Online Visual Tracking Using Temporally Coherent Part Clusters

Wenbo Li[1] Longyin Wen[2*] Mooi Choo Chuah[1] Yi Zhang[3] Zhen Lei[2] Stan Z. Li[2]
[1]Department of Computer Science and Engineering, Lehigh University
{wel514,chuah}@cse.lehigh.edu

[2]National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences
{lywen,zlei,szli}@nlpr.ia.ac.cn

[3]School of Computer Software,
Tianjin University
yizhang@tju.edu.cn

Abstract

Recent advances in visual tracking have focused on handling deformations and occlusions using the part-based appearance model. However, it remains a challenge to come up with a reliable target representation using local parts, and hence existing trackers continue to face drifting problems. To deal with this challenge, we propose a robust online model, formulating the tracking task as a problem of identifying Temporally Coherent Part (TCP) clusters. Specifically, we pose the TCP clusters identification task as a dense neighborhoods searching problem using a relational hypergraph in which the relationship among multiple temporal local parts is encoded as the affinity value of a hyperedge connecting them. Such high-order relationships among multiple local parts across the temporal domain make our tracker more robust towards deformations and occlusions. Extensive experiments on various challenging video sequences demonstrate that our TCP-based method performs better than the state-of-the-art methods.

1. Introduction

Visual tracking is one of the most important components in many computer vision related applications, such as surveillance, human computer interaction, behavior analysis, etc. Although it has been studied for decades, however, given a video stream, tracking an arbitrary object is still a challenging task because of many factors, including deformations, occlusions, appearance variations, lighting changes and little prior knowledge about the shape and appearance of an object. Both deformations and occlusions which exist in many real world tracking problems are considered great challenges in visual tracking. However, only a few works [7, 33, 5] focus on dealing with these two challenges simultaneously. In the work of Chockalingam *et al.* [7], a target contour is tracked to cope with deformations and occlusions. However, this method requires a target that needs to be tracked to keep moving, so it is not robust under the scenario where a target may suddenly stop. Yao *et al.* [33] design a tracker using an online extension of the Deformable Part Model (DPM). Unfortunately, the inherent restrictions of DPM make it hard to determine a reliable target representation and hence it is only robust to small deformations. Cai *et al.* [5] represents a target as a structural graph encoded with the appearance and geometric information. Then, they leverage the pairwise relationships between local parts in two consecutive frames to assist a tracker so that they can deal with the impacts of deformations and occlusions. However, the pairwise relationships are not reliable enough to match truly similar local parts and thus target drifting continues to occur. Hence, having a reliable target representation is the key to successful tracking.

In this paper, we address the issues of deformations and occlusions simultaneously by determining a reliable target representation. To track a target more reliably, we propose the concept of Temporally Coherent Part (TCP) and use TCPs to model a target. Thus, a target can be represented using several TCPs which capture high-order temporal relationships among local parts identified in several consecutive video frames. Such TCPs are identified based on the similarities of their appearance and motion patterns. We design a TCP cluster identification method which is based on the recently proposed dense neighborhoods searching method [18]. Such dense neighborhoods searching method has been proven to be effective in multitarget tracking [31].

The contribution of this work is three-fold: (1) We propose using TCPs to reliably represent a target and use identified TCPs in consecutive video frames to track this target. (2) Two inherent patterns, i.e., appearance and motion patterns, are exploited together to construct a relational hy-

*Corresponding author.

978-1-4799-6683-7/15 $31.00 © 2015 IEEE
DOI 10.1109/WACV.2015.9

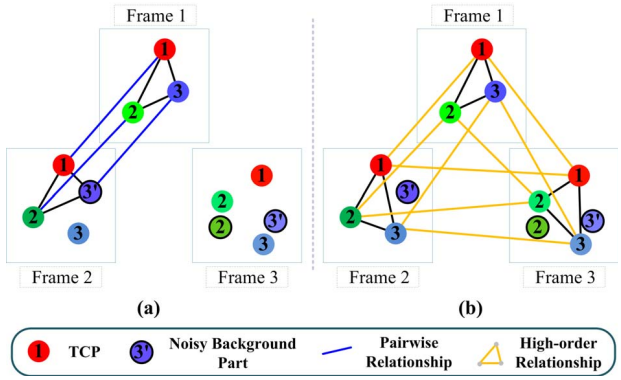

Figure 1. (a) illustrates the pairwise relationship based approach in [5], and (b) our TCP cluster based approach which considers high-order relationship. The circles denote local parts and their color similarity indicates the inherent pattern similarity.

pergraph for identifying TCPs. (3) We conduct extensive experiments using various challenging datasets to demonstrate that our proposed method performs well even in the presence of deformations and occlusions.

2. Related Work

Earlier trackers [8, 21] mostly represent a target using a bounding box, and often a holistic appearance model is used. Although these classical methods are robust to local deformations (as in face tracking) to some extent, they are unsuitable for tracking non-rigid objects that undergo drastic deformations. Some algorithms [7, 20] can cope with deformations by tracking the contours of targets. However, most of them require targets that need to be tracked to keep moving or need prior shape knowledge [9].

Many existing methods focus on improving the robustness of a tracker to occlusions. Adam *et al.* [3] use a patch based appearance model whose dynamic patch template configuration can model spatial structures, and thus their tracker is robust to partial occlusions. Some works [19, 34] model the target as a sparse representation of historical appearance features, which are relatively insensitive to occlusions but such approaches cannot deal with large structural deformations. Kwon and Lee [15] decompose their tracker into several small trackers, enhancing the robustness of the tracker to occlusions. Both Babenko *et al.* [4] and Wen *et al.* [30] present an online method based on multiple instance learning so that the effects of occlusions can be suppressed during the learning process. Others [32, 28, 29] reduce the occlusion impacts by mining the context information to assist tracking. Compared with the above approaches, our tracker not only can handle severe occlusions but cope well with large structural deformations.

Intuitively, color-histogram based model is a feasible solution to handle large structural deformations because the color histogram is independent of the target shape. However, although color-histogram based model covers the geometric variations to some extent, it loses the spatial information of target parts, making its separate use in tracking unstable. In recent years, part based model has attracted much attention for handling deformations due to the following two reasons: (1) Local parts are less sensitive to structural deformations; (2) Compared to rigid templates, the part based representation is expected to contain fewer background regions. There has been continuous efforts in identifying how a target can be represented using local parts. Wang *et al.* [27] represent a target as a collection of individual local parts with few structural constraints. A star model has been proven effective in representing a target for tracking purposes [3, 14, 32, 25]. However, such star model based algorithms model the dependence between local parts and the target center while they ignore those dependencies between local parts. To exploit the inner geometric structure of a target, some researchers model local parts together with their relationships as an undirected graph and then the tracking problem is formulated as a matching problem between an existing undirected graph and several candidate graphs extracted from a subsequent video frame [5, 26].

Recently, some researchers extend the superpixel idea from still images to video segments by considering the issue of temporal coherence [1, 23]. They define a new concept called supervoxel which is a stack of superpixels that represent a video segment. Although the supervoxel based algorithms also mine the temporal coherence of superpixels as in our approach, our method is still quite different from theirs in two aspects: (1) The supervoxel does not have the concept of foreground and background objects while our TCP clusters based algorithm is designed to model a target as a foreground object; (2) The supervoxel can be viewed as a stack of temporally coherent superpixels, i.e., each frame of the video segment that a supervoxel spans must have one superpixel belonging to that supervoxel. By contrast, our algorithm does not set such hard limit for TCP clusters because we believe such relaxation can boost the robustness of our tracker towards deformations and occlusions.

Our work is also related to [11, 10]. Both schemes adopt the Hough voting to detect an approximate target position, and then segment the foreground from the background. However, their schemes work at pixel level while ours works on superpixel level. Thus, our approach works better than theirs when deformations and occlusions occur.

3. TCP Clusters Based Tracker

To our best knowledge, the part based tracker of Cai *et al.* [5] is the most similar approach to ours. We highlight the difference between our approach and that in [5] in Fig. 1. In Fig. 1, we show that local parts are identified to represent a target in three consecutive video frames, Frame 1 to Frame 3. A local part in one frame which is similar to one in

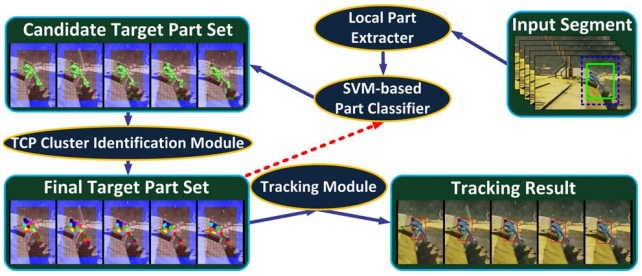

Figure 2. Overall tracking procedure for one video segment. The green rectangle and the blue dotted rectangle represent the search window and the sample window respectively. The green colored local parts are in the *Candidate Target Part Set*. Target parts within a TCP cluster are colored the same in the *Final Target Part Set*.

another frame are marked with the same numeric number. For example, the red circles marked 1 in Frame 1, Frame 2 and Frame 3 are all similar local parts. Some local parts may look similar but not truly similar local parts (i.e. can be used to represent a target), *e.g*. the circles marked 3 with light blue color are truly similar local parts but those marked 3' with darker blue color and darkened edge are noisy background parts.

In [5], the authors obtain pairwise relationships between matching local parts in two consecutive frames to assist in target tracking. Such a relationship is constructed based on the appearance and geometric information. As shown in Fig. 1(a), their method sometimes matches noisy background part with the target part. However, as shown in Fig. 1(b), our approach excludes noisy background parts when identifying the high-order relationships among multiple local parts across the temporal domain. We define the term Temporally Coherent Part (TCP) to represent target parts identified in k-consecutive frames ($k \geq 3$) that are similar. For example, circles marked 1 in Frame 1, Frame 2 and Frame 3 form one TCP cluster. We can use such identified TCP clusters to represent a target more precisely since (a) the high-order relationships we use allow us to impose stronger constraints on inherent patterns of the associated TCPs; (b) such high-order relationships also allow us to exploit motion properties of these TCPs which is impossible for the pairwise relationship based method in [5]. In addition, different from [5] in which the geometric properties are exploited, we assume that local parts within the same frame has no relationships with one another and hence our method has larger tolerance against drastic deformations between consecutive video frames.

The overall procedure of our proposed tracking algorithm for one video segment (which consists of multiple consecutive video frames) is shown schematically in Fig. 2. First, a video sequence is segmented into different overlapping video segments with each segment consisting of k frames. For each video segment, there is a search window and a sample window. The search window includes the target to be tracked. Sample window is an extended search window where local parts will be extracted based on the appearance patterns by *Local Part Extractor*. The extracted local parts are fed into *SVM-based Part Classifier* to identify local parts which can be included in *Candidate Target Part Set*. *Candidate Target Part Set* is then fed into *TCP Cluster Identification Module* which identifies TCP clusters to be included in *Final Target Part Set*. The TCP clusters are identified via a dense neighborhood searching process using a relational affinity hypergraph. The TCP cluster identification algorithm will be further explained in Section 4. In a nut shell, this process builds a hypergraph per video segment where a node corresponds to a candidate target part and the relationships among the candidate target parts are described using hyperedges with weights that represent the likelihood of them belonging to the same TCP cluster. We denote the constructed hypergraph for the k-th video segment using $\mathcal{H}_k = (\mathcal{X}_k, \mathcal{R}_k)$, where $\mathcal{X}_k$ and $\mathcal{R}_k$ denote the node and hyperedge set of the hypergraph respectively. The node set $\mathcal{X}_k$ consists of all nodes in the hypergraph, i.e., $\mathcal{X}_k = \{x_{i,k}\}_{i=1}^{n}$, where $x_{i,k}$ is the i-th node and n is the number of nodes in the hypergraph. $\mathcal{H}_k$ captures the relationships among the candidate target parts.

All identified TCP clusters then form the *Final Target Part Set* which is used in the *Tracking Module* to track a target. The *Final Target Part Set* is used to estimate the location and scale of the target in the current video segment. Such information will be used to update the search and sample windows for the next video segment so that we can better track the target. The output from the TCP clustering process in the current video segment will also be used to update the *SVM-based Part Classifier* for the next video segment. Such update process is illustrated as a red dotted arrow in Fig. 2.

4. TCP Clusters Identification

4.1. Relational Affinity Hypergraph Construction

Since we represent each target part in *Candidate Target Part Set* as a node in the relational affinity hypergraph, we will use the node and target part interchangeably in this section. We first construct a relational affinity hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{R})$ for each video segment. To simplify notation, we drop the segment index k. We use $\boldsymbol{r}^m = (x_1, \ldots, x_m)$ to denote a hyperedge with m nodes. To minimize the computation cost of constructing such a relational hypergraph, we use a distance threshold to constrain its construction process: $\forall\ \boldsymbol{r}^m \in R$, s.t., $\forall\ x_i, x_j \in \boldsymbol{r}^m$, $dist(x_i, x_j) < \varepsilon$. $dist(\cdot,\cdot)$ is the geometric distance between two temporal local parts and $\varepsilon = \gamma\mu_s$ ($\mu_s = 1.5$ is the constant), where γ approximates the average diameter of superpixels found within the search window. $\gamma = \sqrt{W_s \cdot H_s / \theta_s}$, where W_s and H_s denote the width and height of the search window

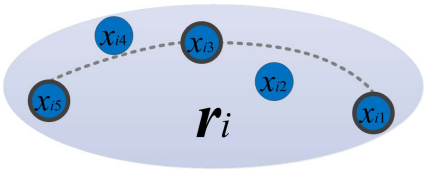

Figure 3. The illustration of hyperedge $\boldsymbol{r}_i$ in which 5 nodes are involved. The nodes which are connected by the fitted trajectory (illustrated as a dotted line) are marked with darken edge.

respectively. θ_s denotes the number of superpixels in the search window.

To determine which hyperedge is more important than others for identifying TCP clusters, we need to assign a weight to each hyperedge to indicate its importance. Before we describe how to compute such weights, we first define a few terminologies which can aid our explanation. In Fig. 3, we illustrate a hyperedge $\boldsymbol{r}_i$ which consists of 5 local parts. Each local part, x_{ij} can be described using $v(x_{ij})$ which is the color histogram of x_{ij}. This color histogram represents appearance information which can help us discriminate the foreground target from the background. Color-histogram based model is also suitable for coping with deformations since it is independent of the shape and structure of a target. Since the hyperedge captures temporal information, we can construct a fitted trajectory by connecting $\lceil \frac{m}{2} \rceil$ nodes. We represent this fitted trajectory as $\widehat{\mathcal{T}}_{\boldsymbol{r}_i}$.

The weight of a hyperedge is an affinity value which is dependent on two factors: (a) the appearance of the local parts and (b) the motion patterns of these parts within the hyperedge. Thus, the affinity value $\mathcal{S}(\boldsymbol{r})$ of a hyperedge $\boldsymbol{r}$ can be computed using the following equation which is a weighted sum of an appearance and motion related term:

$$\begin{aligned} \mathcal{S}(\boldsymbol{r}) &= \omega_1 \cdot \sum_{x_i, x_j \in \boldsymbol{r}} cos\left(v\left(x_i\right), v\left(x_j\right)\right) \\ &+ \omega_2 \cdot \exp\left(-\sigma \cdot \sum_{x_i \in \boldsymbol{r}^{rem}} \| \ell\left(x_i\right) - \ell\left(\widehat{\mathcal{T}}_{\boldsymbol{r}}\right) \|^2\right), \end{aligned} \tag{1}$$

where the first summand denotes the appearance term and the second one denotes the motion term. ω_1 and ω_2 are two preset weight parameters. $cos(\cdot, \cdot)$ denotes the cosine distance between two feature vectors.

The second motion related term is motivated by the following: Typically, in a target tracking application, one would like to see a smooth target trajectory. Since each hyperedge captures information about the motions of the local parts, we want to see such motions being close to a fitted trajectory which is constructed assuming smooth motion of related local parts in that hyperedge.

$\ell(x_i)$ in Equ. 1 denotes the position of local part x_i. $\widehat{\mathcal{T}}_{\boldsymbol{r}}$ denotes the fitted trajectory and $\boldsymbol{r}^{rem}$ denotes the local part set excluding those which form the fitted trajectory. σ is a scalar parameter which ensures that the value of the motion term is in the same ballpark as the appearance term.

4.2. Dense Neighborhoods Searching

The core task of *TCP Cluster Identification Module* is to identify TCP clusters via searching dense neighborhoods on a relational affinity hypergraph. In our approach, a TCP cluster is a dense neighborhood that consists of multiple nodes which yield the maximum normalized affinity value. The number of dense neighborhoods and their sizes are considered as hidden variables in the dense neighborhood searching process and the final values for these two variable will only be obtained after the post processing step, which will be introduced in the next subsection.

To ensure that all dense neighborhoods can be identified, we set every node in the relational hypergraph as a starting point and search dense neighborhoods around each starting point iteratively. Intuitively, if a node belongs to a dense neighborhood, the affinity value between this node and other nodes in this dense neighborhood will usually be large and this dense neighborhood will as a result be identified. Otherwise, this node will be treated as a false positive to this dense neighborhood hence it will not be included as a member of this dense neighborhood in the final results. For a starting point x_o, we intend to determine its neighbors $\mathcal{C}(x_o)$ which form a dense neighborhood with x_o. The optimization problem is formulated as

$$\begin{aligned} &\mathcal{C}^*\left(x_o\right) = \arg\max_{\mathcal{C}(x_o)} \mathcal{M}(\{x_o\} \cup \mathcal{C}(x_o)) \\ &\text{s.t.} \quad \mathcal{C}\left(x_o\right) \subset \mathcal{X},\ x_o \notin \mathcal{C}\left(x_o\right),\ |\mathcal{C}\left(x_o\right)| = k, \end{aligned} \tag{2}$$

where $\mathcal{M}(\{x_o\} \cup \mathcal{C}(x_o))$ denotes the normalized affinity measure function. Equ. 2 can be further formulated as a combinatorial optimization problem which can be solved with the pairwise updating mechanism discussed in [18, 17]. We refer readers to [18, 17] for more details regarding how to solve this optimization problem.

4.3. Post Processing

Intrinsically, the purpose of tracking is to estimate the state of a target, i.e., its centered location and scale. To obtain a reliable target representation using TCP clusters, we need a post processing phase, which comprises of two steps (a) filter out redundant TCP clusters, and (b) add additional relevant local parts excluded by the dense neighborhood searching process. The first step is referred to as the refinement step while the second is referred to as the recovery step. The target part set obtained after the refinement step is denoted as Ψ_t while the newly identified local parts from the recovery step forms Ψ_r set. The final TCP clusters after the post processing phase then constitute our *Final Target Part Set* denoted as Φ. This final target part set Φ is the union of Ψ_t and Ψ_r, i.e., $\Phi = \Psi_t \bigcup \Psi_r$.

Refinement. As stated in the previous subsection, we set each node in the relational hypergraph as a starting point to search dense neighborhoods (TCP clusters). Consequently, one node may appear in multiple clusters in the searching result. Hence we need this refinement step to filter out redundant TCP clusters. After the dense neighborhoods searching step, we obtain a TCP cluster set $\widetilde{\Psi}_t = \{\widetilde{\psi}_{t,i}\}_{i=1}^{n}$ and their associated normalized affinity values. $\widetilde{\Psi}_t$ is sorted to produce an ordered TCP cluster set $\widehat{\Psi}_t = \{\widehat{\psi}_{t,i}\}_{i=1}^{n}$ whose elements are placed in descending order of their normalized affinity values. We initialize the TCP set $\Psi_t = \emptyset$ at first and add the TCP cluster in $\widehat{\Psi}_t$ sequentially. For the i-th component $\widehat{\psi}_{t,i} \in \widehat{\Psi}_t$, we check whether it intersects with existing TCP clusters in Ψ_t. If there is no intersection between $\widehat{\psi}_{t,i}$ and all TCP clusters in Ψ_t, then we add $\widehat{\psi}_{t,i}$ directly to Ψ_t, i.e., $\Psi_t \leftarrow \Psi_t \bigcup \{\widehat{\psi}_{t,i}\}$. Otherwise, we remove the intersection from $\widehat{\psi}_{t,i}$ to get $\widehat{\psi}^*_{t,i}$ and check whether $|\widehat{\psi}^*_{t,i}| \geq m$, recalling from Section 4 that m denotes the hypergraph order. If $|\widehat{\psi}^*_{t,i}| \geq m$, we add $\widehat{\psi}^*_{t,i}$ directly to Ψ_t. Otherwise, we argue that the coherence of local parts in $\widehat{\psi}^*_{t,i}$ is so weak that we can ignore $\widehat{\psi}^*_{t,i}$ and move to the next iteration. Note that although the dense neighborhoods searching algorithm is reliable, we still adopt a conservative approach by limiting the total number of TCP clusters retained in Ψ_t. We denote this maximun size as θ_t i.e. $|\Psi_t| \leq \theta_t$. For each dataset, $\theta_t = \mu_t \sqrt{W_t \cdot H_t / \gamma^2}$ ($\mu_t = 0.65$ is the constant), where W_t and H_t denotes the width and height of the initial bounding box respectively and γ denotes the average diameter of superpixels.

Recovery. All local parts which are excluded from the dense neighborhoods searching process form the non-TCP pool. There exist local parts in this non-TCP pool which are very likely to be foreground parts and hence these parts need to be included into *Final Target Part Set*. Recall that the *SVM-based Part Classifier* is used to identify relevant local parts which are used to represent a target of interest, thus we use the *SVM-based Part Classifier* again to identify useful local parts from the non-TCP pool. Specifically, we include local part x_i into Φ_r if $p(y_{x_i} = 1|v(x_i)) > 0.3$, where $p(y_{x_i} = 1|v(x_i))$ is the likelihood value of x_i being a foreground part given by *SVM-based Part Classifier* and $v(x_i)$ is the appearance feature of x_i.

5. Tracking

Given the reliable target representation (*Final Target Part Set* Φ), the tracking task can be accomplished by determining the optimal state of the target, including the optimal center l_c^* and scale $s^* = (w, h)$, where w and h denotes the width and height of the bounding box respectively.

To obtain the optimal state of the target as in [5], we use a coarse-to-fine strategy involving two phases. Phase One estimates a rough target center by calculating the weighted mean of the center of target parts in Φ:

$$l_c = \frac{1}{S_\omega} \sum_{\mathcal{P}_i \in \Phi} \omega_{\mathcal{P}_i} l_{\mathcal{P}_i}, \tag{3}$$

where $l_{\mathcal{P}_i}$ denotes the center of the i-th part in Φ, $\omega_{\mathcal{P}_i}$ denotes the probability of $\mathcal{P}_i$ to be part of the target and $S_\omega = \sum_{\mathcal{P}_i \in \Phi} \omega_{\mathcal{P}_i}$. Given a rough target center l_c, Phase Two modifies the target center with a local perturbation term δ to a visually better location and adjusts the target scale so that the bounding box can cover more foreground regions. The optimal scale s^* and δ^* can be computed as follows:

$$(\delta^*, s^*) = \arg\max_{\delta, s} \left\{ \alpha \cdot \theta^{pos}\left(l_c + \delta, s\right) - \theta^{neg}\left(l_c + \delta, s\right) \right\}, \tag{4}$$

where $\theta^{pos}(l_c+\delta, s)$ and $\theta^{neg}(l_c+\delta, s)$ respectively denotes the number of positive pixels and negative pixels within the bounding box located at $l_c + \delta$ with scale s. If a pixel is located inside the region a target part covers, it is labeled as a positive pixel; Otherwise, this pixel is labeled as a negative one. α is the term to boost the influence of positive pixels. Then, the optimal target center is $l_c^* = l_c + \delta^*$ and the optimal target scale is s^*. The optimal target center and scale will be used to update the search and sample windows (defined in Section 3) for the next video segment so that we can better track the target.

6. Experiments

We conducted experiments to compare our approach with other existing works using 16 challenging video sequences, all of which are used in prior works, including *lemming* [24], *waterski* [12], *lipinski* [12], *yunakim* [12], *football* [22], *kwan* [22], *transformer* [14], *gymnastics* [14], *diving* [14], *dancer* [25], *bluecar* [5], *avator* [5], *up* [5], *neymar* [5], *cliff-dive* [10] and *torus* [6]. These sequences include many challenging factors which can degrade the tracking accuracy: structural deformation, severe occlusion, abrupt movement, illumination variations, complex background and large scale changes. We compared our tracker to 12 algorithms, including 6 bounding box based trackers [4, 16, 13, 19, 15, 34], 5 part based trackers [3, 25, 14, 27, 5], and a pixel based tracker [10]. As a pixel can be regarded as a degraded part, [10] can also be considered as a part-based tracker. We use two criteria for comparing different schemes, namely (a) average center error in pixels (lower score means better performance) and (b) tracking success rate (higher score means better performance). Note that in each video frame, the tracking is considered successful if the PASCAL VOC overlap measure $\frac{R_T \cap R_{GT}}{R_T \cup R_{GT}}$ is above 0.5, where R_T is the rectangle area of the bounding box from the algorithm and R_{GT} is the rectangle area of the ground truth bounding box.

Table 1. Average center errors (in pixels) comparison.

Seq.	[4]	[16]	[13]	[19]	[15]	[34]	[3]	[25]	[14]	[27]	[10]	[5]	Ours
lemming	14.9	128	167	140	92.4	19.7	82.8	107	158	**7.15**	12.2	8.61	12.1
waterski	17.1	34.1	20.8	42.6	46.0	13.4	78.5	16.0	116	9.57	8.91	8.94	**7.35**
lipinski	33.8	90.8	109	46.6	16.5	23.8	50.5	30.9	14.1	12.3	10.2	**9.17**	9.32
yunakim	59.4	142	39.4	70.6	28.2	46.6	50.4	27.0	-	16.8	13.9	15.6	**13.5**
football	102	236	197	146	118	127	75.8	33.1	30.2	179	84.4	**8.38**	9.87
kwan	15.6	48.0	96.5	48.0	34.3	41.7	13.1	9.21	42.3	15.3	9.58	6.87	**6.57**
transformer	47.7	139	25.5	269	42.9	29.2	36.6	141	23.2	**10.1**	45.0	14.6	10.7
gymnastics	10.7	76.2	17.4	34.8	10.0	56.1	9.82	15.9	10.5	20.0	19.46	**6.41**	7.81
diving	95.6	82.1	102	93.6	65.1	46.7	74.1	67.8	**10.6**	67.6	32.8	11.7	14.2
dancer	16.3	15.1	13.3	10.3	10.8	14.9	19.3	19.1	18.9	**6.62**	11.7	7.14	8.13
bluecar	130	83.3	48.1	90.6	15.2	99.6	92.2	80.8	186	20.1	13.4	**10.5**	11.5
avatar	107	163	162	261	113	161	139	125	18.3	18.3	16.0	10.9	**10.8**
up	150	57.3	34.0	59.0	20.0	23.2	149	66.7	55.0	37.7	79.2	7.33	**6.20**
neymar	214	203	25.8	169	8.13	13.6	-	106	-	6.08	7.72	**5.88**	9.22
cliff-dive	16.1	26.9	55.5	20.4	15.6	15.5	35.2	88.3	-	11.9	17.5	9.98	**8.89**
torus	64.8	57.9	47.1	56.7	3.43	37.0	51.9	-	56.4	5.94	40.5	**2.90**	3.43

Table 2. Successful frame rate (%) comparison.

Seq.	♯	[4]	[16]	[13]	[19]	[15]	[34]	[3]	[25]	[14]	[27]	[10]	[5]	Ours
lemming	**1336**	83.2	21.3	17.5	9.7	38.1	82.5	54.9	38.1	9.0	**93.3**	42.9	**93.3**	92.5
waterski	**95**	60.0	69.5	65.3	61.1	55.8	68.4	46.3	73.7	5.3	89.5	72.6	93.7	**96.5**
lipinski	**660**	47.0	5.3	31.8	20.5	55.3	24.2	34.1	16.7	28.8	13.6	82.6	76.5	**94.7**
yunakim	**571**	13.0	4.3	11.3	2.6	11.3	2.5	9.6	36.1	-	51.3	34.8	87.7	**99.1**
football	**958**	17.2	1.6	3.7	19.3	17.7	14.6	18.8	21.4	24.7	22.4	27.4	**94.3**	85.9
kwan	**751**	66.2	22.5	17.9	36.4	36.4	34.4	79.5	89.4	4.0	31.8	81.5	96.7	**100.0**
transformer	**124**	38.7	39.5	40.3	30.7	40.3	46.8	40.3	25.8	62.9	**100.0**	46.8	**100.0**	**100.0**
gymnastics	**763**	58.8	25.5	22.9	3.3	73.2	21.6	73.9	72.6	84.3	23.5	75.8	84.3	**94.8**
diving	**230**	18.3	14.4	13.0	11.7	13.0	19.6	22.2	13.5	54.6	15.2	17.4	46.1	**72.2**
dancer	**225**	89.3	94.2	73.8	99.1	99.1	98.2	82.7	76.4	63.6	23.1	93.8	98.2	**100.0**
bluecar	**441**	4.5	16.9	51.7	59.6	79.8	2.3	18.0	25.8	2.3	33.7	16.9	**85.5**	85.4
avatar	**134**	11.2	5.2	9.7	11.9	12.7	14.9	5.2	13.4	42.5	38.1	27.6	80.6	**89.6**
up	**190**	20.0	52.1	31.6	44.7	55.3	27.4	27.9	17.9	20.5	37.9	20.0	**96.8**	94.7
neymar	**311**	7.9	3.2	2.5	4.8	77.8	46.0	-	57.4	-	**88.9**	88.3	57.1	**88.9**
cliff-dive	**76**	69.1	64.7	64.7	64.7	64.7	70.6	38.2	17.6	-	88.2	63.2	**97.1**	95.6
torus	**264**	11.7	10.2	12.9	7.6	**99.2**	32.2	45.1	-	33.0	90.9	20.1	98.9	97.0
average	-	38.5	28.2	29.4	30.5	51.9	37.9	39.8	39.7	33.5	52.6	50.7	86.7	**92.9**

6.1. Implementation Details

We want to emphasize that the values of most parameters in our algorithm are fixed in all experiments. The typical values for our parameters used in this paper are presented as follows. The weight parameters in the affinity value calculation of Equ. 1 are: $\omega_1 = 0.075$ and $\omega_2 = 0.25$. The sigma in motion term of Equ. 1 is: $\sigma = 0.0625$. The boosting term α in Equ. 4 is usually set to 5 or 10. The local perturbation term $\delta \in [-\phi, \phi] \times [-\phi, \phi]$, we usually set ϕ to 8 or 12. In our experiments, each video segment consists of 5 consecutive frames and the overlap of two consecutive video segments is 4-frame. In addition, we use 3-rd order hypergraph (i.e. m discussed in Section 4 is set to 3) for all testing sequences. For the oversegmentation method, we employ [2] and its compactness is set to 50. To ensure there are sufficient superpixels to represent a target, the number of superpixels in the search window is usually set between 100 and 200. For the update process, the *SVM-based Part Classifier* is updated every three video segments. For the sake of fair comparison, we use the same initial bounding box for all trackers in all sequences and we use the default parameters of other trackers which are provided in their papers and choose the best one of 5 runs.

6.2. Experimental Analysis

We present our experimental results in Table 1 and Table 2 where the scores of the top two performing methods are colored in red and blue respectively. In addition, we also present in Fig. 4, the performance of the center errors for the top five performing methods on each sequence. Finally, some visual comparison results are displayed in Fig. 5. To aid in the discussion of our results, we group testing sequences based on 4 challenging factors and discuss the capabilities of all evaluated trackers under each factor.

Structural Deformation. When a structural deformation happens, the target appearance features will change rapidly and severely, which makes the bounding box based trackers powerless. For the sequences with local structural deformations, such as *waterski*, *lipinski*, *football*, *kwan*, *gymnastics*, *dancer*, *up* and *neymar*, the bounding box representations [4, 16, 13, 19, 15, 34] can model targets in these sequences well, but the part based trackers [3, 25, 14, 27, 10, 5] still give better performances in general. In the sequences with rapid and severe structural deformations (*e.g. transformer*, *diving*, *cliff-dive*, *yunakim* and *avatar*), overall, the part based trackers perform better than bounding box based trackers [4, 16, 13, 19, 15, 34], as shown in Table 2 and Fig. 4. The performance difference is due to the fact that part based trackers typically focus on the local part appearance instead of the global target appearance and the local part appearance is less sensitive to structural deformations than the global one. However, despite having better overall performance, not every part based tracker is resistant to all the deforming cases in our experiment: [3] and [25] sometimes fail in the tracking task due to the lack of an effective appearance updating mechanism; [14] suffers from its inability to track local parts so it frequently fails to complete the tracking task. Our tracker does well on almost all testing sequences, thanks to our consideration of the high-order relationships among the target parts across the temporal domain. For some testing sequences, *e.g. football*, where our tracker does not yield the best performance, it might be caused by the fact that the bounding boxes generated by our tracking algorithm (depicted in Section 5) are not perfectly aligned with the ground-truth bounding boxes.

Occlusion. The targets in sequences *lemming*, *football*, *neymar*, *bluecar* and *avatar* are sometimes occluded either mildly or severely. The methods [13, 34, 25] without incorporating any mechanism to overcome occlusions cannot locate the target in these sequences. Note that although some trackers [4, 15, 3] claimed that they are insensitive to occlusions, but our results show that they do not have reliable performance in these sequences due to their inability to cope with other challenges, such as structural deformations or complex background. When partial occlusion occurs, similar to [14, 27], our tracker will shrink to the non-occluded parts. However, despite the temporary shrinking under the occlusion, our tracker is able to continue tracking target after the occlusion with the help of *SVM-based Part Classifier* which keeps the initial reliable positive parts as

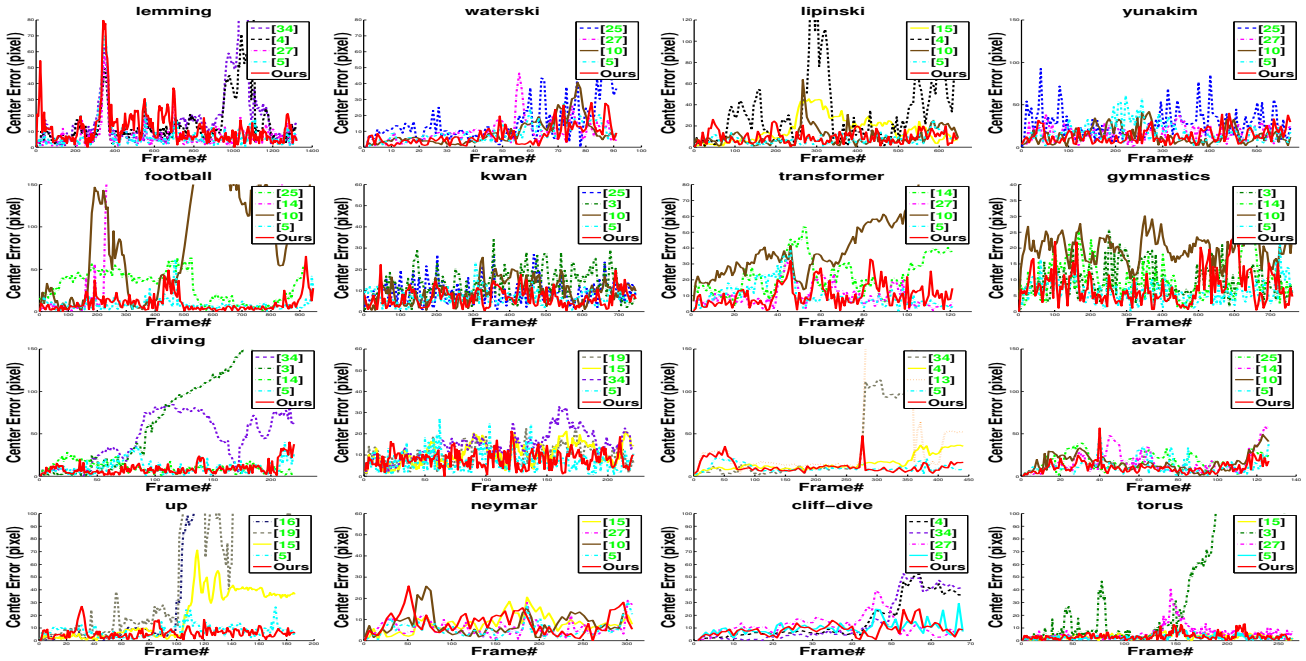

Figure 4. Tracking results of the evaluated trackers. Results of trackers with top five performances on each sequence are displayed.

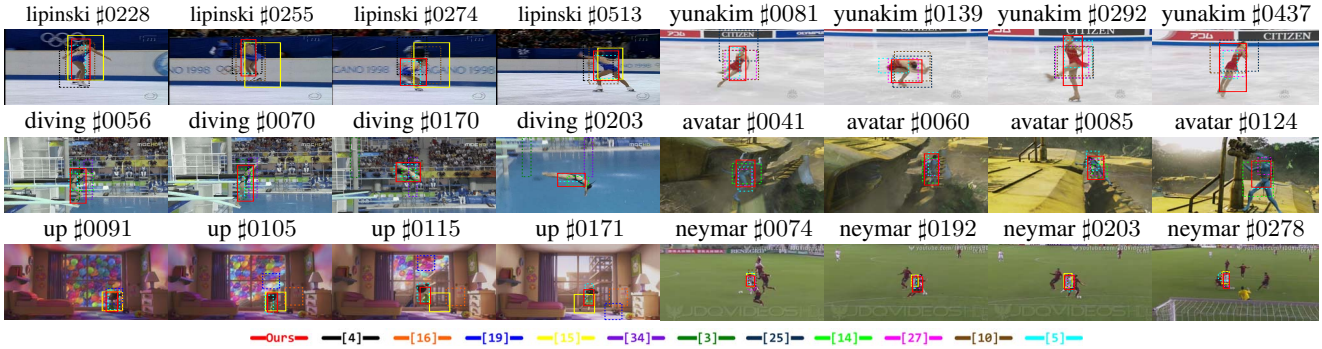

Figure 5. Tracking results of different trackers. Only the trackers with relatively better performance are displayed.

positive training samples throughout the tracking process. For example, in the line chart of "*lemming*" in Fig. 4, between Frame 300 and Frame 400, there exists a fluctuation on the red curve due to an occlusion and after the occlusion, the curve re-approaches the x-axis which indicates that our tracker continues tracking the target reliably.

Abnormal Movement. The abnormal movement is another great challenge for object tracking because it does not meet the typical tracking movement assumption, i.e., the object moves linearly at a constant speed within a short time span. Our testing sequences include several types of abnormal movements, such as quick motion (*avatar*), abrupt motion (*up* and *cliff-dive*), rotation (*yunakim*, *lipinski*, *waterski* and *torus*) and complex motions (*diving*). In *avatar*, many trackers lose the target when the avatar jumps. The targets in *up* and *cliff-dive* move abruptly, *e.g.* when the girl in the *up* sequence suddenly jumps or when the diver in the *cliff-dive* sequence abruptly spreads out her body and descends rapidly. In such time instances, most trackers will suffer from drifts. Similarly, most trackers become unstable when the athlete spins quickly in *yunakim*. *diving* is one of the most challenging sequences with complex motions, i.e., the combination of different types of abnormal movements, including the to-and-fro motion (the take-off process) and the rotation (the twisting process). Note that, for this challenging sequence, our tracker performs much better than the other trackers because it benefits from the appearance based segmentation and the TCP clusters identification technique, both of which make our scheme more immune to the performance degradation caused by abnormal movements.

Illumination Variations. The appearance features are usually seriously impacted by illumination variation. As depicted in Fig. 5, the frequent illumination variations in the video sequence *up* cause many trackers to drift away quickly, such as [27, 25, 19, 7, 13]. Only [5] and our tracker present satisfying performance on *up*. Both of the appear-

ance models in these two trackers use an online updating mechanism which makes them more resistant to illumination variations. Furthermore, besides the timely appearance updating mechanism, both trackers use the appearance-free information (the inner structure in [5] and the exploitation of the motion properties in the relation hypergraph construction of our tracker) to assist tracking, which helps in identifying the target parts with severe appearance changes.

7. Conclusion

In this paper, a TCP cluster based tracker is proposed for determining a reliable representation of a target so that the target can be tracked reliably irrespective of deformations and occlusions. The core of our tracking approach is to identify TCP clusters based on a relational hypergraph, which models the high-order relationships among multiple local parts across the temporal domain. The appearance and motion patterns are subtly integrated in the calculation of the hypergraph affinity values. A post-processing strategy, which involves the refinement and recovery steps, is designed to determine *Final Target Part Set*. Given the *Final Target Part Set*, the optimal target state is determined using a coarse-to-fine strategy. Extensive experiments were conducted using 16 challenging datasets which include severe deformations and occlusions to compare our algorithm with several state-of-the-art methods. The results clearly show the robustness of our proposed tracker.

8. Acknowledgement

This work was supported by the Chinese National Natural Science Foundation Projects ♯61375037 and ♯61473291. W. Li' and M. Chuah's research are partially supported via a Lehigh fellowship and NSF CSR grant 1016296.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2274–2282, 2012.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels. In *Technical Report, EPFL*, volume 2, page 3, 2010.

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

[4] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1619–1632, 2011.

[5] Z. Cai, L. Wen, J. Yang, Z. Lei, and S. Z. Li. Structured visual tracking with dynamic graph. In *ACCV (3)*, pages 86–97, 2012.

[6] L. Cehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *ICCV*, pages 1363–1370, 2011.

[7] P. Chockalingam, S. N. Pradeep, and S. Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *ICCV*, pages 1530–1537, 2009.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, page 2142, 2000.

[9] D. Cremers and G. Funka-Lea. Dynamical statistical shape priors for level set based sequence segmentation. In *VLSM*, pages 210–221, 2005.

[10] S. Duffner and C. Garcia. Pixeltrack: A fast adaptive algorithm for tracking non-rigid objects. In *ICCV*, pages 2480–2487, 2013.

[11] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *ICCV*, pages 81–88, 2011.

[12] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, pages 2141–2148, 2010.

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

[14] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *CVPR*, pages 1208–1215, 2009.

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

[16] J. Lim, D. A. Ross, R.-S. Lin, and M.-H. Yang. Incremental learning for visual tracking. In *NIPS*, 2004.

[17] H. Liu, L. J. Latecki, and S. Yan. Robust clustering as ensembles of affinity relations. In *NIPS*, pages 1414–1422, 2010.

[18] H. Liu, X. Yang, L. J. Latecki, and S. Yan. Dense neighborhoods on affinity graph. In *International Journal of Computer Vision*, pages 65–82, 2012.

[19] X. Mei and H. Ling. Robust visual tracking using $\ell 1$ minimization. In *ICCV*, pages 1436–1443, 2009.

[20] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 266–280, 2000.

[21] P. Prez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV (1)*, pages 661–675, 2002.

[22] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *CVPR*, 2007.

[23] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Temporally consistent superpixels. In *ICCV*, pages 385–392, 2013.

[24] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *CVPR*, pages 723–730, 2010.

[25] S. M. N. Shahed, J. Ho, and M.-H. Yang. Online visual tracking with histograms and articulating blocks. In *Computer Vision and Image Understanding*, pages 901–914, 2010.

[26] F. Tang and H. Tao. Probabilistic object tracking with dynamic attributed relational feature graph. In *IEEE Trans. Circuits Syst. Video Techn.*, pages 1064–1074, 2008.

[27] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011.

[28] L. Wen, Z. Cai, Z. Lei, D. Yi, and S. Z. Li. Online spatio-temporal structural context learning for visual tracking. In *ECCV (4)*, pages 716–729, 2012.

[29] L. Wen, Z. Cai, Z. Lei, D. Yi, and S. Z. Li. Robust online learned spatio-temporal context model for visual tracking. In *IEEE Trans. Image Processing*, volume 23, pages 785–796, 2014.

[30] L. Wen, Z. Cai, M. Yang, Z. Lei, D. Yi, and S. Z. Li. Online multiple instance joint model for visual tracking. In *AVSS*, pages 319–324, 2012.

[31] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li. Multiple target tracking based on undirected hierarchical relation hypergraph. In *CVPR*, pages 4321–4328, 2014.

[32] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1195–1209, 2009.

[33] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel. Part-based visual tracking with online latent structural learning. In *CVPR*, pages 2363–2370, 2013.

[34] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV (3)*, pages 864–877, 2012.