is vulnerable to switching to another similar object. To address these issues, we propose to employ the tracking-learning-detection concept of this tracker to not only explore the structure of positive and negative data but also the more semantic data: *Distracters* and *Supporters*. It is worth noting that our method does not aim to solve the multi target tracking (MTT) problem, where data association is the most important component. In MTT, most methods employ an object detector [21, 17] or background subtraction [29] to find possible object responses, whereas in our case, we automat-



(a) Initialization (b) Exploit all other distracters (c) Tracking the distracters even when the object leaves FoV (d) Reacquire the target without confusion with other distracters

Figure 2. Automatically exploiting distracters. Target is in green, distracters are in yellow.

ically exploit and track all regions with similar appearance to our target. Based on the context model, *i.e*. supporters and distracters, and the appearance model, the tracker can avoid confusion during tracking.

The rest of this paper is organized as follows. The distracter detection is described in section 2. The selection of supporters is discussed in section 3. The experiments are demonstrated in section 4, followed by conclusions and future work.

2. Detection of distracters

2.1. Description

Distracters are regions which have appearance similar appearance to the target and consistently co-occur with it. Usually, distracters are other moving objects sharing the same object category as our target (Figure 2). To build an appearance model to distinguish objects of the same type is equivalent to develop a recognition system which needs a large amount of supervised samples to train. However, in visual tracking, the tracker has temporal and spatial information help to exploit which region is *"dangerous"* to preclude. To prevent our tracker from drifting to these regions, we propose to detect and initiate a simple tracker for each of them so that we can minimize confusion during tracking.

2.2. Detection

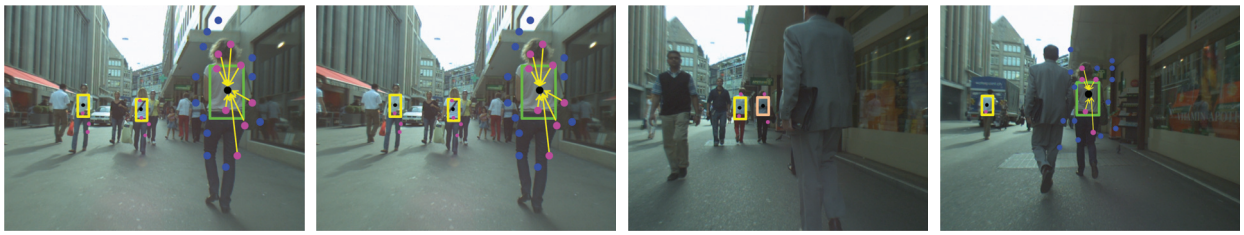
Due to the efficiency of randomized ferns classifier, which is widely used in recognition [30, 5], and tracking [16], we employ it to detect possible distracters in every frame. Randomized ferns were originally proposed by Özuysal *et al*. [24] to increase the speed of randomized forest [6]. Unlike tree-structure in randomized forest, ferns, having non-hierarchical structures, consist of a number of binary testing functions. In our case, each of them corresponds to a set of Binary Pattern features. Each leaf in a fern records the number of added positive and negative samples during training. For a test sample, its evaluation by calculating the binary pattern features leads it to a leaf in the fern. After that, the posterior probability for that input testing sample in feature vector x_i to be labeled as an object ($y = 1$) by a fern j is computed as $Pr_j(y = 1|x_i) = p/(p + n)$, where p and n are the number of positive and negative samples recorded by that leaf. The posterior probability is set to zero if there is no record in that leaf. The final probability is calculated by averaging the posterior probabilities given by all ferns:

$$Pr(y = 1|x_i) = \sum_{1}^{T} Pr_j(y = 1|x_i) \tag{1}$$

where T is the number of ferns. To imrpove the running time, these randomized ferns are shared between our object detector and distracter detector. Each tracker controls the posterior probability by adding its positive and negative samples to the ferns according to the P-*constraints* and N-*constraints*, respectively as in [16]. The P-*constraints* force all samples close to the validated trajectory to have positive label, while N-*constraints* have all patches far from the validated trajectory labeled as negative. Different from [16], we avoid adding hard negative samples to avoid over-fitting. Also, during tracking, when the appearance of a distracter is different from our target, we discard it. Indeed, it helps to emphasize that our focus is on tracking a single target, not on multiple target tracking. This clearly explains the intuition: when several objects have similar appearance to our object, the target tracker pays attention to them; if these distracters change their appearance and no longer look like our object, they can be ignored.

Therefore, a sample is considered a distracter candidate if it passes the random ferns with a probability $Pr(y = 1|x_i) > 0.5$, and is not the target. How to determine a candidate as our target is discussed in Section 4. We maintain an M frames sliding window and count the frequency fd_k of a candidate k based on its appearance consistency spatial consistency related to the target. Then a candidate is classified as a distracter as follows

$$P_d(y_d = 1|x_i) = \begin{cases} 1 & if \quad fd_k > 0.5 \\ & and \quad d(x_i, M) > 0.8 \\ 0 & otherwise \end{cases} \tag{2}$$



(a) Detecting all supporters (b) Learning the active supporters (c) Avoid drifting to other regions while target is under occlusion (d) Reacquire the target with strong supporter model

Figure 3. Learning supporters. Active supporters are pink dots, passive supporters are in blue dots, object center is black dot.

where $P_d(y_d = 1|x_i)$ is the probability for a candidate i in a feature vector x_i having label y_d, while $d(x_i, M)$ is the confidence of this candidate evaluated by the template-based model of the target. The first condition allows to detect distracters which repeatedly co-occur with our target, while the second one helps to exploit distracters having very similar appearance to our target.

3. Selection of supporters

3.1. Description

We aim to build an efficient supporters set which helps to quickly verify the location of the target. Supporters are features which consistently occur around the object as shown in Figure 3. They also have a strong correlation in motion with our target. It is worth noting that our goal is tracking in unconstrained environment with several challenges such as frame-cuts, abrupt motion due to hand-held camera recording. It limits us from using some motion model to predict the location of a target based on the motion of the supporters as in [13] or of the auxiliary objects as in [27]. We also would like to emphasize that our candidate responses are obtained based on detection. The supporters are also detected from the local region around each candidate. After that, these supporter detection responses are matched with the ones from previous frames to find the co-occurrence between them and our target. In fact, from these results, the motion correlations are also inferred without using very complex motion models needed in unconstrained environments. Moreover, unlike the supporters proposed in [13] which are expensive to detect and match in the whole frame, our supporters are efficiently detected and matched around the locations of the very few candidates having high probability to be the target in each frame.

3.2. Selection

To detect supporters, we use the Fast Hessian Detector and employ SURF descriptor as in [4] to describe the region around them. We store all of these supporters in a sliding window of k frames ($k = 5$ in our implementation). There are two types of supporters: *active* and *passive*. The active supporters are the ones co-occurring with our target in high frequency $fs > 0.5$ within the sliding window, while passive ones are the rest. When there are regions having similar appearance to our target but not being tracked by distracter trackers, all of SURF features are detected around these regions. After that, they are matched to our supporter model, which basically are the latest descriptors of the supporters in the sliding window. Finally, the supporting score is computed as follows

$$S_i = \frac{n_{am}}{n_{ta}} \tag{3}$$

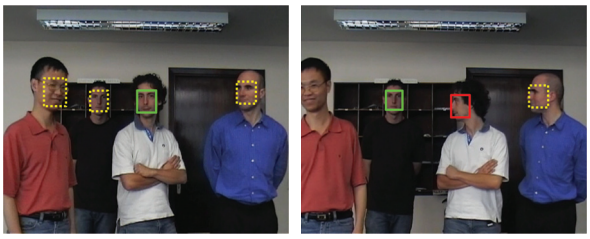
where n_{am} and n_{ta} are the numbers of active matched supporters and total active supporters in the model. A supporter model is considered strong if $S_i > 0.5$ and $n_{ta} > 5$ (to avoid the unstable information within non-textured regions around our target). Then all of the matched results are used to update the supporter model. Note that the unmatched results are also added to the model.

4. Context tracker

This section describes how the context tracker exploits context information while tracking, and takes advantage of them to avoid drift.

Basic tracker.

We use the P-N Tracker [16] as our basic target tracker with several extensions. First, we extend the randomized ferns to accept multiple objects; in fact, it is not equivalent to a multi-class classifier because each object preserves its own posterior probability while they may share the same object type as our target. Second, we applied our new 6bitBP [8] which helps to boost up the speed of the detector. Our 6bitBP makes use of the constant value of each whole patch during evaluation. Third, instead of using only the first initial patch as the object model, which is quite conservative and vulnerable to appearance changes, we use the online template-based object model as in [15]. However, we improve this model by constructing it in binary search tree using k-means. The model is iteratively split into two subsets to form a binary tree. By doing this, the computational complexity to evaluate a sample



(a) Start changing appearance (b) Exceeding the threshold, drifting.

Figure 4. Drifting to another object with the highest score.

is $O(log_n)$ instead of $O(n)$ when using Brute-force. This improvement is important in improving the running time because the online model linearly grows to adapt to appearance changes. It is worth noting that other tracking methods can also be extended using our concepts. However, we choose the PN-Tracker because it uses scanning window to search for all of possible candidates in the whole image which helps to explore the context at the same time. Also, the randomized forest is extendable to reduce the cost of initializing a totally new tracker for a distracter.

Exploiting distracters. As discussed in Section 2, distracters are regions which have similar appearance as our target. In our tracker, a testing sample confidence score is computed using Normalized Cross-Correlation (NCC) between it and the closest image patch in the object model. The region having the highest confidence is considered as the current target if its score is larger than a threshold $\theta = 80\%$. However, in practice, there are several other regions satisfying this condition. After we choose the best candidate as the tracking result (see Algorithm 1), all of other responses are associated to the distracter trackers using greedy association: the tracker producing higher confidence on a patch is associated with higher priority. The remaining regions trigger new distracter trackers. These trackers are formulated similarly to our basic tracker. However, to avoid the increasing number of unnecessary trackers, they are terminated whenever they lose their target.

Exploiting supporters. Assuming that we have the valid target at frame t, the supporters are extracted around the location of that target with a radius R. After that, a sliding window of $k = 5$ frames is used to store and match the previous supporters with the current ones. Each match makes the frequency of that supporter increase by 1.

Context tracker. As discussed in the "exploiting distracters" section, in practice, there are several candidates similar to our target with very high confidence score. In fact, the right candidate may not even obtain the highest score, especially when the appearance is changing (as shown in Figure 4). Without context, the tracker obviously switches to the one with the highest score. Also, in unconstrained environments, our target may leave the FoV, or be completely occluded by other objects. The tracker will simply switch to another region satisfying the threshold θ. Here, our tracker automatically exploits all the distracters and pays attention to them by tracking them simultaneously. Also, our tracker discovers a set of supporters to robustly identify the target among other similar regions. For more information please see Algorithm 1.

Algorithm 1 Context Tracker

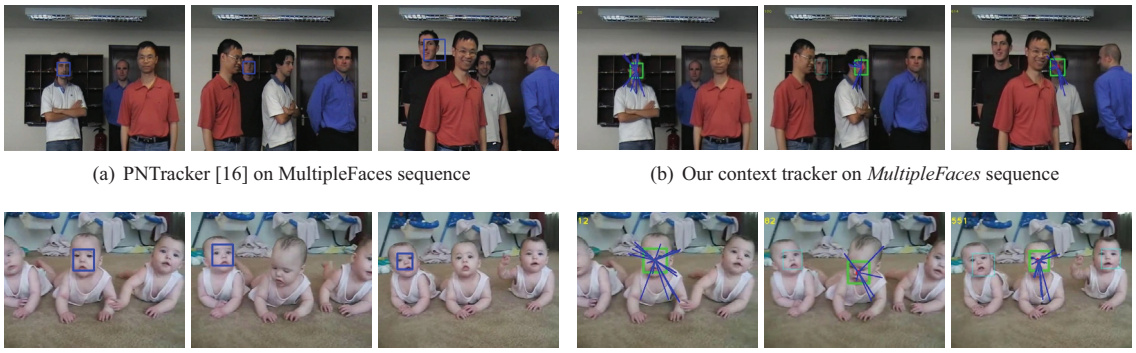
- init target T^*, $Distracters = \{\}$
- detect $Supporters$ around the T^*

while *run* **do**
  - detect object responses using *randomized ferns*
  - detect all possible candidates using *online template-based model*
  foreach *candidate* **do**
    - calculate the supporting score (Eq. 3)
    if *supporting score is strong* **then**
      - add candidate to *Strong Candidates*
    end
  end
  if *Strong Candidates* $\neq \{\emptyset\}$ **then**
    - $T* \leftarrow$ object with max confidence score
  end
  - run *greedy data association* to find T^* and $Distracters$ based on their confidence score.
  foreach *tracker* **do**
    - update its posterior probability (Eq. 1) in *randomized ferns*
    - update its *online template-based object model*
  end
  if $T*$ *has high confidence score* **then**
    - update $Supporters$ (active and passive) by matching with supporters around T^* .
    - update $Distracters$ with new regions has high confidence score $\neq T^*$
  end
end

5. Experiments

5.1. Experiment settings

In our implementation, we use 8 ferns and 4 6bitBP features per fern. All thresholds are fixed as described. The most important threshold is the one which validates the correct target. It is calculated by the *NCC* between a candidate and the online object model. It has been carefully chosen as *80%* according to the experiment demonstrated in [15], which also shown that *LOOP* event outperforms the other growing ones. Our scanning window starts searching the minimum region of 20x20. For a sequence of resolution



(a) PNTracker [16] on MultipleFaces sequence

(b) Our context tracker on *MultipleFaces* sequence

(c) PNTracker [16] on *Babies* sequence

(d) Our context tracker on *Babies* sequence

Figure 5. Comparison between PNTracker [16] and our context tracker on challenging sequences

320x240, the number of search windows is 100k while in 640x480, it is 600k, approximately.

It is worth noting that the complexity of our algorithm is affected by the number of supporters and distracters. In experiments, we observe the maximum number of active supporters is around 30-40 while that of distracters is around 10-15. Hence, we limit the maximum of the most similar distracters to 15, and that of supporters to 30 to guarantee a reasonable running time. Without code optimization, our C++ implementation of the context tracker runs comfortably at 10-25 fps on 320x240 sequences depending on the density of supporters and distracters, which means without using context information, our method runs at 25 fps, while in a heavy case ,the algorithm slows down to 10 fps. To show the performance of our tracker, the experiments are carried out in two different setups: one is to evaluate the contribution of context in our tracker, and the other is to compare ours with various state-of-the-art methods.

5.2. Context performance evaluation

We compare our tracker with and without context elements. The PNTracker is used as reference. It is worth noting that the implementation of PNTracker[1] is the combination implementation of [15, 16]. To emphasize the contribution of context in terms of distracters and supporters, we choose two very challenging sequences which contains similar objects moving: *Multiplefaces*, and *Babies*.

- The *Multiplefaces* sequence drawn from SPEVI data set[2] is very difficult with 4 people moving around. It contains several challenges such as out of plane rotation, total occlusion, target leaving FoV. It is hard to differentiate between the other faces and our target. While our tracker easily ignores the other object using distracters and supporters, the PNTracker occasionally switches to another face during tracking. To avoid the randomization effects, we run each tracker 5 times and observe several drift cases of PNTracker which makes it fail to recover. The results are shown in Figure 5.
- The *Babies* video shows a triplet of three babies playing on the ground. This sequence is really interesting and challenging because they all look alike. PNTracker jumps to the face of another baby as soon as the target has some appearance change. Our tracker successfully keeps following the right target till the end. Several results are demonstrated in Figure 5.

It is important to note that in most of the cases where no strong context exists, our tracker still shows overall better results than PNTracker and outperforms other state-of-the-art methods.

5.3. Comparison with other trackers

To demonstrate the performance of our context tracker we use several recent state-of-the-art methods for comparison including: FragTracker (FT) [1], MILTracker (MILT) [3], CotrainingTracker (CoTT) [28], PNTracker [16], DNBSTracker (DNBST) [19], VTDTracker [18]. All codes com from the original authors.

The chosen data set includes several challenging sequences: *Motocross* and *Carchase* in [16], *Vehicle* in [28], Liquor[3], *ETHPedestrian*[4], *Multifaces*[2], *Clutter* and *Scale*[5], *Animal* used in [18], *and Girl* in [3]. They contain occlusion and object leaving FoV (*Motocross, Carchase, Vehicle, ETHPedestrian, Multifaces, Girl*), very cluttered background (*Carchase, Liquor, ETHPedestrian, Multifaces*) out-of-plane rotation (*Carchase*, *Vehicle*, *Multifaces*, *Girl*), abrupt motion (*Motocross, Clutter, Scale, Animal*), motion blur (*Liquor, Clutter, Animal*). Several of them are recorded in unconstrained environments such as *Motocross, Vehicle, ETHPedestrian, Carchase,* and *Animal*.

[1] PNTracker: http://info.ee.surrey.ac.uk/Personal/Z.Kalal/

[2] SPEVI dataset: http://www.eecs.qmul.ac.uk/ andrea/spevi.html

[3] PROST dataset: http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/pros

[4] ETH dataset: http://www.vision.ee.ethz.ch/ aess/dataset/

[5] http://www.vision.ee.ethz.ch/boostingTrackers/index.htm

Video Sequence	Frames	FT	MILT	CoTT	DNBS	VTD	PNT	**Ours**
Animal	72	69	9	*8*	19	**6**	37	9
Carchase	5000	lost @ 355	lost @ 355	lost @ 409	lost @ 364	lost @ 357	lost @1645	**24**
Clutter	1528	lost @ 1081	lost @ 413	9	*6*	*6*	**4**	*6*
ETHPedestrian	874	lost @ 95	lost @ 95	lost @ 95	lost @ 635	lost @ 95	**10**	*16*
Girl	502	lost @ 248	30	**14**	39	69	19	*18*
Liquor	1407	lost @ 47	lost @ 288	30	lost @ 404	lost @ 404	*21*	**10**
Motocross	2665	lost @137	lost @ 485	lost @ 591	lost @ 10	lost @ 10	**10**	*12*
Multifaces	1006	lost @ 64	lost @ 64	lost @ 394	lost @ 64	lost @ 64	lost @ 97	**26**
Scale	1911	8	11	6	lost @ 269	*3*	6	**2**
Vehicle	946	lost @ 679	lost @ 481	*9*	lost @ 517	lost @ 517	**8**	**8**
Speed (fps, on 320x240)		1.6	14	2	7	0.2	12*	10

Table 1. Average center location error (pixels). Performance comparison between the trackers (FT: FragTracker [1], MILT: MILTracker [3], CoTT: Co-Tracker [28], DNBS: DNBSTracker [19], VTD: Visual Tracking Decomposition [18], PNT: PNTracker [16], and Ours: Our context tracker) in different challenging video sequences. The best performance is in **bold**, the second best is in *italic*. The number in blue color indicates the frame number when the tracker gets lost. The * indicates that the method was implemented on Matlab using C-Mex.

We discuss the metric used in our experiment: the average center location error used in [3, 18]. The reason we adopt this metric instead of the detection-criterion of the VOC challenge [10] is that different trackers require different initial bounding box depending on the nature of their method. For example, template-based methods such as PNTracker [16], FragTracker [1] are initialized by a tight bounding box, while a detection-based method such as Co-Tracker [28] needs a loose one. However, because our chosen data set is very challenging with a number of long-term sequences, most current methods fail somewhere in the middle of a sequence. Therefore, we note the frame number where a tracker starts to lose the object and never reacquires. It means we accept the result of a tracker even when it fails to get the right target in several frames before reacquisition happens. A target is considered "lost" if the overlapping region between its bounding box and the ground-truth is less than 50%.

The quantitative comparisons are shown in Table 1. The running time comparison (in the last row) is for a raw reference as different methods have different search range which impacts the speed greatly. For example, FragTracker [1] and MILTracker [3] use exhaustive search in a small area. Increasing the range slows down the speed significantly as the number of candidates grows. In Co-Tracker [28], the use of particle filter is also affected by the search range implicitly influenced by the number of the particles. PN-Tracker [16] and our tracker scan the whole image to find candidates. The running time of our tracker also depends on the number of distracters discovered by the tracker. Those distracters are few as we observed in general cases. Some snapshots of our tracker and others are given in Figure 6. It shows that our tracker has overall better performance than PNTracker [16] with the help of context, and outperforms all other approaches. Although most of them may work well in controlled environments, it is difficult for them to consistently follow the target in long-term sequences and in unconstrained environments. There are some large numbers in our results ("Carchase", "Motocross") is because it reacquires the object in several frames later than the ground truth, which makes the overall score look not good when we calculate the error using its previous position. It is also important to note that the maximum number of frames run by VTDTracker is 1000 frames; hence, its results shown in Table 1 are the average of 1000 frames at the maximum.

6. Conclusion

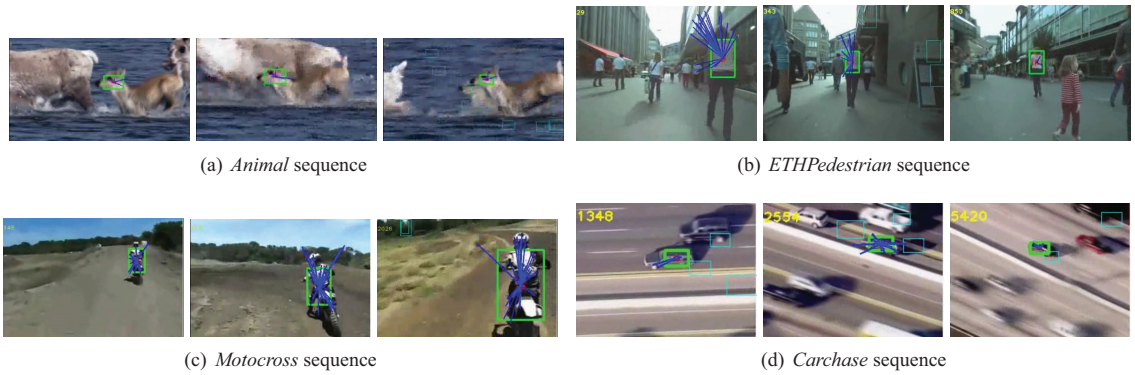
We have proposed a novel tracking method which automatically explores the context information in two semantic terms: distracters and supporters. Our tracker can successfully take advantage of the context to overcome challenges in tracking in unconstrained environments with occlusion, abrupt motion, motion blur, frame-cut. Moreover, with the context, our tracker explicitly handles the situation where several objects similar our target are present. With extensive experiments, we demonstrate that our tracker outperforms most of current state-of-the-art methods.

Currently, our tracker cannot track articulated objects, nor handle fast appearance change. We would like to improve our tracker to handle these issues. Also, we expect to apply our tracker in specific systems such as face tracking and pedestrian tracking systems.

Acknowledgements This research was funded, in part, by MURI-ARO W911NF-06-1-0094. The first author was also supported by the Vietnam Education Foundation.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006.



(a) *Animal* sequence (b) *ETHPedestrian* sequence

(c) *Motocross* sequence (d) *Carchase* sequence

Figure 6. Some snapshots of our context tracker on several sequences.

[2] S. Avidan. Support vector tracking. In *PAMI*, volume 26, pages 1064–1072, 2004.

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, pages 983–990, 2009.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *CVIU*, volume 110, pages 346–359, 2008.

[5] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *ICCV*, pages 1–8, 2007.

[6] L. Breiman. Random forests. In *ML*, volume 45, pages 5–32, 2001.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *PAMI*, volume 25, pages 564–577, 2003.

[8] T. B. Dinh, N. Vo, and G. Medioni. High resolution face sequences from a PTZ network camera. In *FG*, 2011.

[9] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, pages 1271–1278, 2009.

[10] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc2007) results.

[11] J. L. Fan, Y. Wu, and S. Y. Dai. Discriminative spatial attention for robust tracking. In *ECCV*, pages 480–493, 2010.

[12] H. Grabner, C. Leistner, and H. Bishof. Semi-supervised online boosting for robust tracking. In *ECCV*, pages 234–247, 2008.

[13] H. Grabner, J. Matas, L. V. Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, pages 1285–1292, 2010.

[14] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. In *IJCV*, volume 29, pages 5–28, 1998.

[15] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *OLCV*, pages 1417–1424, 2009.

[16] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, 2010.

[17] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *CVPR*, pages 685–692, 2010.

[18] J. S. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[19] A. Li, F. Tang, Y. W. Guo, and H. Tao. Discriminative nonorthogonal binary subspace tracking. In *ECCV*, pages 258–271, 2010.

[20] L. J. Li, R. Socher, and L. F. Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*, pages 2036–2043, 2009.

[21] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*, pages 2953–2960, 2009.

[22] X. Mei and H. Ling. Robust visual tracking using L1 minimization. In *ICCV*, pages 1436–1443, 2009.

[23] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov network. In *CVPR*, pages 975–982, 2009.

[24] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. In *PAMI*, volume 32, pages 448–461, 2010.

[25] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR*, pages 1–8, 2007.

[26] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What helps where and why? semantic relatedness for knowledge transfer. In *CVPR*, pages 910–917, 2010.

[27] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. In *IJCV*, volume 77, pages 125–141, 2008.

[28] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. In *PAMI*, volume 31, pages 1195–1209, 2009.

[29] Q. Yu, T. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers and discriminative trackers. In *ECCV*, pages 678–691, 2008.

[30] Q. Yu and G. Medioni. Multiple-target tracking by spatiotemporal monte carlo markov chain data association. In *PAMI*, volume 31, pages 2196–2210, 2009.