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

Amit Kumar K.C., Damien Delannay, Laurent Jacques, and Christophe De Vleeschouwer
{amit.kc, laurent.jacques, christophe.devleeschouwer}@uclouvain.be,
damien.delannay@keemotion.com

ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium**

Abstract. This paper assumes prior detections of multiple targets at each time instant, and uses a graph-based approach to connect those detections across time, based on their position and appearance estimates. In contrast to most earlier works in the field, our framework has been designed to exploit the appearance features, even when they are only sporadically available, or affected by a non-stationary noise, along the sequence of detections. This is done by implementing an iterative hypothesis testing strategy to progressively aggregate the detections into short trajectories, named tracklets. Specifically, each iteration considers a node, named key-node, and investigates how to link this key-node with other nodes in its neighbourhood, under the assumption that the target appearance is defined by the key-node appearance estimate. This is done through shortest path computation in a temporal neighbourhood of the key-node. The approach is conservative in that it only aggregates the shortest paths that are sufficiently better compared to alternative paths. It is also multi-scale in that the size of the investigated neighbourhood is increased proportionally to the number of detections already aggregated into the key-node. The multi-scale and iterative nature of the process makes it both computationally efficient and effective. Experimental validations are performed extensively on a 15 minutes long real-life basketball dataset, captured by 7 cameras, and also on PETS'09 dataset.

1 Introduction and Overview

Multi-object tracking is a fundamental issue in computer vision. It supports high-level semantic scene analysis in numerous and various applications. Vehicle trajectories are, for example, collected to control traffic monitoring solutions [1]. People displacement analysis is important to improve the security of public spaces [2], or to understand sport actions [3], for example.

Due to the recent improvement in object detection, many detection-based approaches have been proposed to handle the multi-target tracking problem. In

** Part of the work is supported by the Belgian NSF and by the WIST3 Walloon Region project SPORTIC.

such approaches, plausible object locations are first estimated in each individual frame,together with some features characterizing the appearances of the detected objects. Afterwards, graph-based solutions are generally envisioned to match the detections in consecutive frames. For example, in [4], authors explicitly model the spatial layout and mutual occlusion constraints, and use linear programming relaxation to solve the multi object tracking problem. The K-shortest paths approach has been investigated in [5], while the authors in [6] use min-cost flow network and greedy approach to estimate the number of tracks, as well as their birth and death states. In [7], the problem is formulated as a global maximum a posteriori estimation over a directed acyclic hyper graph. Similarly, [8] casts the problem into finding maximum weighted independent sets of a graph.

As a main limitation, most of these methods implicitly assume that the appearance features are known with the same level of reliability all along the time. Therefore, they exploit them in a uniform manner with time. In practice, however, most features are subject to non-stationary noise, and their ability to discriminate objects varies with the scene context. For example, colour histograms appear to be quite noisy in presence of occlusions, and object positions do not help to disambiguate a clutter of detections. In some other cases, highly discriminant appearance features are only available sporadically (and under certain configurations only). For example, in sports, a number on a jersey is visible only when facing the camera. In such time-varying observation process, the task of tracking of objects, while taking into account the position and all the available appearance features, is non-trivial.

To illustrate this, we now explain the limitations of conventional graph-based approaches when dealing with sporadic/noisy features. In short, graph-based approaches assign a node to each detection. Edges are then defined to connect the nodes, and each edge gets a cost that reflects the *dissimilarity* between the two nodes it connects. Afterwards, a (K-)shortest-path algorithm is generally applied to find the trajectories of the (K) targets. The approach has proven to be effective in scenarios for which the features are collected with same level of accuracy for each detection. In contrast, the approach is not appropriate in cases for which appearance features are noisy or missing at some time instants. This is because the (in)consistency of the appearances observed along a path cannot any more be measured simply based on the accumulation of the appearance (dis)similarities measured between pairs of consecutive nodes, since the relevance of those dissimilarity measurement directly depends on the availability and reliability of the corresponding appearance estimates. Hence, most graph-based methods fail to properly exploit noisy or sporadic appearance cues.

To address the above limitations, our paper introduces a new paradigm to aggregate detections into objects trajectories. Similar to numerous previous works, it adopts a graph-based formalism, but fundamentally goes from a paradigm that builds on comparisons of appearances between consecutive nodes towards a new paradigm that investigates the graph under some target appearance hypothesis.

Each iteration of the algorithm works as follows. A node, named key-node, is selected to define a target appearance hypothesis. Given this hypothesis, a

shortest-path computation algorithm is considered to investigate how to aggregate the key-node with its temporal neighbours in the graph, while promoting the nodes that share this target appearance hypothesis. The process is repeated iteratively, each node becoming a key-node at some step of the algorithm, and each tracklet defining the nodes of the updated graph to be used in subsequent iterations of the algorithm. To limit the computational complexity while giving the opportunity to build long trajectories, we adopt a multi-scale strategy to define the size of the temporal neighbourhood to investigate around a key-node. Specifically, the size of the observation window is made proportional to the size of the key-node, i.e. to the number of detections that have been aggregated into the key-node during the earlier steps of the algorithm. In addition, to avoid misleading the overall multi-object tracking process due to a wrong aggregation decision, *e.g.*, caused by some inappropriate appearance hypothesis, the shortest-path connecting the key-node to the extremity of its observation neighbourhood is only validated when it is sufficiently shorter than the alternative paths connecting each one of its extremities to the opposite extremity of the observation window.

The advantages of our proposed approach can be summarized as follows. Primarily, it naturally favours the aggregation of detections that share a similar appearance, even if those detections are not adjacent in time. It can also naturally account for different levels of reliability in the observation process, typically by giving more credit to the reliable appearance measurements when defining the cost associated to the discrepancy between the target appearance hypothesis and a node appearance estimate. Hence, the algorithm becomes able to effectively exploit sporadic or noisy features, which is a significant step forward compared to the state-of-the-art. As a second advantage, the multi-scale and progressive nature of the algorithm not only mitigates its computational load, but also helps in selecting long term matching based on more reliable tracklet appearance estimations, which directly benefits to the overall accuracy of the algorithm.

To the best of our knowledge, the only graph-based previous work exploiting sporadically available appearance cues has been presented in [9]. In this work, the authors assume a discrete set of N possible appearances, and end up in a K-shortest paths computation on a N-layered graph, K being the number of targets, and N corresponding to the number of appearance hypothesis. Our method is more flexible than [9], in the sense that it does not require prior knowledge of the discrete set of possible appearances, as required in [9]. Our approach naturally adapts to the observation of new appearances in the scene, and can handle an arbitrary number of appearances. In addition, our approach is computationally efficient due of its iterative and multi-scale nature. In particular, it avoids the (slow) linear programming optimization required in [9]. Moreover, we avoid the computational burden associated to the construction of a N-layer graph by embedding the hypothesis testing within an iterative local aggregation framework.

The rest of the paper is organized as follows. Section 2 defines the graph terminology. The tracking algorithm is explained in Section 3. Section 4 discusses

the experimental results and demonstrates the approach on a real-life basketball dataset.

2 Graph Formalism and Notations

As an input, the algorithm receives the set of candidate targets detected independently at each time instant, as described in [10]. Apart from the detection time t and the location $\boldsymbol{x}$, the detector computes K appearance features $\boldsymbol{f}_i$ $(1 \leq i \leq K)$ for a target. Since a feature might be noisy or even missing, the detector outputs a confidence value $c_i \in [0, 1]$ for each feature ($c_i = 0$ standing for a missing feature). A detection $\mathbf{d}$ is therefore characterized by the vector

$$\boldsymbol{d} = (t, \boldsymbol{x}, \mathcal{F}, \boldsymbol{c}),$$

where $\mathcal{F} = \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_K\}$ and $\boldsymbol{c} = (c_1, \cdots, c_K)$. The set of detections at a given time t is denoted as $\mathcal{D}^t$. As introduced in Section 1, the proposed algorithm adopts a graph-based formalism to progressively aggregate the detections into *tracklets* using a graph-based formalism. We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ by:

- a set of nodes, with each node corresponding to a tracklet, *i.e.*, $\mathcal{V} = \{v_k : 1 \leq k \leq \#\mathcal{V}\}$,
- a set of edges, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, defining the connectivity between the nodes in $\mathcal{V}$,
- and a set of weights, $\mathcal{W} : \mathcal{E} \to \mathbb{R}^+$, weighting these nodes and edges.

Initially, individual detections define the nodes of the graph. Detections are then aggregated into tracklets, which define the nodes of the updated graph. The proposed iterative aggregation process is presented in details in Section 3, including the definition of cost and edges between nodes. Here, we only introduce the associated terminology. Formally, along the aggregation process, a tracklet v is defined to be collection of chained detections, *i.e.*, $v = \left(\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, \cdots, \boldsymbol{d}^{(N)}\right)$, N being the length of the tracklet, also denoted as $\mathcal{L}(v) = N$. Notice that the chain is ordered, in the sense that the detection times $t_{d^{(i)}}$, $0 < i \leq N$, are such that $t_v^{(s)} = t_{d^{(1)}} < t_{d^{(2)}} < \cdots < t_{d^{(N)}} = t_v^{(e)}$, with $t_v^{(s)}$ and $t_v^{(e)}$ respectively denoting the starting and ending time of the tracklet. Figure 1 depicts how the tracklets are gathered into a graph in the proposed framework.

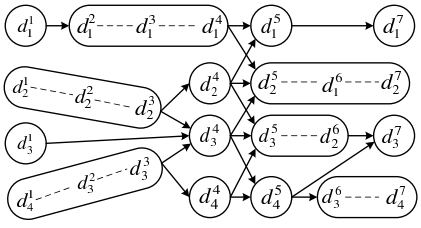

Fig. 1: Our graph formalism. The kth detection at time t is denoted by d_k^t. Some of them are aggregated into tracklets. Each node corresponds to a tracklet. An edge that connects two nodes u and v has a cost $\mathcal{W}(u, v)$.

Notice that pairs of tracklets are connected only between their extremities, in such a way that each connection maintains the increasing ordering of the detection times composing the two tracklets. The weight $\mathcal{W}(u, v)$ is introduced to denote the linking cost between two nodes $u, v \in \mathcal{V}$. It is formally defined in Section 3. In short, it typically decreases with the likelihood that nodes u and v correspond to the same physical target. In addition, we introduce the *inner cost* $\mathcal{W}(v, v)$ of a node v to denote the cost of traversing tracklet v from its starting time to its ending time. It typically depends on the length $\mathcal{L}(v)$ of the node, *i.e.*, $\mathcal{W}(v, v) = \rho\mathcal{L}(v)$, and is introduced to avoid that long nodes create 'short-cuts' in the graph.

In the sequel, we use two more graph notations. Since we consider a recursive algorithm that incorporates new detections at each time instant t, the graph is continuously incremented with time, and is denoted $\mathcal{G}^t$ at a given time t. Second, $\mathcal{G}^t_{[t_1,t_2]}$ represents a graph formed by selecting in $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \mathcal{W}^t)$ the tracklets $v \in \mathcal{V}^t$ having at least one extreme time component inside the temporal window $[t_1, t_2]$. The connectivity $\mathcal{E}^t$ and the weight $\mathcal{W}^t$ are restricted accordingly from these selected tracklets in order to form $\mathcal{E}^t_{[t1,t2]}$ and $\mathcal{W}^t_{[t1,t2]}$.

3 Iterative Hypothesis Testing

The global flow of our proposed iterative aggregation algorithm is presented in Algorithm 1. We observe that the graph manipulated by our method is continuously incremented by the new detections, encountered as time is evolving. Once all the detections computed at time t have been connected to the previous graph, the algorithm iteratively investigates how each node of the graph can be aggregated with its neighbours. This investigation starts with a node, named key-node, and is performed either in the forward or in the backward direction. As controlled by the *dir* flag in Algorithm 1, the direction of investigation changes at each time instant to propagate the appearance hypothesis associated to the key-node both towards the future and the past of this key-node, thereby making the global process symmetric with respect to time.

The remainder of the section details the behaviour of the two functions introduced in Algorithm 1, namely *IncrementGraph* and *IterativeAggregation*. The incrementation of the graph with time is presented in Section 3.1. The core of our proposed multi-scale and iterative aggregation strategy is then explained in Section 3.2.

3.1 Incrementing the Graph with Time

At time $t = 1$, the graph is just a set of detections at that instant, *i.e.*, $\mathcal{G}^1 = \big(\mathcal{D}^1, \emptyset\big)$. At time $t > 1$, the graph is obtained by adding new detections to the so-called previous graph, $\mathcal{G}^{t-1}$, resulting from earlier steps of the algorithm, up to time $t - 1$.

The procedure is referred to as **IncrementGraph** in Algorithm 1. It connects the detections $\mathcal{D}^t$ at time t to the graph $\mathcal{G}^{t-1}$. All nodes ending later than time $t-\tau_{\max}$ are linked to all the current detections. We set $\tau_{\max} = 120$. This allows to investigate connections up to 6 seconds (at the frame rate of 20 fps) in the past, which is sufficient to make the algorithm robust to most practical occurrence of

Algorithm 1 Recursive aggregation of tracklets with time

Input: Set of detections, $\mathcal{D}^t$
Output: Graph at time t, $\mathcal{G}^t$
Procedure:
dir $\leftarrow +1$ {/* Note: dir=1 means forward and -1 means backward aggregation */}
for each time instant t **do**
  if $t = 1$ **then**
    $\mathcal{G}^t = \left(\mathcal{D}^t, \emptyset\right)$
  else
    $\mathcal{G}^t \leftarrow$ IncrementGraph($\mathcal{G}^{t-1}, \mathcal{D}^t$) {/* See Section 3.1 */}
    $\mathcal{G}^t \leftarrow$ IterativeAggregation($\mathcal{G}^t$, dir) {/* See Section 3.2 */}
    dir $\leftarrow$ -dir
  end if
end for

missed detections. We present the effect of this parameter on the performance in the supplementary material. The set of links (or edges), connecting the detections computed at t, is denoted $\mathcal{E}^t$. Those edges are directed and "time-forwarded". Therefore, the graph $\mathcal{G}^t$ is directed and acyclic (DAG), and permits only causal traversals. Nevertheless, the graph can be globally reversed in order to allow anti-causal paths for processing purposes (see Section 3.2). Each edge $e \in \mathcal{E}^t$ is characterized by a cost which measures the distance between two nodes. As the appearance features are often unreliable or even unavailable for most elements in $\mathcal{D}^t$, the distance between a node $v \in \mathcal{V}^{t-1}$ and a new detection $d \in \mathcal{D}^t$, is computed based only on the position and motion related parameters. Specifically, the cost $\mathcal{W}^t(v, d)$ is defined as

$$\mathcal{W}^t(v,d) = \begin{cases} \left[1 + \gamma \times (t - t_v^{(e)} - 1)\right] g_{\text{sp}}(v,d) & \text{if } t - t_v^{(e)} \leq \tau_{\max}, \\ \infty & \text{else,} \end{cases} \tag{1}$$

with the metric g_{sp} measuring the distance between detection d and the predicted position of the object corresponding to node v. It is defined as:

$$g_{\text{sp}}(v,d) = \left\| \boldsymbol{x}_d^{(s)} - \boldsymbol{x}_v^{(e)} - \dot{\boldsymbol{x}}_v^{(e)} \left(t - t_v^{(e)}\right) \right\|_2, \tag{2}$$

where the term $\dot{\boldsymbol{x}}_v^{(e)}$ is the velocity, at the end of tracklet v. It is zero for unit length tracklets, and is computed from the last 2 detections of the tracklet otherwise. The factor $\gamma > 0$, typically set to 3, introduces penalty for missed detections.

Track creation, deletion and duplicate detections Here, we provide a brief remark on how a track is created and deleted. Each node corresponds to a track and we create node as soon as a novel detection is introduced. It is removed either if it is aggregated with other nodes, or if its length is too small compared to its distance to the current time, i.e., $\mathcal{L}(v) \ll t - t_v^{(e)}$. In the latter case, we consider it as false positive.

In a typical tracking scenario, duplicate detections arise which might result in two tracklets that partly overlap in time; despite they correspond to the same object. Such time overlap prevents the desired aggregation of the tracklets in a graph that is only composed of forward links. To mitigate this, we introduce backward links in such a way that the graph is still directed and acyclic. They make the aggregation of overlapped tracklets possible.

3.2 Iterative Tracklet Aggregation

Once the graph has been incremented with novel detections, the objective is to aggregate the nodes that correspond to the same physical object. To exploit appearance cues that are noisy, or only available sporadically. Therefore, we cannot rely on conventional propagation of appearance similarity measures between consecutive nodes. Instead, we promote a novel aggregation paradigm, founded on iterative hypothesis testing process.

Overview of the contribution In this approach, each iteration selects a node, named key-node, and studies how to aggregate this key-node with its forward or backward neighbourhood, under the assumption that the observed key-node appearance defines the reference appearance of the tracked object. Given this hypothesis, paths that go through nodes that do (not) share the key-node appearance are promoted (penalized). This is done simply by decreasing (increasing) the cost to go through a node of the graph when the appearance of that node is similar (different) to that of the key-node. Hence, all appearance cues, even the sparse or inaccurate one, can be exploited to drive the selection of aggregated paths within the graph. Since the process is repeated with each node being the key-node, all observed appearance hypotheses are examined.

The hypothesis testing approach is complemented by a multi-scale strategy. It consists of defining the size of the neighbourhood to be proportional to the length of the key-node. The advantages are twofold. First, the approach makes sense from the tracking efficiency point of view, since longer nodes are more likely to have accumulated reliable and accurate knowledge about their appearance, which should be exploited to connect them with other nodes with the same appearance, even if they are far away. Second, with respect to the computational complexity, the fact that long time frame neighbourhoods are only investigated when detections already got the opportunity to be aggregated into tracklets of sufficient length reduces the actual number of nodes to be considered when dealing with large neighbourhood windows. Moreover, our approach adopts an extreme caution while aggregating the nodes in the graph. A plausible aggregation computed during an iteration will only be validated and integrated to the graph structure for subsequent iterations if it is reliable enough.

We now detail the practical implementation of the the algorithm, which is presented in Algorithm 2.

Multiscale Iterative Hypothesis Testing Formally, the key-node is denoted v_{key}. It is selected at each iteration among the set of nodes, $\mathcal{R}^t$, that have not yet been investigated at time t. The aggregation of the key-node with its neighbours is then investigated in an observation window that precedes or follows

the key-node, depending on the sign of the *dir* flag. The size of the observation window is proportional to the length of the key-node. As explained earlier, this proportional definition allows traversal in different time-scales. We use Δ to denote the observation window interval. Hence, $\Delta = [t^{(e)}_{v_{\text{key}}}, t^{(e)}_{v_{\text{key}}} + \kappa\, \mathcal{L}(v_{\text{key}})]$ in the forward mode ($dir = 1$), or $\Delta = [t^{(s)}_{v_{\text{key}}} - \kappa\, \mathcal{L}(v_{\text{key}}), t^{(s)}_{v_{\text{key}}}]$ in the backward mode ($dir = -1$), where κ is the window proportionality constant. We use $\kappa = 5$. Detailed results on varying κ are provided on the supplementary material.

Given the key-node v_{key} and its observation window Δ, we define the graph $\mathcal{G}_\Delta$ to investigate how the key-node can be aggregated with its neighbours to define an appearance-consistent path under the assumption that the tracked object appearance is defined to be the key-node appearance. The function is named **GraphHypothesis** because it returns the graph that is used to test the key-node appearance hypothesis. The graph $\mathcal{G}_\Delta$ is directly derived from the graph $\mathcal{G}^t$, by cutting $\mathcal{G}^t$ according the limits of the observation window, and updating the inner costs of the nodes within the window to reflect the hypothesis made about the target appearance. In short, the inner cost $\mathcal{W}(v, v)$ of a node $v \in \mathcal{V}_\Delta$ is increased (decreased) if it has a different (similar) appearance than the one of the key-node.

The inference of the tracklet features based on the features observed in the set of detections directly depends on the characteristics of the features observation process. If, for example, the observation process is affected by outliers, a RANSAC approach could help in capturing the right appearance model. On the other hand, if the observations are independent and affected by Gaussian noise, then a weighted average provides an appropriate inference. In this paper, we use a weighted average for the tracklet appearance as an example of practical implementation. Then, the average i^{th} feature of a node v is computed as:

$$\overline{\boldsymbol{f}}_i^{(v)} = \tfrac{1}{C_i} \sum_{t=1}^{\mathcal{L}(v)} c_{i,v}^{(t)} \boldsymbol{f}_{i,v}^{(t)}, \tag{3}$$

where $C_i = \sum_{i=1}^{\mathcal{L}(v)} c_{i,v}^{(t)}$. In particular, $\overline{\boldsymbol{f}}_i^{(\text{ref})}$ denotes the average i^{th} feature of the key-node, used as an hypothesis reference. Let $D(v)$ denote the value by which the inner cost of node v is incremented due to its dissimilarity with the appearance of the key-node. Then,

$$D(v) = N \sum_{i=1}^{K} \underbrace{\left(\alpha_i \left\| \overline{\boldsymbol{f}}_i^{(\text{ref})} - \overline{\boldsymbol{f}}_i^{(v)} \right\|_1 + (1 - \alpha_i) w_i^{(\text{fix})} \right)}_{:= w_i^{(v)}} \tag{4}$$

The parameter α_i is introduced to give less weight to the appearance features that are computed on short tracks as they are prone to error. L_1 norm is chosen for the computational reasons, but could be replaced by other metrics like Bhattacharyya distance, etc.

$$\alpha_i = \begin{cases} 0 & \text{if } C_i \leq C_{\min}, \\ 1 & \text{if } C_i \geq C_{\max}, \\ \frac{C_i - C_{\min}}{C_{\max} - C_{\min}} & \text{otherwise.} \end{cases} \tag{5}$$

Algorithm 2 IterativeAggregation

Input: Graph at time t, $\mathcal{G}^t$; Direction of aggregation, dir
Output: Updated graph at time t, $\mathcal{G}^t$

Procedure:

Initialize: $\mathcal{R}^t \leftarrow \mathcal{V}^t$ {/* $\mathcal{R}^t$ is the set of nodes that are yet to be scheduled for hypothesis testing. */}
while $\mathcal{R}^t \neq \emptyset$ **do**
$v_{\text{key}} \leftarrow \text{Schedule}(\mathcal{R}^t)$
$\Delta \leftarrow$ Limits of the observation window
$\mathcal{G}_\Delta \leftarrow$GraphHypothesis$(\mathcal{G}^t, \Delta, v_{\text{key}})$
$v_{\text{agg}} \leftarrow \text{Aggregate}(v_{\text{key}}, \mathcal{G}_\Delta)$
if $v_{\text{agg}} \neq v_{\text{key}}$ **then**
$\mathcal{G}^t \leftarrow \text{Simplify}\big(\mathcal{G}^t, v_{\text{agg}}\big)$
end if
$\mathcal{R}^t \leftarrow \mathcal{R}^t \setminus v_{\text{agg}}$
end while

Aggregate: *Refer to Figure 2 for the illustration.*
Input: Key-node, v_{key}; Windowed graph for testing key-node hypothesis, $\mathcal{G}_\Delta$
Output: Set of nodes that can be aggregated, v_{agg}
Procedure:
$(S_b, S_{\text{sb}}) \leftarrow$ Best and second best shortest paths from v_{key} to the other extremity of $\mathcal{G}_\Delta$
if $\text{Cost}(S_b) \ll \text{Cost}(S_{\text{sb}})$ **then** {/* Here $\ll$ signifies that path S_b is sufficiently better than path S_{sb}. */}
$\mathcal{G}_\Delta^- \leftarrow$ReverseDirection$(\mathcal{G}_\Delta)$
$(S_{b'}, S_{\text{sb}'}) \leftarrow$ Best and second best shortest paths from v_b to the other extremity of $\mathcal{G}_\Delta^-$
if $\text{Cost}(S_{b'}) \ll \text{Cost}(S_{\text{sb}'})$ **then**
if $v_{b'} = v_{\text{key}}$ **then**
$v_{\text{agg}} \leftarrow S_b$
end if
end if
else
$v_{\text{agg}} \leftarrow v_{\text{key}}$
end if

In Algorithm 2, the function **Schedule** selects a node for hypothesis testing that has not yet been scheduled. Different scheduling mechanisms can be envisioned. However, in this paper, we select the nodes that are "sufficiently long" and "not so far from t". That is, we schedule the nodes in decreasing order of $\mathcal{L}(v)/\max\{1, t - t_v^{(e)}\}$. Since long nodes are more likely to have accumulated sufficient appearance features, they get more priorities. Similarly, by exploring nodes that are recent in time, we prevent the fast growth of the graph.

where $C_{\min}$ and $C_{\max}$ are the limits to define if the feature is considered reliable or not. We set $C_{\min} = 20$ and $C_{\max} = 100$. An analysis of effect of these values on the performance is presented in the supplementary material. When $\alpha_i \to 1$, $w_i^{(v)} \to \left\| \overline{\boldsymbol{f}}_i^{(\text{ref})} - \overline{\boldsymbol{f}}_i^{(v)} \right\|_1$ and when $\alpha_i \to 0$, $w_i^{(v)} \to w_i^{(\text{fix})}$. The term $w_i^{(\text{fix})}$ is introduced so that a node, which definitely looks similar to the key-node ($D(v) \approx 0$), is favoured compared to a node for which no appearance features is available $\left(D(v) \approx N \sum_i w_i^{(\text{fix})}\right)$. It corresponds to the noise level, affecting the feature. Empirically, we set $w_i^{(\text{fix})} = 5$ for all $1 \leq i \leq K$ and is related to the unit of the detection. After the inner costs of the nodes have been incremented by $D(v)$, a shortest path algorithm is applied. For this, the DAG shortest path algorithm is preferred because of the inherent directed and acyclic nature of the graph. The cost of a path is defined to be the sum of costs of the edges and the inner costs of the nodes along it, and is given by the function **Cost** in the algorithm.

Even though it seems that updating the costs requires additional scanning of the graph, it is mitigated by the concept of *visitors* in the shortest path algorithm. The visitors allow to update the costs of the nodes or edges "in place" by invoking various events.

Path Ambiguity Estimation and Validation Having the cost of edges been defined in order to take the displacement as well as the appearance into consideration, the shortest path S_b, which connects the key-node to a set of nodes within the window, reasonably corresponds to a single physical object (same appearance, and coherent motion) and that could thus be aggregated into a single node. To limit the risk of connecting nodes that correspond to two distinct objects, we check the level of ambiguity of the shortest path by comparing its cost to the costs of a set of paths that could constitute reasonable alternative to connect the extremities of the shortest path to the opposite extremity of the observation window. Figure 2 illustrates this process.

This validation process is run in two steps. In the first step, the shortest S_b and the second shortest S_{sb} paths are considered. Moreover, the ends of the best and second-best paths are denoted as v_{b} and v_{sb} respectively. The shortest path S_b is considered being sufficiently better than S_{sb} only if several conditions are met: (i) $\text{Cost}\big(S_{\text{b}}\big) < K_1$, (ii) $\text{Cost}\big(S_{\text{b}}\big)/\text{Cost}\big(S_{\text{sb}}\big) < K_2$, and (iii) $\text{Cost}\big(S_{\text{sb}}\big) > K_3$. The thresholds K_1 and K_3 vary linearly with Δ. We set $K_2 = 1/3$. The sensitivity on varying K_2 is detailed in the supplementary material.

If all conditions are met, the second step of the validation process is considered. For this, the graph is reversed by flipping the direction of all the edges of $\mathcal{G}_\Delta$. It is mentioned as **ReverseDirection** in the algorithm. The shortest ($S_{b'}$) and second shortest ($S_{\text{sb}'}$) paths linking v_{b} with the opposite extremity of the observation window are then computed. If $S_{b'}$ leads to the original key-node, *i.e.*, if $v_{\text{b}'} = v_{\text{key}}$, and if a similar set of conditions hold for $S_{b'}$ and $S_{\text{sb}'}$, then the path S_b is considered to be *unambiguous*, and is replaced by a single node in the path. This procedure is called **Simplify** in the algorithm. It updates the appearance features of the node as in Equation 3 and also the motion parameters. It keeps only the edges connecting the extremities of the aggregated path to the rest of

the graph. Other connections involving intermediate nodes are removed. Since all the nodes along S_b are aggregated into a single node (and thus the intermediate nodes are removed from the graph), it resembles the greedy matching of the nodes and is thus suboptimal.

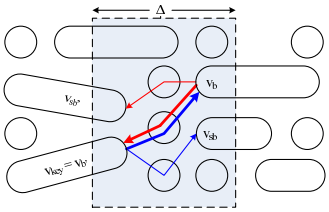

Fig. 2: Illustration of the **Aggregate** function in Algorithm 2. Within the window, the best (thick arrow) and the second best (thin arrow) paths are searched. Blue and red arrows represent forward and backward directions respectively.

4 Evaluation

The proposed algorithm has been evaluated on the APIDIS dataset [11]. The dataset has been generated by 7 cameras distributed around a basket-ball game. The candidate detections on the ground plane at each frame are independently computed, as described in [10]. The ID and the position of players have been manually defined at every second of a 15 minutes period. This provides the reference ground truth used in our evaluation.

In the remainder of the section, we first present the appearance features that are considered to support the tracking algorithm. We then discuss the multi-object tracking evaluation metrics, followed by the results and comparison with other approaches.

4.1 Appearance Features

In the APIDIS dataset, the jersey colour and the digit printed on it ($K = 2$), are computed for each candidate detection. Specifically, a rectangular box is positioned at the expected height of the player shirt. The first feature is the average colour, which is computed as the average blue component divided by the sum of average red and green components, over the silhouette of the player within the rectangular box. However, depending on whether the detection is close (far) from the camera, and also whether the detection is visible (occluded) in each camera view, the measurement is considered (discarded). The second feature is a set of candidate shirt digits, obtained by running a digit-recognition algorithm in the same rectangular region. These features are computed for all available cameras. Finally, the confidence of the feature is computed as the ratio of number of cameras, for which features are computed, to the total number of available cameras. A sample of such a measurement is shown in Figure 3 for a specific player (digit=8, colour=0.25). It can be easily observed that the appearance characteristics are noisy and sometimes missing in our scenario. Indeed, these features cannot always be reliably measured for each frame because of occlusions, illumination change, unfavourable shirt orientation with respect to the camera, etc.

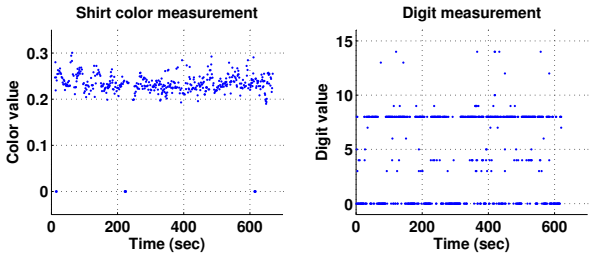

Fig. 3: Shirt colour and digit measurement along time for a player (digit=8, colour=0.25). Zero values indicate that no measurements are available.

4.2 Performance Metrics

We evaluate the performance of the proposed tracking algorithm based on the well-known CLEAR-MOT metrics [12]. A standard metric for evaluating object trackers is the Multiple Object Tracking Accuracy (MOTA), defined as:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{m}_\text{t} + \text{fp}_\text{t} + \text{re}_\text{t} + \text{sw}_\text{t})}{\sum_t \text{g}_\text{t}}, \tag{6}$$

where g_t is the number of ground-truth objects present at time t; m_t, fp_t, re_t and sw_t are the number of misses, false positives, track reinitializations and track switches. We write, $\text{FP} = \sum_t \text{fp}_\text{t}, \text{RE} = \sum_t \text{re}_\text{t}, \text{SW} = \sum_t \text{sw}_\text{t}, \text{MS} = \sum_t \text{m}_\text{t}, \text{GT} = \sum_t \text{g}_\text{t}$. A switching error occurs when the tracker starts following another object, whereas a reinitialization error occurs when the tracker fails to track the object at some time and a new track is assigned for the same object later on. The error due to switching is more problematic as it might lead to significant errors in higher level interpretation of the scene. An error due to a miss means that the tracker does not have any estimate for the corresponding ground truth. Similarly, a false positive represents that the tracker outputs some estimate for which no ground truth position is available.

4.3 Results

Figure 4 compares the performance obtained by the proposed algorithm when different set of appearance features are exploited. The results are obtained by enabling (or disabling) certain features in the algorithm and running on the 15 minutes long video sequence. There are all together 7460 ground truth positions, *i.e.*, GT=7460. As we can see, the switches and re-initializations are reduced substantially. However, the false positives increase slightly. When we incorporate only the digit feature, it can be seen that digit features, even though they are highly sparse, can disambiguate some tracks. However, the improvements are not as ample as those from the colour feature. It can be justified due to the fact that colour feature is available more often than the digit feature, and hence some tracks might not have accumulated sufficient digit information. When both features are used, not only the switches are reduced but also the gaps between tracklets are bridged (thereby, reducing the re-initializations and misses). To compute the tracking effectiveness, the authors of [9] applied their method on the same APIDIS dataset. With their notion of MOTA, we achieve 94.5%, whereas they achieve 92.8%.

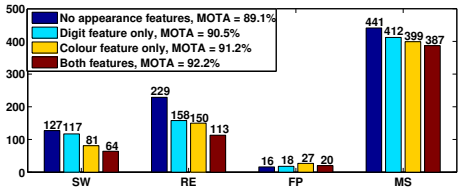

Fig. 4: Components of MOTA metric (15 min video) for different cases. Indeed, exploitation of color and digit features help to reduce the errors.

We compare our results with other approaches namely [13, 14, 5, 7]. These results have been adopted from [7] which have been computed just for 500 frames. Therefore, we also ran our algorithm on the same 500 frames. The results are shown in Table 1. As the results in [7] were computed only with colour, we also used just the colour feature. From Table 1, we can see that the proposed method outperforms several popular state-of-the-art methods, in scenarios relying on target appearance estimation, but also in scenarios that only exploit detections positions as input features. We explain the benefit observed in this latter case by the progressive and conservative nature of our proposed aggregation process. In case of matching ambiguities, instead of validating the shortest path (and possibly committing an error), the algorithm waits for more observations. When validating other unambiguous paths, their motion parameters are estimated more accurately which, in turn, can benefit in resolving the ambiguities.

Table 1: MOTA metric, computed on 500 frames of the APIDIS dataset

Method	**MOTA**
Track-before-detect [13]	0.614
Trajectory association [14]	0.781
K-shortest path [5]	0.586
Second order with exclusion [7]	0.735
Baseline: proposed method (position only)	**0.828**
Proposed method (position+colour feature)	**0.864**

In addition, our approach is applied on camera view 1 of the PETS'09 S2.L1 dataset. First, targets are detected by using the deformable parts model [15]. Then, 8-bin histograms are computed on RGB channels separately, which are then stacked to obtain a 24-bin feature vector. The feature is considered reliable only if the overlap between the bounding boxes is less than 20%. It takes only 42 seconds in MATLAB (3GHz 4-core CPU, 4 GB RAM), to estimate the feature and process all 795 frames. It results in (MOTA, MOTP)=(0.83, 0.74), which is competitive to several other tracking algorithms like Berclaz et al.[5] (0.82, 0.56), Breitenstein et al.[16] (0.75, 0.60) and Andriyenko et al. [17] (0.89, 0.56). These performance metrics are extracted from [17].

5 Conclusion and Future Perspectives

The paper proposed a framework for matching of detections while exploiting partial appearance features. It proceeds with hypothesis testing in an iterative

framework which considers the input data at different time scales. The iterative principle helps in aggregating the appearance observations on tracklets by computing unambiguous local paths, thereby creating nodes with more reliable appearance cues. It also reduces the size of the graphs, and thus the complexity, handled by successive iterations of the algorithm. The multi-scale aspect allows matching decisions to be taken at different time horizons and is elegantly embedded in the framework. Future work will focus on the generalized inference of tracklet appearance and also on investigating different scheduling mechanisms for selecting key-node.

References

1. Ocakli, M., Dermirekler, M.: Video tracker system for traffic monitoring and analysis. In: IEEE Signal Processing and Communication Applications. (2007)
2. Piciarelli, C., Micheloni, C., Foresti, G.: Trajectory based anomalous event detection. IEEE Transactions on Circuits and Systems for Video Technology (2008)
3. Alahi, A., Boursier, Y., Jacques, L., Vandergheynst, P.: Sports players detection and tracking with a mixed network of planar and omnidirectional cameras. In: ICDSC, Como, Italy. (2009)
4. Jiang, H., Fels, S., Little, J.: A linear programming approach for multiple object tracking. In: CVPR. (2007)
5. Berclaz, J., Fleuret, F., Turetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. PAMI **33** (2011) 1806–1819
6. Pirsivash, H., Ramanan, D., Fowlkes, C.C.: Globally-optimal greedy algorithms for tracking a variable number of objects. In: CVPR. (2011)
7. Russell, C., Setti, F., Agapito, L.: Efficient second order multi-target tracking with exclusion constraints. In: BMVC. (2011)
8. Brendel, W., Amer, M., Todorovic, S.: Multiobject tracking as maximum weight independent set. In: CVPR. (2011) 1273 –1280
9. Shitrit, H.B., Berclaz, J., Fleuret, F., Fua, P.: Tracking multiple people under global appearance constraints. In: ICCV. (2011)
10. Delannay, D., Danhier, N., Vleeschouwer, C.D.: Detection and recognition of sports(wo)men from multiple views. In: ICDSC, Como, Italy. (2009)
11. : (http://www.apidis.org/dataset/)
12. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. J. Image Video Process. **2008** (2008) 1:1–1:10
13. Taj, M., Cavallaro, A.: Multi-camera track-before-detect. In: ICDSC, Como, Italy. (2009)
14. Anjum, N., Cavallaro, A.: Trajectory association and fusion across partially overlapping cameras. In: AVSS. (2009) 201–206
15. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **32** (2010) 1627–1645
16. M. D. Breitenstein, F. Reichlin, B.L.E.K.M., Gool, L.V.: Online multiperson tracking-by-detection from a single, uncalibrated camera. PAMI **33** (2011)
17. Anton Andriyenko, K.S., Roth, S.: Discrete-continuous optimization for multi-target tracking. In: CVPR. (2012)