

Discriminative and Efficient Label Propagation on Complementary Graphs for Multi-Object Tracking

Amit Kumar K.C., *Student Member, IEEE*, Laurent Jacques,
and Christophe De Vleeschouwer, *Member, IEEE*

Abstract

Given a set of detections, detected at each time instant independently, we investigate how to associate them across time. This is done by propagating labels on a set of graphs, each graph capturing how either the spatio-temporal or the appearance cues promote the assignment of identical or distinct labels to a pair of detections. The graph construction is motivated by a locally linear embedding of the detection features. Interestingly, the neighborhood of a node in appearance graph is defined to include all the nodes for which the appearance feature is available (even if they are temporally distant). This gives our framework the uncommon ability to exploit the appearance features that are available only sporadically. Once the graphs have been defined, multi-object tracking is formulated as the problem of finding a label assignment that is consistent with the constraints captured each graph, which results into a difference of convex (DC) program. We propose to decompose the global objective function into node-wise sub-problems. This not only allows a computationally efficient solution, but also supports an incremental and scalable construction of the graph, thereby making the framework applicable to large graphs and practical tracking scenarios. Moreover, it opens the possibility of parallel implementation.

Index Terms

Computer vision, label propagation, sporadic features, multi-object tracking, graph labeling



• AKC, LJ and CDV are funded by the F.N.R.S. and are affiliated to the ISPGROUP, ELEN Department, ICTEAM Institute, Université catholique de Louvain (UCL), B-1348, Louvain-la-Neuve, Belgium.
E-mail: {amit.kc, laurent.jacques, christophe.devleeschouwer}@uclouvain.be

Discriminative and Efficient Label Propagation on Complementary Graphs for Multi-Object Tracking

1 INTRODUCTION

IN this paper, we address the problem of multi-object tracking (MOT). We assume that the targets of interest have been detected at each time instant [11], [27], [28] and that some appearance features have been extracted. Given this error-prone information, our objective is to link these detections into consistent trajectories using a graph-based formalism.

Conventionally, a graph-based formalism assigns 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 [12] is typically used to find the trajectories of the (K) targets. Alternatively, other approaches use network flow [19], maximum weighted independent set [24], etc. to solve the same problem. These approaches have been proven to be effective in scenarios for which the features are collected with the same level of accuracy and reliability for each detection. With such a stationary measurement process, the likelihood that the detections along a path correspond to the same physical object can be reasonably estimated based on the accumulation of dissimilarities (similarities) between (close to)consecutive nodes in the path. In contrast, these approaches are not appropriate in cases for which appearance features cannot be measured with same accuracy and reliability in every space and time co-ordinates. Such cases are prevalent in many real-life situations. For example, color histograms tend to be noisy in presence of occlusions. In some other cases, highly discriminative features are available only sporadically. This happens, for example, while imaging biological cells under varying illuminations, each illumination level highlighting certain features of the cell. As another example, a digit, printed on the jersey of a player, is available only when it faces the camera. In such cases, the task of tracking multiple objects, while exploiting such features, becomes non-trivial.

Two works have recently addressed this category of problems. In their formulation, the authors in [20] assume that a discrete set of L possible appearances is known beforehand, which allows the creation of a L -layered graph. In the i -th layer, running through a node is penalized when the appearance of the node is available and differs from the i -th presumed

appearance. Afterwards, a K -shortest path algorithm is applied to find the K shortest paths across the L -layers. This method demonstrates that exploiting sporadic features can significantly improve the tracking performance. However, it is restricted to cases for which the number and the appearance of the targets are known *a priori*.

In contrast, [8] does not make any assumption about the (number of) targets appearances. It proposes a widely applicable iterative hypothesis testing strategy to exploit appearances features that are corrupted by non-stationary noise or are only sporadically available. In short, the authors iteratively consider each node in the graph as a key-node, and investigate how to link this key-node with other nodes in its neighborhood, under the hypothesis that the appearance observed in the key-node position is representative of the actual appearance of the target. The method offers the advantage to handle cases for which the discrete set of possible appearances is not known *a priori*. The greedy and iterative nature of the algorithm makes it efficient from a computational and memory usage perspective (no L -layered graph). Its main disadvantage is that it is greedy and consequently does not guarantee the global optimality of the solution.

In this paper, we extend our initial contribution in [7]. We adopt a graph-based label propagation framework. Therefore, we construct a number of distinct graphs, one for each appearance feature, apart from the usual spatio-temporal graph. Additionally, we also construct an exclusion graph to reflect the fact that two detections that occur at the same time should be assigned to distinct labels. Hence, we construct $K + 2$ ‘complementary’ graphs (one spatio-temporal, K appearance, one exclusion), where $K \ll L$ is the number of appearance features. An example is shown in Figure 1. In case of a sport game, for example, the jersey color and the digit, printed on it, can be considered as two appearance features, and result in two distinct appearance graphs.

During graph construction, a node is assigned to each detection. For all the graphs but the exclusivity one, edges connect pairs of nodes with a weight that increases with the similarity between the nodes in terms of space, time or appearance. The higher the weight, the more likely the two nodes correspond to the same physical target. Exceptionally, the edges of

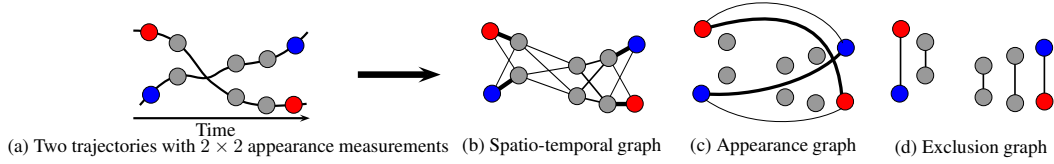


Fig. 1. **(a)** An example with two targets (red and blue) with associated detections at each time. Gray detections mean that no appearance feature is available. **(b)** Spatio-temporal graph that depicts the spatio-temporal association between the nodes, **(c)** Appearance graph that connects nodes even if they are far in time. **(d)** Exclusion graph in which edges connect nodes that coexist at the same time.

the exclusion graph only connect nodes that cannot belong to the same physical target. This is relevant, for example, when the detections occur at the same time.

Given these graphs, MOT problem is formulated as finding a consistent label assignment, which means that (i) the nodes that are sufficiently close in space/time and/or appearance are labeled similarly, and (ii) the nodes that co-exist at the same time are labeled differently. The consistency of labeling is measured by the labeling energy, which accumulates the difference in the labels between a node and other nodes that are connected to it. Due to the definition of weights in our graph, a good labeling should minimize the energy in the spatio-temporal and the appearance graphs while maximizing the energy due to the exclusion graph. Following our initial contribution in [7], our paper formulates the multi-object tracking with sporadic appearance features as a labeling problem in a number of complementary graphs. In addition to [7], it also proposes:

- an efficient solution to the labeling problem, splitting the ‘big’ problem into ‘small’ node-wise problems that can be solved locally, optionally based on a parallel implementation (Section 3),
- an extension of the local label propagation process to handle incremental/on-line tracking scenarios (Section 4).

Those two novel contributions make our proposed framework particularly suitable to practical scenarios.

The rest of the paper is organized as follows. Section 2 formulates the MOT problem as a consistent label assignment problem. Section 3 proposes the solutions to the label assignment problem. A brief review of the related work is presented in Section 5. Experimental results are presented in Section 6. Section 7 concludes our paper.

2 TRACKING PROBLEM FORMULATION

This section first describes the construction of the associated graphs. Afterwards, the multi-object tracking is formulated as a graph-consistent labeling problem.

2.1 Notation

Vectors and matrices are denoted with bold lower-case and upper-case symbols respectively while scalar

values are denoted by light ones. Upper-case calligraphic letters denote sets. Table 1 presents some of the symbols.

TABLE 1
Notations

K	number of appearance features
$\mathbf{x}_i$	feature vector of the i -th sample
$\mathcal{N}_i$	set of neighbors of the i -th sample
$\mathbf{X}^{(i)}$	features of the neighbors of the i -th sample, i.e., $\mathbf{X}^{(i)} := \cup_{j \in \mathcal{N}_i} \{\mathbf{x}_j\}$
$\mathbf{w}_i^*$	optimal reconstruction weights for the i -th sample
$\mathcal{G}$	a graph of node set $\mathcal{V}$, edge set $\mathcal{E}$ and weight $\mathbf{W}$
n	$ \mathcal{V} $, number of nodes
$\mathbf{L}_l^{(+)}$	Laplacian of the l -th graph for $l \in \{0, 1, \dots, K\}$, $l = 0$ for the spatio-temporal graph
$\mathbf{L}^{(-)}$	Laplacian of the exclusion graph
$\mathbf{y}_i$	label distribution assigned to the i -th node
m	size of the label vector.
$\mathbf{Y}$	$= (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)^T$, label assignment matrix
Δ_d	$= \{\mathbf{x} \in \mathbb{R}_+^d : \mathbf{1}^T \mathbf{x} = 1\}$, probability simplex of a given size d
$\mathcal{P}_{nm}$	set of all row-stochastic matrices of size $n \times m$
Parameters	
γ	Scaling factor for time (Section 2.2)
T	Connection window size for spatio-temporal graph (Section 2.2)
$v_{\max}$	Maximum speed for gating constraint (Section 2.2)
α_l	Weight assigned to the l -th labeling energy (Section 2.3)
T_c	Connection window size for appearance graph (Section 4.1)
σ	‘Heat’ parameter (Section 4.1)
T_o	Observation window for bounding complexity (Section 4.2)

2.2 Graph construction

We consider three distinct types of graphs. Hence, each graph should be constructed separately. Nevertheless, the constructions of spatio-temporal and appearance graphs follow the same approach, derived from the locally linear embedding (LLE) technique [26]. It assumes that data points can be accurately reconstructed by a weighted linear combination of their local neighbors. We motivate the linearity assumption by the fact that (i) target motion is linear in a small temporal window, and (ii) appearance features lie on a manifold. The number of neighbors is a design parameter, and should be chosen according to the kind of feature and the problem at hand.

In the following, we represent the feature of the i -th detection by $\mathbf{x}_i$ and that of its neighbors by $\mathbf{X}^{(i)} := \cup_{j \in \mathcal{N}_i} \{\mathbf{x}_j\}$, where $\mathcal{N}_i$ is the set of neighbors of

i. Afterwards, the graph construction is formulated as the problem of finding the reconstruction weights w_i^* that minimizes the following optimization problem

$$w_i^* = \underset{w_i \in \Delta_{|\mathcal{N}_i|}}{\operatorname{argmin}} \|x_i - X^{(i)} w_i\|_2^2 + \frac{\delta}{2} \|w_i\|_2^2, \quad (1)$$

where $\Delta_m := \{w \in \mathbb{R}^m \mid w \succeq 0, \mathbf{1}^\top w = 1\}$ is the probability simplex of a given size m . The reason to constrain the weights to belong to the simplex is that it promotes weight vector sparsity. To see this, we observe that the simplex constraint is equivalent to enforcing positive weights with unit ℓ_1 -norm, and first consider the case with $\delta = 0$ in Equation (1). When minimizing a quadratic fidelity as the one present in the first term of the cost of Equation (1) under such ℓ_1 -norm constraint, the solution is generally restricted to a small dimensional facet of the unit ℓ_1 -norm [38], [39], *i.e.*, a domain where the solution is sparse. We favor sparsity as it leads to an efficient optimization in Section 3. Promoting too much sparsity is however not desired. If a sample is similar to several other samples (*e.g.*, a feature occurs several times along the sequence of detections), the sparse reconstruction selects only one neighbour and ignores the rest. To mitigate this limitation, we add a quadratic part $\frac{\delta}{2} \|w_i\|_2^2$, which offers an additional advantage of making the problem strongly convex, resulting in a unique w_i^* . This can be seen as similar to an elastic net regularization [37] in the sense that the sparsity term is imposed by the constraints. We use $\delta = 10^{-2}$.¹

Once the weights for each data point are computed, we gather them into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where

- $\mathcal{V}$ is the set of nodes, with i -th node corresponding to the i -th detection. We denote the number of nodes by $n = |\mathcal{V}|$.
- $\mathcal{E}$ defines the connectivity between the samples such that an edge (i, j) is created between nodes i and j only when the weight $w_i^*(j)$, resulting from Equation 1, is non-zero, *i.e.*, $\mathcal{E} = \{(i, j) \mid w_i^*(j) > 0\}$.
- $\mathbf{W}$ assigns a weight to each edge such that

$$W_{ij} = \begin{cases} w_i^*(j) & \text{if } (i, j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Now, we explain the specific issues in the construction of each graph.

Spatio-temporal graph. In case of the spatio-temporal graph, x_i is defined by the time instant t_i and the location information c_i (*e.g.*, bounding box of the detections). Hence, $x_i = (\gamma t_i, c_i)^\top$, where γ affects the relative importance of the time difference compared to the location difference between the data points. A non-zero γ ensures that the prediction of the position of a detection from its neighbors is consistent with both location and time-stamps of the neighbors,

1. Effect of choosing different δ is discussed in the supplementary material.

assuming that the targets move at constant velocity in a small temporal neighborhood. We use $\gamma = 3$ pixels/frame. Our experiments (Figure 5) show that this choice has little impact on the performance.

The neighbors $\mathcal{N}_i$ are defined to be the samples whose time indices fall within a small temporal window of size T without falling under the gating constraint defined below to build the exclusion graph. T should be large enough to bridge local missed detections, but also small enough so that linear motion assumption holds. We use $T = 10$ frames.

Appearance graph. In case of the appearance graph, x_i corresponds to an appearance feature (*e.g.*, color histograms, etc.). Since we are considering the fact that a feature might occur only sporadically, $\mathcal{N}_i$ is defined to constitute all the samples except the samples that co-occur with the i -th sample and that do not have appearance features.

Exclusion graph. This graph captures the constraints associated to the fact that some detections cannot share the same labels. For example, two detections that occur at the same time instant should have different labels. This is usually referred to as *time exclusivity*. This information is encoded by setting $W_{ij} = 1$ if $t_i = t_j$. In addition, we can enforce the spatial constraint such that a target cannot have a large spatial displacement over short time interval. We encode this *gating* constraint by setting $W_{ij} = 1$ if $\|c_i - c_j\|_2 > v_{\max}|t_i - t_j|$, where $v_{\max}$ is the maximum speed of the target. Thus, $\mathcal{N}_i$ comprises of the detections that either co-exist with the i -th detection or violate the gating constraint.

2.3 Multi-object tracking as consistent labeling problem

Given a set $\mathcal{V}$ of n vertices (*i.e.*, the detections or the *tracklets* in tracking scenario), we consider that a label assignment $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)^\top$ is defined to assign a m -dimensional² label distribution $\mathbf{y}_i \in \Delta_m$ to the i -th node, where Δ_m is the m -dimensional probability simplex. Each dimension of the label distribution $\mathbf{y}_i$ corresponds to a target. Formally, the k -th dimension, $\mathbf{y}_i(k)$, $k = 1, \dots, m$, can be interpreted as the probability of the node i being the k -th target. Consequently, $\mathbf{Y}$ is a row-stochastic matrix, with each row summing to unity. Therefore, we write $\mathbf{Y} \in \mathcal{P}_{nm}$, where $\mathcal{P}_{nm}$ is the set of all row-stochastic matrices of size $n \times m$. We consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ as explained earlier. This graph is assumed to assign large positive weights to edges that connect vertices that are likely to have similar labels (typically because they are close in time and space, or because they have similar appearance). In [23], a harmonic function approach is introduced to measure the inconsistency

2. Ideally, m should be equal to the number of targets plus one (for false positive). Since, in general, we do not know the number of targets *a priori*, we set $m = n$, considering the worst case in which each detection corresponds to a different target.

of the label assignment matrix $\mathbf{Y}$ with respect to the graph $\mathcal{G}$. Specifically, it measures the ℓ_2 -norm of the difference between the labels assigned to nodes that are connected in the graph $\mathcal{G}$, and the labeling energy, also known as the harmonic energy [23], is defined as

$$E_L(\mathbf{Y}) := \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 = \text{Tr}(\mathbf{Y}^\top \mathbf{L} \mathbf{Y}), \quad (3)$$

where Tr is the trace of a matrix and $\mathbf{L}$ is the graph Laplacian, defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix whose i -th diagonal element is $D_{ii} := \sum_{j \in \mathcal{N}_i} W_{ij}$. Due to the definition of weights in our graphs, we have $D_{ii} = 1$. Therefore, $\mathbf{D}$ is an identity matrix. For a graph with non-negative weights, *i.e.*, $W_{ij} \geq 0$, $\mathbf{L}$ is positive semi-definite and consequently the labeling energy in Equation (3) is convex in $\mathbf{Y}$.

In our framework, we have $K + 2$ distinct graphs. As all the graphs have the same set of nodes, we frequently refer to a graph by its Laplacian $\mathbf{L}$ in the sequel. We represent the exclusion graph by $\mathbf{L}^{(-)}$, and other graphs by $\mathbf{L}_l^{(+)}$, $l \in \{0, \dots, K\}$, where $l = 0$ corresponds to the spatio-temporal graph and $1 \leq l \leq K$ corresponds to the l -th appearance graph. We explicitly introduce the minus (respectively, plus) superscript to emphasize that we would like to maximize (respectively, minimize) the labeling energy on the corresponding graph.

Given the measure of labeling energy on each graph, we want to define a label assignment $\mathbf{Y}^*$ that minimizes the labeling energies due to $\mathbf{L}_l^{(+)}$ and maximizes the labeling energy due to $\mathbf{L}^{(-)}$. Mathematically, we have

$$\begin{aligned} \mathbf{Y}^* &:= \underset{\mathbf{Y} \in \mathcal{P}_{nm}}{\text{argmin}} \sum_{l=0}^K \alpha_l E_{\mathbf{L}_l^{(+)}}(\mathbf{Y}) - E_{\mathbf{L}^{(-)}}(\mathbf{Y}) \\ &= \underset{\mathbf{Y} \in \mathcal{P}_{nm}}{\text{argmin}} E_{\mathbf{L}_{\text{eff}}^{(+)}}(\mathbf{Y}) - E_{\mathbf{L}^{(-)}}(\mathbf{Y}) \end{aligned} \quad (4)$$

where $\mathbf{L}_{\text{eff}}^{(+)} := \sum_{l=0}^K \alpha_l \mathbf{L}_l^{(+)}$, and $\alpha_l \geq 0$ weighs the contribution of labeling energy due to l -th graph. The choice of α_l depends on the scenario at hand, *i.e.*, on the prior knowledge available about the relevance of the features. For example, while tracking sport players, the decrease in labeling energy associated to the color graph is not of primary importance since the players from the same team have similar colors. Hence, detections sharing the same color might correspond to distinct players/labels. In such case, it is meaningful to lower the weight assigned to the color graph as compared to the spatio-temporal graph. In other cases, for which a unique specific color is assigned to each target, a large weight should be assigned to the color graph to force the assignment of distinct labels to detections having different colors. Since $\alpha_l \geq 0$ and $\mathbf{L}_l^{(+)}$ is positive semi-definite, $\mathbf{L}_{\text{eff}}^{(+)}$ is also positive semi-definite. Given $\mathbf{Y}^*$, the i -th node

is assigned the label that corresponds to the largest entry in $\mathbf{y}_i^*$.

3 GRAPH-CONSISTENT LABELS COMPUTATION

In this section, we explain how to compute the solution $\mathbf{Y}^*$ of the problem, defined in Equation (4). First, we present a global label assignment solution, based on the difference of convex programming. Afterwards, we introduce a node-wise optimization approach to solve the problem efficiently.

3.1 Joint label assignment optimization

Let us rewrite Equation (4) as

$$\begin{aligned} \mathbf{Y}^* &= \underset{\mathbf{Y} \in \mathcal{P}}{\text{argmin}} \text{Tr}(\mathbf{Y}^\top \mathbf{L}_{\text{eff}}^{(+)} \mathbf{Y}) - \text{Tr}(\mathbf{Y}^\top \mathbf{L}^{(-)} \mathbf{Y}) \\ &:= \underset{\mathbf{Y} \in \mathcal{P}}{\text{argmin}} [g(\mathbf{Y}) := f(\mathbf{Y}) - h(\mathbf{Y})] \end{aligned} \quad (5)$$

As $\mathbf{L}_{\text{eff}}^{(+)}$ and $\mathbf{L}^{(-)}$ are positive semi-definite matrices, both $f(\mathbf{Y}) := \text{Tr}(\mathbf{Y}^\top \mathbf{L}_{\text{eff}}^{(+)} \mathbf{Y})$ and $h(\mathbf{Y}) := \text{Tr}(\mathbf{Y}^\top \mathbf{L}^{(-)} \mathbf{Y})$ are convex in $\mathbf{Y}$, whereas $g(\mathbf{Y})$ is non-convex. Specifically, Equation (5) belongs to a family of *difference of convex* (DC) problems, and an iterative majorization-minimization algorithm can be used to solve the problem [21], as presented in Algorithm 1. Starting with a random label distribution $\mathbf{Y}^{(1)} \in \mathcal{P}$, the algorithm iteratively linearizes $h(\mathbf{Y})$ around the k -th iterate $\mathbf{Y}^{(k)}$ and solves the resulting convex function $f(\mathbf{Y}) - \nabla h^\top(\mathbf{Y}^{(k)}) \mathbf{Y}$ using the projected gradient method [35]. The number of iterations T_{joint} depends on the convergence tolerance.

Algorithm 1 Joint label assignment optimization

Input

Graph Laplacians: $\{\mathbf{L}_l^{(+)}, l = 0, \dots, K\}, \mathbf{L}^{(-)}$
 Scaling weights: $\{\alpha_l, l = 0, \dots, K\}$
 Number of iterations: T_{joint}

Output

Label assignment matrix: $\mathbf{Y}^*$

Procedure:

Choose an initial solution $\mathbf{Y}^{(1)} \in \mathcal{P}_{nm}$ randomly.

For $k = 1, \dots, T_{\text{joint}}$

 Compute $\nabla h(\mathbf{Y}^{(k)})$, gradient of $h(\mathbf{Y})$ at $\mathbf{Y}^{(k)}$.

 Solve the convex optimization problem

$$\mathbf{Y}^{(k+1)} \leftarrow \underset{\mathbf{Y} \in \mathcal{P}_{nm}}{\text{argmin}} [f(\mathbf{Y}) - \nabla h^\top(\mathbf{Y}^{(k)}) \mathbf{Y}]$$

 by the projected gradient method [35].

End For

Return $\mathbf{Y}^* \leftarrow \mathbf{Y}^{(T_{\text{joint}})}$.

It is worth noting that the gradient of $\text{Tr}(\mathbf{Y}^\top \mathbf{L} \mathbf{Y})$ is $(\mathbf{L} + \mathbf{L}^\top) \mathbf{Y}$. Therefore, both $\mathbf{L}$ and its transpose $\mathbf{L}^\top$ are considered identically during gradient descent.

Complexity analysis: Since there are n nodes, the graph Laplacian is a $n \times n$ matrix. Each node is assigned to a m -dimensional label distribution. Consequently, $\mathbf{Y}$ is a $n \times m$ matrix. The projected gradient method [35] performs gradient descent step followed by projection step for T_p times. Each step has a naive

complexity of $\mathcal{O}(n^2m)$, which can be improved to $\mathcal{O}(kmn)$ if the graph Laplacian is k -sparse. Thus, the overall complexity is $\mathcal{O}(n^2mT_pT_{\text{joint}})$. The parameters T_p and T_{joint} depend on a fixed tolerance value.

The main disadvantage of the above solution is that its computational complexity grows quadratically with the number of nodes. Therefore, it cannot scale to large graphs. Furthermore, it can only handle off-line tracking problems because the optimization problem formulation is based on the whole graph.

In the sequels, we describe how to circumvent these limitations based on a node-wise decomposition.

3.2 Node-wise label assignment optimization

To address the complexity issue of the joint label propagation algorithm, we adopt a node-wise decomposition of the objective function. That is, instead of solving a “big” and “global” optimization problem, each node updates locally and sequentially its label distribution to decrease the global objective. The approach is similar to the Gauss-Seidel iteration (or, co-ordinate descent approach). The advantages of such decomposition are twofold. First, the computational complexity gets significantly reduced, making the framework applicable to large graphs, potentially based on parallel implementation. Second, as we solve the problem by iterating over the nodes, it becomes possible to handle tracking problems for which the graphs grow incrementally, as new detections are gradually computed along the time.

In the remainder of the section, we first explain our proposed efficient and node-wise label propagation solution, and derive the conditions under which the global objective function monotonically decreases. Afterwards, we introduce a strategy to scale up the algorithm using parallel implementation.

3.2.1 Node-wise decomposition

In this section, we first generalize the energy in Equation (3) by replacing the $\frac{1}{2}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2$ term by a convex and symmetric function $\phi(\mathbf{y}_i, \mathbf{y}_j)$. Afterwards, we decompose the global optimization problem in Equation (5) into a node-wise optimization problem such that the high dimensional optimization problem is turned into a sequence of small problems in each node. In doing so, we derive the class of ϕ functions that guarantees monotonic decrease of the objective function.³

Formally, replacing the ℓ_2 -norm by ϕ in Equation (3), we write the objective function in Equation (5) as

$$\begin{aligned} g(\mathbf{Y}) &= \sum_{i=1}^n \sum_{j=1}^n \left[\sum_{l=0}^K \alpha_l W_{ij}^{(l)} - W_{ij}^{(-)} \right] \phi(\mathbf{y}_i, \mathbf{y}_j) \\ &\equiv \sum_{i=1}^n \sum_{j=1}^n W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j), \end{aligned} \quad (6)$$

where we define $W_{ij}^{(\text{eff})} := \sum_{l=0}^K \alpha_l W_{ij}^{(l)} - W_{ij}^{(-)}$. Denoting $\tilde{A}_{ij} := A_{ij} + A_{ji}$, we then isolate the contribution of the p -th node as

$$\begin{aligned} g(\mathbf{Y}) &= \sum_j W_{pj}^{(\text{eff})} \phi(\mathbf{y}_p, \mathbf{y}_j) + \sum_{i \neq p} \sum_j W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) \\ &= \sum_j \tilde{W}_{pj}^{(\text{eff})} \phi(\mathbf{y}_p, \mathbf{y}_j) + \sum_{i \neq p} \sum_{j \neq p} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) \end{aligned} \quad (7)$$

$$= g_p(\mathbf{y}_1, \dots, \mathbf{y}_n) + \sum_{i \neq p} \sum_{j \neq p} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) \quad (8)$$

where we assume $\phi(\mathbf{y}_i, \mathbf{y}_i) = 0$ and $\phi(\mathbf{y}_i, \mathbf{y}_j) = \phi(\mathbf{y}_j, \mathbf{y}_i)$ in Equation (7), and we introduce $g_p(\mathbf{y}_1, \dots, \mathbf{y}_n) := \sum_j \tilde{W}_{pj}^{(\text{eff})} \phi(\mathbf{y}_p, \mathbf{y}_j)$ for brevity in Equation (8).

Given $\mathbf{Y}^{(k)} = (\mathbf{y}_1^{(k)}, \dots, \mathbf{y}_n^{(k)})^\top \in \mathcal{P}_{nm}$, we choose an index $p \in \{1, \dots, n\}$ and compute a new iterate $\mathbf{Y}^{(k+1)} = (\mathbf{y}_1^{(k+1)}, \dots, \mathbf{y}_n^{(k+1)})^\top \in \mathcal{P}_{nm}$ that satisfies

$$\mathbf{y}_i^{(k+1)} \begin{cases} = \mathbf{y}_i^{(k)} & \text{if } i \neq p, \\ \in \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} g_p(\mathbf{y}_1^{(k)}, \dots, \mathbf{y}, \dots, \mathbf{y}_n^{(k)}) & \text{if } i = p. \end{cases} \quad (9)$$

Then, by construction,

$$\begin{aligned} g(\mathbf{Y}^{(k+1)}) &= g_p(\mathbf{Y}^{(k+1)}) + \sum_{i \neq p} \sum_{j \neq p} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i^{(k)}, \mathbf{y}_j^{(k)}) \\ &\leq g_p(\mathbf{Y}^{(k)}) + \sum_{i \neq p} \sum_{j \neq p} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i^{(k)}, \mathbf{y}_j^{(k)}) \\ &= g(\mathbf{Y}^{(k)}) \end{aligned}$$

Therefore, we conclude that under the following assumptions:

- the loss function $\phi(\cdot, \cdot)$ is convex,
- the loss function is *coincident*⁴, i.e., $\phi(\mathbf{y}_i, \mathbf{y}_i) = 0$,
- and the loss function is *symmetric* with respect to its arguments, i.e., $\phi(\mathbf{y}_i, \mathbf{y}_j) = \phi(\mathbf{y}_j, \mathbf{y}_i)$,

the optimization step at any fixed node p

$$\begin{aligned} \mathbf{y}_p^{(k+1)} &\in \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} g_p(\mathbf{y}_1^{(k)}, \dots, \mathbf{y}, \dots, \mathbf{y}_n^{(k)}) \\ &= \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} \sum_j \tilde{W}_{pj}^{(\text{eff})} \phi(\mathbf{y}, \mathbf{y}_j^{(k)}) \end{aligned} \quad (10)$$

monotonically decreases the objective function $g(\mathbf{Y})$. Equation (10) is still a DC problem and it can be solved by using *majorization-minimization* technique, as discussed in Section 3.1. It has to be noted that when ϕ is chosen to be the ℓ_2 -norm, the above conditions are satisfied.

The *label propagation* process is finally achieved by sequentially updating the label distribution over the nodes, possibly $T_{\text{con}} > 1$ times, until $g(\mathbf{Y}^{(k)})$ does not decrease any more. We summarize the overall process in Algorithm 2. Note that we do not assume anything

3. Detailed derivation is provided in the supplementary material.

4. The coincidence property will make the loops irrelevant and generally we do not need loops in the graph.

about the structure of the graph, thereby allowing loops in the graph.

Algorithm 2 Node-wise label assignment algorithm

Input

Weight matrices: $\{\mathbf{W}^{(l)}, l = 0, \dots, K\}, \mathbf{W}^{(-)}$
 Scaling weights: $\{\alpha_l, l = 0, \dots, K\}$
 Number of iterations: T_{con}

Output

Label assignment matrix: $\mathbf{Y}^*$

Procedure

Set $\mathbf{W}^{(\text{eff})} \leftarrow \sum_l \alpha_l \mathbf{W}^{(l)} - \mathbf{W}^{(-)}$

Set $\widetilde{\mathbf{W}}^{(\text{eff})} \leftarrow \mathbf{W}^{(\text{eff})} + \mathbf{W}^{(\text{eff})\top}$

Choose initial solution, $\mathbf{Y}^{(1)} \in \mathcal{P}_{nm}$

Set $k \leftarrow 1$

For $t = 1, \dots, T_{\text{con}}$

 Initialize $\mathcal{U} \leftarrow \mathcal{V}$

While $\mathcal{U} \neq \emptyset$

 Select a node p from $\mathcal{U}$

 Solve $\tilde{\mathbf{y}} \leftarrow \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} \sum_j \widetilde{W}_{pj}^{(\text{eff})} \phi(\mathbf{y}, \mathbf{y}_j^{(k)})$

$\mathbf{Y}^{(k+1)} \leftarrow (\mathbf{y}_1^{(k)}, \dots, \mathbf{y}_{p-1}^{(k)}, \tilde{\mathbf{y}}, \mathbf{y}_{p+1}^{(k)}, \dots, \mathbf{y}_n^{(k)})^\top$

$\mathcal{U} \leftarrow \mathcal{U} \setminus \{p\}$

$k \leftarrow k + 1$

End While

End For

Return $\mathbf{Y}^* \leftarrow \mathbf{Y}^{(T_{\text{con}})}$

Note: we have observed that the order in which p is chosen from $\mathcal{U}$ does not affect the labeling energy much. Consequently, we chose nodes in the sequential order.

Complexity analysis: Each node solves a m -dimensional DC program using the projected gradient method. Let the number of iterations required for the convergence of the projected gradient method be $T_{p'}$, which is comparable to T_p in Section 3.1. The complexity of the DC optimization in a specific node is therefore $\mathcal{O}(mT_{p'})$. Since there are n nodes and since we traverse the nodes T_{con} times, the overall complexity is $\mathcal{O}(mnT_{p'}T_{\text{con}})$. From experiments, we have seen that $T_{\text{con}} \ll T_{\text{joint}}$. Comparing with the complexity of joint approach, which is $\mathcal{O}(n^2mT_pT_{\text{joint}})$, the node-wise decomposition approach has an improvement of $\mathcal{O}(nT_{\text{joint}}/T_{\text{con}})$, which becomes significant as n increases, making it a better choice for large-scale problems as confirmed by our experiments.

3.2.2 Parallel implementation

The node-wise decomposition of the objective function also paves the way for a parallel implementation of the label optimization. This allows our proposed approach to scale up further with the size of the graph. In this section, we first derive a condition under which the parallelization of the coordinate descent decreases the objective function.

We denote the set of nodes for parallel descent by $\mathcal{J}$ and its complement by $\bar{\mathcal{J}} := \mathcal{V} \setminus \mathcal{J}$. Then, we

decompose the objective function as ⁵

$$g(\mathbf{Y}) = \sum_{i \in \mathcal{J}} g_i(\mathbf{Y}) - \sum_{i \in \mathcal{J}} \sum_{j \in \bar{\mathcal{J}}} \widetilde{W}_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) \quad (11)$$

The negative terms in Equation (11) are called *interference* terms. To nullify these terms, we pick up the nodes in $\mathcal{J}$ such that there are no edges between them, i.e., $\forall (i, j) \in \mathcal{J} \times \mathcal{J}, \widetilde{W}_{ij}^{(\text{eff})} = 0$. Under this condition, we can write

$$g(\mathbf{Y}) = \sum_{i \in \mathcal{J}} g_i(\mathbf{Y}) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} W_{ij}^{(\text{eff})} \phi(\mathbf{y}_i, \mathbf{y}_j) \quad (12)$$

and solve the local optimization problem

$$\mathbf{y}_i^{(k+1)} \in \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} g_i(\mathbf{y}_1^{(k)}, \dots, \mathbf{y}, \dots, \mathbf{y}_n^{(k)}) \quad (13)$$

in parallel for each node $i \in \mathcal{J}$. Then, the resulting label assignment matrix $\mathbf{Y}^{(k+1)}$, defined as

$$\mathbf{y}_i^{(k+1)} \begin{cases} \in \underset{\mathbf{y} \in \Delta_m}{\text{argmin}} g_i(\mathbf{y}_1^{(k)}, \dots, \mathbf{y}, \dots, \mathbf{y}_n^{(k)}) & \text{if } i \in \mathcal{J}, \\ = \mathbf{y}_i^{(k)} & \text{otherwise,} \end{cases}$$

decreases monotonically the objective function, i.e., $g(\mathbf{Y}^{(k+1)}) \leq g(\mathbf{Y}^{(k)})$. As a consequence, as long as the nodes that are processed in parallel are not neighbors, a monotonic decrement of the objective function is guaranteed. In Section 6.4, we demonstrate the benefit of parallelization with a simple yet effective batch-based scheduling approach.

4 FROM OFF-LINE TO INCREMENTAL LABEL PROPAGATION

In previous sections, we described the off-line graph construction and label propagation steps. However, in many real-life applications, detections arrive progressively along the time. To handle such scenarios, while being as close as possible to the off-line formalism, we embed the node-wise label propagation within an incremental graph construction process. Once the novel detections arrive, the graph is incremented by incorporating them. Afterwards, we re-optimize the label distribution by iterating over the nodes using the node-wise decomposition.

In the incremental graph construction, we do not have access to the future samples. Consequently, the LLE-based graph construction of Section 2.2 cannot be used. This has two implications. First, we need to define an explicit strategy to gradually incorporate new targets in the scene. Second, the implicit linear motion model cannot be embedded while constructing the spatio-temporal graph since future detection locations are not known at construction time.

The remainder of the section first explains how new detections are connected to the existing nodes.

5. Detailed derivation is provided in the supplementary material.

It then describes how labels are propagated through the incremented graph.

4.1 Incremental Graph Construction

We assume that the detections arrive sequentially along the time. Let the set of detections at time t be denoted by $\mathcal{D}^{(t)}$. Also, let the graph up to time $t - 1$ be $\mathcal{G}^{(t-1)} = (\mathcal{V}^{(t-1)}, \mathcal{E}^{(t-1)}, \mathbf{W}^{(t-1)})$. Since the graph evolves with time, we denote the number of nodes and the size of the label vector explicitly by $n^{(t)}$ and $m^{(t)}$ respectively.

The incrementation differs depending on whether we are dealing with the spatio-temporal graph, the appearance graph(s) or the exclusion graph. In all graphs, the new detections are first added to the set of vertices $\mathcal{V}^{(t-1)}$ to generate $\mathcal{V}^{(t)}$. Edges and weights are incremented separately for each graph as follows:

Exclusion graph. We create new edges between the nodes that occur at time t . Also, we create edges from the nodes at time t to the existing previous nodes if they are not within the *gating region*. Each exclusion edge has a weight 1.

Spatio-temporal and appearance graphs. We connect each node at time t with the nodes in a window $[t - T_c, t]$, where T_c is the connection window size. Large T_c results in dense graphs whereas small T_c results in sparse graphs. Once the neighborhood is defined, we assign a weight W_{ij} between a novel node i and an existing node j as

$$W_{ij} = \begin{cases} \exp(-\frac{1}{\sigma^2} d(\mathbf{x}_i, \mathbf{x}_j)^2) & \text{if } |t_i - t_j| \leq T_c, \\ 0 & \text{otherwise,} \end{cases} \quad (14)$$

where t_i and $\mathbf{x}_i$ denote the time instant and the features of the i -th node respectively, $d(\cdot, \cdot)$ measures the dissimilarity between the features $\mathbf{x}_i$ and $\mathbf{x}_j$, and σ is a scaling parameter. T_c and σ parameters are adapted to each kind of graph. In our experiments, T_c is set to 10 frames for the spatio-temporal graph (as in off-line graph construction), but is extended up to 200 frames in the appearance graph to bridge the gaps caused by the sporadic nature of the feature. The parameter σ should be larger than the typical distance measured between the features of two detections corresponding to the same targets, while being smaller than the typical distance measured between distinct targets. In practice, our values for σ have been selected by looking at the two distributions of distances between pairs of detections that correspond to the same/different targets.⁶ Specifically, we use $\sigma = 20$ in the spatio-temporal graph and $\sigma = 0.05$ in the appearance graph. Also, we use $d(\cdot, \cdot) := \|\cdot - \cdot\|_2$ but any other distance measure can be envisioned.

6. These distributions should ideally be derived from ground-truth data. When such a ground-truth is not available, we shown in the supplementary material that reasonable σ can simply be inferred by comparing two distributions of distances measured between either neighboring or co-existing detections. Another alternative consists in building on reliable tracklets to identify pairs of detection corresponding to similar/distinct targets.

To account for the cases in which some detections (nodes) are likely to correspond to new targets, we introduce a virtual source node in the graph. This source node is connected to every node in the spatio-temporal graph. The weight of the edge connecting the source node to the i -th node is represented by $w_i^{(s)}$. This weight depends on the prior knowledge we might have about where and/or when a target is likely to appear in the field of view. In our case, we consider that a new target appears either in the beginning of the tracking process, or when entering the scene on the borders of the image. Therefore, the weights should be large for the detections that are close to the image border and/or that appear in the beginning of the tracking. For the i -th detection, we compute the smallest distance $d_i^{(\min)}$ from the detection to the image border. Then, we compute $w_i^{(s)}$ by replacing $d(\cdot, \cdot)$ by $d_i^{(\min)}$ in Equation (14). Note that when some prior knowledge is available about the appearance of the targets entering the scene, e.g., because the digit of the players sitting on the dug-out in team sport games is known, edges to the source node could be defined in the appearance graph as well. Once the weights are defined, they are normalized such that $\sum_{j \in \{\mathcal{N}_i \cup s\}} W_{ij} = 1$.

4.2 Label propagation in the incremented graph

After incrementing the graphs, we perform node-wise label propagation. We denote the labels distribution over $\mathcal{G}^{(t)}$ after k iterations of the label propagation process by $\mathbf{Y}^{(t,k)}$. Moreover, $\mathbf{Y}^{(t)}$ denotes the labels distribution after the convergence of the propagation process at time t . We first initialize the label distribution matrix at time t , denoted by $\mathbf{Y}^{(t,1)}$, by augmenting the label distribution matrix at time $t - 1$, denoted by a $n^{(t-1)} \times m^{(t-1)}$ -dimensional matrix $\mathbf{Y}^{(t-1)}$, as follows:

$$\mathbf{Y}^{(t,1)} = \begin{pmatrix} \mathbf{Y}^{(t-1)} & \mathbf{0}_{n^{(t-1)} \times |\mathcal{D}^{(t)}|} \\ \mathbf{U}^{(t)} & \end{pmatrix} \quad (15)$$

where $\mathbf{0}_{n^{(t-1)} \times |\mathcal{D}^{(t)}|}$ is a $n^{(t-1)} \times |\mathcal{D}^{(t)}|$ -dimensional zero matrix and $\mathbf{U}^{(t)}$ is a $|\mathcal{D}^{(t)}| \times (|\mathcal{D}^{(t)}| + m^{(t-1)})$ -dimensional matrix such that $U_{ij}^{(t)} = 1/(|\mathcal{D}^{(t)}| + m^{(t-1)})$. Obviously, $\mathbf{U}^{(t)}$ is a (uniform) row-stochastic matrix, and a uniform label distribution is assigned to the novel nodes.

After initialization, we iterate over all the nodes (except the virtual source node) and solve the node-wise optimization problem, introduced in Section 3.2,

$$\mathbf{y}_i^{(t,k+1)} \in \underset{\mathbf{y} \in \Delta_{m^{(t)}}}{\operatorname{argmin}} g_i(\mathbf{y}_1^{(t,k)}, \dots, \mathbf{y}, \dots, \mathbf{y}_{m^{(t)}}^{(t,k)}) + w_i^{(s)} \phi(\mathbf{y}, \mathbf{e}_i), \quad (16)$$

where $\mathbf{e}_i \in \Delta_{m^{(t)}}$ is a singleton vector having 1 at the i -th index and zero elsewhere. It promotes the assignment of a new label to the i -th node when $w_i^{(s)} \approx 1$.

To bound the complexity of our incremental framework, and to turn it into an on-line procedure, we consider a sliding window $[t - T_o, t]$ and forget the history of the graph outside the window. Afterwards, the distributions of nodes that lie outside the window are frozen, and the node-wise optimization, defined in Equation (16), is only considered for the nodes that belong to the window. The window size T_o trades-off the tracking accuracy and the computational (and memory) resources.

5 RELATED WORK

This section provides a brief review of the recent and related works under the following categories:

Label propagation in graphs. Propagation of labels in a graph is often used in semi-supervised learning approaches, and a concise survey of recent developments in this field can be found in [25] and references therein. In short, most of these approaches assume that the label of a node is approximated as the linear combination of the labels of its neighbours [30]. In [29], the authors use a mixed label propagation in which (i) they measure the bipolar similarity (*e.g.*, Karl Pearson’s correlation coefficient that lies in the range $[-1,1]$) between the samples, and (ii) construct a ‘positive’ and a ‘negative’ graphs based on the sign of the coefficient. Afterwards, they minimize the ratio between labeling energies due to the positive and negative graphs. This is done by semi-definite relaxation to assign a binary label to each node of the graph. Our method differs from [29] both in the definition of the graph similarities, and the label propagation method. Specifically, since we use multi-class labels instead of binary labels, and impose that the label distribution at each node should lie on a probability simplex, our problem is difficult to cast into their formalism.

Message passing. Message passing (belief propagation) approaches have been used to label the nodes in a graph in tracking/recognition [6], [31], image completion scenario [16], etc. Each node gathers messages from its neighbors, optimizes locally a problem, and then transmits its message. This approach has been shown to be exact in trees but the convergence is not guaranteed in presence of loops. In contrast, we do not assume any structure of the graph to guarantee the convergence of our approach.

In [31], a subset of the nodes are initially labelled and then a CRF is used to infer the label of the remaining nodes. For this, the authors compute various appearance features and assume that the features are always available with similar accuracies. Hence, their approach cannot exploit appearance features that are sporadic or affected by non-stationary noise. In [6], the authors utilized such non-stationary and sporadic features to prioritize the propagation of belief related to the label probability distribution. Even though this approach exploits sporadically available appearance features, it relies on the assumption that the target

appearances are known beforehand, which is not the case of our approach.

Mutual exclusion. Mutual exclusion has been considered in [32], [33] to learn discriminative appearance features. In these papers, first of all, a low-level but reliable tracker is used to connect unambiguous detections into tracklets. Afterwards, positive samples are defined by pairs of detections that belong to the same tracklet, while negative samples correspond to pairs that belong to tracklets that likely correspond to distinct objects (because they overlap in time). Lastly, these samples are used to train an AdaBoost [34], which in turn selects the discriminative appearances. This work is orthogonal to our proposal since it could help our approach to select the discriminative features while defining the appearance graph(s).

In [9], [10], the authors define a mutual exclusion term based on the physical distance between two detections that occur at the same time. The term goes to infinity as the distance goes to zero. This is motivated by the fact that two objects cannot occupy the same space simultaneously. Our formulation is different in that our mutual exclusion term is defined in terms of the similarity in the label distribution rather than the position.

Distributed proximal optimization: Our label propagation method by node-wise optimization cannot be truly characterized as a distributed computation but it raises this possibility for future developments. In such a scenario, we noticed that in [15], the authors devise a proximal optimization on graph that has quadratic convergence by using the Nesterov’s method [18]. Knowing if their approach, which assumes positive graph weights for forcing convex optimization, can be adapted to general weights and DC minimization is a matter of future study.

Laplacian eigenmaps latent variable model (LELVM): LELVM [14] defines an out-of-sample mapping of the Laplacian eigenmaps. Given a graph, in which the weight of an edge $x_i \sim x_j$ is constructed as $W_{ij} := \exp(-\|x_i - x_j\|_2^2/\sigma^2)$, the latent points Y are the solution of

$$\begin{aligned} & \text{minimize} && \text{Tr}(Y^\top LY) \\ & \text{subject to} && Y \in \mathbb{R}^{N \times L}, Y^\top DY = I, Y^\top D\mathbf{1} = \mathbf{0}. \end{aligned}$$

where D is a diagonal matrix with its i -th diagonal element defined as $D_{ii} := \sum_j W_{ij}$, and $L := D - W$ is the graph Laplacian. When a new sample x arrives, [14] defines an out-of-sample mapping $F(x) = y$ for a new point x as a semi-supervised learning problem, by recomputing the embedding as in previous equation (*i.e.*, augmenting the graph Laplacian with the new point), but keeping the old embedding fixed. LELVM has been used for tracking human pose in [17]. Our incremental label propagation is similar to LELVM in the sense that we also augment our graph and then solve for the “latent” label distribution. However, LELVM cannot handle newly occurring tar-

gets as it assumes that the new sample x belongs to one of the classes defined by $\mathbf{X}$. Moreover, it keeps the old “latent” distributions unchanged, which is not the case in our approach.

6 EVALUATION

The proposed algorithm has been evaluated on the following well-known and challenging datasets: APIDIS [1], PETS-2009 S2/L1 [2], TUD Stadtmitte [3] and TUD Crossing [4]. APIDIS is a multi-camera sequence acquired during a basketball game, whereas the other three are monocular sequences.

In the remainder of the section, we first describe these datasets. We then discuss the evaluation metrics and the implementation details. Finally, we present our results and compare them with several state-of-the-art methods.

6.1 Datasets

APIDIS dataset. This 1-minute video dataset is generated by 7 cameras, distributed around a basketball court. The candidate detections are computed independently at each time instant based on a ground occupancy map, as described in [27]. For each detection, the jersey color and its digit are computed to define the appearance features. In short, the jersey color is computed as the average blue component divided by the sum of average red and green components, over the foreground silhouette of the player within the detected rectangular box. The digit feature is obtained by running a digit-recognition algorithm [36] in the same rectangular region. The digit feature is inherently sporadic as it is available only when the digit on the jersey faces the camera.

Pedestrian datasets. To evaluate the performance of our method in monocular views, we use publicly available PETS-2009 S2/L1, TUD Stadtmitte and TUD Crossing datasets. The PETS dataset is 795-frames long, with moderate target density. However, the pedestrians wear similar dark clothes, which makes appearance comparison very challenging. TUD Stadtmitte and TUD Crossing are 179 and 201 frames long respectively but the targets frequently occlude each other because of the low view-point. Detection results and the ground-truth are obtained from [5]. Afterwards, 8-bin CIE-LAB color histograms are computed for each channel of each bounding box, resulting in a 24-bin appearance vector. We ignore the histogram(s) if the overlap ratio between any two bounding boxes exceeds 5%. This is done because the histograms are less likely to represent the target color correctly in case of overlap, and might thus lead to wrong associations between the detections. Since the histogram feature is not available for every detection any more, it becomes sporadic.

6.2 Evaluation metrics

We use CLEAR MOT metric [13] to evaluate our approach. It defines two quantities namely multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA).

MOTP is defined as the total error in estimated position for matched⁷ ground-truth and track pairs over all frames, averaged by the total number of matches. MOTA measures the number of misses, false positives, re-initializations and identity switches. A miss means that the tracker does not have a matching estimate for a ground-truth. Similarly, a tracker output is called a false positive when no matching ground truth is available. A switching error occurs when the tracker starts following another object, whereas a re-initialization error occurs when the tracker fails to track the object at same time and a new track is assigned for the same object later. The error due to switching is more problematic as it might lead to significant errors in higher level interpretation.

Usually, MOTA is often preferred over MOTP because MOTP depends on the accuracy of target detector and on the accuracy of the ground-truth annotations. In our table, due to its importance regarding long term tracking capabilities, the number of switching errors (SW) is also reported.

6.3 Implementation details

Both the joint and node-wise label propagation algorithms have been implemented on MATLAB running on a 2.4 GHz quad core CPU with 4 GB RAM⁸. The parallel implementation of the node-wise label propagation has been done separately in C++ using Boost Graph Library and OpenMP.

Pedestrian datasets. For these datasets, a node is assigned to each individual detection. The size of the temporal neighborhood in spatio-temporal graph is chosen to be 10 frames. Thus, $T = 10$. When processing time is an issue, we can envision processing the dataset in batches or running a low-level but reliable tracker first to reduce the complexity (which we perform in the APIDIS dataset).

APIDIS dataset. We first pre-process the data by aggregating some of the detections into tracklets based on a spatio-temporally local but reliable tracker. The local but reliable tracker associates two detections between successive frames into a tracklet when they are separated by less than 15 cm and there is no other detection that is closer than 15 cm from any of them. The resulting tracklets define the nodes in our graphs. The neighborhood of the spatio-temporal graph is defined to connect the tracklets within 100 frames on each side, which allows us to connect tracklets that

7. A tracker output and the ground-truth are defined to be matched if their intersection-over-union ratio exceeds 50% (respectively, if the distance < 30 cm for APIDIS. The threshold value of 30 cm is recommended for APIDIS dataset.).

8. A reference implementation is available at <http://sites.uclouvain.be/ispgroup/index.php/Main/HomePage>.

are up to 4 seconds apart. In the exclusion graph, the neighborhood of a node consists of all the nodes that overlap in time. Finally, the appearance features of a tracklet is inferred by averaging the appearance features of the detections along the tracklet.

Post processing. Once the label propagation step is over, we filter out some tracks that satisfy one of the following criteria:

- the number of detections along the track is less than 10 frames,
- the track is primarily composed of low confidence detections, *i.e.*, if the maximum confidence value along the track is less than 0.8.

The reasons behind these heuristics are that false tracks are usually shorter than regular target tracks and that the false positive detections have lower confidence values, compared to the true detections. This case is prevalent in PETS and both TUD datasets.

A glimpse of running times is presented below:

Dataset	Time taken			
	Low-level tracker	Graph construction	Label propagation	
TUD Stadtmitte	-	2 min	3 min	25 sec
TUD Crossing	-	155 sec	167 sec	31 sec
PETS	-	3 min	40 min	5 min
APIDIS	15 sec	1 min	5 min	1 min

6.4 Results

In this section, we first present the tracking results for our frameworks, applied to offline-constructed graphs. Then, we present the tracking results for the incremental graph construction and label propagation. The computational advantages due to the node-wise decomposition and parallelization are presented afterwards. Then, effects of parameters are discussed. Lastly, some qualitative results are presented.

6.4.1 Tracking results for offline-constructed graphs

To better compare with the literature, we consider two versions of the method. The first one uses only the spatio-temporal information. Thus, we construct only the spatio-temporal and the exclusion graphs. This is equivalent to setting $\alpha_0 = 1$ and $\alpha_p = 0, \forall p \neq 0$ in our algorithm. In contrast, the second one considers both the spatio-temporal and the appearance features. For the TUD Stadtmitte, TUD Crossing and PETS datasets, we use $\alpha_0 > \alpha_1$ (α_0 for the spatio-temporal graph and α_1 for the appearance graph). This constrains the spatio-temporal consistency more strictly than the appearance consistency. The reason is that the targets wear similar clothes and therefore have similar appearances in the datasets. In the experiments, we use $\alpha_0 = 1$ and $\alpha_1 = 0.5$.⁹

We compare our results with several methods such as the continuous energy (CE) minimization [9], the discrete-continuous (D-C) minimization [10], the GMCP tracker [22], the K -shortest paths (KSP) [12],

	Method	MOTA	MOTP	SW
TUD Stadtmitte	Continuous energy [9]	60.5	65.8	7
	Discrete-continuous (D-C) [10]	61.8	63.2	4
	GMCP tracker [22]	77.7	63.4	0
	Joint (no appearance)	62.7	73.5	17
	Joint (with appearance)	79.2	73.9	4
	Node-wise (no appearance)	63.1	73.6	16
TUD Crossing	Node-wise (with appearance)	79.5	73.9	4
	Discrete-continuous (D-C) [10]	57.3	73.7	13
	Continuous energy [9]	61.6	73.2	28
	GMCP tracker [22]	91.63	75.6	0
	Joint (no appearance)	62.5	74.3	12
	Joint (with appearance)	65.4	75.4	8
PETS	Node-wise (no appearance)	62.3	74.3	13
	Node-wise (with appearance)	65.4	75.2	8
	Discrete-continuous (D-C) [10]	89.30	56.40	-
	Continuous energy [9]	81.84	73.93	15
	K-shortest paths [12]	80.00	58.00	28
	GMCP tracker [22]	90.30	69.02	8
	Global appearance (GA) [20]	81.46	58.38	19
	Iterative hypothesis (IH) [8]	83.0	74.0	N/A
	Joint (no appearance)	82.77	71.21	25
	Joint (with appearance)	91.04	70.99	5
	Node-wise (no appearance)	83.07	71.23	25
	Node-wise (with appearance)	91.04	71.00	5

TABLE 2

Tracking results on the TUD Stadtmitte (179 frames), TUD Crossing (201 frames) and PETS 2009-S2/L1 (795 frames) datasets. The D-C, IH, GMCP, KSP and GA results are obtained from [8], [10], [20], [22].

the global appearance constraints (GA) [20] and the iterative hypothesis testing (IH) [8]. The CE and D-C trackers estimate the most probable trajectories by minimizing their energies that consist in a combination of observation energy, dynamic energy, mutual exclusion energy, track persistence energy, etc. In addition, the D-C tracker uses cubic splines for modeling the motion of the target, and favors the reduction of the number of trajectories. GMCP solves greedily a generalized minimum clique problem to extract tracklets that have the most stable appearance features and the most consistent motion. KSP solves a network-flow formulation of the tracking problem and minimizes the sum of pairwise association costs between consecutive detections to estimate K tracks. GA improves KSP by incorporating appearance information. IH embeds an hypothesis testing strategy into a greedy shortest-path computation procedure to exploit the appearance features that are unreliable and/or sporadically available. Since C-E, DC and KSP trackers do not use appearance information, we compare them with the first version of our approach that does not use appearance features. Similarly, since GA, IH and GMCP exploit the appearance features, we compare them to the second version of our approach.

In Table 2, we first observe that the joint and node-wise label optimization approaches give similar performances. For TUD Stadtmitte dataset, our method is better than previous methods both in terms of MOTP and MOTA. This is because our approach is able to connect the detections even if they are far in time, resulting in longer and consistent tracks. However, our method is slightly worse than GMCP in terms

9. We varied $\alpha_1 \in [0.1, 1]$ but did not observe significant performance changes.

of ID switches. This might be because GMCP uses motion information in a global manner to ensure a smooth displacement while connecting the tracklets, which is not the case in our formalism.

In case of TUD Crossing dataset, our method outperforms CE and D-C. Surprisingly, GMCP has reported outstanding results. GMCP does not describe how the detections have been obtained. Our methods use same detections than CE and D-C, which has been obtained from the MOTChallenge [4]. We have observed that removing the unreliable (confidence < 0.6) detections from MOT challenge already reduces the MOTA of an oracle tracker to around 0.7, while keeping all the detections introduces many false positives. Hence, even if we were unable to run the GMCP code to verify it, we suspect that the results reported by GMCP are based on a better set of detections than the MOTChallenge ones. It is explained in detail in the supplementary material.

In case of PETS dataset, again we observe that our proposed approach outperforms most approaches. When the appearance features are ignored, the MOTA metric is better than KSP but worse than D-C. This might be because of the fact that D-C exploits higher-order motion models, whereas our formalism does not. We assert the fact that a linear motion is implicit in our formalism to justify our superior performance against KSP and GA, which do not take the motion information into account. When the appearance information is incorporated, the performance is improved significantly from 82% to 91%. Moreover, the switching error is drastically reduced.

The results for the APIDIS dataset are presented in Table 3. Since GA and IH are the only methods from the literature that are able to exploit sporadic appearance features, we focus the comparison with them. As before, first we computed the results without using any appearances. This is done by setting $\alpha_0 = 1, \alpha_1 = 0, \alpha_2 = 0$, where the indices 0, 1 and 2 correspond to the spatio-temporal, the color and the digit graphs respectively. Afterwards, we use both the digit and the color features. As the color feature is less discriminant (because the players from the same team wear jersey of the same color) than the digit feature, we set $\alpha_1 < \alpha_2$. Empirically, we use $\alpha_0 = 1, \alpha_1 = 0.1, \alpha_2 = 0.5$.

Although our approach performs significantly better than GA, the results are slightly worse than IH. We see two potential reasons for this. First, our graph construction method assumes that the features are always reliable (whenever they are present). This is not the case for the IH that takes into account the confidence of feature measurement while connecting two nodes. Doing so, it lowers the impact of noisy appearance features as compared to the reliable ones. Second, IH associates two nodes only when the connection is sufficiently reliable than alternative connections. This prevents potential track switches, as reflected by the

Method	MOTA	MOTP	SW
IH (no appearance) [8]	85.83	60.83	18
IH (color+digit) [8]	86.19	60.90	12
GA (no appearance) [20]*	72.91	53.13	108
GA (color+digit) [20]*	73.07	53.15	110
Joint (no appearance)	81.27	57.13	49
Joint (color+digit)	83.90	60.04	45
Node-wise (no appearance)	81.4	57.17	49
Node-wise (color+digit)	83.85	60.01	45

TABLE 3

Results on the APIDIS dataset (1500 frames). The tracking results for IH and GA are been provided by the authors. [*] Since the detection results for [20] are different than that for the [8] and ours, we relax the distance threshold to 40 cm (from 30 cm) for the tracking results of [20]. Detailed results are provided in the supplementary material.

switching errors.

6.4.2 Tracking results for incrementally constructed graphs

We constructed the graph as described in Section 4.1 and performed incremental label propagation. The construction of the graph in case of APIDIS dataset is slightly different than the other two datasets. In this case, if new detections can be unambiguously matched to the existing nodes, they are aggregated into a single tracklet. Otherwise, we create new nodes for the detections and connect them with existing nodes. The tracking results are presented in Table 4. We observe that the tracking accuracy of the incremental approach is slightly worse than the off-line method. This reveals the importance of embedding a linear motion model during graph construction.

Dataset	Appearance feature	MOTA	MOTP	SW
PETS	No	79.32	70.70	26
	Yes	86.56	71.40	6
TUD Stadtmitte	No	61.60	73.30	13
	Yes	77.20	73.40	2
TUD Crossing	No	61.2	72.1	19
	Yes	63.4	72.3	12
APIDIS	No	74.40	54.20	52
	Yes (color+digit)	80.23	58.45	47

TABLE 4

Results of the incremental graph construction and label propagation approach.

To trade-off the complexity with the quality of the incremental solution, we considered only the nodes which lie within the observation window $[t - T_o, t]$ to perform label propagation. The rest of the nodes were frozen, meaning that the node-wise optimization was not performed on those nodes. The results are elucidated in Figure 2 for the TUD Stadtmitte dataset. As we can see, the processing time monotonically increases with the size of the observation window. However, the tracking accuracy is improved only upto some value (50 frames in Figure 2) after which it saturates. Alternatively, one could define other heuristic

to freeze the nodes. For example, one could decide to freeze a node if the change in its label distribution over time is smaller than some pre-defined threshold.

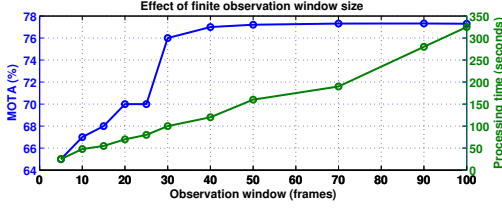


Fig. 2. **Trade-off between the processing time and the tracking accuracy** for different observation window size for the TUD Stadtmittte dataset.

6.4.3 Computational advantages of the node-wise decomposition and parallelization

To study the effect of node-wise decomposition, we constructed the graph off-line with different number of frames. Once the graph was constructed, we used both joint and node-wise approaches for label propagation with 10 random initializations. Afterwards, we computed the processing times for both approaches to reach the same labeling energy (equal to the labeling energy of the joint optimization after convergence). The results are shown in Figure 3. We can see the dramatic improvement in computational speed, especially when the size of the graph increases. We observed that one iteration (over the whole graph) of the node-wise label optimization appears to reduce the labeling energy much faster than one iteration of the joint optimization.

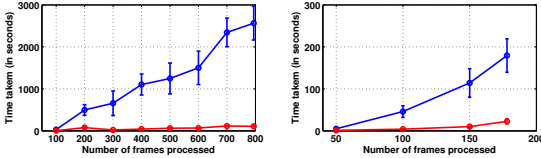


Fig. 3. **Processing times for the joint and the node-wise approaches for different size of the graph.**

To assess the advantages offered by the parallel implementation, we consider a simple scheduling strategy, which directly follows the non-interference condition (see Section 3.2.2) and selects the nodes at random. For each number of processor, we ran the algorithm 10 times and noted the evolution of objective function. The results are depicted in Figure 4. The reported time is different from Figure 3 because of the fact that the parallel implementation is done in C++. Although the parallel implementation decreases the computational time, we observe that the reduction is not proportional to the degree of parallelism. This sub-optimal speed-up factor is due to the fact that we run the algorithm in batches of nodes. As a consequence, the time required to process a batch is governed by the longest time taken by one of its nodes. The algorithm for node-selection strategy and

the distribution of time taken by nodes in the batch are presented in the supplementary material.

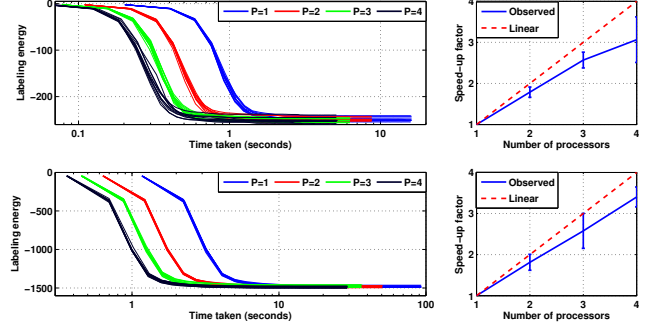


Fig. 4. **Processing time and speed-up factors** for different number of processors ($P = 1$ to $P = 4$) of the TUD Stadtmittte (**top row**) and PETS (**bottom row**) datasets. For each case, we perform 10 runs of the algorithm which are drawn with the same color.

6.4.4 Effect of parameters

Our algorithm has some key parameters. They are listed in Table 1. The effect of α_l and T_o have already been discussed in Section 6.4.1 and Section 6.4.2. In this section, we consider T , T_c , γ and $v_{\max}$ and discuss what are their effects on the performance. For this, only one parameter is changed at a time and all other parameters are fixed at their reference values. Figure 5 presents our results. In all graphs, the blue and green curves depicts the MOTA and the computational time respectively. In the first column, which considers the incremental algorithm, this computational time reflects both the graph construction and the label propagation, since they occur jointly all along the process. In the three last columns, which refer to the off-line algorithm, the green curve measures the graph construction time only.

From Figure 5, we observe that increasing T_c increases the computation time. However, the MOTA is improved only up to some value (100 frames in our experiments) after which it starts decreasing. This is mainly due to the fact that the chances of wrong associations increase with large T_c .

Since the parameters T , γ and $v_{\max}$ do not affect the construction of the appearance graph, we report the time taken for the spatio-temporal graph only. We observe that increasing T increases the connectivity of the graph (which leads to increased time to construct the graph). We observe that the MOTA increases up to certain value of T and then starts decreasing again. On the one hand, when T is small, it might not be effective to bridge the local missed detections. On the other hand, a large T is not only more prone to wrong connections but also might not satisfy the linear motion model assumption. Interestingly, γ does not seem to affect MOTA much. From Figure 5, we also observe that γ does not affect the graph construction time when a small window $T = 10$ is

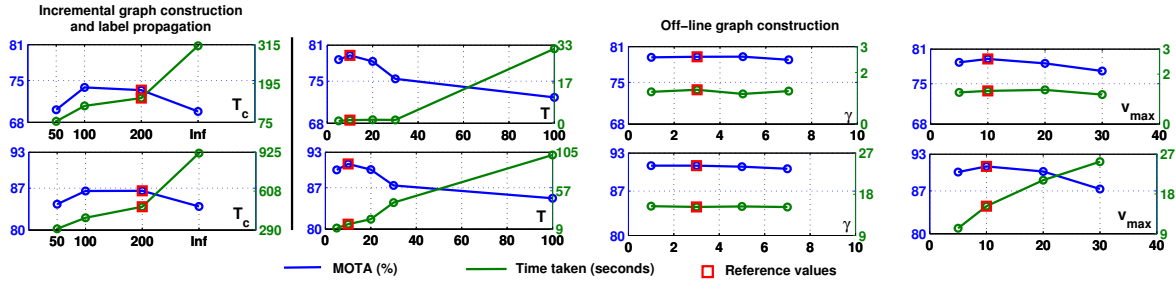


Fig. 5. **Effect of parameters on TUD Stadtmittte (top) and PETS (bottom) datasets.** Each plot shows the effect of changing a single parameter while keeping other parameters fixed. Red squares correspond to the reference parameter values used in our experiments. The parameter is mentioned on the bottom right corner.

considered. However, we have observed that its effect is significant when T increases. As an example, the graph construction time for $\gamma = 1$ is around 10 times more than that for $\gamma = 7$ when T is set to 100 frames for TUD Stadtmittte dataset.¹⁰ This is because a large γ reinforces the implicit linear motion assumption embedded in Equation (1), which in turn restricts the number of neighboring nodes that remain eligible for non-zero weights, leading to sharp reduction in the graph construction time. Finally, reducing $v_{\max}$ typically reduces the time to construct the graph as it discards many detections that violate the gating constraint from the neighborhood. On the flip side, these detections receive non-zero weights in the exclusion graph and they receive different labels, resulting in reduced MOTA when $v_{\max}$ becomes too small, *i.e.*, typically below the reference value of 10. When $v_{\max}$ increases beyond the reference point (in red), it increases the chances of wrong associations, resulting in lower MOTA.

6.4.5 Qualitative results

Now, we present some qualitative results. Figure 6 depicts the detections, constructed graphs and the inferred labels. Due to lack of space, we present the sample frames and discuss the failure cases in the supplementary material.

7 CONCLUSION AND FUTURE WORKS

In this paper, we have focused on the multi-object tracking (MOT) problem under sporadic appearance features. For this purpose, a number of complementary graphs have been constructed to capture the spatio-temporal and the appearance information. Afterwards, MOT has been formulated as a consistent labeling problem in the associated graphs. The proposed solution is based on *difference of convex* programming, for which we have provided both the joint as well as node-wise label optimization solutions. We show that node-wise label propagation allows us to scale up the algorithm with the number of nodes. Two further extensions of the proposed approach have been investigated. First, we have proposed a parallel

implementation of the node-wise label propagation. Second, the node-wise decomposition has been embedded in an incremental graph construction step.

Interesting paths to investigate in future research include the extensions of the framework to embed higher order motion models in the spatio-temporal graph construction, and to handle the range of features confidence levels in a continuous manner. This would be in contrast with our current approach, which turns the variable reliability of the features into sporadic measurements through hard thresholding.

REFERENCES

- [1] <http://sites.uclouvain.be/ispgroup/index.php/Softwares/APIDIS>.
- [2] <http://www.cvg.rdg.ac.uk/PETS2009/>.
- [3] <http://www.d2.mpi-inf.mpg.de/node/428>.
- [4] <http://motchallenge.net/>.
- [5] <http://www.milanton.de/data/>.
- [6] A. K. KC and C. De Vleeschouwer. "Prioritizing the propagation of identity beliefs for multi-object tracking." In *Proc. Brit. Mach. Vis. Conf.*, pp. 117.1–117.11, 2012.
- [7] A. K. KC and C. De Vleeschouwer. "Discriminative label propagation for multi-object tracking with sporadic appearance features." In *Proc. IEEE Conf. Comput. Vis.*, pp. 2000–2007, 2013.
- [8] A. K. KC, D. Delannay, L. Jacques, and C. De Vleeschouwer. "Iterative hypothesis testing for multi-object tracking with noisy/missing appearance features." In *Computer Vision-ACCV 2012 Workshops*, pp. 412–426, 2012.
- [9] A. Andriyenko and K. Schindler. "Multi-target tracking by continuous energy minimization." In *CVPR*, pp. 1265–1272, 2011.
- [10] A. Andriyenko, K. Schindler, and S. Roth. "Discrete-continuous optimization for multi-target tracking." In *CVPR*, pp. 1926–1933, 2012.
- [11] F. Fleuret, J. Berclaz, R. Lengagne and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. In *PAMI*, 30(2), 267–282, 2008
- [12] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. "Multiple object tracking using k-shortest paths optimization" In *PAMI*, 33(9): 1806–1819, 2011.
- [13] K. Bernardin and R. Stiefelhagen. "Evaluating multiple object tracking performance: the CLEAR MOT metrics." *Journal on Image and Video Processing*, Vol. 2008, Article no. 1, pp. 1:1–1:10, Feb. 2008.
- [14] M. Á. Carreira-perpiñán, and Z. Lu. "The Laplacian Eigenmaps Latent Variable Model." *AISTATS*, 2007.
- [15] A. I. Chen and A. Ozdaglar. "A fast distributed proximal-gradient method." In *Communication, Control, and Computing (Ann. Allerton Conf.)*, pp. 601–608. IEEE, 2012.
- [16] P. F. Felzenszwalb and D. P. Huttenlocher. "Efficient belief propagation for early vision." *IJCV*, 70(1):41–54, 2006.
- [17] Z. Lu, C. Sminchisescu, and M. Á. Carreira-perpiñán. "People tracking with the laplacian eigenmaps latent variable model." In *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1705–1712, 2007.

10. This observation is not reported in Figure 5.

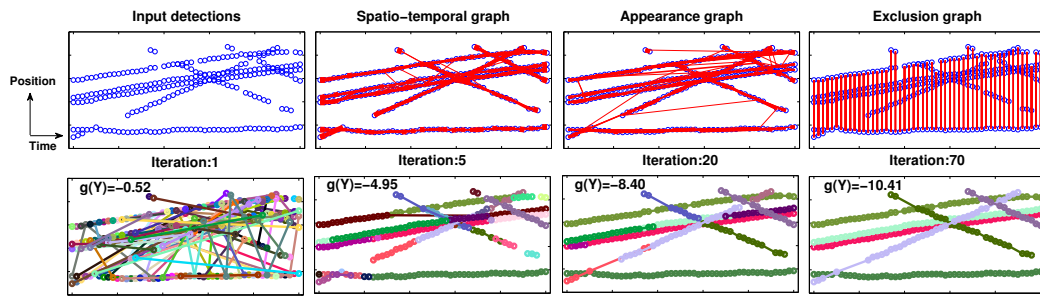


Fig. 6. **Sample graphs and label evolution on a subset of detections from PETS dataset.** Top row shows the input detections and the three constructed graphs. For clarity, edges that have weights smaller than 10^{-2} are suppressed. Bottom row depicts the evolution of label of the nodes along with the corresponding labeling energy.

- [18] Y. Nesterov. "Introductory Lectures on Convex Optimization. A Basic Course." Vol. 87. Springer, 2004.
- [19] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. "Globally-optimal greedy algorithms for tracking a variable number of objects." In *CVPR*, pp. 1201-1208, 2011.
- [20] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua. "Tracking multiple people under global appearance constraints." In *ICCV*, 2011.
- [21] B. K. Sriperumbudur and G. R. G. Lankriet. "On the convergence of the concave-convex procedure." In *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1759-1767, 2009.
- [22] A. R. Zamir, A. Dehghan, and M. Shah. "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs." In *Proc. 12th Eur. Conf. Comp. Vis.*, pp. 343-356, 2012.
- [23] X. Zhu, Z. Ghahramani, J. Lafferty, et al. "Semi-supervised learning using gaussian fields and harmonic functions." In *ICML*, volume 3, pp. 912-919, 2003.
- [24] W. Brendel, M. Amer, S. Todorovic. "Multiobject tracking as maximum weight independent set." In *CVPR*, 2011.
- [25] W. Lu, J. Wang, and S.-F. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9), September 2012.
- [26] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323-2326, December 2000.
- [27] D. Delannay, N. Danhier and C. De Vleeschouwer. Detection and Recognition of Sports(women) from Multiple Views. In *ICDSC*, Como, Italy, 2009.
- [28] S. Khan and M. Shah. Tracking multiple occluding people by localizing on multiple scene planes. In *PAMI*, 31.3 (2009): 505-519.
- [29] W. Tong and R. Jin. Semi-supervised learning by mixed label propagation. In *AAAI*, pages 651-656. AAAI Press, 2007.
- [30] F. Wang and C. Zhang. Label propagation through linear neighborhoods. In *ICML*, 2006.
- [31] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy. Learning to track and identify players from broadcast sports videos. *PAMI*, 35(7), pp. 1704-1716, 2013.
- [32] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by online learned discriminative appearance models. In *CVPR*, pp. 685-692, 2010.
- [33] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking. In *CVPR*, pp. 1217-1224, 2011.
- [34] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 1997.
- [35] P. H. Calamai and J. J. Moré. Projected gradient methods for linearly constrained problems. *Mathematical programming*, 39(1), pp. 93-116, 1987.
- [36] C. Verleysen, and C. De Vleeschouwer. Recognition of sport players numbers using fast color segmentation. In *SPIE-IS&T Electronic Imaging*, 2012.
- [37] H. Zou, and T. Hastie. Regularization and variable selection via the elastic net. In *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005): 301-320.
- [38] S. S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. In *SIAM journal on scientific computing* 20(1), pp. 33-61, 1998.
- [39] R. Tibshirani. Regression shrinkage and selection via the lasso. In *Journal of the Royal Statistical Society: Series B (Methodological)*, pp. 267-288, 1996.



Amit Kumar K.C. received double MS degree in Research on Information and Communication Technologies (MERIT) from Politecnico di Torino (PdT, Italy) and Université catholique de Louvain (UCL, Belgium) in 2010, and PhD in Applied Sciences from UCL in 2015. Since 2015, he has been working in University of California, San Diego (UCSD) as a post-doctoral researcher. His research interests include image segmentation, multi-object tracking, graph formalism and optimization theory. He is a student member of the IEEE.



Laurent Jacques received the BSc in Physics, the MSc and the PhD in Mathematical Physics from the Université catholique de Louvain (UCL), Belgium. Postdoctoral researcher in the ICTEAM institute of UCL from 2005 to 2011, he was funded by the Walloon Region (2005-2006), the Belgian FRS-FNRS (2006-2010, 2011-2012) and by the Belgian Science Policy (Return Grant, BEL-SPO, 2010-2011). Visiting researcher at Rice University (DSP/ECE, Houston, TX, USA) in

spring 2007, he also performed a postdoctoral stay from 2007 to 2009 at the Swiss Federal Institute of Technology (LTS2/EPFL, Switzerland). Since Oct. 2012, he is Professor and FNRS Research Associate in the Image and Signal Processing Group (ISPG) in ICTEAM/UCL. His research focuses on Sparse Representations of signals (1-D, 2-D, sphere), Compressed Sensing theory (reconstruction, quantization) and applications, Inverse Problems in general, and Computer Vision.



Christophe De Vleeschouwer is a Senior Research Associate at the Belgian NSF, and an Associate Professor at UCL (ISPG-Group). He was a senior research engineer with IMEC (1999-2000), a post-doctoral Research Fellow at UC Berkeley (2001-2002) and EPFL (2004), and a visiting scholar at CMU (2014-2015). His main interests concern video and image processing for content management, transmission and interpretation. He is enthusiastic about non-linear

and sparse signal expansion techniques, ensemble of classifiers, multi-view video processing, and graph-based formalization of vision problems. He is the co-author of more than 35 journal papers or book chapters, and holds two patents. He served as an Associate Editor for IEEE Transactions on Multimedia, has been a co-founder of Keemotion (www.keemotion.com), using video analysis for automatic sport coverage. He is a member of the IEEE.