Adaptive Covariance Tracking with Clustering-based Model Update

Lei Qin¹, Fahed Abdallah², and Hichem Snoussi¹
¹ICD/LM2S, UMR CNRS 6279, Université de Technologie de Troyes, Troyes, France
²HEUDIASYC, UMR CNRS 6599, Université de Technologie de Compiègne, Compiègne, France

Abstract—We propose a novel approach to track nonrigid objects using the recently proposed adaptive covariance descriptor [1] with clustering-based model update mechanism. The adaptive covariance descriptor represents an object of interest according to its characteristics in a small-dimensional covariance matrix and possesses higher discriminative power with respect to the original covariance descriptor. A clustering-based update mechanism is then conducted on the target model to adapt to the object appearance changes during the tracking process. We show that by updating with a carefully selected cluster, the update mechanism can efficiently deal with significant appearance deformations and severe occlusions. Comparative experimental results on challenging video sequences demonstrate the effectiveness of the proposed approach.

Keywords: visual tracking, adaptive covariance descriptor, clustering, model update.

1. Introduction

Long-term visual tracking in ever-changing environment is a challenging task which often requires handling target appearance variations. In general, there may be two types of appearance variations: intrinsic and extrinsic. Nonrigid shape deformation and/or pose variation of an object are considered as the intrinsic appearance variations while extrinsic variations are due to the changes resulting from different illumination, camera motion, camera view point, occlusions etc. Effectively modeling such appearance variations plays a critical role in visual tracking.

Many tracking methods have been proposed in the last decades [2]. Most of the top-performing approaches rely on adaptive appearance models which evolve during the tracking process as the appearance of the object changes. Black et al. [3] employ a mixture model to represent and recover the appearance changes in consecutive frames. Jepson et al. [4] develop a more elaborate mixture model with an online EM algorithm to explicitly model appearance changes during tracking. In [5], Zhou et al. embed appearance adaptive models into a particle filter to achieve a robust visual tracking. Li et al. [6] propose an incremental Principal Component Analysis (PCA) algorithm for subspace learning. In [7], a generalized tracking framework based on the incremental image-as-vector subspace learning methods with a sample mean update is presented. Porikli et al. [8] propose a tracking framework using covariance matrix descriptor with mean update in Riemannian manifold. Li et al. [9] represent the target with Log-Euclidean Riemannian eigenspace model and apply R-SVD [10] to update both the sample mean and the eigenbasis online.

For these adaptive appearance model based visual tracking, a tracker labels samples (typically image patches) in each frame as foreground or background. At the end of each frame, the adaptive tracker uses the newly obtained sample-label pairs to presumably improve its prediction rule for the frames to come. However, due to the challenges in natural scenes and accumulated tracking errors, the tracker may gradually diverge. The main reason behind tracker drift is that the tracker is updated using a self-learning policy and the target model is eventually contaminated with non-target samples. Although a number of solutions [11], [12], [13] have been proposed to tackle the tracker drift problem, these approaches are somehow complicated and/or not able to handle severe occlusions as well as significant appearance variations.

In this work, we focus on the problem of tracking an arbitrary object with no prior knowledge other than its location in the first frame. Our goal is to develop an efficient and robust way to keep tracking the object throughout long-term video sequences in the presence of significant appearance variations and severe occlusions. The idea of our approach is two folds: i) employ descriptors or features with high discriminative power and relatively robust to extrinsic variation; ii) update the appearance model in a punctual and careful manner to keep the model adaptive to the intrinsic variations of the target.

The main contribution of this work consists of proposing a tracking approach using adaptive covariance descriptor with a clustering-based model update mechanism. The adaptive covariance descriptor [1] firstly introduced by Qin et al. is a variant of the original covariance descriptor [14], which is capable of fully capturing both statistical and spatial properties of object appearance. The covariance matrix descriptor of [1] is adaptive to the characteristics of the specific target, thereby shows superior appearance representation ability with respect to the original covariance descriptor [14]. In this work, a clustering-based update mechanism is then conducted on the target model at a pre-fixed frequency to adapt to the object appearance changes. We show that by clustering among collected tracking results, target samples and non-target samples will naturally be discriminated. Updating using only clustered target samples will keep the updated
model attached to the original target, avoiding contamination with non-target samples.

The rest of this paper is organized as follows. In section 2, we briefly review the region covariance descriptor, adaptive covariance descriptor as well as their distance metrics. The clustering based model update is introduced and illustrated in detail in section 3. Finally, in section 4, our comparing experimental results convincingly show that by combing adaptive covariance descriptor and clustering-based model update, stable long-term visual tracking can be achieved.

2. Adaptive covariance descriptor for object representation

In this section, we first make a brief review of the covariance matrix descriptor and its corresponding metrics. Then we introduce the adaptive covariance descriptor based target model.

2.1 Covariance matrix descriptor

Denote I as a $W \times H$ one-dimensional intensity or three-dimensional color image, and F as the $W \times H \times m$ dimensional feature image extracted from I,

$$F(x, y) = \Psi(I, x, y),$$  
where $\Psi$ is a function extracting image features such as intensity, color, gradients, and filter responses. For a given rectangular region $R \in I$, denote $\{f_i\}_{i=1,...,N}$ as the $m$-dimensional feature points obtained by $\Psi$ within R. Consequently, the image region can be represented as a $m \times m$ covariance matrix:

$$C = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu)(f_i - \mu)^T$$  
where $\mu$ is the mean of $\{f_i\}_{i=1,...,N}$.

For fast calculation of covariance matrices, integral images [15] are employed as an intermediate image representation. With this representation, any rectangular region sum [14], [15] can be computed in constant time.

2.2 Metrics for covariance matrices

Covariance matrices are symmetric positive definite (SPD) and lie on a connected Riemannian manifold. Typically, there exist two invariant Riemannian metrics: affine-invariant metric [16] and the Log-Euclidean metric [17]. In this work, we adopt the Log-Euclidean metric since distances and Riemannian means under this metric take a much simpler form than those under the affine-invariant metric.

Under the Log-Euclidean Riemannian metric, the distance between two covariance matrices $C_1$ and $C_2$ is given by,

$$d(C_1, C_2) = ||\log(C_2) - \log(C_1)||,$$  
where $|| \cdot ||$ represents the standard Euclidean vector norm and $\log(C)$ is the matrix logarithm.

Taking the Log-Euclidean distance, the log-Euclidean mean of multiple covariance matrices $\{C_1, \ldots, C_n\}$ is obtained in closed form as

$$\bar{C} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \log(C_i)\right).$$  
(4)

More details of the Log-Euclidean Riemannian metric and comparison of the two metrics can be found in [17].

2.3 Adaptive covariance descriptor

Since one can have a template image patch of the target in the initial frame, the idea of adaptive covariance descriptor is to first learn a PCA projection from the template pixel set and then construct a covariance matrix descriptor within the subspace formed by some most significant components.

The construction procedure to build an adaptive covariance descriptor with PCA projection is illustrated in Fig. 1.

As in [1], we first extract features from the raw image patch to form a set of $n$ $(m \times 1)$-vectors (n indicates the number of pixels in this region and $m$ the dimension of features). Based on this point set, we not only construct the original covariance matrix, but also learn a PCA projection. The $m$-dimensional point set is then projected to a subspace by the learned PCA projection resulting in a compact $k$-dimensional point set. Finally, the adaptive covariance matrix descriptor is constructed based on the projected point set.

For a candidate image patch to be compared with the template, the descriptor is similarly computed, except that it employs the PCA projection pre-learned from the template point set. In this way, covariance matrix descriptors are constructed in a subspace dedicated to the object template.

Since adaptive covariance descriptor describes the target with only a few relevant components, it shows to be more discriminative while computationally much faster [1]. For more details on adaptive covariance descriptor and comparison with the original covariance descriptor, we refer the readers to [1].

2.4 Adaptive Covariance Descriptor Based Appearance Model

To handle partial occlusions and increase robustness as well, a target is represented by five patches as in [14], [9], where the whole region captures the global information of the target, and the left, right, top, and bottom halves encode the local information. Each patch is then represented by an adaptive covariance region descriptor. We denote them as $\{C_{(i)} \in \text{Sym}(k)\}_{i=1,...,5}$, where $\text{Sym}(k)$ is the space of real $k \times k$ symmetric matrices.

The five adaptive covariance matrices $\{C_{(i)}\}_{i=1,...,5}$ of the target template (or reference) in initial frame are computed as our initial appearance model. Distance between candidate adaptive covariance matrices $C^c$ and current target appearance
model $C^r$ is measured as,

$$d(c, r) = \sum_{i=1}^{5} (||\log(C^c_i) - \log(C^r_i)||).$$

(5)

At each frame we search in the whole image (or in a local window around the target location in the previous frame, which depends on the implementation) to find the region that has the smallest distance to the current object model. The best matching region determines the location of the object in the current frame.

After a long time period, the target may undergo both intrinsic and extrinsic variations. Update of appearance model is thus necessary.

3. Clustering-based model update

In the practice of visual tracking, a target model can effectively represent the target appearance for a certain duration, and in some cases there are no appropriate positive samples for updating. The former indicates that with relatively robust descriptor, it is not necessary to update a target model too frequently and the latter may mention a severely occluded scene where no good positive samples are available in the frame. We thus tend to update the model at a certain frequency instead of every frame, as long as at least one optimal target sample which crops the target precisely is collected during this period.

Another important issue for accurate updating is to ensure that the model is updated with good positive samples. Tracking results collected during a certain period may contain optimal positive samples but also can have suboptimal or background samples. Previous works usually neglect this issue or simply address it by selecting good samples using a pre-fixed threshold [5]. In other words, updating is performed with samples which have distances to the target model smaller than the threshold. However, during a long-term visual tracking, both background and target appearance are ever-changing. It is thus very difficult or even impossible to estimate a threshold that can separate optimal sample and suboptimal samples effectively for a long time.

To conquer this, our idea is that clustering analysis among the collected samples can naturally align similar optimal samples, suboptimal samples and background samples into different groups. A clustered group whose centroid is most similar to the current target model is chosen. Updating using samples in the selected group will not only keep the model adaptive to appearance changes but also prevent contaminating target model with suboptimal samples. In the following, we give more details of the clustering process with adaptive covariance descriptor based appearance model.

3.1 Preprocessing for clustering

As stated in section 2.4, each sample is represented by five adaptive covariance matrices $C_{i=1...5}$, which lie on a connected Riemannian manifold. Thanks to the Log-Euclidean transform, the covariance matrices can be transformed into vector space ($\log(C_{i=1...5})$), then unfolded and concatenated to take a vector form.

As time progresses, collected samples constitute a sample set. When the pre-fixed frequency is met, a clustering analysis can be performed among these samples. In practice, the concatenated vectors are high dimensional and may cause the clustering process time-consuming. A PCA reduction can be employed a priori to obtain compact vectors while preserving dominant information. In fact, Ding et al. proved in [18] that principal components are the continuous solutions to the discrete cluster memberships indicators for K-means clustering. It is plausible that clustering in the projected subspace may improve the clustering accuracy [18], [19].
3.2 Clustering-based model update

The next step is to cluster the preprocessed samples. Since samples may represent background, partially occluded target or precisely cropped target etc., it is difficult to predict the number of clusters that are present. Hence, a standard clustering approach such as K-means is not appropriate. The mean shift clustering algorithm [20], which is an iterative gradient ascent method for finding local density maxima, was used instead. It does not require prior knowledge of the number of clusters and does not constrain the shape of the clusters. The data association criteria is based on the underlying probability of the data points. The algorithm begins by placing a window (usually a hyper-sphere) around each point in the feature space. On each iteration, each window moves in the direction of the mean shift vector which is computed as follows:

\[ y_{t+1} = \frac{1}{\Theta_{\lambda}} \sum_{x \in \Theta_{\lambda}} (y_t - x) \]

where \( y_t \) is the window center at iteration \( t \), and \( \Theta_{\lambda} \) is the set of points in the hyper-sphere window of radius \( \lambda \). It is also possible to use a kernel function to weight points according to how far they are from the window center. The windows eventually converge towards local density maxima yielding the cluster centroids. Thus, the mean shift clustering algorithm avoids the issue of knowing the number of clusters at the price of introducing another bandwidth parameter \( \lambda \). This parameter, however, is easier to tune regarding all possible inputs [21].

After clustering, the cluster whose centroid has the smallest distance to current target model is selected. Mean of the samples in the selected cluster using Log-Euclidean transform is written as,

\[ C_i^s = \exp(\frac{1}{T} \sum_{d=1}^{T} \log(C_i^d)), \quad i = 1 \ldots 5, \]

where \( T \) is the number of members inside the selected cluster.

The initial target appearance model \( C_{r_0} \) is always kept during the tracking process. Estimated target model \( \hat{C}^r \) is determined as linear combination of initial target model \( C_{r_0} \), current model \( C^r \) and mean of samples \( C_i^s \) in the selected group:

\[ \hat{C}_i^r = \exp(\alpha \cdot \log(C_{r_0}^i) + \beta \cdot \log(C_i^r) + \gamma \cdot \log(\hat{C}_i^r)), \quad i = 1 \ldots 5, \]

subject to \( \alpha + \beta + \gamma = 1.0; \quad 0 \leq \alpha, \beta, \gamma \leq 1.0. \) (9)

The benefit of introducing \( \alpha, \beta \) and \( \gamma \) here is that it increases the flexibility of the model, which can be regarded as Updating Rate. A Large value of \( \alpha \) indicates insistence to the initial model, while a large \( \gamma \) corresponds to quick adaptation to new observations.

Finally, model update is accomplished by setting:

\[ C^r = \hat{C}^r. \] (10)

It is interesting to note that clustering among samples that are collected in long-term tracking also provides a natural way to aggregate samples representing different poses of the target, which may be helpful for building multiple-pose classifiers for target recognition or re-identification. We remain this as our future work.

4. Experiments

We evaluate the proposed method on three challenging sequences and compare it with a state-of-art tracker and other update policies. The sequences are collected from public dataset and self-captured videos, where the tracking problem is considered to be difficult due to occlusions, appearance variations and cluttered scenes. We set the update frequency to 10 frames and \( \lambda \) (bandwidth) of the mean shift clustering to 0.65 for 10-dimensional vectors (after PCA dimension reduction). The linear combination coefficients \( \alpha, \beta \) and \( \gamma \) in (9) are set to 0.12, 0.33 and 0.55 respectively based on experimental experience. In order to accelerate the tracking process, we search in a local window around previous tracking result to find the best match instead of searching the whole frame. The algorithm runs at 10-20 frames per second on a desktop with Intel 2.67 GHz processor and 8 GB memory.

We first compare the proposed method with the tracking algorithm using appearance-adaptive model and particle filter [5] (denoted as “AapTracker” in Table 1) and then specifically access the clustering-based update mechanism (denoted as “AptCov+cu”) with respect to other update policies. The first update policy for comparison is to fully adapt to changes every frame using mean of current tracking result and the last model updated [22] (denoted as “AdpCov+fu” in Table 1) and the other one is to use fixed template without update (denoted as “AdpCov+nu”). Some information about the sequences and Percentage of Correctly tracked Frames (PCF) of all these algorithms are summarized in Table 1. PCF measures the percentage of correctly tracked frames over all frames on the sequence. We display some tracking results on “outdoor2” and ”player” sequences with different update policies in Fig. 2 and Fig. 3 respectively. The “AdpCov+nu” tracker drifts to non-target as the target undergoes significant appearance deformation (see frame 200 and 300 in Fig. 2) and/or there exist similar non-targets in the scene (from frame 441 in Fig. 3), while the “AdpCov+fu” tracker leads to tracking failure as its target model is eventually contaminated resulting from occlusions (from frame 100 in Fig. 2) or accumulated tracking errors (from frame 711 in Fig. 3).

Moreover, Relative Position Errors (RPE) is computed to quantitatively evaluate tracking performance. RPE is defined
Fig. 2: Tracking results on “outdoor2” sequence using different update policies. First row: “AdpCov+cu”. Second row: “AdpCov+nu”. Third row: “AdpCov+fu”. The blue rectangles indicate the search windows and green rectangles represent the tracking results.

Table 1: Sequence Information and Tracking Performances

<table>
<thead>
<tr>
<th>Sequence</th>
<th>outdoor1</th>
<th>outdoor2</th>
<th>player</th>
</tr>
</thead>
<tbody>
<tr>
<td># of frames</td>
<td>205</td>
<td>305</td>
<td>1226</td>
</tr>
<tr>
<td>Frame size</td>
<td>640×480</td>
<td>640×480</td>
<td>720×576</td>
</tr>
<tr>
<td>Initial target size</td>
<td>19×90</td>
<td>16×70</td>
<td>14×59</td>
</tr>
<tr>
<td>Severe Occlusion</td>
<td>Once</td>
<td>4 times</td>
<td>Once</td>
</tr>
<tr>
<td>AapTracker</td>
<td>56.59%</td>
<td>50.82%</td>
<td>26.51%</td>
</tr>
<tr>
<td>AdpCov+cu</td>
<td>97.09%</td>
<td>96.07%</td>
<td>99.18%</td>
</tr>
<tr>
<td>AdpCov+nu</td>
<td>92.23%</td>
<td>60.66%</td>
<td>33.85%</td>
</tr>
<tr>
<td>AdpCov+fu</td>
<td>12.14%</td>
<td>12.46%</td>
<td>55.06%</td>
</tr>
</tbody>
</table>

as the ratio of the center distance between and to the size of the ground truth. RPE on “outdoor2” sequence of the compared algorithms are demonstrated in Fig. 4. We can easily see from these results that the proposed method outperforms all its competitors and stable long-term tracking is achieved despite severe occlusions and significant shape deformations.

5. Conclusions

We have presented a visual tracking algorithm combining the adaptive covariance descriptor and a clustering-based model update mechanism. We show that clustering on collected tracking samples can play a natural role in aggregating optimal positive samples. Update with these clustered optimal samples efficiently keeps the model adapt to target appearance changes and prevents it from being contaminated. Superior performance achieved on challenging long-term video sequences demonstrates the promise of the proposed approach.

References

Fig. 3: Tracking results on “player” sequence using different update policies. First row: “AdpCov+cu”. Second row: “AdpCov+nu”, tracker drifts to another player of the same team from frame 441. Third row: “AdpCov+fu”, tracker diverges to a player of the other team from frame 711.

Fig. 4: Relative Positive Errors for each frame in “outdoor2” sequence.


