Recurrent Tracking using Multifold Consistency

Pan Pan
University of Illinois at Chicago
Chicago, IL
ppan3@uic.edu

Fatih Porikli
Mitsubishi Electric Research Labs
fatih@merl.com

Dan Schonfeld
University of Illinois at Chicago
dans@uic.edu

Abstract

We present an adaptive object tracking algorithm that is based on a novel consistency measurement computed recursively over a multifold of forward and backward frames. To obtain the multifold consistency score, the target object given for the current frame is searched in the consecutive frames forward in time without updating the object model. Then, a reinitialized model using the last observation is traced backward in time up to the current frame. Our hypothesis is that the object states before and after this process should be consistent for a successful tracking and a disagreement in states indicates a possible error. We utilize this score of each separate object to adjust the complexity of the corresponding core trackers, such as particle filters and mean-shift variants, and switch between these methods recurrently to extract the most reliable object tracks. Our results show the proposed recurrent tracking technique is capable of producing longer and more accurate trajectories, which is otherwise not possible for non-adaptive counterparts.

1. Introduction

Visual object tracking is one of the most essential and challenging tasks in computer vision. So far, a great amount of tracking algorithms has been proposed to bring the performance into a more desirable level in real world applications.

Tracking can be considered as estimation of the state given all the measurements up to that moment, or equivalently constructing the probability density function of object location. When the measurement noise are assumed to be Gaussian, the optimal solution is provided by the Kalman filter. When the state space is discrete and consists of a finite number of states, Markovian filters can be applied for tracking. The most general class of filters is represented by particle filters, which are based on Monte Carlo integration methods. The current density of the state (which can be location, size, speed, boundary [12]) is represented by a set of random samples with associated weights and the new density is computed based on these samples and weights. However, it is based on random sampling that may cause sample degeneracy. Besides, there is a tradeoff between sampling efficiency and computational burden, especially for higher dimensional representations. In contrast, the mean-shift tracker is a non-parametric density gradient estimator that is iteratively executed within the local search kernels [3],[11]. It models the object probability density in terms of color histogram, and moves the object region towards the largest gradient direction. Thus, it is computationally simple. Nevertheless, if the object relocation between the successive frames is larger than the kernel size, it fails to detect the object. Since the histograms are used to determine the likelihood, the gradient estimation and convergence becomes inaccurate in case the object and background distributions are similar.

There are previous attempts to gauge the tracking quality and adapt the tracking algorithm. For instance, a region-based correlation and edge-based adaptive contour approach is proposed in [13]. Velocity and feature information are used to decide when the corresponding algorithm tracks well and when it fails. Combining sum-of-squared differences and mean-shift tracker was investigated in [1]. For particle filtering, the variance of the current estimated target state is utilized in [2] to switch between two complementary sampling algorithms based on the variance of the current estimated target state. Survival diagnostic and survival rate were presented as quantities to assess the efficiency of particles filters and used in partitioned sampling in [9]. To adjust the number of particles, sum of weights [7], Kullback-Leibler divergence [6], and entropy [14] are often incorporated. More recently, the variance of the proposal density [10] is considered to optimize particle allocations. Another approach [8] presents a probabilistic framework to combine multiple algorithms through their explicit probability distribution functions or sample-sets without employing an auxiliary confidence measure. A method to track an object from both the beginning and end key frames is proposed in [15], where they use trajectory segment analysis to deal with occlusion and sudden motion. Computing the optimal trajectories among frames [5] is also considered to improve tracking. Backward tracking is integrated in [4] to detect the newly appearing Maximally Stable Extremal Regions. Tracking along forward and time reversed Markov chain is used in [16] to evaluate the tracking performance. Yet, most methods impose hard thresholds and inherently cause drift. Often they have no mechanism to recognize whether they are on the target or have already lost the track of the object.

In spite of the chivalrous effort to develop the most accurate tracking method in last two decades, it would be

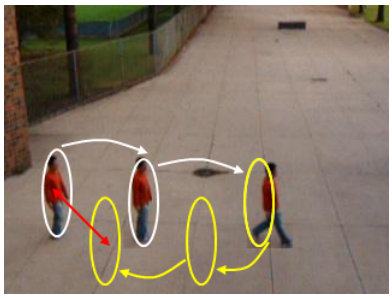


Figure 1. Multiple frames are shown. In case the tracker fails the consistency will be low as the object states (locations in this example) before and after the fold do not match.

premature to claim that a single technique can handle successfully any real world conditions. There are unfortunately many natural reasons to fail a tracker including irregular and fast object motion, partial and full occlusions, object appearance changes due to illumination variations, drastic pose and size transformations, indistinguishable backgrounds, and crowded scenes. Likewise, each core tracking method has its own advantages and shortcomings.

A rational approach is to take advantage of each core tracker and adaptively select the one that would provide the correct solution for the underlying scenario and conditions. To achieve this, we need to know how accurately a tracker would perform without having a ground truth to compare against its results.

1.1. Our method

To overcome the shortcomings of the above methods, we propose a novel measure, called as multifold consistency, to assess the quality of the tracking. Instead of monitoring the instantaneous probability density surfaces or depending on the rather rigid object model comparisons, we properly evaluate the performance of core trackers by observing their individual tracking behaviors in forward and backward in time as illustrated in Figure 1.

Main contributions of our work can be summarized as follows: i) We introduce a multifold consistency score computed for each tracked object. This enables estimating the instantaneous condition of the tracking and discovering potential failures for each individual object. ii) In addition, we make use of this information into a recurrent tracking framework to obtain the most accurate results that may be possible by a a pool of given trackers. The multifold consistency is utilized as the basic principle to automatically adapt parameters as well as switch among different algorithms. This resolves the most demanding problem in the engineering of a tracking systems, the fine tuning of the parameters. iii) Unlike the conventional methods that depend on the object appearance similarity, our tracking approach can identify occlusions and determine whether an object left the scene or not.

The rest of the paper is organized as follows. In section 2, the multifold consistency is defined. The recurrent tracking framework is described in section 3, and the experimental results are discussed in section 4.

2. Multifold Consistency

We introduce an instantaneous quality measure, called as multifold consistency, that can assess the current performance of a chosen tracking algorithm automatically without a ground truth. In other words, this measure determines how consistently a tracker performs for a certain object in the corresponding video segment.

We consider tracking not only forward in time (future frames) but also backward (past frames). Depending on the application requirements, tracking may pace with a permissible latency using a small number of buffered future frames, which is common for most video surveillance systems. This may not be possible for the causal systems that employ the estimated object position to restraint the motion of the camera, for example in the case of aerial vehicles that track moving targets. Thus, we present alternative consistency score definitions applicable for the causal systems that use only the past frames and for the systems that operate under latency.

Let's assume without loosing generality that the state X_t of an object of interest is known for the current frame I_t. The state may consist of the geometric characteristics such as image plane location, size, orientation, and shape parameters. Each object x is represented by a model β_t, which can be into a form of probability distribution function, i.e. a histogram, or a template. Here we assume the initial state is given, yet the discussion can be extended for automatically detected objects, for example, using a classifier and background subtraction. We have no additional constraint on whether the object is visible in all frames. In fact, our method can identify and resolve such situations.

We denote the current tracking algorithm as $f_i(\theta_i)$ with the specific parameterization θ_i within the pool of alternative tracking algorithms $f_1, ..., f_K$. Our formulation is not restricted to any tracking method and we can include any algorithm in our pool. For simplicity, we dropped the parameters θ and index i in the rest of this discussion. The tracker f estimates the state of the object at frame I_{t+1}

$$X_{t+1} = f^{+}(X_t, \beta_t) \tag{1}$$

using its model β_t where f^{+} indicates a forward track in time and f^{-} means backward, i.e. $X_{t-1} = f^{-}(X_t, \beta_t)$.

Since object model β may change in time due to the illumination and pose variations, trackers often require the object model to be updated using the estimated state

$$\beta_{t+1} \leftarrow y(\beta_t, X_{t+1}) \tag{2}$$

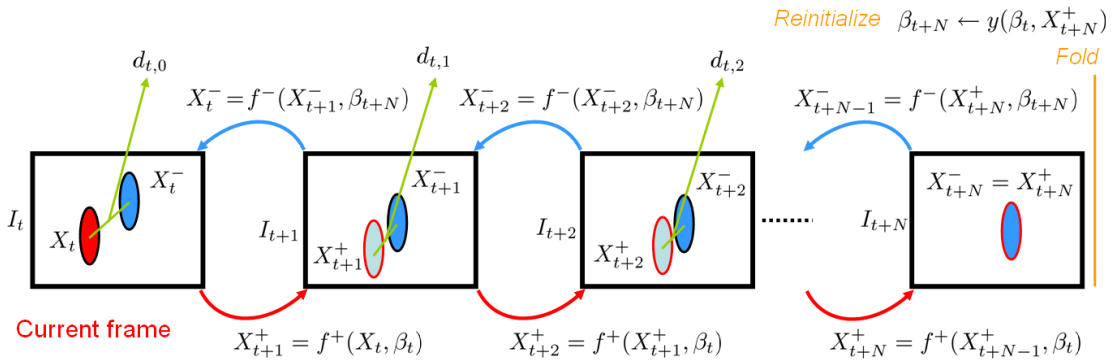


Figure 2. Computation of the multifold consistency coefficients for the latency scenario.

where y is a model building function, e.g. color histogram computation, which may optionally utilize the existing model to allow smooth model changes.

What we mean by the term *multifold* is the tracking of an object in forward and then *folding* the temporal direction and tracking it backward in time in multiple frames. Our intuition is that competent forward tracking ($I_t \rightarrow I_N$) algorithm before the fold will estimate the object state correctly in that frame (I_N). When we reinitialize the object model at that state the model will be valid. Thus, when we track this reset model back to the first frame ($I_t \leftarrow I_N$) the state will match to the state that we started. Therefore, for an accurate (wrong) tracking the state distance will be small (large) and the consistency coefficients will be high (low).

Considering the latency scenario *(case-I)*, the consistency score $C_t(x, f)$ of an object x for the a selected tracker f at time t is determined by first tracking the object x along the consecutive frames $I_{t+1}, ..., I_{t+N}$ in forward direction $X^+_{t+n} = f^+(X^+_{t+n-1}, \beta_t)$ with the original object model β_t. We denote the forward tracked states as X^+. At the final frame I_N, we reset the object model to $\beta_{t+N} \leftarrow y(\beta_t, X^+_{t+N})$. We track back the object, this case with the updated model, to obtain states $X^-_{t+N}, ..., X^-_{t+1}$ down to the frame I_t. In other words, during back tracking the object model is kept identical to the reset model β_{t+N}. Each time we compare the forward and backward estimated states to obtain the corresponding state distance coefficients as

$$d_{t,n} = ||X^+_{t+n} - X^-_{t+n}||. \tag{3}$$

where $n = 0, .., N-1$. Above, the back tracked states are $X^-_t = f^-(X^-_{t+1}, \beta_{t+N})$,.., $X^-_{t+N-1} = f^-(X^-_{t+N}, \beta_{t+N})$. Note that the forward tracked and backward tracked states are same at I_N, i.e. $X^-_{t+N} = X^+_{t+N}$. This process is illustrated in Figure 2. One definition of the multifold consistency is then given as

$$C_t(x, f) = 1 - d_{t,0} = 1 - ||X_t - X^-_t|| \tag{4}$$

where $d_{t,0}$ is normalized to $[0, 1]$. Since the intermediate states are already extracted when the reset model is tracked back, the consistency can also be defined over all frames

$$C_t(x, f) = \frac{1}{N} \sum_{n=0}^{N-1} (1 - d_{t,n}). \tag{5}$$

One can also initialize the object model at every frame and track back the object, that is, fold the time, to obtain a vector of distances. Denoting the C_N as the consistency score when we fold the time at frame I_N, the vector form becomes $[C_N\ C_{N-1}\ ..\ C_1]^T$. This means that we will now run multiple back trackers, which may be computationally prohibitive for some applications. The norms in equations (4)-(5) require $2(N-1)$ inter-frame correspondences, however the vector norm needs $N(N-1)$ of them.

For the causal settings *(case-II)*, we have only the newly acquired frame I_{t+1} to be processed and a moving set of previous frames $I_{t-N+1}, ..., I_{t-1}$. As before, we estimate the state of the object in the new frame by forward tracking. We fold the time, and reset the object model using the estimated state. This folding is done just once. Then, we track back the reinitialized object model β_{t+1} to the I_{t-N+1} frame. At each frame, we compare the previously determined state X_{t-n} to the back tracked state X^-_{t-n} to determine the state distance

$$d_{t,n} = ||X_{t+n} - X^-_{t+n}|| \tag{6}$$

for $n = 0, .., N-1$. This case, the multifold consistency is defined at the deepest frame I_{t-N+1}

$$C_t(x, f) = 1 - d_{t,-N+1}. \tag{7}$$

The causal case is shown in Figure 3.

The value of N is of great importance for the computational load and saliency of the consistency score. It is equal to the expected lifetime of the tracker, that is, the average accurate tracking duration for possible objects.

The philosophy of our multifold consistency can be considered similar with that in communication systems, where channel refers to the medium used to convey information

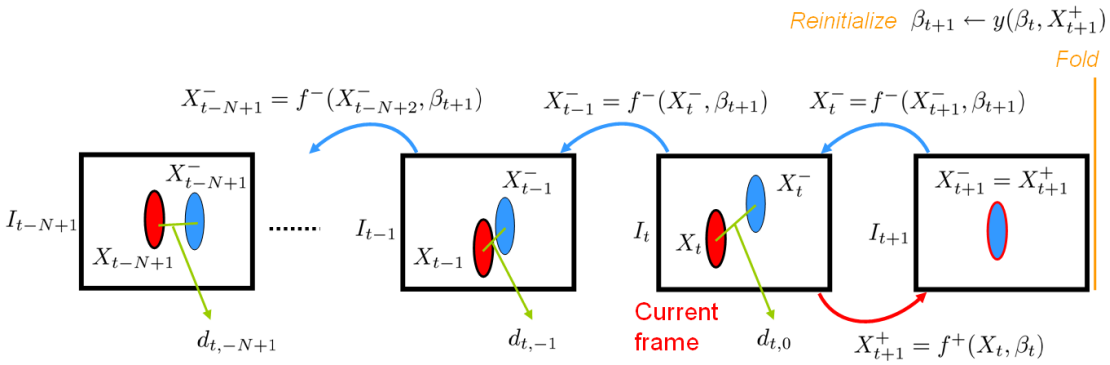


Figure 3. Computation of the multifold consistency coefficients for the causal case.

from a sender to a receiver and in order to assess the properties of a channel, the information at the receiver is compared with the information at the sender by asking the receiver to send the received signal back, as our time foldings and backward trackings.

2.1. State Distance

In its simplest form, the state X_t is the image coordinates of the object window. For more complicated tracking tasks, the state can be composed of other geometric features including the size, shape, mesh, and motion parameters.

One slight difference with our distance norm and the conventional Euclidean distance is that we aim to evaluate how well two states match but not to measure exactly how far they are from each other in case they do not agree at all. In other words, we are interested in the amount of *overlap* in the states. Our reasoning is that once the tracker makes an error and looses the target object, i.e. the estimated state does not match to the original state, it does little matter whether the estimated state is more or less different than the original.

This impelled us to adopt a shifted logistic (sigmoid) distance norm. For an M-dimensional state vector $X_i = [a^i_1 \,..\, a^i_M]$ the distance is defined as

$$||X_i - X_j|| \leftarrow sig(||X_i - X_j||) = sig(\sum_{m=1}^{M} w_m |a_m - a^j_m|)$$

where $sig(\alpha) = 2(1 + e^{-(\alpha-\alpha_0)})^{-1}$ and w_m's are the mixing weights such as $\sum w_m = 1$, and α_0 is a positive value. We normalize the distance onto range $[0, 1]$ to get the multifold consistency scores within the same (but inversely proportional) $[0, 1]$ interval. The choice of α_0 is scalable with the state, e.g. the size of the object. Also, combining all different features into a single measure requires appropriate weighting parameters w_m's of the corresponding distances, e.g. between the size and rotation, or between individual affine motion parameters. Determining such optimal weights is application dependent, and outside the scope of this paper.

In addition to the geometric parameters, appearance e.g. histograms or templates can also be incorporated into the distance. However, using appearance to assess a tracker's instantaneous performance, as the existing methods often depend on, is inherently problematic especially when the estimated state is wrong but has similar appearance, which is not an issue for our approach.

3. Recurrent Tracking

Having a measure of the instantaneous quality score for a tracker enables us to switch between different algorithms automatically. In addition, this let us to increase the computational load whenever it is necessary. We adapt the parameters to achieve the best possible tracking results that can be obtained using the available pool of algorithms. Concentrating only on the inter-frame correspondence, we utilized the conventional approaches including the different versions of the mean-shift kernel estimate and particle filters in the pool of the alternative tracking methods. Yet, our technique is applicable to other tracking algorithm.

In an opportunistic style, we first compute the multifold consistency for the largest number of frames, which is $n = N$, to enable larger jumps when the tracking is concluded to be valid. In other words, the fold time is set to N. We compare the consistency score with a tight threshold τ_n proportional to the n. If the consistency is sufficiently large, which indicates the tracking was successful, we jump to that frame, reset the object state and model. Otherwise, we decrease the fold time, $n \leftarrow n - 1$, until there are n_{stop} frame left, which causes the framework to switch the tracking algorithm. When a more complex algorithm has to be chosen, the fold time is set to N. Any valid tracking switches back the tracker to the simplest algorithm in the pool. For the case of no acceptable consistency score is obtained, i.e. $\max C_t(x, f_i) < \tau_{stop}$, we terminate

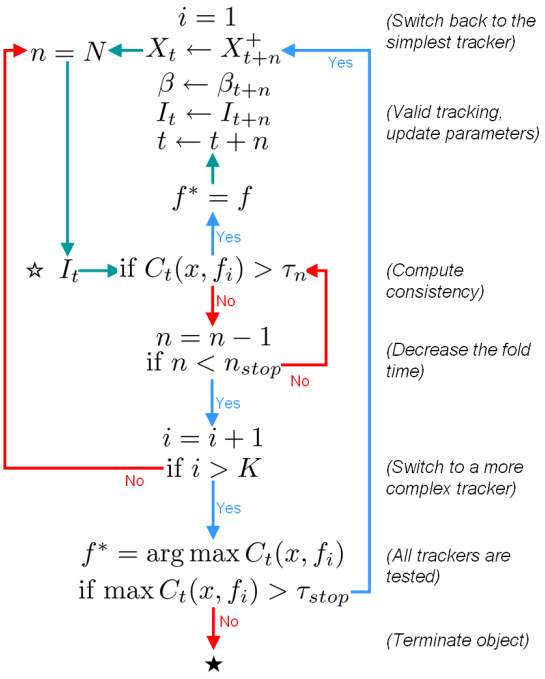


Figure 4. Recurrent tracker first tries all different fold times then starts switching between the pool of algorithms for the latency scenario. White (black) star points the start (end) of the flow.

the object. The flow diagram of this process is shown in Figure 4. As mentioned, this technique tries to relieve the computational load for the latency scenario (*case-I*). The value of n_{stop} is chosen such that when we give up the current tracker and update to a more complicated one. It is determined by the change of the tracking difficulty of the objects. In order to avoid drift caused by the small error, n_{stop} is usually a small value larger than zero, e.g. 6 10.

For the causal systems, we choose rather a greedy technique. Instead of changing the fold time when the current algorithm fails, we only switch to a more complex algorithm to estimate the most accurate state for the frame I_{t+1}.

As it may already noted, these two solutions are not the only possible approaches, in fact the causal case can be considered as a special form of the latency scenario where we cannot change the previous tracking estimates.

4. Experimental Results

We tested the recurrent tracking methods on several synthetic and real-world sequences to demonstrate the performance. In experiments, the state X of objects is described by an ellipsoid as $X = [c_x, c_y]$ or $X = [c_x, c_y, a, \rho]$ where (c_x, c_y) is the center, a is the short axis, and ρ is the rotation angle of the ellipse. The aspect ratio of the long and short axes is kept the same which is calculated at the initialization.

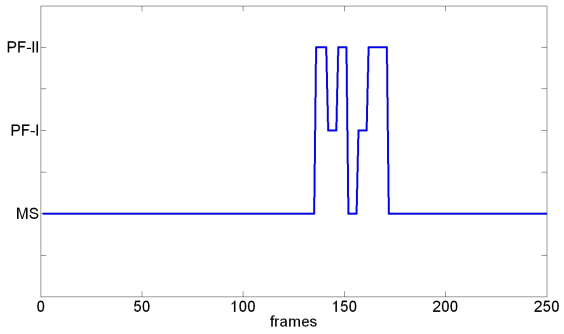


Figure 5. Recurrent tracker states for `Football`.

To make a fair comparison between the pooled trackers of the mean-shift and the particle filter variants, we used color histograms to model and compute the likelihood only within the intermediate stages of the inter-frame correspondence for all algorithms. Using different object descriptors such as histogram of oriented gradients, shape, eigenmodels, etc. would not change the relative rankings of these pooled trackers.

In our experiments the pooled trackers contain the mean-shift algorithm and the particle filters with varying number of particles. The computationally fastest tracker in out pool is the mean-shift algorithm, which does local iterative density estimates. Therefore, it can track an object if only it has overlapping areas between the consecutive frames. Due to this motion constraint, we consider the mean-shift also as the basic tracker in our pool. Particle filters that use more particles are considered to be more complex trackers than the ones using lesser number of particles, which also complies with the Bayesian theory.

`Football` sequence is a challenging synthetic sequence with a football moving randomly in a time-varying cluttered noisy background. The tracker pool included the mean-shift with Gaussian kernel and the particle filters with 200 and 1700 particles. We set the fold time to 6 frames, the mutual consistency threshold to $\tau_n = 0.95$, and the minimum number of frames to switch to a more complex tracker (Figure 4) to $n_{stop} = 6$. The particle filter with 1700 particles generated the most accurate results without losing any objects as expected. The recurrent tracker adaptively switched to this version of the particle filter and achieved a similar mean squared error while having tracked all objects. The actual switching response is shown in Figure 5. Yet, the computational load of the recurrent tracker is around the half of the corresponding particle filter.

Figure 7 (PETS09 sequence) shows a person walk across another person with similar appearance. Our method successfully detects the loss of tracking as shown the drop of mutual consistency score. And the drop is more sensitive than the appearance based similarity score.

Table 1. Performance on `Football` sequence

	CPU time (millisecond per frame)	Performance
Mean-shift	34.15	Fails after 137 frames
Particle filter I	54.82	Fails after 138 frames
Particle filter II	291.41	Average MSE 0.94
Recurrent tracker	158.09	Average MSE 1.15

S.S.: 0.859	0.797	0.551	0.576	0.576
M.C.: 0.972	0.963	0.048	0.014	0.052

Figure 7. A person walks across another person with similar appearance. Our method detects the tracking confusion. SS represents similarity score based on color information, and MC denotes multifold consistency.

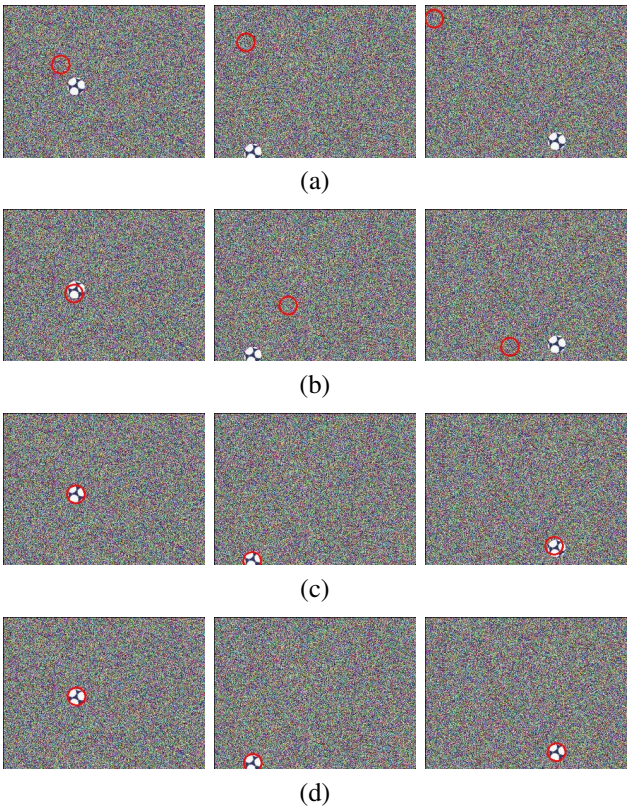


Figure 6. Tracking results of (a) the mean-shift and (b) Particle filter I (c) Particle filter II (d) recurrent tracking on `Football` sequence.

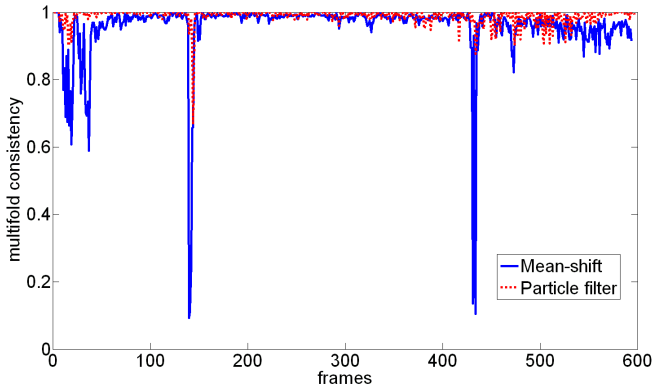
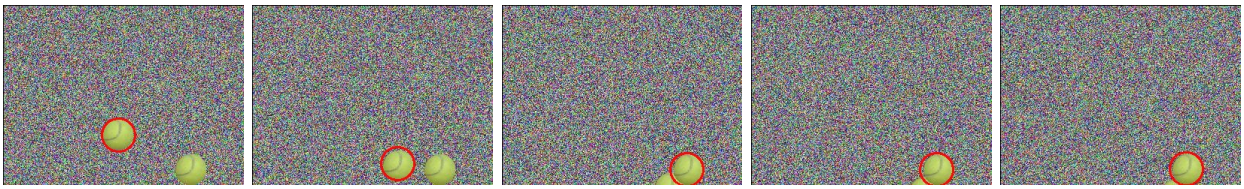


Figure 9. Multifold consistency for two algorithms extracted from `campus` sequence using Eq. (4). We obtained these graphs by initializating the object at every frame (for strictly illustration purposes as the recurrent tracking does not compute the consistency score for all algorithms at every frame). As visible, the computationally simpler algorithm causes more failures.

Figure 8 shows a nearby identical ball exists at the location the first ball exits from the image. Our method competently detects the exit event. The mutual consistency score suddenly drops when our algorithm detects that ball disappears, while the appearance based similarity score cannot. We accomplish this without having to make motion history assumptions as Kalman smoothing.

Figure 9 shows the multifold consistency curves obtained for the mean-shift and particle filters (300 particles) for the `campus` sequence that depicts a person walks and runs in succession. The mean-shift tracking is generally faster than the particle filter, but it can not deal with large motion.



S.S.: 0.825	0.791	0.882	0.863	0.857
M.C.: 0.998	0.994	0.106	0.123	0.132

Figure 8. An object is exiting out of the scene at a location where there is a nearby identical object. SS represents the similarity score based on color information, and MC denotes the multifold consistency. Our method detects the exit event as its value suddenly drops, which indicates tracker is not on the original object any longer. In comparison, the appearance based similarity score fails to detect the exit event; its scores are still high, which means it considers the tracking is on the original object.

However, the particle filter can compensate for the large motion in case the sampling space (in this example the spatial window) is big proportionally with the number of particles. We present a comparison of the conventional mean-shift tracking and the recurrent tracking for this sequence in Figure 10. As shown, our technique switches to the particle filter whenever the multifold consistency is low, and generates perfect tracking results.

As shown in Figures 11 and 12, the recurrent tracking outperformed the mean-shift and the particle filter in a side-by-side comparisons. Our results demonstrate that the proposed consistency score is an accurate and effective measure of quality and the opportunistic recurrent tracking is a competent switching technique.

5. Conclusions

We described a novel instantaneous quality score computation method and utilized it to improve the object tracking performance by adaptively selecting the most accurate tracker. Our solution does not have any limitations and it is able to incorporate any available core tracking algorithms. Since we rank the pool of trackers according to their computational loads, our method provides a novel approach to minimize the complexity while achieving the best results. As a future study, we will extend the recurrent tracking to multi-hypothesis testing and graph-based tracklet connectivity techniques.

References

[1] R. V. Babu, P. Pérez, and P. Bouthemy. Robust tracking with motion estimation and local kernel-based color modeling. *Image and Vision Computing*, 25:1205–1216, 2007.

[2] T. Bando, T. Shibata, K. Doya, and S. Ishii. Switching particle filter for efficient visual tracking. *Robotics and Autonomous Systems*, 54:873–884, 2006.

[3] D. Comaniciu, V. Ramesh, and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[4] M. Donoser and H. Bischof. Efficient maximally stable extremal region (mser) tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[5] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.

[6] D. Fox. Adapting the sample size in particle filters through kldsampling. *International Journal of Robotics Research*, 22(12):985–1003, 2003.

[7] D. Koller and R. Fratkina. Using learning for approximation in stochastic processes. In *International Conference on Machine Learning*, pages 287–295, 1998.

[8] I. Leichter, M. Lindenbaum, and E. Rivlin. A probabilistic framework for combining tracking algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[9] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *European Conference on Computer Vision*, pages 3–19, 2000.

[10] P. Pan and D. Schonfeld. Dynamic proposal variance and optimal particle allocation in particle filtering for video tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(9):1268–1279, 2008.

[11] F. Porikli and O. Tuzel. Multi-kernel object tracking. *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[12] P. Prez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of the IEEE*, 92(3):495–513, 2004.

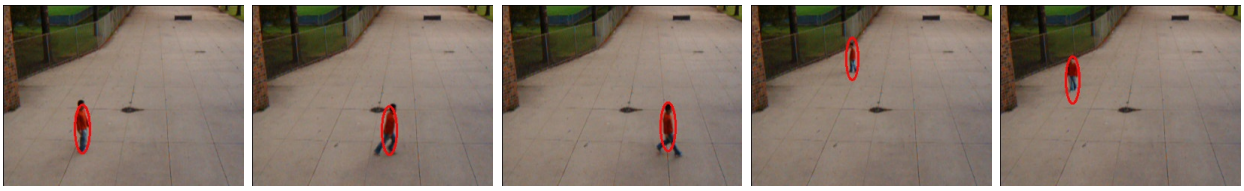
[13] K. Shearer, K. D. Wong, and S. Venkatesh. Combining multiple tracking algorithms for improved general performance. *Pattern Recognition*, 34:1257–1269, 2001.

[14] A. Soto. Self adaptive particle filter. In *International Joint Conferences on Artificial Intelligence*, 2005.

[15] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum. Bidirectional tracking using trajectory segment analysis. In *IEEE International Conference on Computer Vision*, pages 717–724, 2005.

[16] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. In situ evaluation of tracking algorithms using time reversed chains. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

(a)



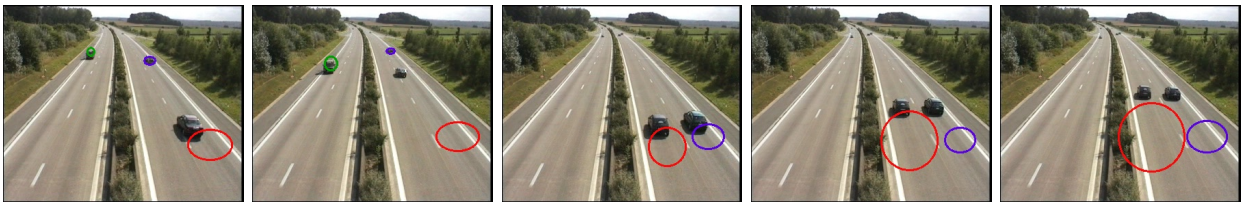
(b)

Figure 10. Experimental results of (a) the mean-shift and (b) recurrent tracking on `Campus` sequence.

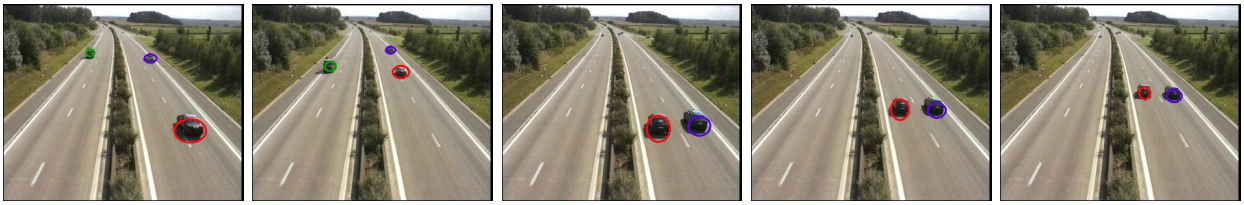
(a)

(b)

Figure 11. Results of (a) the mean-shift, and (b) recurrent tracking.



(a)



(b)

Figure 12. Results of (a) the conventional particle filter, and (b) recurrent tracking.