Supplementary material: Iterative Hypothesis Testing for Multi-object Tracking with Noisy/Missing Appearance Features

Anonymous ACCV 2012 submission

Paper ID 633

In this supplementary paper, we first explain the experimental set up. Then, we provide the running time and the complexity analysis of the proposed algorithm. Some failure cases are presented afterwards. Finally, we provide the sensitivity of the MOTA metrics with respect to some key parameters of the algorithm.

1 Experimental set up

The proposed approach has been implemented in C++. It utilizes Boost Graph Library for representing the graph. The DAG shortest path algorithm is provided in the library. All experiments are performed on a desktop computer with 3GHz quad-core CPU, 4 GB of RAM, and running under Linux.

2 Running time

The running time of the proposed approach is shown in Figure 1. It presents the time taken to process detections in the video sequence (ordinate) versus the length of the video (abscissa).

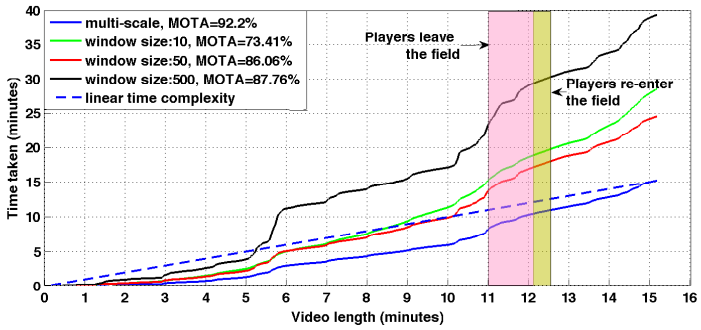


Fig. 1: Running time for multi-scale as well as fixed window approaches. Dotted line represents the linear time complexity (for comparison purpose). Shaded rectangles represent the intervals in which players exit the field and enter again.

From the Figure 1, we can see that the multi-scale strategy indeed reduces the computational complexity. This shows that the multi-scale nature of the

proposed approach is efficient in terms of computational load. Note that the MOTA metrics of the system for different window sizes are different too.

To limit the complexity of the system (and hence the running time), we can discard some nodes in the graph which are older (in time) by τ_{old} frames. Alternatively, the system can be reset when the the detection module does not provide any outputs. This is the case when the players exit the game (number of detections=0). In fact, the players leave the court at 11th minute of the game and re-enter at 12th minute, shown with shaded rectangles in Figure 1. We can reset the tracking process when the players leave the field and start again once they enter again.

3 Complexity of Iterative Hypothesis Testing

Let us assume that there are n nodes in the graph. Each node is considered iteratively as the key-node and a window is considered. Let d be the average number of detections per frame and $\delta = \kappa\mathcal{L}(v_{\text{key}})$ be the size of the window. In the worst case (assuming that no detections have been aggregated into tracklets), there will be $n_\delta = d\delta$ nodes within the window. Also, let m_δ be the number of edges inside the window. The DAG shortest path method is used to find the shortest path. The complexity of DAG shortest path is $\mathcal{O}(n_\delta + m_\delta)$. Since there are n nodes in the overall graph, the overall worst-case complexity will be $\mathcal{O}(n(n_\delta + m_\delta))$. As $n_\delta < n$, the complexity is lower than $\mathcal{O}(n^2)$. Experimentally, the worst case complexity is found to be $\mathcal{O}(n^{1.475})$, which is shown in Figure 2. The reduction of this computational complexity can be accredited to the multi-scale nature of the algorithm.

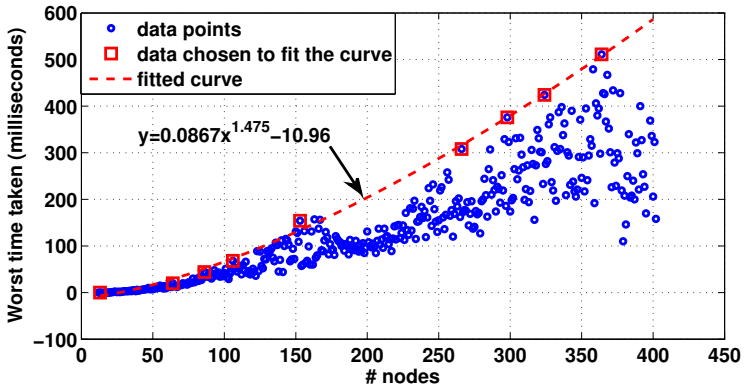


Fig. 2: Complexity of the iterative aggregation algorithm. The parameters of the best fitted line $[y = ax^b + c]$ are $a = 0.08672$, $b = 1.475$ and $c = -10.96$ with 95% confidence in the intervals $(-0.0332, 0.2067)$, $(1.243, 1.707)$, and $(-34.94, 13.03)$ respectively.

4 Failure Cases

We observe that the tracking failures can occur in following cases:

Duplicate detections: Because of the imperfection of the detector, a target might be detected multiple times. Such duplicate detections can be observed when there is a clutter. Unlike typical false positive detections, they are coherent and, hence, persist longer in time. In such situation, a new track is assigned to them. A typical example is shown in Figure 3. The problem of duplicate detections can be mitigated by introducing backward edges in the graph. However, it no longer makes the graph DAG.

(a) Frame 221 (b) Frame 222

Fig. 3: Duplicate detections. (a) Tracks 2 and 3 approach close. Notice that 2 is assigned to the yellow player (b) A new track 13 appears for the same yellow player.

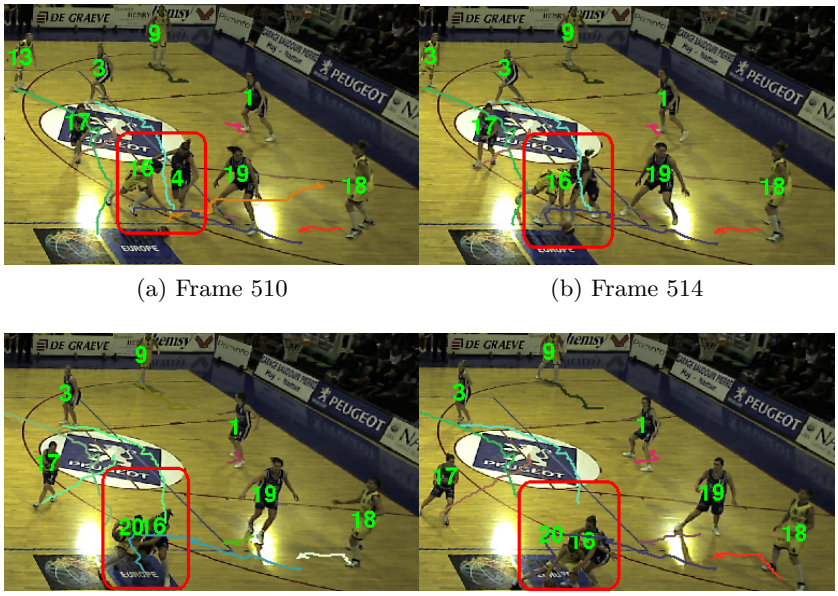
Track reinitialization and switch: When the targets are in clutter, the appearance features become noisy. Moreover, there are often some mis-detections as well as false detections. In such situations, the tracks are sometimes not well resolved, thereby, resulting in the reinitialization and sometimes switching of the track identities. An example is shown in Figure 4.

5 Performance with respect to various parameters

Some key parameters of our algorithm are:

- Connection horizon for new detections, $\tau_{\max}$
- Missed detection coefficient, γ
- Window proportionality constant, κ
- Best cost to second best cost ratio, K_2
- Lower and upper thresholds for the computation of reliability of the feature, $C_{\min}, C_{\max}$

The results for the quiescent point ($\tau_{\max} = 120, \gamma = 3, \kappa = 5, K_2 = 1/3, C_{\min} = 20, C_{\max} = 100$) are (FP=20, MS=387, RE=113, SW=64, MOTA=0.922). These results are shaded in gray in the subsequent tables. Now, we present the performance of the algorithm (in terms of MOTA components) with respect to these parameters individually. For this, only one parameter is changed at a time and all other parameters are fixed at their quiescent values.



(a) Frame 510 (b) Frame 514

(c) Frame 523 (d) Frame 533

Fig. 4: Identity switch and reinitialization errors. (a) Tracks 16 and 4 come close to each other. Note that track numbers 16 and 4 are assigned to the yellow and the blue players respectively. (b) Track 4 vanishes (*miss*). (c) Track number 20 is assigned for the yellow player (*reinitialization*). Moreover, track 16 (which was assigned to the yellow player) is now assigned to the blue player (*switch*). (d) Tracks 16 and 20 after the errors.

– **Connection horizon for new detections**, $\tau_{\max}$: It defines the maximum time in past, of the nodes, to which new detections are connected with a single hop, while incrementing the graph. It is defined formally in Equation 1 of the main paper. A larger $\tau_{\max}$ makes the system robust to the missed detections as it allows connections that are far apart in time. Therefore, higher values of $\tau_{\max}$ result in less misses and also in less re-initializations. However, higher values also result in denser graph which, in turn, increases the computational complexity. The results of effect of $\tau_{\max}$ on the performance of the algorithm are shown in Table 1.

Table 1: Performance with respect to $\tau_{\max}$.

$\tau_{\max}$	**30**	**60**	**120**	**240**
FP	7	24	20	22
MS	426	410	387	380
RE	162	137	113	111
SW	74	64	64	76

– **Missed detection coefficient**, γ: It penalizes the missed detections by increasing the cost of an edge proportionally to the time interval between the nodes. Please refer to Equation 1 in the main paper. Larger values of γ imply that consecutive connections are preferred over long "single hop" connections (thus, reduces misses). However, larger γ makes the algorithm less robust to missed detections (thereby, increasing misses). Hence, there is a trade-off between the γ and the error due to misses. The results are shown in Table 2 which confirm our intuition about the γ.

Table 2: Performance with respect to γ

γ	**1**	**2**	**3**	**4**	**5**
FP	34	20	20	20	11
MS	429	382	387	382	399
RE	125	116	113	118	118
SW	72	83	64	70	71

– **Window proportionality constant**, κ: It controls the extent of the neighborhood to be investigated around the key-node. Lower values of κ mean that the neighborhood is quite local around the key-node, i.e., nodes which are far apart are not investigated. Consequently, the tracker commits less switching error and less false positives but at the expense of more errors due to misses and reinitializations, as presented in Table 3.

Table 3: Performance with respect to κ.

κ	**1**	**3**	**5**	**7**
FP	14	19	20	33
MS	417	408	387	380
RE	120	126	113	110
SW	62	64	64	73

– **Best-cost to second-best-cost ratio**, K_2: This is a very crucial parameter since it controls the ambiguity of computed shortest path. Smaller values imply that the computed shortest path is considered reliable only when the best cost is very small as compared to the second best cost. Consequently, only highly reliable nodes are connected and error due to switching decreases. At the same time, many plausible aggregations will be discarded, resulting in the increase of errors due to misses and re-initializations. The results are shown in Table 4.

Table 4: Performance with respect to K_2.

K_2	**1/1.5**	**1/2**	**1/3**	**1/5**
FP	70	32	20	25
MS	363	377	387	445
RE	101	107	113	149
SW	97	81	64	60

– **Lower and upper thresholds for the computation of reliability of the feature**, $(C_{\min}, C_{\max})$: They are the lower and upper thresholds for the

ACCV2012
#633

computation of reliability of the feature, as introduced in the Equation 5 in he main paper. In the following table, we present the effect of these thresholds for the digit feature. Obviously, if we choose these values conservatively, the switching error will be lower. But, the errors due to misses and re-initializations will increase, as shown in Table 5.

Table 5: Performance with respect to $(C_{\min}, C_{\max})$.

$(N_{\min}, N_{\max})$	(20,100)	(20,70)	(20,50)	(5,50)	(20,30)
FP	20	19	19	22	19
MS	387	400	402	390	402
RE	113	127	116	133	117
SW	64	65	70	67	75