UNIVERSITÉ CATHOLIQUE DE LOUVAIN
ECOLE POLYTECHNIQUE DE LOUVAIN
ICTEAM INSTITUTE

Detection-based Multi-Object Tracking in Presence of Unreliable Appearance Features

Amit Kumar K.C.

Thesis submitted in partial fulfilment of the
requirements for the Ph.D. Degree in Applied Sciences

Thesis Committee

Advisor:	Christophe De Vleeschouwer, UCL, Belgium
Jury:	Laurent Jacques, UCL, Belgium
	Jean-François Delaigle, COFELY, Belgium
	Andrea Cavallaro, QMUL, UK
	François Fleuret, IDIAP/EPFL, Switzerland
President:	David Bol, UCL, Belgium

Louvain-la-Neuve, Belgium.

To my parents.

Abstract

Multi-object tracking (MOT) is the task of estimating the trajectory of several objects as they move around a scene. MOT has gained in interest due to its potential in many disciplines such as surveillance, sport analysis, human computer interface, biology, etc.

This thesis considers MOT in a scene captured by one or several sensors. It assumes that prior detections of the targets are available, and a set of features characterizing the appearance of the detections, have been extracted. In contrast to previous related works, we aim at formalizing the scenarios in which the reliability or even the availability of such appearance features vary over time. Our contributions, briefly explained below, are all related to the exploitation of sporadic and noisy features in a graph-based framework.

Our first major contribution proposes an iterative hypothesis testing (IHT) framework that embeds shortest-path computations into a hypothesis testing procedure. Each hypothesis assumes that the appearance of the target is defined by that of a selected node, called key-node. Given this assumption, the cost of going through a node that is different (similar) from the key-node is increased (decreased) to favor the selection of a path that is consistent with the appearance of the key-node. The shortest-path is validated only when it is sufficiently better than any alternative path. Doing so, we progressively aggregate the detections into tracklets while taking advantage of their features, even when they are sporadic and/or affected by non-stationary noise. In the subsequent target recognition, we utilize the unreliability of the appearance features to prioritize the message passing while assigning reliable identities to the tracklets.

As a second contribution, we propose a discriminative label propagation (DLP) framework to propagate labels across the detections in a way that is consistent with a number of complementary graphs that captures the various relationships, *e.g.*, similarities or dissimilarities between the detections

in terms of space, time and/or appearance between the detections. The resulting cost function is a difference of convex functions, which is efficiently solved using *majorization-minimization* techniques. We propose to decompose this 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.

Acknowledgments

First and foremost, I am grateful to my advisor Christophe De Vleeschouwer. This thesis would not have been possible without his constant supervision and motivation. He has also inspired me to stay positive at 'difficult' times during my PhD.

I am thankful to the Belgian National Science Foundation (F.R.S.-FNRS) for providing me financial resources and "la liberté de chercher". Some financial supports from the "Région Wallonne" project SPORTIC and Wallonie-Bruxelles International (WBI) are also greatly acknowledged.

I would like to thank Laurent Jacques for inspiring me with optimization theory, specifically the proximal operators. He has always given me good answers whenever I had some doubts/questions.

I am grateful to the jury members Andrea Cavallaro, Jean-François Delaigle and François Fleuret for accepting to evaluate this work. I believe that the quality of this thesis has improved greatly because of their constructive comments. In addition, I am grateful to Andrea for giving me the opportunity to spend a 6-month research stay in his lab at the Queen Mary, University of London (QMUL).

I have had a great opportunity to share my office A.160 with wonderful colleagues. In particular, I am thankful to Cédric Verleysen, Maxime Taquet, Damien Jacobs and Arnaud Browet. I have enjoyed having discussion about both technical and non-technical stuff with them. Cédric has been an awesome colleague over the years. Extra thanks go to him for helping me out with the French documents. Arnaud has helped me with LaTex and TikZ. I am grateful to Pascaline Parisot and Damien Delannay who provided me data and code for APIDIS project. I would like to convey special thanks to Pascaline Parisot for those wonderful cakes, proof-reading my thesis and frequent supervision of my works, specially in the beginning. I thank François Hubin, Patricia Focant and Jean Deschuyter for their constant administrative and technical help.

During my PhD, I have received support from my friends from Louvain-la-Neuve. My colleagues from the Stévin building (Kaori, Sébastien, Quentin, Augustin, Pierre-Yves, Stéphanie, Amir, Valerio, Mohieddine, Adriana, Mireia, Kévin, Arsalan, Donia) have contributed to enrich this amazing experience. I am grateful to all the friends from LLN, specially Shambhu, Subir, Vaibhav, Sandeep, Bhavesh, Aneesh, Sumit, Jeevan and others for supporting in various ways. I would like to thank Sumit for proof-reading some chapters.

Last but not least, I wish to give the largest part of my thanks to my family, specially my parents (Kedar Bahadur K.C. and Ishwori K.C.) and my wife (Sirjana K.C.) for their constant love and care.

Contents

List of Figures

List of Algorithms

List of Tables

List of symbols and acronyms

Acronyms

APIDIS Autonomous Production of Images based on Distributed and Intelligent Sensing (p. 48)

BP Belief Propagation (p. 11)

CLEAR Classification of Events, Activities and Relationships (p. 21)

CNN Convolutional Neural Network (p. 14)

DAG Directed and Acyclic Graph (p. 9)

DC Difference of Convex (p. 83)

DLP Discriminative Label Propagation (p. 6)

DP Dynamic Programming (p. 12)

DPM Deformable Parts Model (p. 14)

GAC Global Appearance Constraints (p. 6)

GMCP Generalized Minimum Clique Problem (p. 17)

GMOTA Global Multiple Object Tracking Accuracy (p. 23)

HOG Histogram of Oriented Gradients (p. 14)

IHT Iterative Hypothesis Testing (p. 5)

LELVM Laplacian Eigenmaps Latent Variable Model (p. 93)

LLE Locally Linear Embedding (p. 79)

MAP Maximum a Posteriori (p. 16)

METE Multiple Extended-target Tracking Error (p. 24)

MHT Multiple Hypothesis Tracking (p. 15)

MOT Multi-Object Tracking (p. 1)

MOTA Multiple Object Tracking Accuracy (p. 21)

MOTP Multiple Object Tracking Precision (p. 21)

MSER Maximally Stable Extremal Regions (p. 18)

MTT Multi-Target Tracking (p. 1)

MWIS Maximum Weight Independent Set (p. 16)

OLDAM On-line Learned Discriminative Appearance Model (p. 17)

OSPA Optimal Sub-Pattern Assignment (p. 23)

SIFT Scale Invariant Feature Transform (p. 18)

SVM Support Vector Machine (p. 14)

TSP Travelling Salesman Problem (p. 29)

Symbols

$\boldsymbol{A}$ Adjacency matrix (p. 7)

$\boldsymbol{b}_v$ Belief vector of node v (p. 12)

$\boldsymbol{C}$ Capacity matrix (p. 10)

$\boldsymbol{D}$ Degree matrix (p. 9)

$\boldsymbol{f}_i$ i-th appearance feature (p. 34)

$\boldsymbol{L}$ Graph Laplacian (p. 9)

$\boldsymbol{m}_{u\to v}^{(t)}$ Message from node u to node v at time t (p. 11)

$\boldsymbol{W}$ Weight matrix (p. 8)

$\boldsymbol{Y}$ Label assignment matrix (p. 9)

$\boldsymbol{y}_i$ Label vector assigned to node i (p. 9)

$\mathcal{E}$ Set of edges (p. 7)

$\mathcal{G}$ A graph (p. 7)

$\mathcal{J}$ Set of nodes chosen for parallel coordinate descent (p. 87)

$\mathcal{L}$ Label set (p. 11)

$\mathcal{N}_i$ Set of neighbors of node i (p. 8)

$\mathcal{V}$ Set of vertices/nodes (p. 7)

Π_m Set of permutations of $\{1, 2, \cdots, m\}$ (p. 23)

$C(T_i)$ Cost of set T_i (p. 28)

c_i Confidence value of the i-th feature (p. 34)

G_t Ground-truth at time t (p. 21)

K Number of disjoint sets/tracks/cliques (p. 28)

N Number of appearance features (p. 34)

P_{st} Path from node s to node t (p. 9)

Introduction and Motivation

1

1.1 Context and Scope

Video object tracking is the task of estimating the trajectory of one or several objects as they move around a scene, captured by one or several sensors. Typically, when the sensor is a camera, it usually relies on appearance features (shape, color, texture, size, etc.) to follow the objects across time. Video tracking problem is usually categorized into *single object tracking* and *multi-object tracking* (MOT) or *multi-target tracking* (MTT).

In single object tracking, a target of interest is followed in the video sequence, whereas in multi-object tracking, all targets of interest are followed simultaneously. The key difference between multi-object tracking and tracking independently multiple targets is that the latter approach considers other targets as 'noises'. Consequently, it fails to exploit the (exclusivity) relationship between the targets. In contrast, MOT jointly considers the relationships between the targets and, therefore, results in more accurate tracking performance. In this thesis, we focus on MOT only.

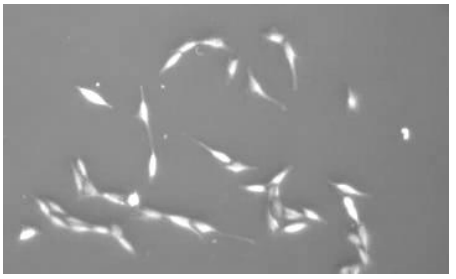
MOT has gained numerous attention due to its potential in many disciplines, ranging from surveillance to biology (see Figure 1.1). Visual surveillance applications analyze people's displacements. Tracking body parts allows humans to interact with computer by gesture. Vehicle trajectories can be used to monitor the traffic [1]. Tracking of players [2] in sport events helps to analyze and interpret the game. In the biomedical area, the mobility of cells is considered to assess tissue-repair, and predict diseases in early stages [3, 4]. In material engineering, tracking the evolution of cracks is pivotal in the study of the fracture mechanism in metals [5].

Many approaches have been proposed in the literature for object tracking.

(a) Sport analysis [6]

(b) Surveillance [7]

(c) Animal behavior [8]



(d) Biology [9]

Figure 1.1: **Some application scenarios of MOT**. *Best viewed in color.*

Roughly, they can be categorized into *propagation-based* and *detection-based*[1] solutions. Propagation-based tracking approaches propagate the state (namely, position, size, etc.) of the objects over time using local prediction and convergence mechanism [13, 14, 15, 16, 17]. Even though these approaches have been successfully used to track single (or few) objects, these approaches are not very well suited to multiple objects due to rapid increase in complexity. In detection-based tracking approaches, target locations are first estimated in each time instant. Once objects have been detected, a set of target appearance features (*e.g.*, color, size, shape, etc.) is extracted, and the tracking problem reduces to the task of linking these detections into tracks of single physical objects using these location and appearance features.

We limit our scope to the scenarios for which objects-of-interest are first detected at each time instant. This is because detections are essential for automatic initiation (respectively, termination) of the tracking process when objects are discovered in the scene (respectively, disappear from the scene), *e.g.*, birth/death of cells. More importantly, efficient and effective algorithms have

[1] also called association-based tracking (ABT) [10], tracking-by-detection [11], track-after-detect [12], etc.

been designed in many computer vision fields to detect objects-of-interest. For example, our host laboratory has demonstrated that people can be detected reliably based on a distributed network of cameras [18, 19]. Besides, specific shape and size of the targets make them easy to detect with ad-hoc filters [20, 21, 22].

1.2 Motivation and research statement

When targets are isolated, the position feature is sufficient to aggregate them into consistent object trajectories. However, when the targets come closer to each other, position feature alone is not sufficient to follow the targets accurately. Appearance features are then crucial to disambiguate a clutter of targets.

Different features have different discriminative abilities and different levels of observation frequency and reliability. For example, position feature is quite frequent but poorly discriminant in clutters. Color features are also frequently available, but have variable discriminative power depending on the application context. In some cases, highly discriminant features are available only sporadically. This occurs, for example, when biological cells are captured under changing illumination, where each lightning condition emphasizes some specific characteristics of the cell. This case also happens when a feature is only visible in some observation configurations (*e.g.*, a number worn by a player is only visible when facing the camera, faces are identifiable only when the person turns towards the camera properly). These features are unobserved for long duration. In summary, features relevance, meaning their availability and their ability to discriminate targets, is not uniform and varies along time and with the scene context. Examples of such situations are depicted in Figure 1.2.

Most previous approaches assume that the features should only be similar for detections that are close in time. Thus, dissimilarities between far away detections are not considered. Under this assumption, the dissimilarity of a track can be represented by the accumulation of dissimilarities between the detections that are observed in close time. This assumption allows one to simplify the tracking problem (see Chapter 3). The validity of such assumption is justified in many conventional tracking scenarios where the appearance features vary smoothly along time. In those cases, comparing appearances that are far apart in time does not bring much cues about whether those two detections

(a) Face (b) Digit (c) Color

Figure 1.2: **Appearance features can be noisy and/or sporadic.** Face (in (a)) and digit features (in (b)) are observed only when they face the camera. Color feature (in (c)) is noisy and unreliable because of occlusion. *Best viewed in color.*

correspond to the same physical target or not.

In contrast, these methods cannot exploit the information captured by the detections that are distant in time. This is relevant specially when (i) some highly discriminant features (*e.g.*, digit features, faces, etc.) are available only sporadically and/or, (ii) some appearance features are not uniformly reliable along the scene. In such cases, the (in)consistency of the appearances along a path cannot be any more measured simply based on the accumulation of the appearance (dis)similarities between consecutive detections. This is because those dissimilarity measurements might be unreliable and/or purely unavailable. Not using those features, however, strongly penalizes tracking performances in some scenarios. For example, using digit feature on the jersey allows one to directly recognize the player and to track him reliably.

Hence, the federating objective of our work is to design a multi-object tracker that is able to exploit appearance cues that are noisy and/or only available sporadically.

1.3 Contributions

As explained earlier, most of the detection-based tracking approaches are not suitable to handle such noisy/sporadic appearance features. Our contributions, briefly presented below, are all related to the exploitation of those features. Similar to many approaches, we adopt graph-based formalism[2].

[2]The demo videos are available at http://sites.uclouvain.be/ispgroup/index.php/Research/MultiObjectTracking.

1.3.1 Iterative hypothesis testing framework

We formulate the problem by embedding a shortest-path computation into a series of tests or *hypotheses*. Each hypothesis investigates if a detection, so called *key detection*, can *unambiguously* be connected with other detections based on the similarity in terms of space, time and appearance. The hypothesis starts with the assumption that the key detection appearance reflects target appearance. Given such an assumption about target appearance, it is possible to increase or decrease the cost of going through a detection irrespective of its location in the sequence by comparing its appearance features and the assumed target appearance features. This hypothesis is validated only when the shortest-path connecting the key detection to the extremity of an observation window is sufficiently better than any alternative path, under the target appearance assumption.

In practice, we iterate over the detections, which is why we refer to this approach as to the *iterative hypothesis testing* (IHT) approach. It is further discussed and positioned with respect to previous arts in Chapter 3. Experimental results have shown a dramatic improvement in the tracking performance in several datasets. Part of this chapter has been published in [23] and a journal version is in preparation.

1.3.2 Priority identity propagation for sport player recognition

This work builds on our IHT framework and addresses the problem of recognizing sport players from their tracklets. Specifically, given an identity estimate of a detection/tracklet, we investigate how we can propagate this identity estimate to other detections. Because of the noise and sporadicity of the appearance features, some detections will have more reliable identity estimate than others. This measure of reliability is exploited to prioritize the propagation or identities. We have achieved a player recognition rate around 89%. This work has been published in [24].

1.3.3 Discriminative label propagation framework

Our goal is to assign the same label to detections that correspond to the same physical target. For this purpose, labels are assigned consistent with constraints that are formulated in terms of space, time and appearance. For example, detections that co-exist at the same time instant should be labeled differently because a target cannot occur at two locations at the same time. Detec-

tions that have similar appearance features should be labeled similarly even if they are far apart in time. This allows us to exploit the appearance features even if they are sporadic.

In practice, we construct a number of graphs that reflect these constraints and propagate labels over the graphs. We refer to this as *discriminative label propagation* (DLP). This is discussed in detail in Chapter 5. This work has been published in [25] and a journal paper is under review in IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).

1.4 Organization of the thesis

The remainder of this thesis is structured as follows. Chapter 2 first introduces basic elements of graph theory. Then, it presents a basic pipeline for detection-based MOT, and defines some terminologies related to it. Eventually, it briefly reviews previous works related to object detection and graph-based multi-object tracking formalisms.

Chapter 3 explains in detail our iterative hypothesis testing (IHT) framework. We demonstrate its advantages compared to two popular algorithms for computing shortest paths, namely the K-shortest paths (KSP) [26] and the global appearance constraints (GAC) [27] algorithms.

Chapter 4 explains how prioritizing the propagation of identity beliefs helps in recognizing individual players in sports scenes.

Chapter 5 presents our discriminative label propagation (DLP) framework. Specifically, it (i) explains the construction of a number of complementary graphs that captures the labeling constraints associated to relationships between the detections, (ii) formulates the label propagation as a difference of convex problem, and (iii) presents an efficient algorithm to solve the problem, including in an on-line/incremental context.

Finally, we conclude the thesis with future perspectives in Chapter 6. A description of datasets, used in this thesis, is provided in Appendix C.

Related Works

2

This chapter introduces some basic elements of graph theory, including some algorithms on graphs that are used in multi-object tracking. It then presents the detection based tracking paradigm, and reviews several approaches used in the literature to address the object detection problem and the MOT question using graph formalisms.

2.1 Elements of graph theory

This section first introduces graph terminology. Afterwards, it reviews some algorithms on graphs which are related to multi-object tracking.

2.1.1 Terminology

A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical representation of pairwise interaction between n individual agents and is defined by a set of *vertices* (or *nodes*) $\mathcal{V} = \{1, \cdots, n\}$ and a set of *edges* (or *links*) $\mathcal{E} = \{(i, j) \mid i, j \in \mathcal{V}\}$. Nodes i and j are said to be *neighbors* if $(i, j) \in \mathcal{E}$. We denote the number of nodes and edges by $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$, respectively.

A graph can be represented by its *adjacency matrix* $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ as:

$$A(i, j) = a_{ij} := \begin{cases} 1 & \text{if } (i, j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

A graph is called *undirected* if the edges have no orientation, *i.e.*, for all $i, j \in \mathcal{V}$, we have $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$. The adjacency matrix of an undirected graph is *symmetric*, *i.e.*, $\boldsymbol{A} = \boldsymbol{A}^\top$. When the orientation of the edges matters,

the graph is called *directed*. In this case, each edge $(i, j) \in \mathcal{E}$ has a source i and destination j. The adjacency matrix of a directed graph is not necessarily symmetric. An edge that connects a node to itself, *i.e.*, $(i, i) \in \mathcal{E}$, is called a *self-loop*.

One can associate a weight w_{ij} to an edge $(i, j) \in \mathcal{E}$ representing the intensity of interaction between nodes i and j. The resulting graph is said to be *weighted*. A weighted graph can be represented by a weighted adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ such that $W(i, j) = w_{ij} \neq 0$ if $a_{ij} = 1$. In the remainder of the thesis, we assume that the graphs are weighted because an unweighted graph can be represented with $w_{ij} = 1 \; \forall (i, j) \in \mathcal{E}$.

The set of in- and out-neighbors of a node i in a directed graph are respectively defined as:

$$\mathcal{N}_i^{(in)} := \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\},$$
$$\mathcal{N}_i^{(out)} := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}.$$

The in- and out-degrees of a node i are defined as:

$$d_i^{(in)} := \sum_{j \in \mathcal{N}_i^{(in)}} w_{ji},$$
$$d_i^{(out)} := \sum_{j \in \mathcal{N}_i^{(out)}} w_{ij}.$$

In case of undirected graph, the set of neighbors and the degree of the i-th node are defined respectively as:

$$\mathcal{N}_i := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\},$$
$$d_i := \sum_{j \in \mathcal{N}_i} w_{ij}.$$

2.1.2 Some algorithms on graphs

In this subsection, we briefly describe some graph-based algorithms that are commonly used for multi-object tracking.

Shortest path

Given two nodes $s, t \in \mathcal{V}$, we define a *path* P_{st} of length k as:

$$P_{st} := \{v_1 = s, v_2, \cdots, v_{k-1}, v_k = t\}$$

such that $(v_i, v_{i+1}) \in \mathcal{E}\ \forall i$. There can be multiple paths between s and t. We define the set of possible paths between s and t by $\mathcal{P}_{st} := \{P_{st}\}$. When the weight w_{ij} at each edge $(i,j) \in \mathcal{E}$ is defined as a *distance* or *cost* function, the *shortest-path* from s to t is the path that has the smallest overall distance, *i.e.*,:

$$P_{st}^{(min)} := \underset{P_{st} \in \mathcal{P}_{st}}{\operatorname{argmin}} \sum_{i=1}^{k-1} w_{v_i, v_{i+1}}.$$

When the weights are non-negative, Dijkstra's algorithm [28] can be used to find the shortest-paths between any pairs of nodes. Given a single node as 'origin' node, it finds the shortest paths from the origin node to all other nodes. The complexity of Dijkstra's algorithm is $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$. When the graph is directed and acyclic (DAG), *i.e.*, without any cycles, a topological sorting[1], which runs at $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}|)$, can be used instead of Dijkstra's algorithm.

Laplacian and labeling energy

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, the graph *Laplacian* $\boldsymbol{L}$ is defined as:

$$\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{W}$$

where $\boldsymbol{D} := \mathbf{diag}(d_1, \cdots, d_n)$ is a diagonal matrix for which the i-th diagonal element is the degree of the i-th node. It is commonly called the *degree* matrix. From the definition, it follows that the ij-th element of $\boldsymbol{L}$ is

$$L_{ij} := \begin{cases} d_i - w_{ii} & \text{if } i = j, \\ -w_{ij} & \text{if } (i,j) \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

Let a label assignment $\boldsymbol{Y} = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n)^\top$ assigns a K-dimensional vector $\boldsymbol{y}_i$ to the i-th node. Then, the inconsistency of the label assignment matrix $\boldsymbol{Y}$ with respect to the graph $\mathcal{G}$ can be measured using *harmonic function approach*, introduced in [29], as:

$$E_{\boldsymbol{L}}(\boldsymbol{Y}) := \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} w_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L} \boldsymbol{Y}),$$

where **Tr** is the trace of a matrix. Since the graph has non-negative weights, the graph Laplacian $\boldsymbol{L}$ is *positive semi-definite* and consequently, the energy $E_{\boldsymbol{L}}(\boldsymbol{Y})$ is *convex* in $\boldsymbol{Y}$.

[1] Topological sort of a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a linear ordering of all its nodes such that if $\mathcal{G}$ has an edge (u, v), then u appears before v in the ordering.

Flow network

A *flow network* is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{C}, \boldsymbol{W})$ in which each edge $(i, j) \in \mathcal{E}$ has a capacity $c_{ij} \in \mathbb{R}_+$. Let us consider two vertices: a source s and a sink/terminal t. A flow in the network is a non-negative real function $f : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_+$ that satisfies the following properties for all edges $(i, j) \in \mathcal{E}$:

- **Capacity constraints:** The flow along an edge cannot exceed its capacity, *i.e.*, $f_{ij} \leq c_{ij}$.
- **Flow conservation:** The net flow to a node is zero, except for the source and the terminal, which 'produces' and 'consumes' flow respectively. Hence, $\sum_{k \in \mathcal{N}_i^{(in)}} f_{ki} = \sum_{j \in \mathcal{N}_i^{(out)}} f_{ij}$ for all $i \in \mathcal{V} \setminus \{s, t\}$.

When it is possible to reduce multiple edges in different directions to a single, oriented edge, *i.e.*, the graph is 'reduced', it is common to introduce the *skew-symmetry* property of flow. It states that the flow from i to j must be opposite to that from j to i, *i.e.*, $f_{ij} = -f_{ji}$.

A flow $\boldsymbol{f}$ that satisfies all above constraints is said to be *feasible*. Since the cost of sending f_{ij} units of flow through an edge $(i, j) \in \mathcal{E}$ is $w_{ij} f_{ij}$, the cost of a feasible flow $\boldsymbol{f}$ is $\sum_{(i,j) \in \mathcal{E}} w_{ij} f_{ij}$. Then, the problem of sending d units of flow from s to t as cheaply as possible is referred to as the *min-cost flow problem*. Mathematically, it can be formalized as:

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{(i,j) \in \mathcal{E}} w_{ij} f_{ij}, \\
\text{subject to} \quad & 0 \leq f_{ij} \leq c_{ij}, \\
& \textstyle\sum_{k \in \mathcal{N}_i^{(in)}} f_{ki} = \sum_{j \in \mathcal{N}_i^{(out)}} f_{ij} \quad \forall i \in \mathcal{V} \setminus \{s, t\}, \\
& f_{ij} = -f_{ji}, \\
& \textstyle\sum_{i \in \mathcal{N}_s^{(out)}} f_{si} = d, \\
& \textstyle\sum_{j \in \mathcal{N}_t^{(in)}} f_{jt} = d.
\end{aligned}
\tag{2.1}
$$

The min cost flow problem can be solved by *linear programming*. Besides, there exists other graph-related algorithms like Ford-Fulkerson algorithm [30], push-relabel algorithm [31, 32], etc. to solve this problem.

Random field

A conditional random field (CRF) [33] models dependencies between data. Given sets of observations $\boldsymbol{X}$ and hidden (random) variables $\boldsymbol{Y}$, a CRF models the conditional probability distribution $\mathbf{prob}(\boldsymbol{Y}|\boldsymbol{X})$ with an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where

– a node $v \in V$ corresponds to the v-th random variable $\boldsymbol{Y}_v$,

– an edge $(u, v) \in \mathcal{E}$ encodes the statistical dependence between u and v such that $\boldsymbol{Y}$ obeys Markov property with respect to $\mathcal{G}$. In other words, $\boldsymbol{Y}$ satisfies $\mathbf{prob}(\boldsymbol{Y}_v|\boldsymbol{X}, \boldsymbol{Y}_u, u \neq v) = \mathbf{prob}(\boldsymbol{Y}_v|\boldsymbol{X}, \boldsymbol{Y}_u, u \in \mathcal{N}_v)$.

Then, the inference problem can be written as

$$\boldsymbol{Y}^\star = \underset{\boldsymbol{Y}}{\operatorname{argmax}}\, \mathbf{prob}(\boldsymbol{Y}|\boldsymbol{X}) \tag{2.2}$$

Assuming that the joint probability $\mathbf{prob}(\boldsymbol{Y}|\boldsymbol{X})$ can be written in terms of unary ϕ and pairwise ψ potentials, we write Equation 2.2 as:

$$\boldsymbol{Y}^\star = \underset{\boldsymbol{Y}}{\operatorname{argmax}} \prod_{v \in \mathcal{V}} \phi(\boldsymbol{y}_v|\boldsymbol{X}) \prod_{(u,v) \in \mathcal{E}} \psi(\boldsymbol{y}_u, \boldsymbol{y}_v|\boldsymbol{X}), \tag{2.3}$$

Equation 2.3 can be equivalently written as a minimization problem as

$$\boldsymbol{Y}^\star \equiv \underset{\boldsymbol{Y}}{\operatorname{argmin}} \sum_{v \in \mathcal{V}} U(\boldsymbol{y}_v|\boldsymbol{X}) + \sum_{(u,v) \in \mathcal{E}} B(\boldsymbol{y}_u, \boldsymbol{y}_v|\boldsymbol{X}) \tag{2.4}$$

where $U := -\log \phi$ and $B := -\log \psi$. Designing a good CRF for inference requires careful construction of U (or, ϕ) and B (or, ψ). Alternatively, a CRF can be learned through training. For example, Szummer *et al.* [34] learn a CRF using graph-cuts [35, 36]. Inference on a CRF is usually done via graph-cuts [36] or belief propagation (BP) [37]. Since we use it in Chapter 4, we describe BP briefly.

Belief propagation

Belief propagation, proposed by Judea Pearl in 1982, is a *message passing* algorithm for performing inference on graphical models. It estimates the marginal distribution for each unobserved node, conditionally to any observed nodes. The algorithm works by passing real valued functions called *messages* along the edges between the hidden nodes. Let us denote the set of all possible states (*e.g.*, labels) by $\mathcal{L}$. Let $\boldsymbol{m}^{(t)}_{u \to v}$ be the $|\mathcal{L}|$-dimensional message that the node u sends to a neighboring node v at iteration t. Intuitively, $\boldsymbol{m}^{(t)}_{u \to v}(\boldsymbol{y}_v)$ is the belief that node u thinks about the state $\boldsymbol{y}_v \in \mathcal{L}$ of node v at any iteration t. Each message is initialized uniformly. Afterwards, each node gathers messages from its neighbors and then transmits its message. The messages are updated in sum-product form at each iteration as:

$$\boldsymbol{m}^{(t)}_{u \to v}(\boldsymbol{y}_v) \propto \sum_{\boldsymbol{y}_u \in \mathcal{L}} \Big[\psi(\boldsymbol{y}_u, \boldsymbol{y}_v|\boldsymbol{X}) \underbrace{\phi(\boldsymbol{y}_u) \prod_{s \in \mathcal{N}_u \setminus v} \boldsymbol{m}^{(t-1)}_{s \to u}(\boldsymbol{y}_u)}_{:= \boldsymbol{h}_u(\boldsymbol{y}_u)} \Big] \tag{2.5}$$

where $\boldsymbol{h}_u(\boldsymbol{y}_u)$ is the information gathered at node u about the label $\boldsymbol{y}_u$. It is also referred to as the *pre-message* for $\boldsymbol{y}_u$. Alternatively, the summation term in Equation 2.5 is replaced by a max term and is referred as *max-product form* [38].

After T iterations, a belief vector $\boldsymbol{b}_v$ is computed for each node as:

$$\boldsymbol{b}_v^{(T)}(\boldsymbol{y}_v) \propto \phi(\boldsymbol{y}_v) \prod_{s \in \mathcal{N}_v} \boldsymbol{m}_{s \to u}^{(T)}(\boldsymbol{y}_v) \tag{2.6}$$

Finally, the normalized belief vector $\boldsymbol{b}_v^{(T)}$ provides the estimate of the identity distribution of the node v. In general, the complexity of message passing is $\mathcal{O}\big(|\mathcal{V}||\mathcal{L}|^2 T\big)$. It requires $\mathcal{O}\big(|\mathcal{L}|^2\big)$ to compute each message, there are $|\mathcal{V}|$ messages to compute at each iteration, and there are T iterations. The message construction can be performed in $\mathcal{O}(|\mathcal{L}|)$ if ψ has linear or quadratic form [39].

The belief propagation is proved to be exact when there are no cycles in graph. In such cases, it is equivalent to dynamic programming (DP). In case of cyclic graphs, BP provides only approximate solutions. In practice, however, the approximate solutions are often good.

2.2 Detection-based multi-object tracking paradigm

In this section, we briefly define some terminologies that are widely used in detection-based multi-object tracking. Figure 2.1 depicts a general detection-based tracking pipeline. In detection-based tracking approaches, plausible object locations are first estimated in each individual frame, based on the exploitation of discriminant appearance features like shape, size, color, gradients, etc. (see Section 2.3). Once objects have been detected, the tracking problem reduces to a data association problem, which has to link those detections into tracks of single physical objects. Graph-based solutions are generally envisioned to solve this problem (see Section 2.4).

Detection-based multi-object tracking largely refers to the following terminologies:

- **Detections** or detection responses are outputs of an object detector which is trained for specific objects-of-interest, *e.g.*, human, cells, face, vehicle, etc. We review several approaches for detectors in the next section. Detections can be *hard*, meaning that the detector takes a binary decision about the location and/or size (*i.e.*, *state*) of the target, or *soft*, meaning that the detector outputs a confidence value along with the state of the

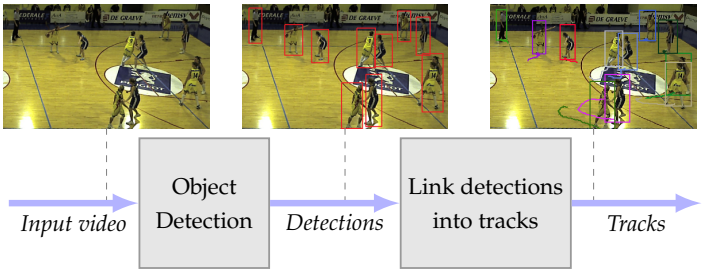


Figure 2.1: **Block diagram of detection-based tracking approach.**

target. At each detection, a number of appearance features, *e.g.*, color histograms, texture, shape, etc. are extracted.

- **Tracks** are sequences of detections along time. They include at most one detection at each time instant, and are supposed to aggregate detections corresponding to the same physical target.
- **Tracklets** are fragments of tracks that are assumed to have originated from the same target. In other words, a tracklet is a sequence of detections that *reliably* correspond to the same target.

These concepts are illustrated in Figure 2.2.

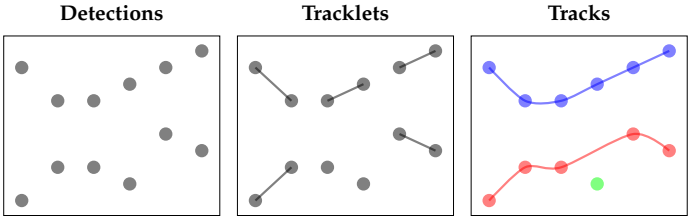


Figure 2.2: **Detections (*left*), tracklets (*center*), and tracks (*right*) for 6 time instances**. Different colors represent different targets.

2.3 Related works in object detection

We now briefly introduce various approaches for the detection of objects-of-interest.

The most computationally efficient detectors generally rely on a *foreground mask*. A foreground mask is obtained by thresholding the difference between

the observed image and a background model [40, 41]. The foreground mask is supposed to detect the moving areas in the view, and can be compared with the model of the object silhouette in order to localize the objects in the image [21, 42, 43]. These detectors usually suffer from significant false and/or missed detections in presence of multiple moving and/or interacting objects, or when the objects are stationary. To address these limitations, modern approaches (i) merge the information carried by the foreground masks computed from multiple and complementary views of the same scene [44, 45, 18, 2, 19], or (ii) exploit visual classifiers to capture the specificities of the object visual appearance [46, 22].

On the one hand, several strategies have been considered to fuse the masks from multiple views [44, 45, 18, 2, 19, 43]. They generally rely on the definition of a ground occupancy probability map, which exploits the verticality of people's silhouettes or 3D geometry approximation [43] to estimate the likelihood of whether a particular ground plane position is occupied or not .

On the other hand, significant works have been done to detect people or objects of interest based on their visual appearance. Modern approaches make an extensive use of training samples, to learn how the object is defined in terms of topologically organized components [22] and/or in terms of texture statistics [47]. The pioneering work of Viola and Jones [48] illustrates the success of those approaches to detect objects in images. It relies on boosting strategies to select and combine a large number of weak binary tests to decide whether the content of a (sub-)image corresponds to the object-of-interest or not. Dalal and Triggs [46] have proposed a sliding window detector that extracts a feature vector based on the local orientation gradients, referred to as the histogram of oriented gradients (HOG). Afterwards, a support vector machine (SVM) decides whether the feature vector corresponds to a pedestrian or not. Another popular work on object detection is by Felzenszwalb *et al.* [22]. Their work, called the deformable parts model (DPM), treats an object as a constellation of spatially consistent individual parts. Doing so, the detector is able to represent high variation in object classes and achieves state-of-the-art results in object detection challenges. Therefore, DPM is often used as a preferred people detector for detection-based tracking approaches. Moreover, DPM can be adapted to exploit various constraints. For example, Tang *et al.* [49] observe that typical occlusions are due to overlaps between people and tailor the existing DPM to detect pedestrians at various occlusion levels.

Recently, there has been a growing interest in convolutional neural network (CNN) [50, 51] for object detection. Girschick *et al.* [52] have used CNN

for object detection. Given an image, they first generate many (around 2000) region proposals. Afterwards, a CNN is used to compute features. Finally, a class-specific classifier is used to classify the regions into different classes (*e.g.*, aeroplane, person, etc.). They have shown a dramatic improvement in object detection accuracy.

2.4 Related works in graph-based multi-object tracking

In this section, we review various graph-based algorithms that have been proposed in the literature to address the MOT problem. Broadly, these approaches can be classified into the following categories:

- Multiple hypothesis tracking
- Hierarchical data association
- Conditional random field
- Network model
- Message passing methods
- Other methods

2.4.1 Multiple hypothesis tracking

In 1979, Reid [53] proposed an algorithm, called multiple hypothesis tracking (MHT) for tracking multiple targets in a cluttered environment. An efficient solution to MHT was proposed by Cox and Hingorani [54] in which the authors provide k-best hypotheses in polynomial time using Murty's algorithm [55].

When new detections are received, probabilities are calculated for the hypotheses that the detections correspond to one of the previously known targets, or to new targets or to false detections. Each hypothesis is associated with a probability score. The hypotheses are represented by a *tree* where a path from the *root* node to a *leaf* node defines a time-sequence of hypothesis. Subsequent steps of the algorithm analyze how each of these hypotheses are consistent with future observations, leading to an exponential growth of the hypotheses with time. To circumvent this, unlikely hypotheses are pruned

and hypotheses with similar states are merged. Furthermore, the entire set of targets and measurements is divided into clusters that are solved independently.

Our IHT is fundamentally different from the MHT. In IHT, a "hypothesis" assumes that the appearance of the chosen key-node defines the target appearance. Under this appearance hypothesis, the shortest-path from/to the key-node is computed as a track. It either keeps or rejects the track, based on a testing phase, checking that no 'reasonably good' alternative path exists.

2.4.2 Hierarchical data association

Most hierarchical data association approaches first construct reliable short tracklets, and then progressively concatenate them into long trajectories. In these approaches, a node corresponds to a tracklet and each edge that connects two nodes gets a weight that measures the (dis)similarity between these two nodes in terms of space, time and/or appearance. These approaches differ in the way short traklets are computed (*e.g.*, based on shortest-paths) and the way weights between the tracklets/detections are defined. Nevertheless, these methods benefit from the fact that (i) the tracking problem is reduced to a series of smaller sub-problems, and (ii) the weight (or affinity) between the nodes is refined at each level.

Huang *et al.* [56] perform data association on a three-level hierarchy starting from a conservative low-level linking of the detector responses to form two-frame tracklets. In mid-level, these tracklets are further associated to form longer ones based on a dynamic model and a refined appearance computation. The association between the tracklets is formulated as a maximum a posteriori (MAP) problem and is solved by the Hungarian algorithm [57]. Finally, high-level data association infers scene information such as scene occluders and entry/exit areas and produces final tracks.

Li *et al.* [58] use boosting technique to learn the similarity score between the tracklets instead of selecting them heuristically. They formulate the tracklet association problem as a joint problem of ranking and classification, where the ranking part aims at favoring correct tracklet associations against other plausible associations, and the classification part rejects wrong associations.

Brendel *et al.* [59] formulate the multi-object tracking problem as the problem of finding the maximum weight independent set (MWIS). They provide a polynomial-time MWIS algorithm and show that it converges to a local optimum. Long-term occlusions are handled by iteratively repeating MWIS to

progressively merge smaller tracks into longer ones.

Zamir *et al.* [60] construct a graph by connecting all detections, except the ones that occur at the same time instant, within a temporal window. Each edge is weighted by the appearance and motion dissimilarities. The appearance cost of any 'feasible' solution (which corresponds to a tracklet) is based on comparing all pairs of detections within the tracklet, irrespective of the temporal ordering. The solution is obtained by solving a generalized minimum clique problem (GMCP), which greedily extracts tracklets that have the most stable appearance features and the most consistent motion. The same procedure is repeated to generate long trajectories. Short- and long- term occlusions are handled by using hypothetical nodes.

All these methods depend on appearance information to be available for each detection (or tracklet). Therefore, their applicability is limited to the scenarios where the appearance features can be extracted with the same reliability. Our iterative hypothesis testing approach (see Chapter 3) also works hierarchically, but supports sporadic and/or non-stationary observation processes.

2.4.3 Conditional random fields

In case of multi-object tracking, $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent the detections (or tracklets or even pairs of tracklets) and their hidden labels respectively. The unary function $\phi(\boldsymbol{y}_v|\boldsymbol{X})$ measures the likelihood that the node $v \in \mathcal{V}$ can be labeled $\boldsymbol{y}_v$ given the observation $\boldsymbol{X}$. Similarly, $\psi(\boldsymbol{y}_u, \boldsymbol{y}_v|\boldsymbol{X})$ stands for the compatibility between nodes u and v having labels $\boldsymbol{y}_u$ and $\boldsymbol{y}_v$ respectively.

In Yang and Nevatia's CRF model [61, 62], each node represents a pair of tracklets. The inference problem is to find the binary label of each node that indicates whether the two tracklets, corresponding to this node, can be linked (label=1) or not (label=0). The unary potential is based on the similarity between the two tracklets in terms of appearance and motion. Specifically, given all tracklets, a linear motion model and an appearance model (called *online learned discriminative appearance model* (OLDAM)) are learned to define the similarity between the tracklet pairs. Edges are introduced to better discriminate different targets, especially difficult pairs, which are spatially near targets with similar appearance. The pairwise potential of an edge represents the possible correlation (*e.g.*, two targets moving closely together) between two pairs of tracklets. These pairwise potentials are learned for each edge separately.

CRFs have also been used to identify the players in broadcast sport videos

by Lu *et al.* [63, 64]. The authors first associate the detections into tracklets and then extract a set of features like SIFT [65] interest points, MSER [65] regions, color histograms, etc. from these tracklets. These feature vectors and the corresponding labels of the players define the observed and hidden variables of the CRF, respectively. Multi-class logistic regression is chosen for the inference, where the parameters of the model are learned during a training phase. In [63], they use belief propagation [37] to infer the labels, whereas linear formulation is opted in [64].

Milan *et al.* [66] use CRF to model the detection- and trajectory- level exclusivity constraints. At the detection-level, a fixed exclusion cost is used between the detections that occur at the same time, whereas at the trajectory-level, the exclusion cost is defined in terms of spatio-temporal overlap between the trajectories. They perform statistical analysis of the ground-truth trajectories to derive appropriate CRF potentials.

Our work on player recognition is also built on the CRF model in which the objective is to infer the identities on a set of player tracklets, using prioritized belief propagation as an inference method.

2.4.4 Network models

When using flow network in multi-object tracking scenarios, nodes usually represent the detections or the tracklets. The flow is modeled as the binary variable indicating whether a flow is present (flow=1) or not (flow=0). This corresponds to whether two nodes can be linked (flow=1) or not (flow=0).

Jiang *et al.* [67] explicitly model the spatial layout and mutual occlusion constraints in a network structure, and use linear programming relaxation to solve this multi-object tracking problem. Zhang *et al.* [68] use a min-cost network with non-overlapping constraints on trajectories. The network is augmented with explicit occlusion model to long-term inter-object occlusions. Afterwards, the global optimal trajectory association is found by solving a linear program. Another globally optimal solution is presented by Pirsiavash *et al.* [69] that sequentially estimates tracks using shortest path computations on a flow network. The authors also give a near-optimal algorithm based on dynamic programming which is linear with the number of objects and the sequence length. Butt and Collins [70] modify the usual min-cost network to incorporate higher order motion. Doing so, they introduce exclusivity constraints that render the problem unsolvable by min-cost network flow directly. Consequently, they propose an iterative algorithm that relaxes the extra

constraints using Lagrangian relaxation, which produces a series of problems solvable by min-cost flow.

Berclaz *et al.* [26] reformulate the network flow problem as a problem of finding K-shortest paths. This novel formulation is much more efficient than the general linear program (LP) formulation technique. This work is later extended by Ben Shitrit *et al.* [27, 71] to incorporate long-range appearance model, called global appearance constraints (GAC). Assuming that they have prior knowledge about L target appearances, they create L-layered graph (each corresponding to one target appearance) and compute the K-shortest paths in the resulting network. In order to reduce the huge complexity, they apply heuristics like pruning the graph, grouping the detections into tracklets, etc. in [71].

2.4.5 Message passing methods

Message passing approaches have been used to label the nodes in a graph in tracking/recognition scenario [72, 73, 64]. 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 [37].

In [64], a subset of the nodes are initially labeled 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 [24], we utilize such non-stationary and sporadic features to prioritize the propagation of belief related to the label probability distribution.

Nilius *et al.* [72] construct a Bayesian network, called a track graph, where nodes represent either single target trajectories or merged tracks. Assuming that the feature vectors of isolated tracks are reliable, they define a similarity measure between them. Later, they formulate the identity linking as a Bayesian inference problem to find the most probable set of paths. For this, they use standard message passing technique. To keep the inference tractable the graph complexity is reduced by removing dependencies that are distant in time. Russell *et al.* [73] formulate the tracking problem as a problem of estimating MAP solution over a directed acyclic hyper-graph, where they capture both the second order motion information and mutual exclusivity constraints. The solution is again obtained by message passing technique. Their approach

runs linearly in the number of objects to be tracked, possible locations of an object, and the number of frames.

2.4.6 Other methods

Song *et al.* [74] propose a stochastic graph evolution framework to associate tracklets into longer tracks by analyzing the statistical properties of individual tracklets as well as each proposed long-term tracks. They assume that the appearances of the tracklets within a window follow a Gaussian distribution with the mean and variance computed from a chosen tracklet (similar to our key-node). Under this assumption, they also modulate the similarity between the chosen tracklet and other tracklets. These tracklets are associated into *potential* tracks by using Hungarian algorithm [57]. To compute the quality of data association, the goodness of a candidate track is estimated by comparing the variance of appearance features at each edge along the track. Specifically, the appearances of all tracklets within the track are clustered into two groups with respect to an edge. Then, a *tracklet association cost* (TAC) is defined at the edge as the ratio of variance of appearances between the clusters to the sum of variance of appearances within the clusters. Given a measure of quality of data association based on TAC, they perform Metropolis-Hasting (MH) sampling method to search for an optimal association. This definition is similar to the Fischer's linear discriminant function. Our method differs from it in two ways. Not only the assumption about Gaussian distribution of the appearances does not hold in case of sporadic appearances, but also TAC cannot handle unreliable or sporadic features simply because these features might be unreliable and/or unavailable for most of the duration of the track.

2.5 Evaluation criteria

To assess the correctness of a multi-object tracker, it is necessary to quantify its performance. We distinguish these evaluation methodologies as (i) *frame-level evaluation* and (ii) *trajectory-level evaluation*. Frame-level evaluation methods such as [75, 76] compute errors at each frame and then aggregate them along time, whereas trajectory-level methods such as [77, 58] compute errors on a track basis. In other words, frame-level evaluations indicate how well the ground-truth targets have been tracked at each frame, whereas trajectory-level evaluations indicate how completely the ground-truth trajectories have been tracked.

In any multi-object tracking evaluation protocol, one usually needs at hand (i) tracker outputs or hypotheses, (ii) a set of ground-truth objects, and (iii) a measure of distance between the estimated and actual state of the target. Let $G_t = \{g_t^j | j = 1, ..., n_t\}$ be the set of ground-truth objects at time $t \in [1, T_{\text{obs}}]$, where T_{obs} is the entire observation interval. The j-th object $g_t^j = (\ell_t^j, \boldsymbol{x}_t^j)$ has a label ℓ_t^j and a location $\boldsymbol{x}_t^j$. Similarly, let $H_t = \{h_t^i | i = 1, ..., m_t\}$ represent the output of the multi-object tracker (or, hypotheses) at time t. The i-th hypothesis $h_t^i = (\hat{\ell}_t^i, \boldsymbol{z}_t^i)$ has a label estimate $\hat{\ell}_t^i$ and a location $\boldsymbol{z}_t^i$. Usually, $\boldsymbol{x}_t^j$ and $\boldsymbol{z}_t^i$ correspond to the points on the ground plane or bounding boxes. The label estimate of a tracker might or might not correspond to an identity of a target.

The distance between a ground-truth object g_t^j and a hypothesis h_t^i, represented by dist_{ij}, is typically defined as the Euclidean distance or the *intersection over union* distance between the bounding boxes, *i.e.*,

$$\text{dist}_{ij} := \begin{cases} \|\boldsymbol{z}_t^i - \boldsymbol{x}_t^j\| & \text{for point targets,} \\ 1 - \frac{|\boldsymbol{z}_t^i \cap \boldsymbol{x}_t^j|}{|\boldsymbol{z}_t^i \cup \boldsymbol{x}_t^j|} & \text{for region targets.} \end{cases} \tag{2.7}$$

Now, we discuss in brief various frame- and trajectory-level evaluation methodologies.

2.5.1 Frame-level evaluation

A widely used protocol for evaluating the performance of multi-object tracking is CLEAR-MOT [75]. It defines two quantities namely multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA). MOTP measures the tracker's ability to localize a target, whereas MOTA measures the number of errors committed by the tracker during tracking.

To define these two terms accurately, we need to define how a correspondence (or, match) between a ground-truth and a tracker hypothesis is established. On the one hand, when the tracker outputs an estimate of the target identity, the matching procedure is straight-forward. Specifically, we consider that g_t^i and h_t^i are matched if $\text{dist}_{ij} < T_{\text{dist}}$. One the other hand, when the label estimate of the hypothesis does not correspond to the identity of a target, a *match* between a ground-truth and a hypothesis is defined as follows. If a target is already being tracked, then g_t^j is said to be matched to h_t^i if $\text{dist}_{ij} < T_{\text{dist}}$. Otherwise, g_t^i is matched to the closest unassigned hypothesis h_t^i if $\text{dist}_{ij} < T_{\text{dist}}$. This is formulated as an *assignment problem* between the unassigned target-hypothesis pairs and is solved by using Hungarian algorithm [57]. We replace j by $\pi(i)$ (or, replace i by $\pi(j)$) to denote that the j-th

target is matched to the i-th hypothesis.

- MOTP is the total error in estimated position for matched target-track pairs over all frames, averaged by the total number of matches made. It is defined as

$$MOTP := \frac{\sum_{t,i} \text{dist}_{i,\pi(i)}}{\sum_t c_t},$$

  where c_t is the number of matches at time t. Ideally, MOTP should be 0. We point out that MOTP is a rough estimate of the performance because it heavily depends not only on the quality of ground-truth annotations but also on the object detector's performance.

- MOTA measures the discrete number of errors made by the tracker. Mathematically, it is defined as

$$\begin{aligned} MOTA &= 1 - \frac{\sum_t (ms_t + fp_t + mm_t)}{\sum_t |G_t|}, \\ &\equiv 1 - \frac{MS + FP + MM}{GT}, \end{aligned}$$

  where $|G_t|$ is the number of ground-truth objects present at time t, $GT = \sum_t |G_t|$, and ms_t, fp_t and mm_t are misses, false positives and mismatches at time t respectively. We write $MS = \sum_t ms_t$, $FP = \sum_t fp_t$ and $MM = \sum_t mm_t$ to denote the time-accumulated errors. These error terms are illustrated in Figure 2.3 and are defined below:

  - **Missed detection (ms_t)** is a g_t^j for which no matching hypothesis h_t^i is detected.
  - **False positive (fp_t)** is a h_t^i for which there is no matching ground-truth g_t^j.
  - **Mismatch (mm_t)**[2] corresponds to h_t^i for which there exists a matching ground-truth g_t^j such that $\ell_t^j \neq \hat{\ell}_t^i$. The mismatch error can be further differentiated into a *switching error* (sw_t) and a *reinitialization error* (re_t). We write $RE = \sum_t re_t$ and $SW = \sum_t sw_t$. A switching error occurs when the tracker starts following another object. 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.

[2] In [24], we call this error as *wrong identification* in the context of sport player recognition.

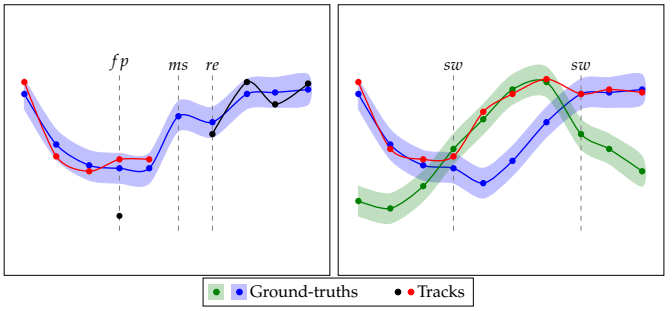


Figure 2.3: **Components of MOTA metric.** fp: false positive, ms: missed detection, re: reinitialization, sw: identity switch. For simplicity, we drop the time subscript t. *Best viewed in color.*

Kasturi *et al.* [78] have defined MOTA slightly differently as

$$MOTA := 1 - \frac{\sum_t c_{ms}(ms_t) + c_{fp}(fp_t) + c_{mm}(1 + mm_t)}{\sum_t |G_t|}, \tag{2.8}$$

where they define the weighting functions as $c_{ms} = c_{fp} = I_d$ (identity function) and $c_{mm} = \log_{10}$. The introduction of 1 in Equation 2.8 is because of the log function.

Ben Shitrit *et al.* [27] observe that the mismatch error is not appropriate to evaluate applications for which preserving identities of the targets is crucial. This is because the mm_t term penalizes only the instantaneous identity switches. To address this, they introduce *global MOTA* (GMOTA) in which the mismatch error is computed along the duration of the track rather than between consecutive time instances. An example is shown in Figure 2.4.

Another popular measure for performance evaluation of multi-target tracking is optimal sub-pattern assignment (OSPA). This OSPA metric[3], proposed by Ristic *et al.* [76], defines the tracking error as

$$D_{p,c}(H_t, G_t) := \left[\frac{1}{q_t} \left(\min_{\pi \in \Pi_{m_t}} \sum_{j=1}^{n_t} \left(d_c(g_t^j, h_t^{\pi(j)}) \right)^p + |n_t - m_t| \cdot c^p \right) \right]^{1/p}, \tag{2.9}$$

where

- $q_t := \max(n_t, m_t)$.
- $d_c(g_t^j, h_t^i) := \min(c, \text{dist}_{ij})$ is the *cut-off distance* with $c > 0$ being the user-defined cut-off parameter;

[3] It is a true metric.

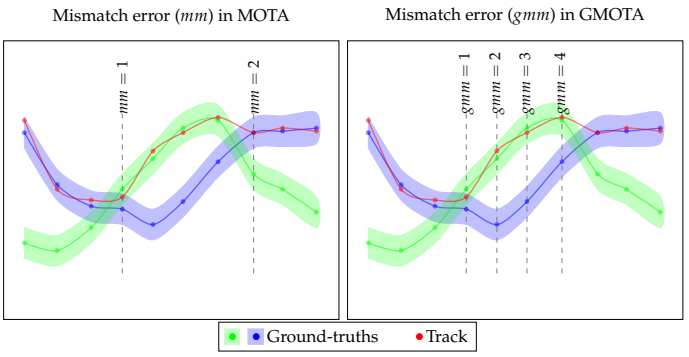


Figure 2.4: **Mismatch errors in MOTA and GMOTA**. Unlike MOTA, in which identity discrepancies are counted between consecutive time instances only, GMOTA counts the identity discrepancies over the duration of the track. Dashed vertical lines represent the counting of these errors.

- $\text{dist}_{ij} := \left[\|\boldsymbol{x}_t^j - \boldsymbol{z}_t^i\|_p + \alpha\delta[\ell_t^j \neq \hat{\ell}_t^i]\right]^{1/p}$ is the *base distance* in which δ is the Kronecker delta, and the first and second terms correspond to the *localization* and *labeling* errors respectively with a balancing term $\alpha \in [0, c]$.
- Π_{m_t} represents the set of all permutations of $\{1, 2, \cdots, m_t\}$;
- $1 \leq p < \infty$ is the OSPA metric order parameter.

The minimization step in Equation 2.9 is performed by using Hungarian algorithm.

Nawaz *et al.* [79] observe that the MOTA is not numerically lower bounded, while the OSPA does not handle change in target sizes. Furthermore, these measures are parameter dependent. To address these limitations, they have proposed multiple extended-target tracking error (METE). Specifically, they define the *accuracy error*, A_t, and *cardinality error*, C_t, as

$$A_t := \min_{\pi \in \Pi_{p_t}} \sum_{j=1}^{q_t} \text{dist}_{\pi(j),j}, \tag{2.10}$$

$$C_t := |m_t - n_t|, \tag{2.11}$$

where $p_t := \max(n_t, m_t)$, $q_t := \min(n_t, m_t)$, and combine them into a parameterless METE as

$$METE_t := \frac{A_t + C_t}{p_t}. \tag{2.12}$$

$METE_t \in [0, 1]$: the lower $METE_t$, the better the tracking result. The computation of accuracy error in Equation 2.10 involves solving an assignment problem and is done by using Hungarian algorithm.

2.5.2 Trajectory-level evaluation

Wu and Nevatia [77] assess the performance on entire trajectories rather than on frame-by-frame assignment. Their definition has been refined by Li *et al.* [58].

- **Mostly Tracked (MT)** means that more than 80% of the ground truth trajectory is tracked.
- **Mostly Lost (ML)** means that more than 80% of the trajectory is lost.
- **Partially Tracked (PT)** means that the trajectory is tracked more than 20% but less than 80%.
- **Fragmentation (Frag)** is the total number of times a ground-truth trajectory is interrupted in tracking. In other words, it is the total number of times a ground-truth trajectory changes its status from 'tracked' to 'not tracked' or vice versa.
- **ID switches (IDS)** is the total number of times a tracked track changes its matched ground truth trajectory.[4]

An illustration of these measures is shown in Figure 2.5.

Additionally, Song *et al.* [74] have proposed two more measures to evaulate the ability of the trackers to recover from the occlusions[5]. They are:

- **RS** is the ratio of tracks which are correctly recovered from short occlusion.
- **RL** is the ratio of tracks which are correctly recovered from long occlusion.

2.6 Conclusion

In this chapter, we introduced some basic elements of graph theory and some algorithms on graph that are used to address multi-object tracking problem.

[4] In [77], IDS is defined as "the number of times two tracks switch their identities".
[5] However, they do not specify how they distinguish short- and long-term occlusions.

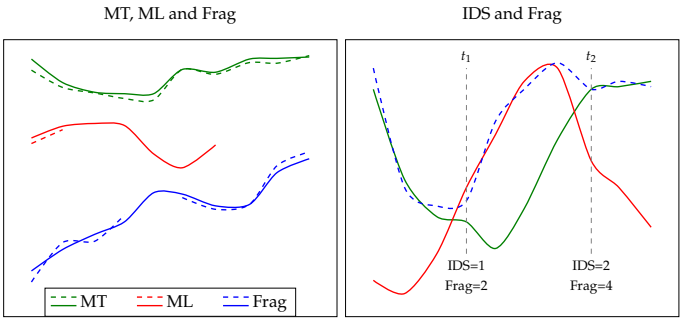


Figure 2.5: **Measures in trajectory-level evaluation.** MT: mostly tracked, ML: mostly lost, Frag: fragmentations and IDS: ID switches. Solid and dashed curves represent the ground-truths and the tracks respectively. **Left**: MT=1, ML=1, Frag=2. **Right**: We can see that each ground-truth trajectory is interrupted twice. Also, the identity of the track is switched twice. Therefore, Frag=4, IDS=2. *Viewed best in color.*

Afterwards, we presented a pipeline underlying detection-based tracking. We provided a brief review of related works for object detection and graph-based MOT. Finally, we described the evaluation methodologies for MOT.

Iterative hypothesis testing for tracking with noisy/missing features

3

In this chapter, we discuss in detail our first contribution, called *iterative hypothesis testing* strategy. We first discuss why many conventional graph-based multi-object trackers cannot handle unreliable and/or sporadic appearance features. Afterwards, we propose our novel iterative hypothesis testing strategy and demonstrate its efficiency and effectiveness on both toy and real-life data.

3.1 Problem statement

In this section, we introduce a generic formulation of multi-object tracking problem, and explain the simplification by previous arts. Afterwards, we point the major shortcoming of these methods in presence of noisy and/or sporadic appearance features. Finally, we propose our novel iterative hypothesis testing strategy.

3.1.1 Multi-object tracking problem formulation

Multi-object tracking (MOT) 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 [80], or to understand sport actions [2], for example.

Due to recent improvements in object detection, many detection-based approaches have been proposed to handle the MOT problem. In such approaches, plausible object locations are first estimated in each individual frame and some features, characterizing the appearances of the detected objects, are extracted. Afterwards, the MOT problem is formulated as the problem of grouping these detections into a minimum number of disjoint trajectories, each trajectory corresponding to a single physical entity. This *data association problem* is usually handled by graph-based solutions. First, a graph is defined to connect a set of nodes that correspond to the detections (or unambiguous association of detections, named *tracklets*). Each edge gets a weight that reflects either distance (or dissimilarity) or similarity in terms of spatio-temporal displacement and/or appearance between the two nodes it connects. Afterwards, multi-object tracking can be naturally formulated its general form as the problem of finding K disjoint sets (or cliques) of nodes such that

- each set contains at most one detection at each time instant,
- the elements of a set are consistent in terms of appearance and spatio-temporal features, and
- the number of sets is limited, based on prior knowledge about the number of targets, or simply by penalizing the increase of K.

Formally, this can be written as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{K} C(T_i) + \lambda g(K), \\ \text{subject to} & T_i \cap T_j = \emptyset, \forall i \neq j, \\ & \cup_{i=1}^{K} T_i = \mathcal{V}, \\ & t_u \neq t_v \forall u, v \in T_i \quad \forall i = 1, \cdots, K; u \neq v, \end{array} \tag{3.1}$$

where $\mathcal{V}$ represents the set of all nodes, $g(K)$ represents the regularization term such that it increases with K, $C(T_i)$ represents the dissimilarity cost within the i-th set T_i, and t_u represents the associated time tag of node u. In Equation (3.1), the first two constraints require that the sets $\{T_i\}_{i=1}^{K}$ define a valid partition, whereas the last constraint requires that T_i cannot have multiple detections from the same time instant. The cost $C(T_i)$ should be defined such that it decreases (increases) when the detections in T_i have a small (large) dissimilarity between them, reflecting (in)consistent associations. The quality of the solution relies on the definition of $C(T_i)$. Ideally, if there are n nodes within T_i, the dissimilarity function should consider all $n(n-1)/2$ [1] of pairs of nodes,

[1] a symmetric (or time-causal) pairwise cost

and associate to each of them a cost that increases with the likelihood that the nodes in a pair correspond to two distinct targets. That is,

$$C(T_i) := \sum_{\substack{u,v \in T_i \\ u \neq v}} w_{uv}, \tag{3.2}$$

where w_{uv} is defined to decrease with the likelihood that the nodes u and v correspond to the same physical object in terms of space, time and/or appearance. Obviously, w_{uv} should increase with the appearance dissimilarity and the spatial distance between nodes u and v. More importantly, its definition should also account

1. for the time elapsed between u and v (because a larger time interval makes it more likely that a target has moved or changed in appearance, hence reducing w_{uv} for a given observed dissimilarity), and

2. for the confidence we have in the observations (an unreliable feature should not lead to definitive conclusion about whether the nodes correspond to the same target or not).

In the following, we refer to these two observations as Observation 1 and Observation 2 respectively.

3.1.2 Previous art simplification and related issues

Given the definition of $C(T_i)$, provided in Equation (3.2), solving Equation (3.1) however rapidly becomes computationally intractable. As investigated by Zamir *et al.* [60], the problem becomes equivalent to the travelling salesman problem (TSP), which is known to be NP-complete. Therefore, most previous works build on the Observation 1 above, namely w_{uv} should only be large for nodes that are close in time, to simplify the problem. Specifically, they ignore dissimilarities between far away nodes and only consider for each node u the cost $w_{uv^\star}$ induced by its immediately subsequent node $v^\star$ in T_i. Formally,

$$C(T_i) := \sum_{u,v^\star \in T_i} w_{uv^\star}, \tag{3.3}$$

where $v^\star := \text{argmin}_{v \in T_i, t_v > t_u} (t_v - t_u)$ is the node in T_i that is temporally closest to u.

Doing so, Equation (3.1) becomes easy to solve, since it basically reduces to finding a set of paths with a minimal cumulative cost. This can be solved by

using a greedy shortest-paths computation [28], or by running the K-shortest paths (KSP) algorithm [26]. Apart from the KSP, several other algorithms such as network flow algorithm [68], robust hierarchical association [56], etc. can be envisioned to estimate the K tracks under this simplification assumption. These approaches have been proven to be effective in a variety of scenarios since the assumption about the prominence of the links connecting close observations is valid in many practical association problems.

This simplification, however, fails to correctly model the tracking problem when the cost w_{uv} of the links that connect nodes that are distant in time becomes important compared to links between subsequent observations. This typically happens when discriminant features are observed with variable level of reliability along the time. In this case, due to the Observation 2 (avoid making a decision based on unreliable features), the weight w_{uv} becomes larger between far away, but reliably observed, nodes than between close nodes with noisy features. Such cases are prevalent in numerous practical scenarios. For example, color 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. Face identity is available only when a person is turning towards the camera properly.

In such time-varying observation process, the task of tracking multiple objects, while taking into account the position and all the available appearance features, cannot be addressed properly with the formulation in Equation (3.3). This is due to the fact that the consistency of a track cannot be measured by the mere accumulation of (dis)similarities between the consecutive nodes in the track, simply because the appearance features might be unreliable or even purely unavailable in some nodes. This major shortcoming of conventional graph-based tracking is illustrated in Figure 3.1.

Figure 3.1(a) depicts the ground-truth trajectories of a red and a green target, as well as the appearance observed in each time frame for each of the target. The color of the node indicates whether the color of the target is available (red or green) or unavailable/unreliable (gray). The problem, defined by Equations (3.1) and (3.2), is depicted in Figure 3.1(b). Edge cost is zero when connecting two nodes with the same color, intermediate (and function of spatio-temporal measurements) when the color information is lacking for one of the nodes, and infinite when the connected nodes have distinct colors. For readability, only the edges connecting the detections that are observed at

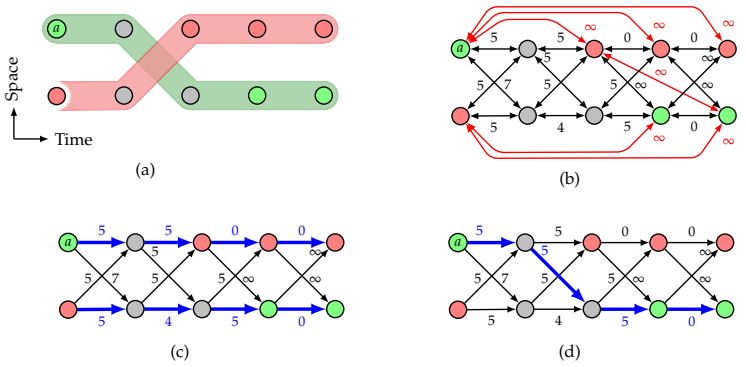


Figure 3.1: **Problem of conventional tracking method in presence of sporadic appearance features. (a)** Detections and trajectories corresponding to two targets (red and green) for 5 consecutive frames are shown. Gray nodes do not have appearance features. For readability, **(b)** only depicts a subset of edges of the fully connected graph: only edges between the nodes in consecutive time instances (in black) and edges with infinite cost (in red). Note that red edges connect nodes even if they are distant in time. **(c)** Results of conventional tracking algorithm with the 'simplification' assumption (see text for explanation). **(d)** Given the appearance of the key-node a, it is possible to simply increase (respectively, decrease) the cost of going through the nodes that are dissimilar (respectively, similar) in the graph irrespective of whether the nodes are temporally close or far. The resulting shortest-path, shown by thick blue arrow, from a is consistent with the appearances. *Best viewed in color.*

consecutive times are depicted (in black), plus the edges with infinite weight (in red). Other w_{uv} are negligible due to the fact that w_{uv} has to decrease as time elapses between u and v (first observation discussed above). The solution to problem (3.1), computed from this graph, using exhaustive search approach, corresponds to the desired tracks and is depicted in Figure 3.1(a). In contrast, making the simplification assumption presented in (3.3) and thus omitting all links between non-consecutive nodes, fails to track the target correctly. This is depicted in Figure 3.1(c), where we observe that a conventional (K-)shortest approach ends up in associating red and green nodes.

In this toy-example, if we are specifically interested in tracking the green target observed in the node a depicted on the top left of Figure 3.1(d), a trivial solution to solve the problem based on a shortest-path computation will consist in increasing/decreasing the cost of an edge when it enters a red/green node, wherever they occur along the track. In that way, the shortest-path is

consistent with the color observations. In this chapter, we propose to extend this trivial single-target tracking solution to a multi-object tracking context, in which no prior knowledge is available about the actual appearance of the targets, and in which the appearance measurements are subject to noise (no guarantee that the measurement in a node defines the actual appearance of the corresponding target).

3.1.3 Contribution

We therefore propose a new paradigm to aggregate detections into objects trajectories. It extends the trivial solution depicted in Figure 3.1(d) by promoting the implementation of an *iterative hypothesis testing* (IHT) strategy to aggregate the detections into short trajectories.

Each iteration of the algorithm works as follows. A node, named key-node (node *a* in Figure 3.1(d)), is selected to define a target appearance hypothesis. Given this hypothesis, a shortest-path algorithm is considered to investigate how to aggregate the key-node with its temporal neighbors in the graph, while promoting the nodes that share its appearance, just as for node *a* in Figure 3.1(d). The process is repeated iteratively, each node possibly becoming a key-node at some step of the algorithm. 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 its neighborhood is only validated when it is 'sufficiently shorter' than alternative paths. In particular, the second shortest-path is also considered within the same observation window. Then, the path reliability is estimated based on the comparison of the costs of the shortest- and the second shortest-paths. The criteria to validate the shortest-path are very strict in the beginning of the iterative process but are then progressively relaxed as the iterations proceed. This progressive relaxation makes the process *greedy* in the sense that most reliable tracklets will be extracted first, independently of the order in which nodes are scheduled as key-nodes.

Furthermore, we adapt the observation window to the size of the key-node (*i.e.*, number of detections already aggregated into the key-node) making the process *multi-scale*. The advantages are two-fold. First, it reduces complexity by aggregating the nodes locally before considering larger observation windows. Second, it gives the opportunity to investigate long time horizons based on more reliable appearance information (since appearance has been accumulated on many frames for large key-nodes), which benefits the tracking accu-

racy.

The proposed approach naturally accounts 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.

The rest of the chapter is organized as follows. Section 3.3 defines the graph terminology. Our iterative hypothesis testing algorithm is described and discussed in Section 3.4. Section 3.6 presents the experimental results and demonstrates the efficiency and effectiveness of our approach both on a synthetic and a real-life basketball dataset.

3.2 Related works

In this section, we review the few works that have been proposed to address the multi-object tracking in presence of noisy and/or sporadic features.

To the best of our knowledge, the only graph-based previous work exploiting sporadically available appearance cues is global appearance constraint (GAC) by Ben Shitrit *et al.* [27]. In this work, the authors assume prior knowledge of 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 possible target appearances. In contrast, to avoid the computational burden associated to the construction of a N-layer graph, and to handle cases for which the possible set of appearances is not a known and finite discrete set, we embed the hypothesis testing within an iterative local aggregation framework. We show in our validation that this results in significant accuracy improvements.

In addition, there is another work that can probably handle the variable confidence in feature measurements even though they do not address it explicitly.

Zamir *et al.* [60] first divide the whole video in fixed sized segments (typically 50 frames long). Afterwards, they adopt similar formalism as in Equation 3.2 on the segments. The solution is obtained by solving a generalized minimum clique problem (GMCP), which greedily extracts tracklets that have the most stable appearance features and the most consistent motion. The same procedure is repeated in a hierarchical manner to generate long trajectories.

Since they work on fixed temporal segments, an error in estimating a tracklet gets propagated in subsequent hierarchies. This is problematic specially because the tracklet extraction step is not 'conservative'. In contrast, our approach works conservatively on longer and longer time windows (see Section 3.4.2). Moreover, they do not handle unreliable appearance features.

3.3 Graph formalism and notations

As an input, the algorithm receives the set of candidate targets detected independently at each time instant, as described in [18]. Apart from the detection time t and the location $\boldsymbol{x}$, the detector computes N appearance features $\boldsymbol{f}_i$ ($1 \leq i \leq N$) 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 $\boldsymbol{d}$ is therefore characterized by the vector

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

where $\mathcal{F} = \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_N\}$ and $\boldsymbol{c} = (c_1, \cdots, c_N)$. The set of detections at a given time t is denoted as $\mathcal{D}^t$. As introduced earlier, the proposed algorithm adopts a graph-based formalism. We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ 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, $\boldsymbol{W} : \mathcal{E} \rightarrow \mathbb{R}_{++}$, 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 3.4.2, including the definition of cost and edges between nodes. Here, we only introduce the associated terminology. Formally, a tracklet v is defined to be collection of chained detections, *i.e.*, $v = \left(\boldsymbol{d}^1, \boldsymbol{d}^2, \cdots, \boldsymbol{d}^{|v|}\right)$, $|v|$ being the length of the tracklet. Notice that the chain is ordered in the sense that the detection times $t_{d^{(i)}}, i \in [1, |v|]$ are such that $t_v^{(s)} = t_{d^1} < t_{d^2} < \cdots < t_{d^{|v|}} = t_v^{(e)}$, with $t_v^{(s)}$ and $t_v^{(e)}$ respectively denoting the starting and ending time of the tracklet.

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 w_{uv} is introduced to denote the linking cost between two nodes $u, v \in \mathcal{V}$. It is formally defined in Section 3.4.1. In short, it typically decreases with the likelihood that the nodes u and v correspond to the same physical target. In addition, we introduce the *inner cost* w_v[2] of a node v to denote the cost of traversing tracklet v from its starting time to its ending time. It is introduced to avoid that long nodes create *short-cuts* in the graph. Since the edges are directed and "time-forwarded" (see Section 3.4.1), the graph $\mathcal{G}$ is directed and acyclic (DAG)[3], and permits only causal traversals. Nevertheless, the graph can be globally reversed in order to allow anti-causal paths for processing purposes. We denote such reversed graph as $\mathcal{G}^-$.

In the sequel, we use two more graph notations. First, $\mathcal{G}_\delta$ represents a windowed-graph formed by selecting in $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ the tracklets $v \in \mathcal{V}$ having at least one extreme time component inside the temporal window δ. The connectivity $\mathcal{E}$ and the weight $\boldsymbol{W}$ are restricted accordingly from these selected tracklets in order to form $\mathcal{E}_\delta$ and $\boldsymbol{W}_\delta$ respectively. Second, in case of incremental tracking, the algorithm incorporates new detections at each time instant t and the graph is continuously incremented with time. We denote the graph at time t by $\mathcal{G}^t$. The corresponding vertices and edges are denoted by $\mathcal{V}^t$ and $\mathcal{E}^t$ respectively.

Figure 3.2 depicts how the tracklets are gathered into a graph in the proposed framework.

3.4 Iterative hypothesis testing algorithm

This section first explains the construction of graph. Afterwards, it presents our proposed algorithm, and outlines its characteristics.

3.4.1 Graph construction

As introduced earlier, our nodes correspond to tracklets. We create a directed edge from u to v only if $0 < t_v^{(s)} - t_u^{(e)} \leq \tau_{\text{max}}$, *i.e.*, node v occurs after u and the time interval is smaller than τ_{max}. The weight of the edge w_{uv} is defined

[2]Note that w_v is not the self-loop of v.

[3]We prefer DAG because the shortest-path can be computed by topological ordering (*e.g.*, depth-first search) which is more efficient than Dijkstra's algorithm. We incorporate the inner cost of a node *on-the-fly* when it is 'being discovered' during the topological ordering. It is made possible by *visitors* in the Boost Graph Library.

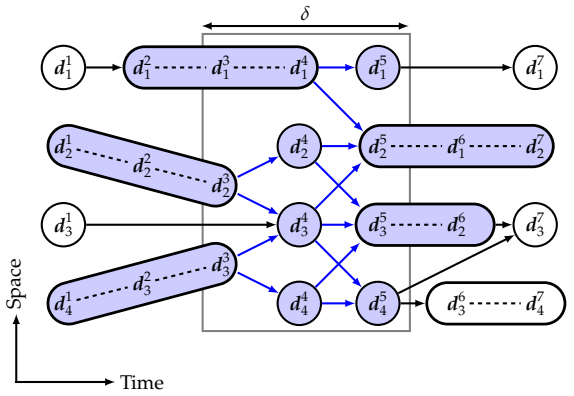


Figure 3.2: **Graph formalism for iterative hypothesis testing.** The k-th detection at time t is denoted by $\boldsymbol{d}_k^t$. Some of them are aggregated into tracklets. Each node corresponds to a tracklet. An edge that connects two nodes u and v has a cost w_{uv}. The windowed-graph $\mathcal{G}_\delta$ is comprised of the nodes and edges (which are drawn in blue) within the observation window δ. *Best viewed in color.*

solely by the spatio-temporal displacement between u and v, *i.e.*,

$$w_{uv} := [1 + \gamma \cdot (t_v^{(s)} - t_u^{(e)} - 1)] g_{\text{sp}}(u, v), \tag{3.4}$$

where the factor $\gamma > 0$ introduces penalty for missed detections, and $g_{\text{sp}}(u, v)$ measures the distance between v and the predicted position of the object corresponding to node u. It is defined as

$$g_{\text{sp}}(u, v) := \left\| \boldsymbol{y}_v^{(s)} - \boldsymbol{y}_u^{(e)} - \dot{\boldsymbol{y}}_u^{(e)} \left(t_v^{(s)} - t_u^{(e)} \right) \right\|_2, \tag{3.5}$$

where the term $\dot{\boldsymbol{y}}_u^{(e)}$ is the velocity, at the end of tracklet u. It is zero for unit length tracklets, and is computed from the last 2 detections of the tracklet otherwise. Since the edges are directed and "time-forwarded", the graph $\mathcal{G}$ is directed and acyclic (DAG).

3.4.2 Iterative hypothesis testing

Our major objective is to design a detections aggregation method that is able to exploit appearance cues that are noisy, or only available sporadically. Therefore, as explained above, 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 neighborhood, 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.

Two subtle mechanisms largely contribute to the success of our approach:

- The first and primary one lies in the conservativeness adopted to turn the path aggregating the key-node with its neighborhood into a single tracklet node for subsequent iterations. Actually, this path is only validated if it is sufficiently better than alternative paths. Importantly, the notion of 'sufficiently good', which is formally defined below, is progressively relaxed along the iterative process. This makes the overall algorithm greedy, in the sense that the less ambiguous paths are validated first, thereby making the solution reasonably independent of the order in which nodes are scheduled as key-node and appearance hypothesis are tested;

- The second one consists in defining the size of the key-node neighborhood proportionally to the length of the key-node. This makes the aggregation multi-scale, which benefits both the accuracy and the computational efficiency, since the individual detections get the opportunity to be aggregated into tracklets before investigating large time horizons, leading to less nodes and more accurate appearance estimation on large time frames.

The global flow of our proposed iterative aggregation algorithm is presented in Algorithm 1.

Given a graph, the algorithm iteratively investigates how a key-node can be aggregated with its neighbours. As controlled by the *dir* flag in Algorithm 1, the direction of investigation changes at each iteration to propagate the appearance hypothesis associated to the key-node both towards the future

Algorithm 1 Iterative Hypothesis Testing

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, number of iterations `MAX_ITER`
Ensure: Updated graph after `MAX_ITER` iterations
Procedure:

```
dir ← +1
for l = 1, ··· , MAX_ITER do
  Initialize: R ← V
  while R ≠ ∅ do
    v_key ← Schedule(R)
    v_agg ← HypothesisTesting(G, v_key, dir, K_1^(l), K_2^(l))
    if v_agg ≠ v_key then
      G ← Simplify(G, v_agg)
    end if
    R ← R \ v_agg
  end while
  (K_1^(l), K_2^(l)) ← Relax(K_1^(l−1), K_2^(l−1))
  dir ← −dir
end for
```

and the past of this key-node, thereby making the global process symmetric with respect to time.

In Algorithm 1, the function `Schedule` selects a node for hypothesis testing that has not yet been scheduled. In this chapter, we select the nodes on decreasing order of their lengths because long nodes are more likely to have accumulated reliable appearance information. Our experimental results have shown that the node scheduling strategy does not affect the performance much.

The remainder of this section details the practical implementation of the core of our proposed multi-scale and iterative aggregation strategy, namely `HypothesisTesting`. It is detailed in Algorithm 2 and involves both (i) the computation of the shortest-path connecting the key-node to its neighborhood, under target appearance hypothesis, and (ii) the validation or rejection of this path as a tracklet for subsequent iterations of Algorithm 1.

Multi-scale tracklet aggregation

Formally, the key-node is denoted v_{key}. It is selected at each iteration among the set of nodes, $\mathcal{R}$, that have not yet been investigated. The aggregation of

the key-node with its neighbors 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. We use δ to denote the observation window interval and $|\delta|$ to denote its size. Hence, $\delta = [t^{(e)}_{v_{\text{key}}}, t^{(e)}_{v_{\text{key}}} + \kappa \cdot |v_{\text{key}}|]$ in the forward mode ($dir = 1$), or $\delta = [t^{(s)}_{v_{\text{key}}} - \kappa \cdot |v_{\text{key}}|, t^{(s)}_{v_{\text{key}}}]$ in the backward mode ($dir = -1$), where $\kappa \in \mathbb{R}_+$ is the window proportionality constant.

Algorithm 2 `HypothesisTesting`

Require: Graph $\mathcal{G}$, key-node v_{key}, direction flag *dir*, validation parameters K_1, K_2

Ensure: Nodes that can be aggregated v_{agg}

Procedure:

$\delta \leftarrow$ Limits of observation window (*see text*)
$\mathcal{G}_\delta \leftarrow$ `GraphHypothesis`$(\mathcal{G}, \delta, v_{\text{key}})$
$(S_b, S_{sb}) \leftarrow$ Shortest- and second shortest-paths from v_{key}
if `isUnambiguous`(S_b, S_{sb}) **then** *# Refer to Figure 3.3 for illustration*
  $\mathcal{G}_\delta^- \leftarrow$ `ReverseDirection`$(\mathcal{G}_\delta)$
  $(S_{b'}, S_{sb'}) \leftarrow$ Shortest- and second shortest-paths from v_b
  if `isUnambiguous`$(S_{b'}, S_{\text{sb}'})$ **then**
    $v_{agg} \leftarrow S_b$
  end if
else
  $v_{agg} \leftarrow v_{key}$
end if
return v_{agg}

`isUnambiguous`(S_b, S_{sb})
  return $\text{cost}(S_b) < K_1 \cdot |\delta|$ and $\text{cost}(S_b)/\text{cost}(S_{sb}) < K_2$

Given the key-node v_{key} and the observation window δ, we define a graph $\mathcal{G}_\delta$ to investigate how the key-node can be aggregated with its neighbors to define an appearance-consistent path under the assumption that the target appearance is defined by the key-node appearance. In Algorithm 2, the function returning $\mathcal{G}_\delta$ 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 $\mathcal{G}$, by cutting $\mathcal{G}$ 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 w_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.

Given the appearance features of individual detections, the inference of the tracklet appearance features directly depends on the characteristics of the features observation process. If, for example, the observation process is affected by outliers, a RANSAC [81] 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. Here, we use a weighted average for the tracklet appearance as an example of possible practical implementation. Then, the average i^{th} feature of a node v is computed as

$$\overline{f}_i^{(v)} = \frac{1}{C_i} \sum_{t=1}^{|v|} c_{i,t}^{(v)} f_{i,t}^{(v)}, \tag{3.6}$$

where $C_i = \sum_{t=1}^{|v|} c_{i,t}^{(v)}$. In particular, $\overline{f}_i^{(\text{key})}$ 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 respect to the key-node appearance. We define

$$D(v) = \sum_{i=1}^{N} \underbrace{\left[\alpha_{\text{key}}\alpha_i\lambda_i \left\|\overline{f}_i^{(\text{key})} - \overline{f}_i^{(v)}\right\|_1 + (1 - \alpha_{\text{key}}\alpha_i) w_i^{(\text{fix})}\right]}_{w_i^{(v)}}, \tag{3.7}$$

where λ_i weights the contribution of the i-th feature. The parameter α_i is introduced to give less weight to unreliable appearance features. The definition of α_i depends on confidence values $c_{i,t}^{(v)}$ of individual detection. However, the precise definition depends on the application at hand. We show in validation section a possible definition of α_i.

From Equation 3.7, when $\alpha_{\text{key}}\alpha_i \to 1$, $w_i^{(v)} \to \lambda_i \left\|\overline{f}_i^{(\text{key})} - \overline{f}_i^{(v)}\right\|_1$ and when $\alpha_{\text{key}}\alpha_i \to 0$, $w_i^{(v)} \to w_i^{(\text{fix})}$. The term $w_i^{(\text{fix})}$ is introduced so that a node that definitely looks similar to the key-node ($D(v) \approx 0$) is favored compared to a node for which no appearance features is available $\left(D(v) \approx \sum_i w_i^{(\text{fix})}\right)$. It corresponds to the noise level, affecting the feature. Empirically, we set $w_i^{(\text{fix})} = 5$ for all $1 \leq i \leq N$.

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 tracklet validation

Having the cost of edges defined to take the displacement as well as the appearance into consideration, the shortest-path S_b, which connects the key-node to the extremity of the observation window, reasonably corresponds to a single physical object (same appearance, and coherent motion) and could thus be aggregated into a single node.

However, 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 alternative paths. Figure 3.3 illustrates this process.

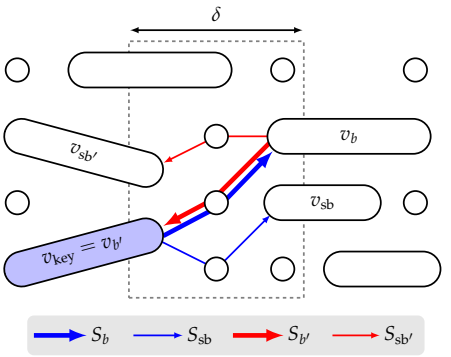


Figure 3.3: **Illustration of the validation of the hypothesis**. Within the window, the best (thick arrow) and the second best (thin arrow) paths (denoted by S_b and S_{sb} respectively) are searched. Blue and red arrows represent forward and backward directions respectively. *Best viewed in color.*

It runs in two steps. In the first step, the shortest S_b and the second shortest S_{sb} paths are considered. 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 unambiguous only if two conditions are met: (i) $\text{cost}(S_b) < K_1 \cdot |\delta|$, and (ii) $\text{cost}(S_b)/\text{cost}(S_{sb}) < K_2$.

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 graph for subsequent iterations of the IHT. This procedure is called `Simplify` in the Algorithm 1. It updates the appearance features of the node as in Equation 3.6 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.

It should be noted that IHT makes the implicit assumption that the trajectory associated to a target is unique in the graph. It means that, for each target, there are no multiple paths with similar costs in the graph. In practice, however, different paths starting from the same key-node, but ending in a different node at the extremity of the observation window, might be quite similar in the sense that they share identical nodes or detections that are very close to each other. Those similar paths are not of interest to the testing phase because they basically define a similar trajectory than the shortest-path. Hence, we consider as the second path a path that ends in a different node than the shortest path, and that is sufficiently different (*i.e.*, does not share many nodes) from the shortest-path.

Choosing small (large) values of K_1 and K_2 makes the constraint more (less) conservative. In the beginning of the algorithm, we start with small values of K_1 and K_2. As the iteration proceeds, we progressively relax the validation criteria. This makes the overall IHT algorithm greedy, in the sense that the less ambiguous paths are validated first, thereby making the solution reasonably independent of the order in which nodes are scheduled as key-node and appearance hypotheses are tested. This progressive relaxation of the key-node path validation constraint is denoted by the function `Relax` in Algorithm 1. An example of relaxation scheme is described in results section.

3.5 From off-line to incremental IHT

Because we iterate over the nodes, our IHT naturally extends to the incremental scenarios in which the detections arrive sequentially over time. Compared

to the off-line approach, there are however few subtleties. They are:

- **Incrementing the graph:** At time $t = 1$, the graph is just a set of detections at that instant, *i.e.*, $\mathcal{G}^1 = (\mathcal{D}^1, \emptyset)$. At time $t > 1$, the graph is obtained by adding new detections $\mathcal{D}^t$ to the so-called previous graph $\mathcal{G}^{t-1}$, resulting from earlier steps of the algorithm, up to time $t-1$. All nodes ending later than time $t - \tau_{\max}$ are linked to all the current detections. The weight of each edge is computed as in Equation 3.4.

- **Scheduling of the nodes:** Unlike the off-line approach, we schedule the 'recent' nodes first. This is done to prevent the fast growth of the graph at each time. Specifically, we schedule the nodes in decreasing order of $|v| / \max\{1, t - t_v^{(e)}\}$ so that the 'recent' and 'sufficiently long' nodes are selected first.

- **Relaxing the validation criteria:** We maintain a 'sliding window' $[t - \delta_{\text{slide}}, t]$ where δ_{slide} is the length of the sliding window. Inside (respectively, outside) the sliding window, we impose conservative (respectively, relaxed) criteria for K_1 and K_2. We use $\delta_{\text{slide}} = 200$ frames.

3.6 Evaluation

We test our proposed IHT algorithm on a toy example and also on the real-life APIDIS [6] and PETS [82] datasets. We compare IHT with the relavant works pointed in Sections 3.1 and 3.2, namely KSP [26], GAC [27] and GMCP [60]. The toy example helps us to highlight the benefits of our progressive aggregation paradigm, while the experiments on real-life examples demonstrate the practical relevance of our approach.

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

In the remainder of the section, we first compare IHT specifically with KSP and GAC on a toy example. KSP serves as a 'representative' method that exploits appearance features in a uniform manner. Afterwards, we discuss the results and compare with other approaches on APIDIS dataset.

3.6.1 Toy example

We consider 3 targets whose ground-truth locations $\{y_1, y_2, y_3\}$ at time instances $k \in \{0, 1, \cdots, 10\}$ are obtained by

$$y_1 := 50 \sin\left(\frac{2\pi k}{10}\right),$$
$$y_2 := 50 \cos\left(\frac{2\pi k}{10}\right),$$
$$y_3 := -20 - 50 \sin\left(\frac{2\pi k}{8}\right). \tag{3.8}$$

The appearance feature of the i-th target, denoted as f_i, is modeled by a 2 state automata as shown in Figure 3.4. The appearances of the state 1 and

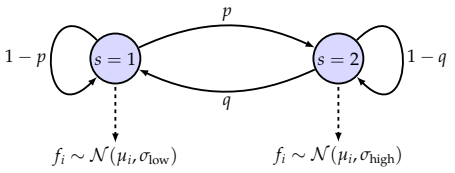


Figure 3.4: **2 state automata for modelling the appearance of the i-th target**. When the automata is at state $s = 1$ (respectively, $s = 2$), the appearance f_i follows a normal distribution with mean μ_i and standard deviation σ_{low} (respectively, σ_{high}). *Best viewed in color.*

2 are modelled as $\mathcal{N}(\mu_i, \sigma_{\text{low}})$ and $\mathcal{N}(\mu_i, \sigma_{\text{high}})$ respectively. We use $\mu_i \in \{0, 120, 240\}$ and $\sigma_{\text{low}} = 10$ and $\sigma_{\text{high}} = 100$. In order to simplify the transition probability matrix, we fix $q = 0.5$ and vary p.

The detections and ground-truth trajectories are shown in Figure 3.5.

Implementation details:

We create a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ where

- $\mathcal{V}$ is the set nodes with i-th node corresponding to the i-th detection. The i-th node is characterized by the time instant t_i, position y_i, appearance f_i (and, optionally feature confidence $c_i \in [0, 1]$).
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ defines the connectivity between the detections. We create edges between the nodes that occur at consecutive time instants, *i.e.*, $\mathcal{E} := \{(i, j) | i \in \mathcal{V}, j \in \mathcal{V}, t_j - t_i = 1\}$.
- $\boldsymbol{W}$ assigns a cost w_{ij} to each edge $(i, j) \in \mathcal{E}$. We write the cost w_{ij} as $w_{ij} := w_{ij}^{(s)} + w_{ij}^{(a)}$, where $w_{ij}^{(s)}$ and $w_{ij}^{(a)}$ are the spatio-temporal and

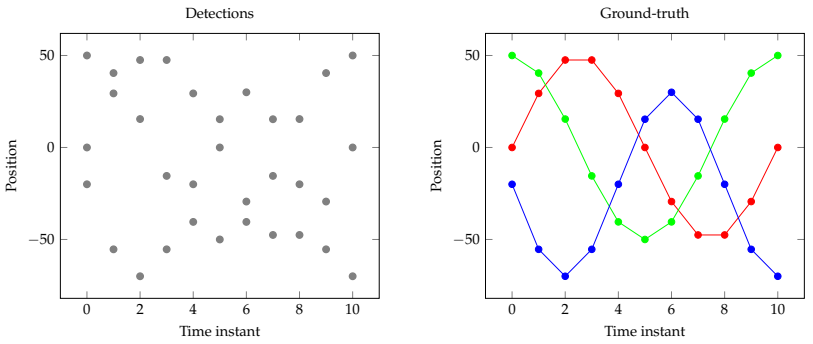


Figure 3.5: **Detections and ground-truth trajectories of the toy example**. *Best viewed in color.*

appearance costs respectively. The spatio-temporal cost is defined as $w_{ij}^{(s)} := \|y_j - y_i\|_2$. The definition appearance cost differs from one algorithm to another. First of all, given two appearance features f_i and f_j, the appearance dissimilarity d_{ij} is computed as

$$d_{ij} := 1 - \left| \cos\left(\frac{\pi(f_j - f_i)}{180} \right) \right|.$$

Then, we define the appearance cost $w_{ij}^{(a)}$ as in Equation (3.7):

Algorithm	**Reference appearance**	**Appearance cost**, $w_{ij}^{(a)}$
KSP	None	$c_i c_j d_{ij} + (1 - c_i c_j) w^{(\text{fix})}$
GAC	l-th global appearance, f_l	$c_i d_{il} + (1 - c_i) w^{(\text{fix})}$
IHT	l-th key-node appearance, f_l	$c_i c_l d_{il} + (1 - c_i c_l) w^{(\text{fix})}$

where $w^{(\text{fix})} \geq 0$ is a fixed cost, introduced to associate a fixed cost to nodes for which the appearance is unknown or unreliable. We use $w^{(\text{fix})} = 10$ in our experiments.

– For GAC and KSP, we connect the *source* node to the nodes that occur at time $k = 0$ and the nodes that occur at time $k = 10$ to the *terminal* node. The costs of these edges are set to zero. In practice, the set of global appearances considered by GAC are either known *a priori* (*e.g.*, provided by oracle), or have been estimated from the measurements (*e.g.*, using k-means with 3 clusters in our toy-example case).

For GMCP, we compute the cost of a candidate solution T_i as

$$C(T_i) := \sum_{u \in T_i} \Big[\sum_{\substack{v \in T_i \\ v \neq u}} w_{uv}^{(a)} + w_{uv^\star}^{(s)} \Big],$$

where

$$v^{\star} = \underset{\substack{v \in T_i \\ v \neq u \\ t_v > t_u}}{\arg\min} (t_v - t_u).$$

As told in Section 3.1.2, this GMCP formulation corresponds to the traveling salesman problem (TSP), which is known to be NP-complete. Therefore, it has to be solved using approximated greedy solution. We have followed the authors in [60], and have adopted the popular opt-2 solution [83]. The solution to the GMCP problem is obtained greedily as follows. We initialize the solution by computing the shortest-path from the source to terminal node on the graph, used in KSP. This is referred to as *current solution*. To find the *neighbor solutions*, we (i) traverse through the current solution, (ii) replace a node along the current solution by other nodes that co-exist with it, and (iii) compute the corresponding cost. Afterwards, we choose the path among the neighbor solutions that has the least cost. If this cost is smaller than the cost of current solution, we replace the current solution by it.

Results: In our simulations, we consider two experimental set-ups. In the first set-up, we consider that we have no knowledge about the reliability of the measurement process (*e.g.*, about the state of the automata). Hence, we set $c_i = 1$ for all detections. In the second set-up, the state of the automata is assumed to be known and we set c_i as

$$c_i = \begin{cases} 0.8 & \text{if } s = 1, \\ 0.1 & \text{if } s = 2. \end{cases}$$

We vary the transition probability p from 0 to 0.9 with an increment of 0.1. For each value of p, we generate 100 realizations of the target appearances and apply IHT, GAC and KSP algorithms to compute tracks for each of them. Afterwards, we compute MOTA score for each algorithm. The results are shown in Figure 3.6.

From the figure, we see that taking the confidence of the feature measurement into account indeed helps to disambiguate the data association (as reflected in the MOTA scores). When we do not take into account the confidence information, all IHT, GAC and KSP perform similarly, with IHT performing slightly better than the other two algorithms. The performance improves significantly when the confidence measure is incorporated. Surprisingly, the GMCP performs the worst even though its problem formulation is close to ideal. Such an inferior performance of GMCP can be accredited to the fact

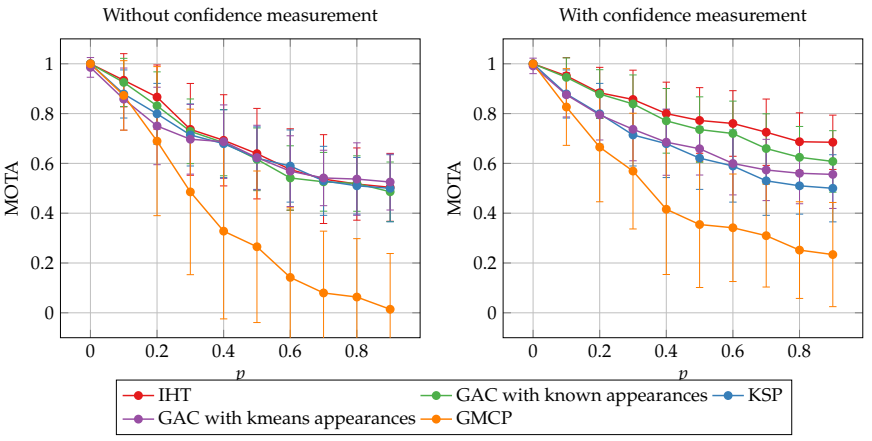


Figure 3.6: **Performance of IHT, GAC, KSP and GMCP on toy example** with and without taking the confidence of feature measurement information. *Best viewed in color.*

that each 'track' is extracted greedily and locally from the set of nodes. Unlike GAC, there is no notion of global solution. Unlike IHT, it does not validate the goodness of the extracted track.

It is worth noting that the performance of GAC is strongly dependent on the prior knowledge of the 3 global appearances. The performance in Figure 3.6 indeed appears to degrade significantly when the 3 appearances are estimated from the measurements (based on k-means clustering, with k=3).

Our IHT algorithm has two distinct steps: (i) node scheduling, and (ii) hypothesis validation. To study the importance of these steps, we envision the following set-ups. We schedule the nodes either at *random* or in *descending* order of appearance confidence. In addition, we validate the shortest-path either *conservatively*, as described by Figure 3.3, or *always*, meaning that we systematically define a new tracklet based on the shortest-path. The results are presented in Figure 3.7.

The results show that the node scheduling has negligible impact on the performance of the IHT algorithm. On the other hand, the conservative validation of the shortest path has a drastic influence on the performance of IHT. By comparing Figure 3.7 with Figure 3.6, we observe that IHT performs worse than KSP when we validate the shortest path immediately. This is not surprising because IHT investigates greedily on a local section of the graph whereas KSP works globally on the whole graph.

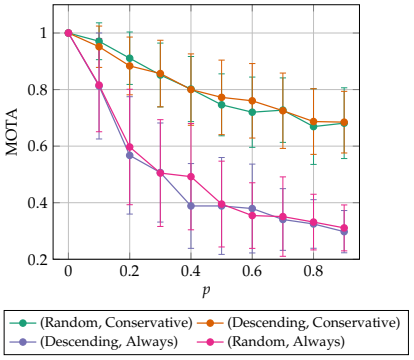


Figure 3.7: **Effect of scheduling of nodes and validation strategy on the performance of IHT**. *Best viewed in color.*

3.6.2 Results for offline IHT

We report the performance of off-line IHT on 1 minute video (1500 frames at 25 frames per second) of APIDIS dataset. The APIDIS dataset and the generation of detections is described in Appendex C. In short, the candidate detections are computed independently at each time instant based on a ground occupancy map, as described in [18]. For each detection, the jersey color and its digit are computed to define the appearance features.

The parameter α_i, introduced in Equation 3.7, is defined for APIDIS dataset as follows

$$\alpha_i = \begin{cases} 0 & \text{if } C_i \leq C_{\text{min}}, \\ 1 & \text{if } C_i \geq C_{\text{max}}, \\ \frac{C_i - C_{\text{min}}}{C_{\text{max}} - C_{\text{min}}} & \text{otherwise.} \end{cases} \tag{3.9}$$

where C_{min} and C_{max} are the limits to define if the feature is considered reliable or not. Some parameters of our algorithm are chosen as

Parameter	Description	Value
τ_{max}	Connection horizon for new detections	120
γ	Missed detection coefficient	3
κ	Window proportionality constant	5
$(C_{\text{min}}, C_{\text{max}})$	Thresholds for feature reliability	(20,100)
`MAX_ITER`	Maximum number of iterations	50

Figure 3.8 compares the performance obtained by the proposed algorithm when different sets of appearance features are exploited. The results are obtained by enabling (or disabling) certain features in the algorithm and running

on the 1 minute long video sequence. There are all together GT=544 ground truth positions.

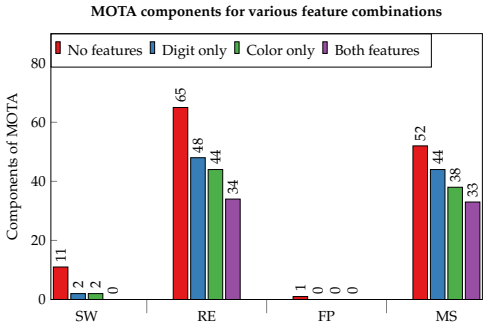


Figure 3.8: **Components of MOTA metric for various feature combinations on a 1 minute long video for off-line IHT.** The MOTA scores for all 4 cases are as follows: (i) no feature: 76.71%, (ii) digit feature only: 83.03%, (iii) color feature only: 84.84%, and (iv) both features: 87.91% . *Best viewed in color.*

As we can see, the switches and re-initializations are reduced substantially when the appearance features are used. It can also be seen that the digit features, even though they are highly sparse, can disambiguate some tracks. However, the improvements are not as ample as those from the color feature. 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).

In order to study the effect of the progressive relaxation of the validation criteria, we envision the following experimental set-ups. First, we fix the values for K_1 and K_2 such that the validation criteria are 'most conservative' (*i.e.*, small values of K_1 and K_2) and 'least conservative' (*i.e.*, large values of K_1 and K_2). Specifically, we set $(K_1, K_2) = (5, 1/4)$ for most conservative and $(K_1, K_2) = (30, 1/1.1)$ for least conservative criteria. For progressive relaxation, we linearly increase K_1 from 5 to 30 in 50 iterations and increase K_2 linearly from $1/4$ to $1/1.1$ in 20 iterations and then fix $K_2 = 1/1.1$ for the rest 30 iterations. We increase K_2 faster than K_1 because the primary condition to validate a path is its low cost. The results are depicted in Figure 3.9.

From Figure 3.9, we see that relaxing the validation criteria indeed helps to improve the tracking results in the sense that it avoids identity switches (just as for a highly conservative criteria), while maintaining re-initialization and misses at the level obtained with a less conservative criteria. Hence, it keeps

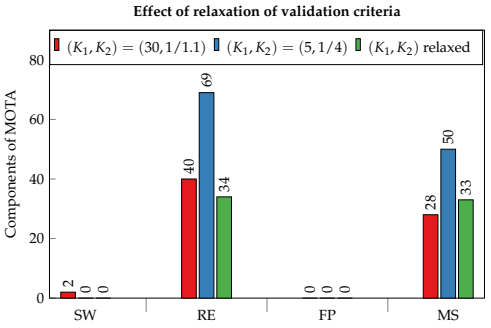


Figure 3.9: **Components of MOTA metric for different values of (K_1, K_2) on a 1 minute long video for off-line IHT.** Red, blue and green bars correspond to the 'least conservative', 'most conservative' and 'progressively relaxed' validation criteria respectively. The MOTA scores for all 3 cases are as follows: (i) least conservative: 87.36%, (ii) most conservative: 78.52%, and (iii) progressively relaxed: 87.91%. *Best viewed in color.*

the best out of the two criteria.

In Table 3.1, we compare IHT with GAC on a 1 minute video of APIDIS[4]. The results for GAC and KSP have been kindly provided by the authors of GAC. Regarding the comparison, note that KSP does not use appearance feature, whereas GAC uses digit and color features. Hence, we compare KSP with IHT that does not use appearance feature and GAC with IHT that uses both color and digit features.

Method	**MOTA %**	**MOTP**	**SW**
KSP [26]	72.91	14.06	108
GAC [27]	73.07	14.06	110
IHT (no appearance)	76.71	10.39	11
IHT (color+digit)	**86.19**	**10.37**	**0**

Table 3.1: Results on 1 minute video of APIDIS dataset. We compare KSP with IHT (no appearance) and GAC with IHT(color+digit).

From Table 3.1, we can see that the proposed method outperforms KSP and GAC. We explain the benefit observed in this latter case by the progressive and conservative nature of our proposed aggregation process.

[4] [27] has reported results on 1 minute video of APIDIS.

3.6.3 Results for incremental IHT

In this section, we present tracking results on both 1 minute and 15 minutes long video of APIDIS, and PETS dataset.

Results on APIDIS dataset

- **Tracking results for 1 minute long video:** The tracking results for the incremental IHT for various feature combinations is depicted in Figure 3.10. From Figure 3.10, we can see that the errors get significantly reduced when the appearances features, even if they are sporadic, are incorporated.

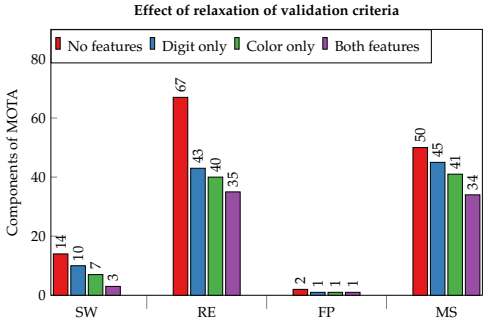


Figure 3.10: **Components of MOTA metric for various feature combinations on a 1 minute long video for incremental IHT.** The MOTA scores for all 4 cases are as follows: (i) no feature: 75.99%, (ii) digit feature only: 82.13%, (iii) color feature only: 83.94%, and (iv) both features: 86.82% . *Best viewed in color.*

We compare the performance of incremental IHT with off-line version in Table 3.2.

From Table 3.2, we can see that the incremental IHT performs almost similar to off-line approach. However, switching error (SW) is significantly better in off-line case.

- **Tracking results for 15 minute long video:**

  In case of incremental IHT, we can afford to run the algorithm for the 15 minutes of the video. Figure 3.11 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

	Off-line IHT		Incremental IHT	
	No features	**Both features**	**No features**	**Both features**
FP	1	0	2	1
MS	52	33	50	34
RE	65	34	67	35
SW	11	0	14	3
MOTP	10.39	10.37	10.64	10.46
MOTA %	76.71	87.91	75.99	86.82

Table 3.2: Comparison of off-line and incremental IHT on 1 minute video of APIDIS dataset.

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 the digit feature only, it can be seen that digit feature, even though it is highly sparse, can disambiguate some tracks. However, the improvements are not as ample as those from the color feature. It can be justified due to the fact that color feature is available more often than the digit feature. 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).

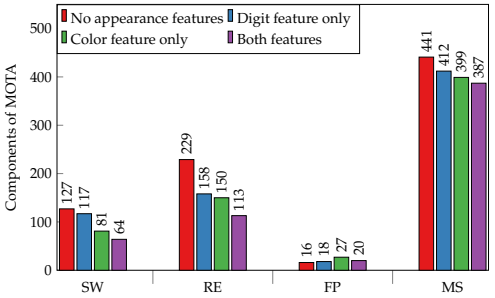


Figure 3.11: **Components of MOTA metric for different cases on a 15 minutes long video.** Indeed, exploitation of color and digit features help to reduce the errors. The MOTA scores for all 4 cases are as follows: (i) No appearance: 89.1%, (ii) Digit only: 90.5%, (iii) Color only: 91.2%, and (iv) Both features: 92.2%. *Best viewed in color.*

Results on PETS dataset

We apply on camera view 1 of the PETS-09 S2.L1 dataset [82] . First, targets are detected by using the deformable parts model [22]. Then, 8-bin histograms are computed on RGB channels separately, which are then stacked to obtain a 24-bin feature vector. The feature is discarded if the overlap between the bounding boxes exceeds 20%. This makes the resulting feature vector sporadic. The results are shown in Table 3.3. The results for discrete-continuous optimization [84], continuous energy [85] are extracted from [84], whereas the results for GAC [27] and KSP [26] are extracted from [27].

Method	**MOTA %**	**MOTP %**
KSP [26]	80.00	58.00
Continuous energy [85]	81.84	73.93
Tracking-by-Detection [86]	82.00	56.00
Discrete-continuous optimization [84]	**89.30**	56.40
GAC [27]	81.46	58.38
IHT (no appearance)	81.18	74.54
IHT (with color histogram)	83.10	**74.56**

Table 3.3: Tracking results on PETS dataset. To compare MOTP with other methods, it has been redefined so that it corresponds to the total overlap between the bounding boxes of the ground-truth and the track, divided by the total number of matches made.

We can see that our incremental IHT compares favorably with several methods. It is worth mentioning that it takes only 42 seconds in MATLAB to estimate the feature and process all 795 frames.

3.6.4 Computational complexity and advantages of multi-scale processing

To study the effect of multi-scale nature of our algorithm, we conduct two experimental set-ups on 15 minutes video of APIDIS dataset. First, we note the number of nodes in the graph and the (maximum) time taken by the hypothesis testing step of IHT at each iteration. Second, we estimate the time taken by IHT with fixed (specifically, $|\delta| \in \{10, 50, 500\}$) and adaptive (*i.e.*, $|\delta| = \kappa \cdot |v_{\text{key}}|$) observation window sizes. The results are shown in Figure 3.12. From the figure, we observe that the worst case complexity is found

to be $\mathcal{O}(n^{1.475})$. Besides, the multi-scale nature of the algorithm not only reduces the computational time but also improves the tracking accuracy.

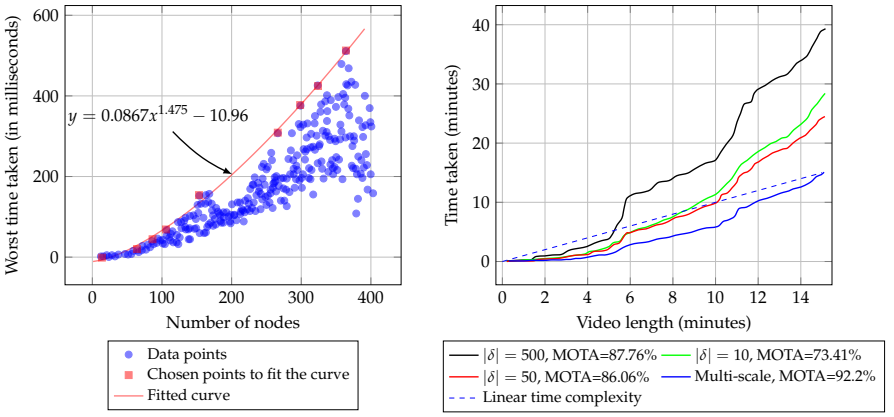


Figure 3.12: **Computational complexity of IHT. (Left)**: The complexity of IHT is estimated to be $\mathcal{O}(n^{1.475})$. **(Right)**: Running time for multi-scale as well as fixed-window approaches. *Best viewed in color.*

3.6.5 Effect of parameters

In this section, we present the performance of incremental IHT algorithm (in terms of MOTA components) with respect to some key parameters. They are:

Parameter	Description
$\tau_{\max}$	Connection horizon for new detections (Section 3.4.1)
γ	Missed detection coefficient (Section 3.4.1)
κ	Window proportionality constant (Section 3.4.2)
$(C_{\min}, C_{\max})$	Lower and upper thresholds to compute the reliability (Section 3.6.2)
(K_1, K_2)	Factors to validate the shortest-path (Section 3.4.2)

The results for the working point ($\tau_{\max} = 120, \gamma = 3, \kappa = 5, C_{\min} = 20, C_{\max} = 100$) are ($FP = 20, MS = 387, RE = 113, SW = 64, MOTA = 92.2\%$) for incremental IHT on APIDIS dataset. Now, we present the performance of the IHT algorithm (in terms of MOTA components) with respect to these parameters individually. For this, only one parameter is changed at a time and all other parameters are fixed at their quiescent values. The results are shown in Table 3.4.

	τ_{max}				γ				
	30	**60**	**120**	**240**	**1**	**2**	**3**	**4**	**5**
FP	7	24	20	22	34	20	20	20	11
MS	426	410	387	380	429	382	387	382	399
RE	162	137	113	111	125	116	113	118	118
SW	74	64	64	76	72	83	64	70	71

	κ				(C_{min}, C_{max})				
	1	**3**	**5**	**7**	**(20,70)**	**(20,100)**	**(20,50)**	**(5,50)**	**(20,30)**
FP	14	19	20	33	19	20	19	22	19
MS	417	408	387	380	400	387	402	390	402
RE	120	126	113	110	127	113	116	133	117
SW	62	64	64	73	65	64	70	67	75

Table 3.4: Effect of $\tau_{max}, \gamma, \kappa$ and (C_{min}, C_{max}) on 15 minutes video of APIDIS dataset.

In addition, we also study how various values of K_1 and K_2 affect the performance of IHT. For this, different values of K_1 and K_2 are chosen and are kept fixed throughout the iteration. Recall that K_1 constrains the cost of the shortest-path, whereas K_2 constrains the ratio of costs between the shortest-path and the second shortest-path. Decreasing K_1 and/or K_2 will therefore make the IHT conservative, leading to less false positives and less identity switches. On the flip side, misses and re-initialization errors will increase. Table 3.5 presents the resulting performance of the IHT .

	K_1				K_2				**Relaxed**
	2	**5**	**15**	**30**	**1/1.5**	**1/2**	**1/3**	**1/5**	
FP	14	18	24	30	70	32	18	17	20
MS	417	401	372	350	363	377	401	445	387
RE	133	120	103	97	101	107	120	149	113
SW	41	60	69	78	97	81	60	58	64

Table 3.5: Effect of K_1 and K_2 on 15 minutes video of APIDIS dataset. For comparison, we also present the results for the case in which K_1 and K_2 are progressively relaxed.

From Table 3.5, we can observe that both K_1 and K_2 affect the performance. As expected, low (respectively, high) values of K_1 and K_2 result in less (respectively, more) false positives and switching errors but more (respectively,

less) misses and re-initializations, allowing us to trade-off the errors. We propose two alternatives to choose these parameters depending on the problem at hand. First, if the objective is to have conservative tracking in which the resulting tracklets are reliable, it is suggested to choose low values of K_1 and K_2. This option is suitable if one envisions to process these trackets in the next step so as to stitch them into long trajectories. Second, we propose to start with small values of K_1 and K_2 and then progressively relax as the iteration proceeds. This option is suitable when long (and probably erroneous) trajectories are preferred.

3.6.6 Qualitative results

Some sample frames with tracking results[5] for PETS dataset are presented in Figure 3.13. In each frame, a target has been assigned a track number.

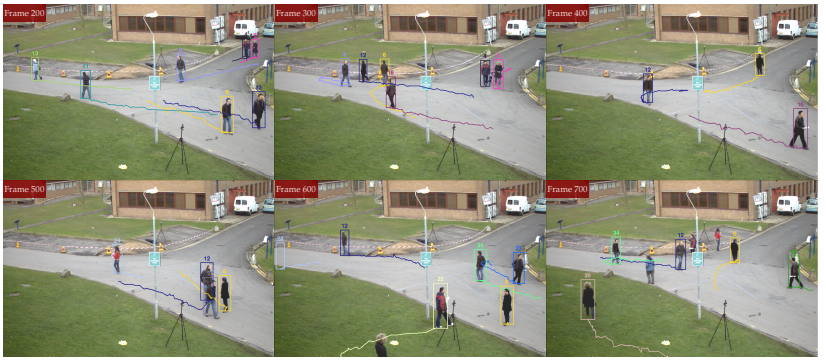


Figure 3.13: **Sample frames on the PETS dataset for IHT.** For visual clarity, a tail of 50 frames is also shown. *Best viewed in color.*

Figure 3.14 presents the sample frames for APIDIS dataset. Results are shown on virtual camera view, which is obtained by stitching camera views 1 and 6.

We observe following failure cases:

- **Duplicate detections**: Because of the imperfection of the detector, a target might be detected multiple times. Such duplicate detections can be observed when there is a clutter. Unlike typical false positive detections, they are coherent and, hence, persist longer in time. In such situation,

[5] A demo video is available at http://sites.uclouvain.be/ispgroup/index.php/Research/MultiObjectTracking.

Figure 3.14: **Sample results on a virtual camera of the APIDIS dataset**. For visual purpose, a track segment from past 75 frames has also been drawn. *Best viewed in color.*

a new track is assigned to them. A typical example is shown in Figure 3.15. The problem of duplicate detections can be mitigated by introducing backward edges in the graph. However, it no longer makes the graph DAG.

- **Track reinitialization and switch**: When the targets are in clutter, the appearance features become noisy. Moreover, there are often some misdetections as well as false detections. In such situations, the tracks are sometimes not well resolved, thereby, resulting in the reinitialization and sometimes switching of the track identities. An example is shown in Figure 3.16.

Figure 3.15: **Duplicate detections**. Tracks 2 and 3 approach close in frame 221. Notice that 2 is assigned to the yellow player. In frame 222, a new track 13 appears for the same yellow player. *Best viewed in color.*

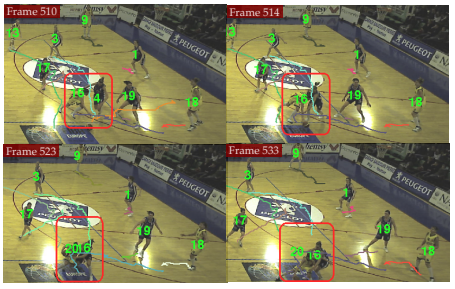


Figure 3.16: **Identity switch and re-initialization errors**. Tracks 16 and 4 come close to each other in frame 510. Note that track numbers 16 and 4 are assigned to the yellow and the blue players respectively. In frame 514, track 4 vanishes (*miss*). Track number 20 is assigned for the yellow player in frame 523 (*reinitialization*). Moreover, track 16 (which was assigned to the yellow player) is now assigned to the blue player (*switch*). Finally, tracks 16 and 20 continue in frame 533. *Best viewed in color.*

3.7 Conclusion and future perspectives

This chapter proposed a novel framework for matching of detections while exploiting unreliable and/or sporadic 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.

From Tracking to Recognition

4

In previous chapter, we have noticed that sporadically available features can be exploited to track multiple targets in videos. It should be noted that the 'conservativeness' of the iterative hypothesis testing (IHT) might prevent the association of tracklets into a long trajectory for each target.

In this chapter, we describe how we can build on such conservative but reliable tracklets to estimate their identities, thereby grouping them into consistent trajectories. Doing so, we solve both recognition as well as track-completion problems. For this purpose, we exploit the tracklet appearances to drive the *belief propagation* in a graph. In short, we first estimate the initial identity distribution to each tracklet from the accumulated appearance features. Since these appearance features can be noisy and/or sporadic, some tracklets will have more *ambiguous* identity distribution than others. Afterwards, we exchange messages between the tracklets to infer the identity of the more ambiguous tracklets from the less ambiguous ones. We demonstrate how prioritizing the propagation of messages between the tracklets based on their reliability helps in achieving better results.

4.1 Introduction

In this chapter, we see multi-target tracking and identification as a two-stage process. In the first stage, the plausible target candidates are first detected independently in each frame, *e.g.*, based on [18]. Each detection is characterized by a set of features and their corresponding confidence values. Typically, the confidence assigned to a feature depends on various factors, *e.g.*, whether the detection is occluded and/or visible on the camera view or not, whether the detection is close to the camera view or not, etc. Afterwards, the detec-

tions are aggregated into tracklets, which are defined to group consecutive detections that obviously correspond to the same physical object. The practical implementation of this aggregation process is a research topic by itself. We built on the solution described in [23] for this step, but any other alternative could be envisioned [26, 69]. The benefits obtained from such aggregation process are twofold. First, it reduces the number of entities that have to be processed in the second stage. Second, it provides more reliable and more accurate knowledge about the target appearance, since a target is observed several times along its tracklet.

In the second stage, which embeds the main contributions of this chapter, a graph-based belief propagation formalism is considered to estimate the identity of each tracklet. Each node in the graph corresponds to a tracklet, and is assigned a probability distribution of identities, based on the confidence assigned to the observed tracklet appearance. Typically, a low confidence in the tracklet appearance measurement, or a measurement that is similar to several targets, both result into a flat and thus ambiguous identity distribution for the tracklet. An edge between two tracklets represents that their identities are dependent, meaning that the knowledge of the identity of one tracklet brings some information about the identity of the other. This usually happens in two cases: (a) when the tracklets co-exist at the same time, and (b) when the tracklets are sufficiently close in space, time and/or appearance. The belief propagation module exploits the graph structure and the similarity between the nodes to compute the posterior probability distribution of identities at the nodes. We view this as an inference problem, looking for the most likely identity of a tracklet, given the identities of the other tracklets. As a main contribution, this chapter introduces an original scheduling mechanism to order the propagation of identity beliefs along the graph. In short, the intuition behind our approach consists in propagating the less ambiguous identity information before more ambiguous identity cues.

The ability to drive the belief propagation, based on the level of ambiguity associated to the identity of each node, differentiates our work from most earlier works dealing with identity assignment. Two examples of related earlier works are [72] and [63]. In [72], the authors assume that a track-graph, denoting when targets are isolated and describing when they interact, exists. Assuming that the feature vectors of isolated tracks are reliable, they define a measure of similarity between them. Later, they formulate the identity linking as a Bayesian inference problem in order to find the most probable set of paths. For this, they use standard message passing technique. In [63], the authors use

a conditional random field to identify players in broadcast sport videos. For this, they use a set of features, like SIFT interest points, MSER regions, color histograms, etc. in order to propagate easy-to-classify images of a player to other images.

We differentiate our work from the above works in the sense that the authors in both papers assume that the feature descriptors have the same reliability throughout the time. Therefore, those earlier works do not account for the ambiguity of the identity assignment during message passing.

The remainder of the chapter is organized as follows. Section 4.2 first reviews the work of [23], to explain how the detections are aggregated into tracklets, and how each tracklet gets an identity distribution based on its appearance. Section 4.3 provides a brief introduction to the standard belief propagation, and then describes the proposed priority-based belief propagation. Finally, an experimental validation is provided and discussed in Section 4.4.

4.2 Tracklet definition and prior identity distribution

In this section, we introduce how the inputs to our identity assignment problem are computed. Assuming that the candidate targets are independently detected at each time instant, we briefly explain how they are aggregated into short tracks of detections that quite likely correspond to the same physical object (called tracklets). We then define how the prior identity probability distribution of each tracklet is estimated, based on the accumulation of appearance cues observed along the tracklet.

4.2.1 Tracklet definition

Given a set of candidate detections, the tracking is generally formulated as a data-association problem in a graph [69, 67, 23]. As introduced earlier, the detections have a set of appearance features and their corresponding confidence values. The confidence value of a feature reflects the reliability of its measurement. To link such detections, we follow an iterative aggregation strategy, as in [23]. Starting with a graph in which each detection is a node, the strategy progressively aggregates the nodes of the graph into bigger nodes, named tracklets. Each node corresponds to a tracklet. Specifically, each iteration considers a node, named key-node, and investigates how to link it with either previous or subsequent nodes, assuming that the appearance of the key-node is the appearance of the target. This hypothesis testing procedure computes a

list of shortest-paths to/from the key-node. Only the path that is significantly better than the other alternative paths defines a tracklet. The main advantage of such a strategy is that it can benefit from the appearance features that are sporadically available, or affected by a non-stationary noise, along the sequence of detections.

The outcome of the aggregation process is a set of tracklets. Formally, each tracklet v is characterized by the following features:

- The positions of the starting and ending, denoted as $\boldsymbol{x}_v^{(s)}$ and $\boldsymbol{x}_v^{(e)}$ respectively,
- The starting and ending time of the tracklet, denoted as $t_v^{(s)}$ and $t_v^{(e)}$ respectively,
- The average appearance features of the object detected along the tracklet, and their corresponding confidence values. These features give an initial estimate of its identity distribution, as discussed in the next section.

4.2.2 Assigning identity distribution based on appearance features

In the previous section, we have presented how the tracklets are defined. This section explains how to assign an identity distribution to each of them.

We assume that there are M targets, each of them being characterized by N appearance features, which are assumed to be known *a priori*. Nevertheless, the appearance features can be learned automatically too [87].

Let the i-th feature of the j-th target be denoted by $\boldsymbol{f}_i^{(j)}$, $1 \leq j \leq M$. Then, the feature set for the j-th target is $\mathcal{F}^{(j)} = \{\boldsymbol{f}_1^{(j)}, ..., \boldsymbol{f}_N^{(j)}\}$. For example, in a basketball match, color and digit on the jersey of the player can be considered as features. In this case, $N = 2$ and $\mathcal{F} = \{\text{color}, \text{digit}\}$.

The appearance of a tracklet is defined by averaging the appearance features, measured in each detection of the tracklet. Let the average appearance features for a tracklet v be denoted by $\overline{\mathcal{F}}^{(v)} = \{\overline{\boldsymbol{f}}_1^{(v)}, ..., \overline{\boldsymbol{f}}_N^{(v)}\}$. This allows us to define the probability of the tracklet v having identity j, denoted by $p_v(j)$, as

$$p_v(j) \propto \prod_{i=1}^{N} \exp\left(-\frac{\parallel \boldsymbol{f}_i^{(j)} - \overline{\boldsymbol{f}}_i^{(v)} \parallel_1}{\tau_i^{(v)}}\right) \qquad \text{for } 1 \leq j \leq M \tag{4.1}$$

where $\tau_i^{(v)}$ weights the influence of the i-th feature on identity assignment, and is related to the confidence assigned to the ith feature of the tracklet v. It

decreases when the appearance feature becomes more accurate and reliable. When $\tau_i^{(v)}$ is very large or when the appearance of a tracklet is far from all target appearances, the probability distribution, in Equation 4.1, tends to uniform distribution, *i.e.*, the identity assignment becomes ambiguous as all identities are equally likely. Conversely, when the tracklet appearance is closer to a target appearance, the probability distribution becomes peaky around that identity, implying that the identity assignment is less ambiguous. Consequently, depending on the observed appearance features and their confidence values, some tracklets have less ambiguous identity distributions than others.

4.3 Belief propagation

In this section, belief propagation is considered to exchange identity information between the tracklets. The purpose is to compute posterior identity probabilities by merging the prior identity distributions and exploiting the graph structure. We first survey the principles of belief propagation. We then present how the belief propagation graph is constructed in our application scenario. Eventually, we introduce the main contribution of our paper, which lies in an original priority-based scheduling mechanism to select the nodes from which the identity information is propagated.

4.3.1 Standard belief propagation

In this section, we briefly recall the framework for the belief propagation technique. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given, where $\mathcal{V}$ is the set of nodes in the graph and $\mathcal{E}$ represents the association between the nodes. The neighborhood of node $v \in \mathcal{V}$ is denoted by $\mathcal{N}_v$.

We assume that each node $v \in \mathcal{V}$ and each edge $(u, v) \in \mathcal{E}$ are associated with potential functions ϕ_v and ϕ_{uv} respectively. The purpose of belief propagation is to find a labeling function l that labels each node $v \in \mathcal{V}$ with a label $l_v \in \mathcal{L}$, $|\mathcal{L}| = M$ being the total number of labels, so as to maximize the joint likelihood function:

$$p(l) \propto \prod_{v \in \mathcal{V}} \left[\phi_v(l_v) \prod_{u \in \mathcal{N}_v} \phi_{uv}(l_u, l_v) \right] \tag{4.2}$$

Generally, it is done iteratively by exchanging "messages" between the nodes. Let $\boldsymbol{m}_{u \to v}^{(t)}$ be the message that the node u sends to a neighboring node v at iteration t. Intuitively, $\boldsymbol{m}_{u \to v}^{(t)}(l_v)$ is the belief that node u thinks about the label l_v

of node v at any iteration t. Each message is initialized uniformly. Afterwards, new messages are updated (in sum-product form) at each iteration as:

$$\boldsymbol{m}_{u\to v}^{(t)}(l_v) \propto \sum_{l_u\in\mathcal{L}} \big[\phi_{uv}(l_u, l_v)\, \underbrace{\phi_u(l_u) \prod_{s\in\mathcal{N}_u\setminus v} m_{s\to u}^{(t-1)}(l_u)}_{:=\boldsymbol{h}_u(l_u)}\big], \tag{4.3}$$

where $\boldsymbol{h}_u(l_u)$ is referred to as the *pre-message* of u. After T iterations, a belief vector $\boldsymbol{b}_v$ is computed for each node as:

$$\boldsymbol{b}_v^{(T)}(l_v) \propto \phi_v(l_v) \prod_{s\in\mathcal{N}_v} \boldsymbol{m}_{s\to v}^{(T)}(l_v) \tag{4.4}$$

4.3.2 Graph of identity beliefs and definition of potential terms

In this section, we will first explain the graph formalism for the belief propagation. Later, we will explain how the potential terms are constructed in our application scenario.

As discussed in Section 4.2, the output of aggregation step is a set of tracklets. Each tracklet has its starting and ending time-stamps and positions. Moreover, each tracklet is assigned an initial identity distribution, based on the observed appearance features. These tracklets are gathered into a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes, with each node corresponding to a tracklet; $\mathcal{E}$ is a set of edges, defining the connectivity between the nodes in $\mathcal{V}$. An edge between nodes u and v implies that their identities are dependent. Thus, knowing the identity of one brings some information about the other. Specifically, the support of the tracklets can be used to enforce the constraint that two tracklets, which co-exist at the same time, should belong to two different physical targets. This defines a *mutex* edge between them. Additionally, the knowledge of the extremities of the tracklets (and their appearances) enables us to estimate the proximity between them such that if the tracklets are sufficiently close, they are likely to share the same identity. In contrast, if they are significantly far apart, they are encouraged to have different identities. This defines a *temporal* edge between them. An example is elucidated in Figure 4.1.

Now, we explain how we associate the potential terms to each node and edge. The unary potential term $\phi_v(l_v)$ is defined to be the likelihood of the node $v \in \mathcal{V}$ having a label l_v. As the prior identity distribution $p_v(l_v)$, computed by Equation 4.1, represents how likely the label l_v is, we use it as the estimate of the unary potential. That is, $\phi_v(l_v) = p_v(l_v)$, $l_v \in \mathcal{L}$.

The pairwise potential term ϕ_{uv} is defined to reflect that the identities of u and v are dependent. Typically, in our practical scenario, it is defined such

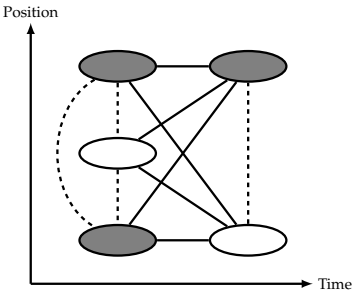


Figure 4.1: **Graph structure.** Each node has an identity distribution: **white** - a peaky distribution, **black** - a flat distribution, and **gray** - an intermediate distribution. Solid lines represent temporal edges whereas mutex edges are shown in dashed lines.

that, when u and v are likely to correspond to the same physical target (*e.g.*, because they are close in appearance or in space and time), $\phi_{uv}(l_u, l_v)$ tends to zero for $l_u \neq l_v$, and to 1 for $l_u = l_v$. In contrast, when they are likely to correspond to different physical targets (*e.g.*, because they co-exist in time), $\phi_{uv}(l_u, l_v)$ should be defined so that $\phi_{uv}(l_u, l_v)$ tends to zero for $l_u = l_v$, and to 1 for $l_u \neq l_v$.

Following those general principles, we define the potentials over the mutex and temporal edges in our graph structure as follows. In case of mutex edges, u and v should have different labels. Therefore,

$$\phi_{uv}(l_u, l_v) = \begin{cases} \epsilon & \text{if } l_u = l_v \\ 1 - \epsilon & \text{otherwise,} \end{cases} \tag{4.5}$$

where ϵ is a small positive number. Setting $\epsilon = 0$ enforces that the identities of the nodes are unique at a given time. However, it is possible that two nodes share the same identity. This, for example, happens when a tracklet incorrectly aggregates (see Section 4.2) detections that correspond to different physical targets, resulting in a mixed identity distribution. In such cases, imposing $\epsilon = 0$ might lower the performance of the system. In our settings, we use $\epsilon = 0.1$.

In case of temporal edges, we express ϕ_{uv} in terms of the *distance* d_{uv} between them. When the distance between the nodes is small, they should be encouraged to share the same label and vice versa. We define

$$\phi_{uv}(l_u, l_v) = \begin{cases} \exp(-d_{uv}/\tau_{\text{dist}}) & \text{if } l_u = l_v \\ 1 - \exp(-d_{uv}/\tau_{\text{dist}}) & \text{otherwise,} \end{cases} \tag{4.6}$$

where τ_{dist} is a constant. Now, we turn our attention towards the definition of the distance between the nodes. Two cases are distinguished. On the one hand, if both u and v have reliable identity estimate, then the computation of the Bhattacharyya distance between the belief vectors $\boldsymbol{b}_u$ and $\boldsymbol{b}_v$ is used to define the distance between the nodes. We denote the entropy of belief vector of a node u by $E(u)$. Then, if $E(u) < \tau_{\text{TH}}$ and $E(v) < \tau_{\text{TH}}$ then,

$$d_{uv} = \left[1 - \sum_{i=1}^{|\mathcal{L}|} \sqrt{b_u(i) b_v(i)}\right]^{1/2} \tag{4.7}$$

We set $\tau_{\text{TH}} = 0.5$ and $\tau_{\text{dist}} = 0.3$. The choice of τ_{TH} is not critical, as long as it is chosen small enough, as illustrated by the the results.

On the other hand, if one of the nodes does not have reliable identity estimate, then the computation of the Bhattacharyya distance between the belief vectors is irrelevant. In such cases, when the nodes are close in time (*i.e.*, $|t_v^{(s)} - t_u^{(e)}| < \tau_{\text{max}}$), the position information is used to measure their distance d_{uv}. If they are far in space, they should have different identities, and conversely if they are close in space, they should be encouraged to share the same identity. We set $\tau_{\text{max}} = 120$ which allows to investigate nodes that are upto 6 seconds (at the frame rate of 20 fps) far apart, and the distance is computed as:

$$d_{uv} = \left\| \boldsymbol{x}_v^{(s)} - \boldsymbol{x}_u^{(e)} \right\|_2 \tag{4.8}$$

We use $\tau_{\text{dist}} = 450$.

In contrast, when the nodes are far in time, even the position cannot guide the definition of the distance. In this case, no message is exchanged between the nodes as it does not help to disambiguate the possible labels of the nodes.

4.3.3 Priority scheduling of belief message exchanges

The standard belief propagation (BP) technique, described in Section 4.3.1, does not prioritize nodes, which means that the nodes are selected in an arbitrary order to send messages to their neighbors. Moreover, all the nodes transmit messages to their neighbors.

However, in our graph, some nodes are less ambiguous about their identities than others. Such non-uniform distribution of ambiguity over the nodes should allow faster convergence of the BP because the messages sent by less ambiguous nodes are more informative. Consequently, they can help the more ambiguous neighbors to disambiguate their labels. Therefore, it sounds natural to prioritize the nodes such that less ambiguous nodes transmit their messages first. Interestingly, the authors in [88] have followed the same intuition

to solve an image completion problem. We have adapted the principle of belief propagation prioritization to our identity assignment context.

As already mentioned, priority is related to the ambiguity of the node identity. The definition of ambiguity (and hence the priority) of a node v depends on the peakedness of the current belief vector $\boldsymbol{b}_v$ that has been estimated by the BP algorithm. We use *entropy* of the belief vector, defined as $H(v) := -\sum_{i=1}^{|\mathcal{L}|} \boldsymbol{b}_v(i) \log(\boldsymbol{b}_v(i))$, to measure the ambiguity of node v. The entropy is maximum when the belief vector has a flat distribution and decreases with the peakedness of the distribution. Therefore, the node v is assigned a higher priority if it has lower entropy and vice versa. We have tested other approaches for the priority (*e.g.*, cardinality of the confusion set [88], kurtosis, etc.). However, the results are not significantly better than the ones obtained with the entropy measure.

Apart from priority-based scheduling, to avoid propagating confusing information, the construction of exchanged messages is also affected by the ambiguity of the information available in each node. Specifically, during the construction of message $\boldsymbol{m}_{u\to q}$, the node u is supposed to gather messages from all its neighbors, except q, to construct $\boldsymbol{h}_u$. However, since there can be some nodes in the neighborhood of u which are more ambiguous than u, the messages coming from those nodes could be considered as being uninformative, or even confusing, since they are nearly flat, meaning that all labels are equally likely. Thus, we only consider those nodes that are less ambiguous than u to compute $\boldsymbol{h}_u$. The practical implementation of this principle works as follows. Each node is assigned a *committed* flag, initialized to *false* in the beginning of each iteration of the message exchange process. Once a node is scheduled to exchange messages with its neighbors, its flag is set to *true*. In this way, the less ambiguous neighbors of u have their committed flags set to true as they have been scheduled before u. Similarly, the more ambiguous nodes will have their committed flags set to false. Following the same kind of idea, the message $\boldsymbol{m}_{u\to q}$ is constructed and sent only if q is more ambiguous than u, *i.e.*, if the committed flag of q is false. This is illustrated in Figure 4.2.

The algorithm for the priority based belief propagation is presented in Algorithm 3.

After T iterations, we assign the label l_v to a node v, as

$$l_v = \begin{cases} l_x^* & \text{if } b_v^{(T)}(l_x^*) > \kappa b_v^{(T)}(l_y^*), \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{4.9}$$

where $l_x^* = \arg\max_{l_x \in \mathcal{L}} b_v^{(T)}(l_x)$ and $l_y^* = \arg\max_{l_y \in \mathcal{L} \setminus l_x^*} b_v^{(T)}(l_y)$. We use $\kappa =$

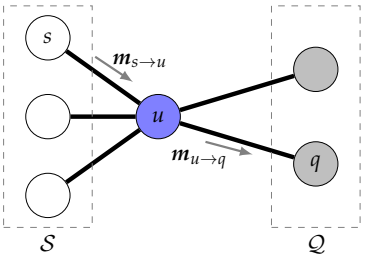


Figure 4.2: **Message construction and dissemination at node u (in blue)**. The node u gathers information from its less ambiguous neighbors, $\mathcal{S}$ (in **white**). Afterwards, u transmits message to its more ambiguous neighbors, $\mathcal{Q}$ (in **gray**).

4.

4.4 Evaluation

We apply the proposed method on the output of the detector and tracklet aggregation process, as described in [23]. We run the algorithm on 10 minutes of APIDIS dataset [6]. The distribution of the cameras is not symmetrical, meaning that there are more cameras on one side than the other. Hence, the observation of appearance features is more reliable on one side of the court than on the other side. The identity and the position of the players have been manually defined at every second. This provides the reference ground truth used in our evaluation.

In the rest of the section, the results for different configurations of the graph and different belief propagation algorithms are presented.

4.4.1 Quantitative results

In order to elucidate the effect of message passing in the performance of the system, we explored the following variations of the identity assignment algorithm: (a) **No BP** (no message passing is done, equivalent to treating each node independently); (b) **Standard BP** (nodes are chosen either sequentially or in an arbitrary order); and (c) **Priority BP** (nodes are prioritized in increasing order of ambiguities). Table 4.1 presents the metrics for the above message passing techniques.

From the Table 4.1, we can observe that the message passing between the nodes indeed boosts the accuracy of the system from 68.14% to 89.04% at the

Algorithm 3 Priority Belief Propagation

Initialize: $b_v(l_v) \leftarrow p_v(l_v)$ and $\phi_v(l_v) \leftarrow p_v(l_v) \quad \forall v \in \mathcal{V}, l_v \in \mathcal{L}$
for $t = 1$ to T **do**
  v.committed $\leftarrow$ false $\forall v \in \mathcal{V}$
  $\mathcal{R} \leftarrow \mathcal{V}$
  while $\mathcal{R} \neq \emptyset$ **do**
    $u \leftarrow$**Schedule**$(\mathcal{R})$ # *Prioritize node u according to its level of ambiguity*
    u.committed $\leftarrow$ true
    $\mathcal{S} \leftarrow \{s | s \in \mathcal{N}_u, s.\text{committed=true}\}$ # *Less ambiguous neighbors of u*
    $\boldsymbol{h}_u \leftarrow$ Compute the pre-message of u from $\mathcal{S}$
    $\mathcal{Q} \leftarrow \{v | v \in \mathcal{N}_u, v.\text{committed=false}\}$ # *More ambiguous neighbors of u*
    for all $v \in \mathcal{Q}$ **do**
      $\boldsymbol{m}_{u \to v}^{(t)} \leftarrow$ **ComputeMessage**$(u, v, \boldsymbol{h}_u, \tau_{\max})$
      $\boldsymbol{b}_v \leftarrow$ Compute belief for node v
    end for
    $\mathcal{R} \leftarrow \mathcal{R} \setminus u$
  end while
end for
Assign identity to each node $v \in \mathcal{V}$ according to the Equation 4.9.

cost of a slight increase in the mismatch[1] error from 1.48% to 2.54%. Interestingly, we can observe that the results are better for random access of the nodes as compared to the sequential access. More importantly, the scheduling of the nodes improves the performance drastically as compared to the arbitrary access of the nodes.

In Figure 4.3, we can observe how the average entropy of the system evolves at each iteration. We can see that the priority BP not only attains the lowest entropy but also converges rapidly. It shows that the order in which the nodes transmit message indeed affects the convergence of the system, thereby, affecting the quality of the solution.

In addition, we explored how the graphical model affects the performance of the priority BP. For this purpose, we have run priority BP on other three different variations of the full graph. They are defined as: (a) **No edges** (nodes are not connected by edges, and are hence independent); (b) **Mutex edges only** (with only the mutex edges between the nodes); (c) **Temporal edges only** (with only the temporal edges between the nodes).

[1] We used the term *wrong identification* in [24].

	Accuracy(%)	FP(%)	MM(%)	MS(%)
No BP	68.14	0.13	1.48	30.25
Std. BP (sequential)	73.59	0.15	1.76	24.50
Std. BP (random)	83.69	0.16	2.36	13.79
Priority BP	**89.04**	**0.15**	**2.54**	**8.27**

Table 4.1: Comparison of performance metrics for different node scheduling approaches.

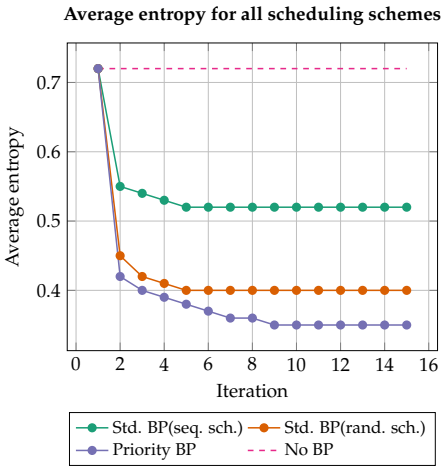


Figure 4.3: **Evolution of average entropy for different node scheduling mechanisms.** *Best viewed in color.*

The performances of all models are shown in Table 4.2. We can draw two main conclusions. First, adding edges to the baseline system allows exploiting the correlation between the nodes. Second, both the temporal edges and the mutex edges help in improving the performances.

Performance metrics for all configurations of the graph

Table 4.3 presents the results of all three message passing techniques, described above. We can see that the yellow team is recognized better than the blue team. This might be accredited to asymmetry of the position of the cameras around the court.

	Accuracy(%)	FP(%)	MM(%)	MS(%)
No edges	68.14	0.13	1.48	30.25
Mutex edges	80.32	0.15	2.03	17.50
Temporal edges	84.49	0.15	2.71	12.65
Both edges	**89.04**	**0.15**	**2.54**	**8.27**

Table 4.2: Performance metrics for priority-based BP on four graphical models.

Team	Metrics	No BP	Std. BP	Priority BP		
				Mutex	Temporal	Both
Blue	Accuracy(%)	48.74	55.54	70.11	77.45	86.37
	FP(%)	0.24	0.24	0.24	0.24	0.24
	MM(%)	0.71	1.04	1.82	2.26	2.42
	MS(%)	50.31	43.18	27.83	20.05	10.91
Yellow	Accuracy(%)	87.54	91.64	90.53	91.54	91.71
	FP(%)	0.03	0.07	0.07	0.07	0.07
	MM(%)	2.25	2.48	2.25	3.16	2.65
	MS(%)	10.18	5.81	7.15	5.23	5.58
Avg.	Accuracy(%)	68.14	73.59	80.32	84.49	**89.04**
	FP(%)	0.13	0.15	0.15	0.15	**0.15**
	MM%)	1.48	1.76	2.03	2.71	**2.54**
	MS(%)	30.25	24.50	17.50	12.65	**8.27**

Table 4.3: Performance metrics for all configurations.

Performance with respect to the entropy threshold

This threshold is used in the definition of the pairwise potential to determine whether the current identity estimate is reliable or not. If the identities of two nodes are reliable, then the potential function is defined by the Bhattacharyya distance between their beliefs. Otherwise, the position feature is used for the definition.

When this threshold is small, the potential function between the nodes is governed by the identity estimate only when the nodes have small entropy. That is, these nodes are already unambiguous, and the exchange of message between them does not bring extra information. This can be observed from the Table 4.4, in which the metrics do not significantly differ for $\tau_{\text{TH}} \leq 0.3$.

Conversely, when the threshold is high, many temporal edges will be driven by the Bhattacharyya distance, and even if the spatio-temporal information is discriminant, ignored and replaced by the ambiguous identity information. Consequently, the system will not be able to exploit the position features. More importantly, unreliable messages (and possibly wrong messages) might be exchanged between the nodes, which, in turn, might degrade the performance. This can be seen from the Table 4.4 in which the metrics get worse for $\tau_{\text{TH}} > 0.5$.

	0.05	**0.1**	**0.3**	**0.5**	**0.6**	**0.8**
Accuracy(%)	87.79	87.79	87.79	89.04	87.08	84.07
FP(%)	0.15	0.15	0.15	0.15	0.15	0.27
MM(%)	2.54	2.54	2.54	2.54	2.54	2.57
MS(%)	9.52	9.52	9.52	8.27	10.23	13.09

Table 4.4: Performance with respect to the threshold on entropy τ_{TH}.

Impact of the node ambiguity level estimation method

The results presented so far are obtained by scheduling the nodes according to their entropies. Our objective is to prioritize the nodes that have a peaked distribution over the ones with flat distribution. Thus, we can define various metrics that estimate the measure of peakedness of the identity distribution. Apart from the entropy, we have tested two methods to estimate the level of ambiguity of the nodes. They are the cardinality of the confusion set and the kurtosis of the belief vector.

Confusion set

The notion of confusion set has been introduced in [88]. Given the belief vector $\boldsymbol{b}_v$, one way to count the confidence of this node is simply to count the number of likely labels, *e.g.*, those whose belief value exceed a certain threshold τ_b. More number of labels imply that the node identity is more ambiguous. Labels with high probability exist for that node and hence it is more ambiguous. Of course, the relative beliefs $b_v^{(\text{rel})}(l_v) = b_v(l_v) - b_v^{(\text{min})}$ (where, $b_v^{(\text{min})} = \min_{l_v} b_v(l_v)$) matter in this case. Thus, the set $\mathcal{CS}(v) = \{l_v \in \mathcal{L} \mid b_v^{(\text{rel})}(l_v) \geq \tau_b\}$ corresponds to the confusion set of v, and its cardinality $|\mathcal{CS}(v)|$ is an estimate of the ambiguity of v.

The performance of the system with various values of the τ_b is presented in Table 4.5.

	0.01	**0.03**	**0.1**	**0.3**
Accuracy(%)	90.16	89.05	86.23	69.84
FP(%)	0.15	0.15	0.15	0.13
MM(%)	2.52	2.45	2.24	1.85
MS(%)	7.17	8.35	11.38	28.18

Table 4.5: Performance metrics with respect to τ_b.

From the table, it can be deduced that the larger value of τ_b implies that the cardinality of the confusion set will be low and thus many nodes will have similar priorities, which, in turn, deteriorate the performance of the system.

Kurtosis of the belief vector

Another method to estimate the peakedness of the belief vector is to compute its kurtosis. The kurtosis is the fourth central moment, divided by fourth power of the standard deviation of a distribution. It is a descriptor of the shape of the distribution and higher kurtosis values correspond to peaky distributions. Thus, we use the kurtosis as the estimate of the level of ambiguity of the identity distribution and schedule the nodes in decreasing order of kurtosis. The results of kurtosis for assigning priority to nodes are as follows:

Accuracy=89.09% **FP**=0.15% **MM**=2.54% **MS**=8.22%.

4.4.2 Qualitative results

In this section, we provide some qualitative results.

Performance metrics

Figure 4.4 depicts the performance metrics for the targets across time. Even though there are 21 targets in the ground truth, there are 2 referees and 4 spectators (which correspond to ID 1, 10, 12, 14, 15 and 16) for which there is no appearance features. Therefore, the identities for these targets have not been computed.

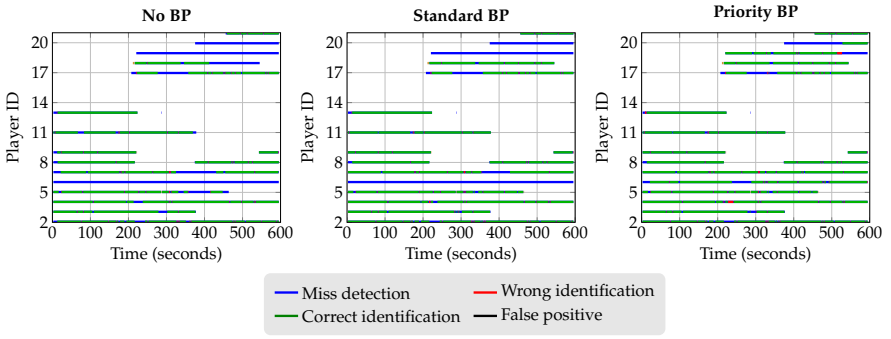


Figure 4.4: **Performance metrics for priority belief propagation across time for each target.** Each line corresponds to a single player track. **Left**: Baseline system where no belief propagation is performed (accuracy = 68.14%). **Middle**: nodes are scheduled in the order they are stored in memory and hence no priority (standard BP) (accuracy = 73.59%). **Right**: Nodes are prioritized according to their level of ambiguity (priority BP) (accuracy = 89.04%).

Sample frames

Some sample frames[2] are shown in Figure 4.5.

4.5 Conclusion

In this chapter, we presented an approach to solve identity (or, label) assignment problem in a scenario for which target candidates have been detected and observed independently with various degree of reliability in the belief propagation framework. Messages are transmitted between the detections to infer the identity of ambiguous observations, based on the reliable identities available from non-ambiguous appearance features. We show that the order in which messages are exchanged between the nodes affects the quality of the solution. We have proposed a priority-based node scheduling mechanisms to favour the transmission of information from less ambiguous to more ambiguous nodes. The above approach has been applied on a real-life basketball game to recognize the players. The correct recognition rate of 89% demonstrates the effectiveness and efficiency of the proposed approach.

[2] A demo video is available at http://sites.uclouvain.be/ispgroup/index.php/Research/MultiObjectTracking.

Figure 4.5: **Sample frames for priority belief propagation**. At each frame, estimated IDs of the players are shown, along with a 50-frame long track. **Green**- correct recognitions; **Red**-wrong identifications; **Blue**-misses. As we can see, referees do not have distinct appearance features and thus their identities are almost uniformly distributed among the blue players. *Best viewed in color.*

Discriminative label propagation for tracking with missing features

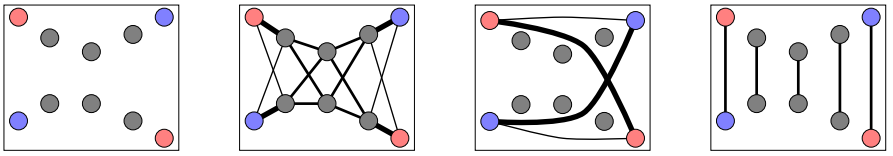
5

5.1 Introduction

In this chapter, we present an alternate formulation of the multi-object tracking problem, for which an efficient solution can be computed and that does not require prior knowledge of the possible appearances. We adopt a graph-based label propagation framework. 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 in order to reflect the fact that two detections that occur at the same time should be assigned to distinct labels. Hence, we construct $N + 2$ 'complementary' graphs (one spatio-temporal, N appearance, one exclusion), where N is the number of appearance features. An example is shown in Figure 5.1. 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 except the exclusion 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 the exclusion graph only connect nodes that cannot belong to the same physical target. This is justified/relevant, for example, when the detections occur at the same time.

Given these graphs, the tracking problem is formulated as finding a consistent label assignment, which means that (i) the nodes that are sufficiently



(a) Detections (b) Spatio-temporal graph (c) Appearance graph (d) Exclusion graph

Figure 5.1: **Graph construction. (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. In graphs (b) and (c), thickness of an edge is proportional to its weight. *Best viewed in color.*

close in space and time are labeled similarly, (ii) the nodes that are close (respectively, far) in appearance are labeled similarly (respectively, differently), and (iii) 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 graphs, a good labeling should minimize the energy in the spatio-temporal and the appearance graphs while maximizing the energy due to the exclusion graph. This chapter includes the following contributions:

- Formulation of the multi-object tracking with sporadic appearance features as a labeling problem in a number of complementary graphs (Section 5.2).
- 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 5.3).
- An extension of the local label propagation process to handle incremental tracking scenarios (Section 5.4).

The rest of the chapter is organized as follows: the formulation of the tracking problem is presented in Section 5.2. The proposed solution is detailed in Section 5.3. Afterwards, we move to the incremental graph construction and label propagation formalism in Section 5.4. A brief review of the related work is presented in Section 5.5. Experimental results are presented in Section 5.6. Section 5.7 concludes the chapter.

5.2 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.

5.2.1 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 [89]. 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 $\boldsymbol{f}_i$ and we concatenate the feature of its neighbors into $\boldsymbol{F}^{(i)}$ as $\boldsymbol{F}^{(i)} := (\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_{|\mathcal{N}_i|})$, where $\mathcal{N}_i$ is the set of neighbors of i. Afterwards, the graph construction can be formulated as the problem of finding the vector of reconstruction weights $\boldsymbol{w}_i^\star$ that minimizes the following optimization problem

$$\boldsymbol{w}_i^\star = \underset{\boldsymbol{w}_i \in \Delta_{|\mathcal{N}_i|}}{\operatorname{argmin}} \|\boldsymbol{f}_i - \boldsymbol{F}^{(i)}\boldsymbol{w}_i\|_2^2 + \tfrac{\lambda}{2}\|\boldsymbol{w}_i\|_2^2, \tag{5.1}$$

where $\Delta_m := \{\boldsymbol{w} \in \mathbb{R}^m \mid \boldsymbol{w} \succeq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{w} = 1\}$ is the probability simplex of size m. The reason to constrain the weights to belong to the simplex, *i.e.*, to choose the weights to be non-negative and to sum to unity, is that it promotes weight vector sparsity, leading to an efficient optimization in Section 5.3. This can be shown by taking the Lagrange multipliers of the positive simplex which amounts to promote a small ℓ_1-norm of the weights in addition to the cost minimization used in Equation 5.1. 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. In order to mitigate this limitation, we add a quadratic part $\frac{\lambda}{2}\|\boldsymbol{w}_i\|_2^2$, which offers an additional advantage of making the problem strongly convex, resulting in a unique $\boldsymbol{w}_i^\star$. This can be seen as similar to an elastic net regularization [90] in the sense that the sparsity term is imposed by the constraints. We use $\lambda = 10^{-2}\|\boldsymbol{f}_i\|_2$. By taking the

parameter λ proportional to $\|\boldsymbol{f}_i\|_2$, the optimization becomes independent of a scaling of $\boldsymbol{f}_i$.

Once the weights for each data point are computed, we gather them into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, 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 $\boldsymbol{w}_i^\star(j)$, resulting from Equation 5.1, is non-zero, *i.e.*, $\mathcal{E} = \{(i, j) \mid \boldsymbol{w}_i^\star(j) > 0\}$.
- $\boldsymbol{W}$ assigns a weight to each edge such that

$$w_{ij} = \begin{cases} \boldsymbol{w}_i^\star(j) & \text{if } (i, j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

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

Spatio-temporal graph. In case of the spatio-temporal graph, $\boldsymbol{f}_i$ is defined by the time instant t_i and the location information $\boldsymbol{x}_i$ (*e.g.*, bounding box of the detections). Hence, $\boldsymbol{f}_i = (\gamma t_i, \boldsymbol{x}_i)^\top$, 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, assuming that the targets move at constant velocity in a small temporal neighborhood. We use $\gamma = 3$ pixels/frame. The neighbors $\mathcal{N}_i$ are defined to be the samples whose time indices fall within a small temporal window of size T except t_i and which do not violate the spatio-temporal constraints imposed by the exclusion graph (see below). 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. Since the window includes the samples from both the past and the future, a linear motion model is implicitly embedded in our method.

Appearance graph. In case of the appearance graph, $\boldsymbol{f}_i$ 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 $t_i = t_j$. Similarly, we can enforce *gating constraint* so that two detections that violate the maximum speed constraint cannot belong to the same trajectory, *i.e*, $w_{ij} = 1$ if $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 > v_{\max}|t_i - t_j|$, 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.

5.2.2 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 a label assignment $\boldsymbol{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_n)^\top$ that is defined to assign a m-dimensional[1] label distribution $\boldsymbol{y}_i \in \Delta_m$ to the i-th node, where Δ_m is a m-dimensional probability simplex. Each dimension of the label distribution $\boldsymbol{y}_i$ corresponds to a target. Formally, the k-th dimension, $\boldsymbol{y}_i(k), k = 1, \cdots, m$, can be interpreted as the probability of the node i being the k-th target. Consequently, $\boldsymbol{Y}$ is a row-stochastic matrix, with each row summing to unity. Therefore, we write $\boldsymbol{Y} \in \mathcal{P}_{nm}$, where $\mathcal{P}_{nm}$ is the set of all row-stochastic matrices of size $n \times m$. We consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ 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 [29], a harmonic function approach is introduced to measure the inconsistency of the label assignment matrix $\boldsymbol{Y}$ 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[2] is defined as

$$E_{\boldsymbol{L}}(\boldsymbol{Y}) := \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L} \boldsymbol{Y}), \tag{5.3}$$

where $\mathbf{Tr}$ is the trace of a matrix and $\boldsymbol{L}$ is the graph Laplacian, defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix whose i-th diagonal element is $d_{ii} := \sum_{j \in \mathcal{N}_i} w_{ij}$.[3] For a graph with non-negative weights, *i.e.*, $w_{ij} \geq 0$, $\boldsymbol{L}$ is positive semi-definite and consequently the labeling energy in Equation (5.3) is convex in $\boldsymbol{Y}$.

[1] In the case where m is not known *a priori*, we set $m = n$, considering the worst case in which each detection corresponds to a different target.

[2] In the literature, it is commonly referred to as the *harmonic energy*.

[3] Due to the definition of weights in our graphs, we have $d_{ii} = 1$. Therefore, $\boldsymbol{D}$ is an identity matrix.

In our framework, we have $N+2$ distinct graphs. As all the graphs have the same set of nodes, we frequently refer to a graph by its Laplacian $\boldsymbol{L}$ in the sequel. We represent the exclusion graph by $\boldsymbol{L}^{(-)}$, and other graphs by $\boldsymbol{L}_l^{(+)}, l \in \{0, ..., N\}$, where $l = 0$ corresponds to the spatio-temporal graph and $1 \leq l \leq N$ corresponds to the l-th appearance graph. We explicitly introduce the minus (respectively, plus) superscript in order 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 $\boldsymbol{Y}^\star$ that minimizes the labeling energies due to $\boldsymbol{L}_l^{(+)}$ and maximizes the labeling energy due to $\boldsymbol{L}^{(-)}$. Mathematically, we have

$$\begin{aligned} \boldsymbol{Y}^\star &:= \underset{\boldsymbol{Y} \in \mathcal{P}_{nm}}{\operatorname{argmin}} \sum_{l=0}^{N} \alpha_l E_{\boldsymbol{L}_l^{(+)}}(\boldsymbol{Y}) - E_{\boldsymbol{L}^{(-)}}(\boldsymbol{Y}) \\ &= \underset{\boldsymbol{Y} \in \mathcal{P}_{nm}}{\operatorname{argmin}} E_{\boldsymbol{L}_{\text{eff}}^{(+)}}(\boldsymbol{Y}) - E_{\boldsymbol{L}^{(-)}}(\boldsymbol{Y}) \end{aligned} \tag{5.4}$$

where $\boldsymbol{L}_{\text{eff}}^{(+)} := \sum_{l=0}^{N} \alpha_l \boldsymbol{L}_p^{(+)}$, 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. In practice, the relevance of a labeling energy depends on how its associated feature is related to targets identities. 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. Since $\alpha_l \geq 0$ and $\boldsymbol{L}_l^{(+)}$ is positive semi-definite, $\boldsymbol{L}_{\text{eff}}^{(+)}$ is also positive semi-definite. Given $\boldsymbol{Y}^\star$, the i-th node is assigned the label that corresponds to the largest entry in $\boldsymbol{y}_i^\star$.

5.3 Graph-consistent labels computation

In this section, we explain how to compute the solution $\boldsymbol{Y}^\star$ of the problem, defined in Equation (5.4). 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.

5.3.1 Joint label assignment optimization

Let us rewrite Equation (5.4) as

$$\begin{aligned} \boldsymbol{Y}^{\star} &= \underset{\boldsymbol{Y}\in\mathcal{P}}{\operatorname{argmin}} \operatorname{\mathbf{Tr}}(\boldsymbol{Y}^{\top}\boldsymbol{L}_{\text{eff}}^{(+)}\boldsymbol{Y}) - \operatorname{\mathbf{Tr}}(\boldsymbol{Y}^{\top}\boldsymbol{L}^{(-)}\boldsymbol{Y}) \\ &:= \underset{\boldsymbol{Y}\in\mathcal{P}}{\operatorname{argmin}} \big[g(\boldsymbol{Y}) := f(\boldsymbol{Y}) - h(\boldsymbol{Y})\big], \end{aligned} \tag{5.5}$$

where $f(\boldsymbol{Y}) := \operatorname{\mathbf{Tr}}(\boldsymbol{Y}^{\top}\boldsymbol{L}_{\text{eff}}^{(+)}\boldsymbol{Y})$ and $h(\boldsymbol{Y}) := \operatorname{\mathbf{Tr}}(\boldsymbol{Y}^{\top}\boldsymbol{L}^{(-)}\boldsymbol{Y})$. As $\boldsymbol{L}_{\text{eff}}^{(+)}$ and $\boldsymbol{L}^{(-)}$ are positive semi-definite matrices, both $f(\boldsymbol{Y})$ and $h(\boldsymbol{Y})$ are convex in $\boldsymbol{Y}$, whereas $g(\boldsymbol{Y})$ is non-convex. Specifically, Equation (5.5) belongs to a family of problems, called *difference of convex* (DC) programming, and an iterative *majorization minimization* [91] algorithm can be used to solve the problem, as presented in Algorithm 4. Starting with a random label distribution $\boldsymbol{Y}^{(1)} \in \mathcal{P}$, the algorithm iteratively linearizes $h(\boldsymbol{Y})$ around the k-th iterate $\boldsymbol{Y}^{(k)}$ and solves the resulting convex function $f(\boldsymbol{Y}) - \nabla h^{\top}\left(\boldsymbol{Y}^{(k)}\right)\boldsymbol{Y}$ using the projected gradient method [92]. The number of iterations T_{joint} depends on the convergence tolerance.

Algorithm 4 Joint label assignment optimization

Input

Graph Laplacians: $\{\boldsymbol{L}_l^{(+)}, l = 0, ..., N\}, \boldsymbol{L}^{(-)}$

Scaling weights: $\{\alpha_l, l = 0, ..., N\}$

Number of iterations: T_{joint}

Output

Label assignment matrix: $\boldsymbol{Y}^{\star}$

Procedure:

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

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

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

Solve the convex optimization problem

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

by the projected gradient method [92].

End For

Return $\boldsymbol{Y}^{\star} \leftarrow \boldsymbol{Y}^{(T_{\text{joint}}+1)}$.

It is worth noting that the gradient of $\operatorname{\mathbf{Tr}}(\boldsymbol{Y}^{\top}\boldsymbol{L}\boldsymbol{Y})$ is $(\boldsymbol{L} + \boldsymbol{L}^{\top})\boldsymbol{Y}$. Therefore, both $\boldsymbol{L}$ and its transpose $\boldsymbol{L}^{\top}$ 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, $\boldsymbol{Y}$ is a $n \times m$ matrix. The projected gradient method [92] performs gradient descent step followed by projection step for T_p times. Each step has a naive complexity of $\mathcal{O}(n^2 m)$.[4] Thus, the overall complexity is $\mathcal{O}(n^2 m T_p T_{\text{joint}})$. 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.

5.3.2 Node-wise label assignment optimization

In order 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 in order 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.

Node-wise decomposition

In this section, we first generalize the energy in Equation (5.3) by replacing the $\frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$ term by a convex and symmetric function $\phi(\boldsymbol{y}_i, \boldsymbol{y}_j)$. After-

[4] This complexity can be improved to $\mathcal{O}(kmn)$ if the graph Laplacian is k-sparse, which is often the case.

wards, we decompose the global optimization problem in Equation (5.5) 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.

Formally, replacing the ℓ_2-norm by ϕ in Equation (5.3), we write the objective function in Equation (5.5) as

$$g(\boldsymbol{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left[\sum_{l=0}^{N}\alpha_l w_{ij}^{(l)} - w_{ij}^{(-)}\right]\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \equiv \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^{(\text{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j), \tag{5.6}$$

where we define $w_{ij}^{(\text{eff})} := \sum_{l=0}^{N}\alpha_l w_{ij}^{(l)} - w_{ij}^{(-)}$. Denoting $\widetilde{a}_{ij} := a_{ij} + a_{ji}$, we then isolate the contribution of the p-th node as

$$\begin{aligned} g(\boldsymbol{Y}) &= \sum_j w_{pj}^{(\text{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p}\sum_j w_{ij}^{(\text{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \\ &= \sum_j \widetilde{w}_{pj}^{(\text{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p}\sum_{j\neq p} w_{ij}^{(\text{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \qquad (5.7) \\ &= g_p(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n) + \sum_{i\neq p}\sum_{j\neq p} w_{ij}^{(\text{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \qquad (5.8) \end{aligned}$$

where we assume $\phi(\boldsymbol{y}_i, \boldsymbol{y}_i) = 0$ and $\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) = \phi(\boldsymbol{y}_j, \boldsymbol{y}_i)$ in Equation (5.7), and we introduce $g_p(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n) := \sum_j \widetilde{w}_{pj}^{(\text{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j)$ for brevity in Equation (5.8).

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

$$\boldsymbol{y}_i^{(k+1)} \begin{cases} = \boldsymbol{y}_i^{(k)} & \text{if } i \neq p, \\ \in \underset{\boldsymbol{y}\in\Delta_m}{\operatorname{argmin}}\, g_i(\boldsymbol{y}_1^{(k)}, \cdots, \boldsymbol{y}, \cdots, \boldsymbol{y}_n^{(k)}) & \text{if } i = p. \end{cases} \tag{5.9}$$

Then, by construction,

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

Therefore, we conclude that under the following assumptions:

- the loss function $\phi(\cdot, \cdot)$ is convex,

- the loss function is *coincident*[5], *i.e.*, $\phi(\boldsymbol{y}_i, \boldsymbol{y}_i) = 0$,
- and the loss function is *symmetric* with respect to its arguments, *i.e.*, $\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) = \phi(\boldsymbol{y}_j, \boldsymbol{y}_i)$,

the optimization step at any fixed node p

$$\begin{aligned} \boldsymbol{y}_p^{(k+1)} &\in \operatorname*{argmin}_{\boldsymbol{y} \in \Delta_m} g_p(\boldsymbol{y}_1^{(k)}, \cdots, \boldsymbol{y}, \cdots, \boldsymbol{y}_n^{(k)}) \\ &= \operatorname*{argmin}_{\boldsymbol{y} \in \Delta_m} \sum_j \widetilde{w}_{pj}^{(\text{eff})} \, \phi(\boldsymbol{y}, \boldsymbol{y}_j^{(k)}) \end{aligned} \tag{5.10}$$

monotonically decreases the objective function $g(\boldsymbol{Y})$. Equation (5.10) is still a DC problem and it can be solved by using *majorization-minimization* technique, as discussed in Section 5.3.1. It should 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 $g(\boldsymbol{Y}^{(k)})$ does not decrease any more. We summarize the overall process in Algorithm 5. Note that we do not assume anything about the structure of the graph, thereby allowing loops in the graph.

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 5.3.1. 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 $\mathcal{O}(n^2 m T_p T_{\text{joint}})$, the node-wise decomposition approach has an improvement of $\mathcal{O}(nT_{\text{joint}}/T_{\text{con}})$. Therefore, the improvement becomes significant when n increases, making it a better choice for large-scale problems as confirmed by our experiments.

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

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

Algorithm 5 Node-wise label assignment algorithm

Input
Weight matrices: $\{\boldsymbol{W}^{(l)}, l = 0, \cdots K\}, \boldsymbol{W}^{(-)}$
Scaling weights: $\{\alpha_l, l = 0, \cdots K\}$
Number of iterations: T_{con}
Output
Label assignment matrix: $\boldsymbol{Y}^\star$
Procedure
Set $\boldsymbol{W}^{(\text{eff})} \leftarrow \sum_l \alpha_l \boldsymbol{W}^{(l)} - \boldsymbol{W}^{(-)}$
Set $\widetilde{\boldsymbol{W}}^{(\text{eff})} \leftarrow \boldsymbol{W}^{(\text{eff})} + \boldsymbol{W}^{(\text{eff})\top}$
Choose initial solution, $\boldsymbol{Y}^{(1)} \in \mathcal{P}_{nm}$
Set $k \leftarrow 1$
For $t = 1, \cdots, T_{\text{con}}$
  Initialize $\mathcal{U} \leftarrow \mathcal{V}$
  While $\mathcal{U} \neq \emptyset$
    Select a node p from $\mathcal{U}$
    Solve $\tilde{\boldsymbol{y}} \leftarrow \underset{\boldsymbol{y} \in \Delta_m}{\text{argmin}} \sum_j \widetilde{W}_{pj}^{(\text{eff})} \phi(\boldsymbol{y}, \boldsymbol{y}_j^{(k)})$
    $\boldsymbol{Y}^{(k+1)} \leftarrow (\boldsymbol{y}_1^{(k)}, \cdots, \boldsymbol{y}_{p-1}^{(k)}, \tilde{\boldsymbol{y}}, \boldsymbol{y}_{p+1}^{(k)}, \cdots, \boldsymbol{y}_n^{(k)})^\top$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{p\}$
    $k \leftarrow k + 1$
  End While
End For
Return $\boldsymbol{Y}^\star \leftarrow \boldsymbol{Y}^{(T_{\text{con}})}$

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.

derive a condition under which the parallelization of the coordinate descent decreases the objective function.

Let us choose the set of nodes $\mathcal{J}$ for parallel coordinate descent. We represent its complement by $\bar{\mathcal{J}} := \mathcal{V} \setminus \mathcal{J}$. Then, we can decompose the objective function as [6]

$$g(\boldsymbol{Y}) = \sum_{i \in \mathcal{J}} g_i(\boldsymbol{Y}) - \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}} \widetilde{w}_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{5.11}$$

The negative terms in Equation (5.11) are called *interference* terms. To nullify these terms, we pickup 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

[6] Detailed derivation is provided in Appendix A.

write

$$g(\boldsymbol{Y}) = \sum_{i \in \mathcal{J}} g_i(\boldsymbol{Y}) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{5.12}$$

and solve the local optimization problem

$$\boldsymbol{y}_i^{(k+1)} \in \underset{\boldsymbol{y} \in \Delta_m}{\operatorname{argmin}} \, g_i(\boldsymbol{y}_1^{(k)}, \cdots, \boldsymbol{y}, \cdots, \boldsymbol{y}_n^{(k)}) \tag{5.13}$$

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

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

decreases monotonically the objective function, *i.e.*, $g(\boldsymbol{Y}^{(k+1)}) \leq g(\boldsymbol{Y}^{(k)})$. 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 5.6.2, we demonstrate the benefit of parallelization with a simple yet effective batch-based scheduling approach.

5.4 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 (5.2.1) 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. It then describes how labels are propagated through the incremented graph.

5.4.1 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 $\mathcal{G}^{(t-1)} = (\mathcal{V}^{(t-1)}, \mathcal{E}^{(t-1)}, \boldsymbol{W}^{(t-1)})$. As the graph evolves with time, it is implicit that the number of nodes n and the size of the label vector m are dynamic quantities. For the sake of simplicity, we write $n := n(t)$ and $m := m(t)$.

Since we have 3 different kinds of graphs, namely the spatio-temporal graph, the appearance graph(s) and the exclusion graph, the incrementation is different for each kind of graph. Nonetheless, 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.

In case of the 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.

For the spatio-temporal and the 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 and controls the size of the graph. 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

$$w_{ij} = \begin{cases} \exp(-\frac{1}{\sigma^2} d(\boldsymbol{f}_i, \boldsymbol{f}_j)^2) & \text{if } |t_i - t_j| \leq T_c, \\ 0 & \text{otherwise,} \end{cases} \tag{5.14}$$

where t_i and $\boldsymbol{f}_i$ denote the time instant and the features of the i-th node respectively, $d(\cdot,\cdot)$ measures the dissimilarity between the features $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$, 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 or different targets. 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.

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 unique 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 (5.14). 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 $\sum_{j\in\{\mathcal{N}_i\cup s\}} w_{ij} = 1$.

5.4.2 Label propagation in the incremented graph

After incrementing the graphs, we perform node-wise label propagation as defined by Algorithm 6. We denote the labels distribution over $\mathcal{G}^{(t)}$ after k iterations of the label propagation process by $\boldsymbol{Y}^{(t,k)}$. Moreover, $\boldsymbol{Y}^{(t)}$ 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 $\boldsymbol{Y}^{(t,1)}$, by augmenting the label distribution matrix at time $t-1$, denoted by a $n^{(t-1)} \times m^{(t-1)}$-dimensional matrix $\boldsymbol{Y}^{(t-1)}$, as follows:

$$\boldsymbol{Y}^{(t,1)} = \begin{pmatrix} \boldsymbol{Y}^{(t-1)} \mid \mathbf{0}_{n^{(t-1)}\times|\mathcal{D}^{(t)}|} \\ \boldsymbol{U}^{(t)} \end{pmatrix} \tag{5.15}$$

where $\mathbf{0}_{n^{(t-1)}\times|\mathcal{D}^{(t)}|}$ is a $n^{(t-1)} \times |\mathcal{D}^{(t)}|$-dimensional zero matrix and $\boldsymbol{U}^{(t)}$ is a $|\mathcal{D}^{(t)}| \times \left(|\mathcal{D}^{(t)}| + m^{(t-1)}\right)$-dimensional matrix such that $U_{ij}^{(t)} = 1/(|\mathcal{D}^{(t)}| + m^{(t-1)})$. Obviously, $\boldsymbol{U}^{(t)}$ 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

5.3.2,

$$\boldsymbol{y}_i^{(t,k+1)} \in \underset{\boldsymbol{y}\in\Delta_m^{(t)}}{\operatorname{argmin}}\, g_i(\boldsymbol{y}_1^{(t,k)}, \cdots, \boldsymbol{y}, \cdots, \boldsymbol{y}_m^{(t,k)}) + w_i^{(s)}\phi(\boldsymbol{y}, \boldsymbol{e}_i), \tag{5.16}$$

where $\boldsymbol{e}_i \in \Delta_m^{(t)}$ 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 it is close to the spatio-temporal border (*i.e.*, when $w_i^{(s)} \approx 1$).

Algorithm 6 Incremental graph construction and label propagation algorithm at time t

Input:
$\mathcal{D}^{(t)}$: detections at time t,
$\mathcal{G}^{(t-1)}$: graph at time $t-1$,
$\boldsymbol{Y}^{(t-1)}$: label assignment matrix at time $t-1$.
T_{con}: number of iterations for 'consensus'
Output:
$\mathcal{G}^{(t)}$: graph at time t,
$\boldsymbol{Y}^{(t)}$: label assignment matrix at time t.
Procedure:
$\mathcal{G}^{(t)} = \texttt{connectGraph}(\mathcal{G}^{(t-1)}, \mathcal{D}^{(t)})$ # *Refer to Section 5.4.1.*
Label propagation:
Initialize $\boldsymbol{Y}^{(t,1)}$.
Set $\boldsymbol{Z} \leftarrow \boldsymbol{Y}^{(t,1)}$.
For $k = 1, \cdots, T_{\text{con}}$
  For each node $i = 1, \cdots, n$
    $\tilde{\boldsymbol{z}} = \texttt{LocalLabelUpdate}(\boldsymbol{Z}, \mathcal{G}^{(t)})$
    $\boldsymbol{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{i-1}, \tilde{\boldsymbol{z}}, \boldsymbol{z}_{i+1}, \cdots, \boldsymbol{z}_n]^\top$
  End For
End For
Set $\boldsymbol{Y}^{(t)} \leftarrow \boldsymbol{Z}$

In Algorithm 6, `LocalLabelUpdate` corresponds to the local label optimization performed by each node as in Equation (5.16) respectively.

In order to bound the complexity of our incremental framework, and to turn it into to 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 (5.16), 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.

5.5 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 has been extensively studied in machine learning in the framework of semi-supervised learning approach, and a concise survey of recent developments in this field can be found in [93] 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 [94]. In [95], 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' graph 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 in order to assign a binary label to each node of the graph. Our method differs from [95] 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. Therefore, we adopt difference of convex programming approach to solve our problem.

Message passing. Message passing (belief propagation) approaches have been used to label the nodes in a graph in tracking/recognition scenario [64, 24] and in disparity estimation, image completion scenario [39]. 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 [64], a subset of the nodes is initially labelled and then a conditional random field 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 [24], the authors utilized such non-stationary and sporadic features in order 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. The exploitation of a specific constraint associated to the structure of the graph (*e.g.*, the exclusivity constraint associated to the detections that coexist in time) has been considered in [96, 97] in order 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 [98], 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 [85, 84], 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 [99], the authors devise a proximal optimization on graph that has quadratic convergence by using the Nesterov's method [100]. 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 [101] defines an out-of-sample mapping of the Laplacian eigenmaps. Given a graph, in which the weight of an edge $\boldsymbol{x}_i \sim \boldsymbol{x}_j$ is constructed as $w_{ij} := \exp(-\|\boldsymbol{f}_i -$

$\boldsymbol{f}_j\|_2^2/\sigma^2$), the latent points $\boldsymbol{Y}$ are the solution of

$$\begin{aligned} &\text{minimize} && \text{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}\boldsymbol{Y}) \\ &\text{subject to} && \boldsymbol{Y} \in \mathbb{R}^{n\times|\mathcal{L}|}, \boldsymbol{Y}^\top \boldsymbol{D}\boldsymbol{Y} = \boldsymbol{I}, \boldsymbol{Y}^\top \boldsymbol{D}\boldsymbol{1} = \boldsymbol{0}. \end{aligned}$$

where $\boldsymbol{D}$ is a diagonal matrix with its i-th diagonal element defined as $D_{ii} := \sum_j w_{ij}$, and $\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{W}$ is the graph Laplacian. When a new sample $\boldsymbol{f}$ arrives, [101] defines an out-of-sample mapping $F(\boldsymbol{f}) = \boldsymbol{y}$ for a new point $\boldsymbol{f}$ 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 [102]. 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 targets as it assumes that the new sample $\boldsymbol{f}$ belongs to one of the classes defined by $\mathcal{L}$. Moreover, it keeps the old "latent" distributions unchanged, which is not the case in our approach.

5.6 Evaluation

The proposed algorithm has been evaluated on the following well-known and challenging datasets: APIDIS [6], PETS-2009 S2/L1 [82] and TUD Stadtmitte [103]. APIDIS is a multi-camera sequence acquired during a basketball game, whereas the other two are monocular sequences.

5.6.1 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 RAM. 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 advantages are twofold. On the one hand, it reduces the number of nodes in the graph, thereby reducing the complexity of the algorithm. On the other hand, it helps to aggregate the appearance feature(s) along the tracklet in order to infer the appearance more accurately. 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 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 is, we label a track to be a false positive if one of the two criteria is satisfied:

- the track-length, defined as the number of detections along the track, is less than 10 frames,
- the track is primarily composed of low confidence detections. This is done by checking if the maximum confidence value along the track is less than 0.8. This case is prevalent in PETS and TUD datasets.

We employ the above heuristics because of the fact that false tracks that result from consistent false positive detections are usually shorter than regular target tracks and that the false positive detections have lower confidence values, compared to the true detections.

A glimpse of running times is shown in Table 5.1.

	Time taken			
	Low-level	Graph	Label propagation	
Dataset	tracker	construction	Joint	Nodewise
TUD	-	2 min	3 min	25 sec
PETS	-	3 min	40 min	5 min
APIDIS	15 sec	1 min	5 min	1 min

Table 5.1: Time taken by various stages of the algorithm on the examined datasets.

5.6.2 Results

In this section, we first present the tracking results for our label propagation 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. Lastly, some qualitative results are presented.

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 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 $\alpha_1 = 0.5$.[7]

We compare our results with several methods such as the continuous energy (CE) minimization [85], the discrete-continuous (D-C) minimization [84], the GMCP tracker [60], the K-shortest paths (KSP) [26], the global appearance constraints (GA) [27] and the iterative hypothesis testing (IHT) [23]. The CE and D-C trackers estimate the most probable trajectories by minimizing their energies. These energy terms 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. The GMCP tracker solves greedily a generalized minimum clique problem to extract tracklets that have the most stable appearance features and the most consistent motion. KSP tracker 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 long-range appearance information. IHT embeds an hypothesis testing strategy into a greedy shortest-path computation procedure to exploit the appearance features that are unreliable and/or sporadically available.

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

In Tables 5.2 and 5.3, we first observe that the joint and node-wise label optimization approaches give similar performances. Regarding the comparison with previous approaches, note that KSP, CE and D-C do not exploit appearance features. Hence, we compare them to the first version of our approach, which does not use the appearance features. Similarly, since GA, IHT and GMCP exploit the appearance features, we compare them to the second version of our approach.

The tracking results for the TUD Stadtmitte dataset are presented in Table 5.2. From Table 5.2, we see that our method is better than previous meth-

Method	MOTA	MOTP	SW
Continuous energy [85]	60.5	65.8	7
Discrete-continuous (D-C) [84]	61.8	63.2	4
GMCP tracker [60]	77.7	63.4	**0**
Joint (no appearance)	62.6	73.5	17
Joint (with appearance)	79.3	**73.9**	4
Node-wise (no appearance)	63.0	73.6	16
Node-wise (with appearance)	**79.6**	73.9	4

Table 5.2: Performance on the TUD Statdmitte dataset. The D-C and GMCP results are copied from [84] and [60].

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

The results on the PETS-2009 S2/L1 dataset are presented in Table 5.3.

From Table 5.3, again we observe that our proposed approach outperforms most contemporary 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 in order 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 5.4. Since GA and IHT are the only methods from the literature that are able to exploit sporadic

Method	MOTA	MOTP	SW
Discrete-continuous (D-C) [84]	89.30	56.40	-
Continuous energy [85]	81.84	73.93	15
KSP [26]	80.00	58.00	28
GMCP tracker [60]	90.30	69.02	8
GAC [27]	81.46	58.38	19
IHT [23]	83.0	**74.0**	-
Joint (no appearance)	82.75	71.21	25
Joint (with appearance)	91.01	70.99	**5**
Node-wise (no appearance)	83.0	71.23	25
Node-wise (with appearance)	**91.03**	71.00	**5**

Table 5.3: Tracking results on the PETS 2009-S2/L1 dataset. The D-C, IHT, GMCP and GA results are obtained from [84, 23, 60, 27].

appearnace 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 $\alpha_0 = 1, \alpha_1 = 0.1, \alpha_2 = 0.5$.

Method	MOTA	MOTP	SW
IHT (no appearance) [23]	85.83	60.83	18
IHT (color+digit) [23]	**86.19**	**60.90**	**12**
GA (no appearance) [27]*	72.91	53.13	108
GA (color+digit) [27]*	73.07	53.15	110
Joint (no appearance)	81.25	57.13	49
Joint (color+digit)	83.80	60.01	45
Node-wise (no appearance)	81.3	57.15	49
Node-wise (color+digit)	83.82	60.00	45

Table 5.4: Results on the APIDIS dataset (1500 frames). The tracking results have been provided by the authors of [23, 27]. [*] Since the detection results for [27] are different than that for the [23] and ours, we relax the distance threshold to 40 cm (from 30 cm) for the tracking results of [27].

As mentioned earlier, the detector used in [27] is different from the one used in [23], and in our approach. Based on the MOTP scores of each method, we suspect that the detector used in [27] is less accurate than the one used in [23] and in our approach. Therefore, for a fairer comparison, we relaxed the distance threshold for [27] from 30 cm (a value recommended for the APIDIS

dataset) to 40 cm while computing the MOTA metrics. To better understand the impact of this threshold, we provide a detailed comparison of the three approaches for different values of the threshold in Table 5.5.

		Distance Threshold								
		30 cm			40 cm			50 cm		
		MOTA	MOTP	SW	MOTA	MOTP	SW	MOTA	MOTP	SW
GA [27]	no appearance	47.86	46.50	98	72.91	53.13	108	83.66	59.41	70
	color+digit	47.81	46.51	106	73.07	53.15	110	84.00	59.47	69
IHT [23]	no appearance	85.83	60.83	18	93.34	68.42	18	96.16	73.84	19
	color+digit	86.19	60.90	12	93.45	68.51	12	96.26	73.79	12
DLP (joint)	no appearance	81.25	57.13	49	85.50	73.14	50	87.86	77.40	52
	color+digit	83.80	60.01	45	86.71	73.81	45	88.08	77.56	46
DLP (incremental)	no appearance	74.40	54.20	52	82.47	70.15	56	85.87	76.32	57
	color+digit	80.23	58.45	47	84.58	72.19	51	86.54	74.45	54

Table 5.5: Comparison of performance on the APIDIS dataset for different distance thresholds.

Even though our approach performs significantly better than GA [27], the results are slightly worse than IHT [23]. 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 iterative hypothesis testing 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, the iterative hypothesis testing framework associates two nodes only when the connection is sufficiently reliable than alternative connections. This prevents potential track switches. This is well-reflected by the switching errors.

Tracking results for incrementally constructed graphs

We constructed the graph as described in Section 5.4.1 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 5.6. 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.

In order to trade-off the complexity with the quality of the incremental

	MOTA	MOTP	SW
PETS dataset			
No appearance	79.32	70.70	26
With appearance	86.56	71.40	6
TUD dataset			
No appearance	61.60	73.30	13
With appearance	77.20	73.40	2
APIDIS dataset			
No appearance	74.40	54.20	52
With appearance (color+digit)	80.23	58.45	47

Table 5.6: Results of the incremental graph construction and label propagation approach.

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 5.2 for the TUD Statdmitte dataset. As we can see, the processing time monotonically increases with the size of the observation window. However, this is not the case with tracking accuracy. It initially increases with respect to the observation window but saturates after some value. Alternatively, one could define other heuristic in order 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.

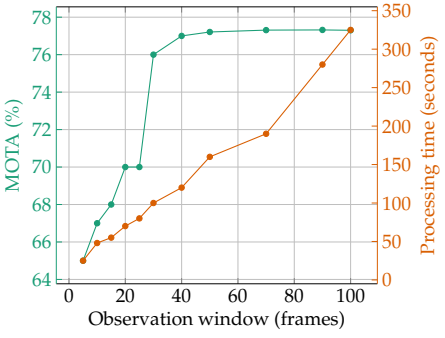


Figure 5.2: **Trade-off between the processing time and the tracking accuracy** for different observation window size for the TUD Stadtmitte dataset. *Best viewed in color.*

Computational advantages of the node-wise decomposition and parallelization

In order 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 5.3. 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.

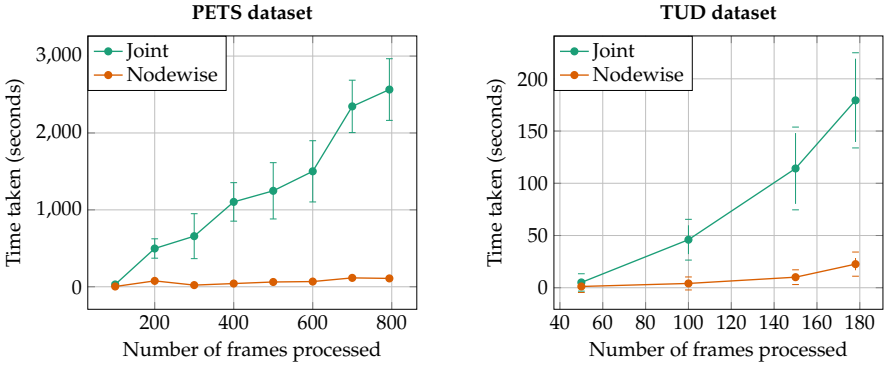


Figure 5.3: **Processing times for the joint and the node-wise approaches** for different size of the graph. *Best viewed in color.*

To assess the advantages offered by the parallel implementation, we consider a simple scheduling strategy. We assume batch processing of nodes, and Algorithm 7 presents a simple (yet effective) method to choose the nodes to be processed in parallel by the P processors at each round of the batch process. We represent the neighbors of a node i as $\mathcal{N}_i$, *i.e.*, $\mathcal{N}_i := \{j | \widetilde{w}_{ij}^{(\text{eff})} \neq 0\}$. Let $\mathcal{U}$ and $\mathcal{J}$ respectively denote the list of the unprocessed nodes and the set of nodes to be processed at the next batch. Initially, we set $\mathcal{U} = \mathcal{V}$. Our node selection strategy is directly based on the non-inference condition, derived in Section 5.3.2, and selects non-interfering nodes at random.

For each number of processor, we ran the algorithm 10 times and noted

Algorithm 7 Node selection strategy for parallelization

Initialize list of unprocessed nodes, $\mathcal{U} \leftarrow \mathcal{V}$.
While $\mathcal{U} \neq \emptyset$
  Initialize $\mathcal{J} \leftarrow \emptyset$.
  Initialize list of available nodes, $\mathcal{A} \leftarrow \mathcal{U}$.
  For $k = 1, ..., P$
    If $\mathcal{A} \neq \emptyset$
      Select a node i from $\mathcal{A}$ at random.
      $\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$.
      $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i \cup \mathcal{N}_i\}$.
    End If
  End For
  Solve the problem in Equation (5.13) in parallel.
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{J}$
End While

Note: $\mathcal{J}$ might not always contain P nodes at each batch. This is mainly due to the fact that if $\mathcal{J}$ contains node i for parallel processing, we cannot choose any node $j \in \mathcal{N}_i$. In our experiments, we observe that $|\mathcal{J}| = P$ only 98% of the time.

the evolution of objective function along time. The results are depicted in Figure 5.4. Note that the time is different from Figure 5.3. This is because of the fact that the parallel implementation is done in C++.

Although we can see that the computational time gets reduced, 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 $|\mathcal{J}|$ nodes. As a consequence, the time required to process a batch is governed by the longest time taken to process by one of its nodes. This is illustrated in Figure 5.5.

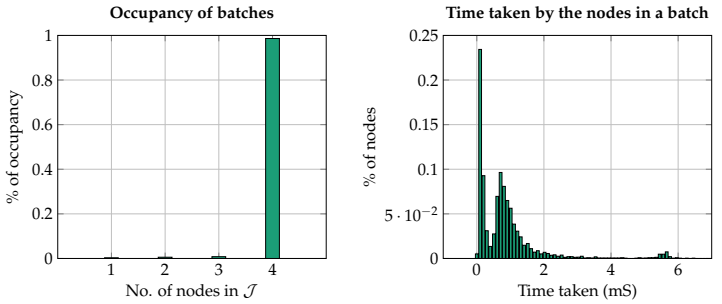


Figure 5.5: **Occupancy and time taken by the nodes in a batch.** (**Left:**) we see that $\mathcal{J}$ contains $P = 4$ nodes almost 98% of the time. (**Right:**) Some of the nodes in the batch take longer time than others. These nodes pose the bottleneck to the speed-up factor. *Best viewed in color.*

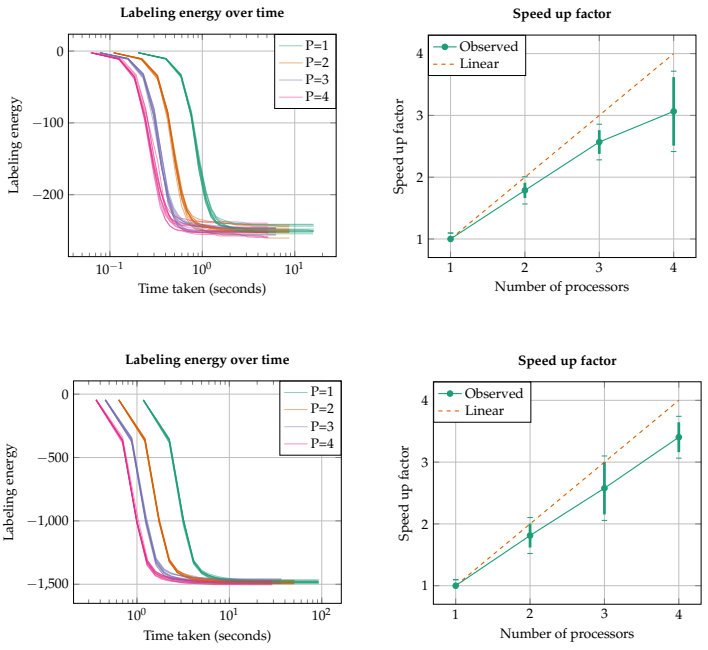


Figure 5.4: **Processing time and speed-up factors of the node-wise label propagation** for different number of processors ($P = 1$ to $P = 4$). For each case, we perform 10 runs of the algorithm which are drawn with the same color. **Top row**: TUD dataset, **bottom row**: PETS dataset. *Best viewed in color.*

Effect of parameters

Our algorithm has several parameters. They are:

γ	Scaling factor for time (Section 2.2)
T	Connection window size (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 (Section 2.2)
σ	'Heat' parameter (Section 4.2)
T_o	Observation window for bounding complexity (Section 4.2)

The effect of α_l and T_o have already been discussed in Section 5.6.2. In this section, we present and discuss how these parameters are chosen and/or what are their effects on the performance.

The MOTA for the quiescent point ($\gamma = 3, t = 10, v_{\max} = 10$) for the TUD dataset is 77.2%. In this section, we present and discuss the effect of these

parameters individually. For this, only one parameter is changed at a time and all other parameters are fixed at their quiescent values.

In Table 5.7, we present the effect of T, γ and $v_{\max}$ on the MOTA as well as the time taken for graph construction step for TUD dataset. Increasing $v_{\max}$ typically reduces the time to construct the graph as it will discard 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. 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, large T not only is more prone to wrong connections but also might not satisfy the linear motion model assumption. Interestingly, γ does not seem to affect MOTA much. However, we have observed that setting a large γ restricts the number of neighboring nodes that remain eligible for non-zero weights, which in practice will sharply reduce the graph construction time. For example, graph construction step for ($\gamma = 1, T = 100$) was around 10 times slower than that for ($\gamma = 7, T = 100$).

	T					γ				$v_{\max}$			
	5	**10**	**20**	**30**	**100**	**1**	**3**	**5**	**7**	**5**	**10**	**20**	**30**
MOTA (%)	76.9	77.2	76.7	76.3	70.2	77.1	77.2	77.3	77.2	74.3	77.2	75.8	65.6
Time (Seconds)	1.03	1.33	1.48	1.42	31.46	1.25	1.33	1.17	1.28	1.27	1.33	1.37	1.18

Table 5.7: Effect of parameters on the TUD dataset.

We report the effect of connection window size on the performance in Table 5.8.

	T_c, frames	50	100	200	∞
PETS	**MOTA (%)**	84.32	86.52	86.56	83.95
	Time (Seconds)	300	390	480	918
TUD	**MOTA**	70.11	72.86	73.40	69.82
	Time (Seconds)	78	126	150	312

Table 5.8: Effect of connection window size for the appearance graph on MOTA and overall (graph construction and label optimization) time. $T_c = \infty$ means that all the past detections have been considered.

From Table 5.8, we observe that increasing T_c increases the computation

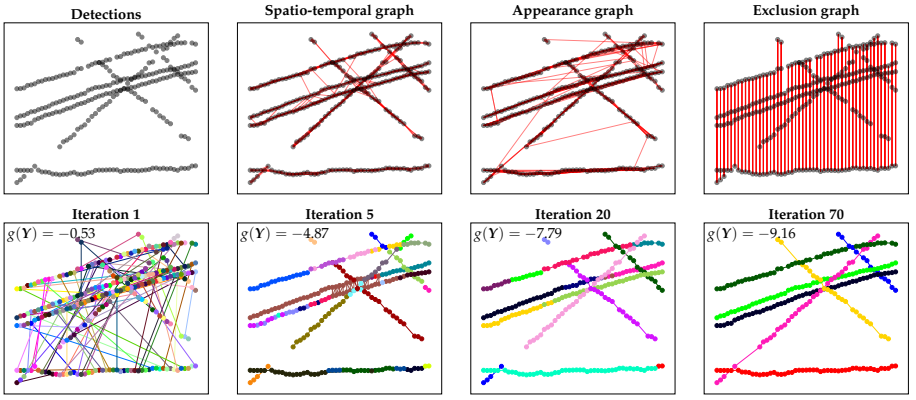


Figure 5.6: **Sample graphs and label evolution on a subset of detections from PETS dataset**. The abscissa and the ordinate correspond to time and position respectively. The top row shows the input detections and the three constructed graphs. For clarity, edges that have weights smaller than 10^{-2} are suppressed. The bottom row depicts the evolution of label of the nodes along with the corresponding labeling energy. Initially, each node is assigned a random label distribution. As the iteration progresses, the nodes are labelled consistently. *Best viewed in color.*

time. However, the MOTA is improved only up to some value (200 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.

Qualitative results

Now, we present some qualitative results[8]. Figure 5.6 depicts the detections, constructed graphs and the inferred labels.

Some sample frames of our tracking results are presented in Figure 5.7. In case of the APIDIS dataset, the frames from camera 1 and 6 are stitched in order to provide an entire view of the field.

Two typical failure cases in our tracking system are depicted in Figures 5.8 and 5.9. In Figure 5.8, an identity switch is depicted. The identities of two targets are momentarily switched. This might be because of the fact that we do not consider the appearance feature if the overlap between their bounding boxes exceeds 5%. Therefore, neither the position nor the appearance disambiguates the identities of the targets. Later on, when the targets are separated,

[8] A demo video is available at http://sites.uclouvain.be/ispgroup/index.php/Research/MultiObjectTracking.

Figure 5.7: **Sample frames** from the PETS2009-S2/L1 (first row), the TUD Stadmitte (second row) and the APIDIS (third and fourth rows) datasets. For the sake of clarity, a tail of 50 frames is added. The numbers represent the distinct IDs of the tracks. *Best viewed in color.*

the algorithm is able to assign the correct label to the targets.

Figure 5.8: **Instantaneous identity switch.** In frame 703, targets 18 and 22 come close. Their bounding boxes overlap significantly in the frame 708 and their identities are momentarily switched. Afterwards, the targets separate and their identities are retained in frames 710 and 712. *Best viewed in color.*

Figure 5.9 shows an example of false positive. This happens because of the spurious detections provided by the object detector. Our current approach does not model such spurious detections, which are typically characterized by their low confidence values, explicitly.

Figure 5.9: **False positive**. A false target 3 appears in frame 7073 and lasts until frame 7094. *Best viewed in color.*

5.7 Conclusion and future works

In this chapter, we have focused on the problem of multi-object tracking 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, the multi-object tracking has been formulated as a consistent labeling problem in the associated graphs. The proposed solution is based on DC (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.

Conclusions and Future Works

6

6.1 Conclusions

Multi-object tracking (MOT) is an important issue in computer vision. It has numerous applications in multiple disciplines such as player tracking in sport analysis, people's tracking in surveillance, cell movement in biology, crack evolution in material science, etc.

We focused on detection-based multi-object tracking approach. In such approach, objects-of-interest are first detected at each time instant. Afterwards, MOT is formulated as the problem of linking these detections into tracks of single targets using location and appearance features using graph-based formalisms. Previous graph-based methods impose strict assumptions about the features reliability and availability, resulting in a simplified formulation. These simplifications are not applicable in the scenarios for which the level of noise affecting the feature observation process is not constant for al detections, resulting in features affected by non-stationary noise or even sporadic in time and space.

In this thesis, we have addressed the MOT problem in presence of such unreliable features. First, we formulated the problem as an *iterative hypothesis testing* (IHT) strategy in Chapter 3. A node was selected to define a possible target appearance hypothesis. Given this hypothesis, a shortest-path computation was performed to check if the node could be aggregated to its neighboring nodes unambiguously. The criteria for checking the ambiguity of the shortest-path was relaxed progressively. We observe that the proposed IHT is not only effective in exploiting such noisy and/or sporadic appearance features, but also computationally efficient due to its multi-scale strategy.

Chapter 4 builds on top of the IHT and addresses the problem of assigning labels (or, identities) to the tracklets in the context of sport player recognition.

As the appearance features cannot be detected with the same reliability along the sequence, the identity of a tracklet, inferred from the accumulated appearance features, is not always unambiguous. In order to infer the identities of the ambiguous tracklets, messages are exchanged between the tracklets (or, nodes). We have shown that the order in which messages are exchanged between the tracklets affects the quality of the solution. Specifically, we have proposed to prioritize the nodes so that (i) less ambiguous nodes transmit their messages first, and (ii) messages are gathered only from the less ambiguous nodes. We have observed that such prioritizing not only converges faster but also provides better recognition rate.

In Chapter 5, we have formulated the MOT problem as a labeling problem in a number of complementary graphs. First of all, a number of graphs is constructed by exploiting various relationships between the detections such as spatial proximity, appearance similarity and exclusivity. Afterwards, labels are assigned to the nodes such that the nodes that are similar in terms of space and appearance are labeled similarly, whereas the nodes that violate spatio-temporal constraints (*e.g.*, nodes that co-exist at the same time) receive different labels. Our formulation leads to a difference of convex problem that can be solved by the *majorization-minimization* technique. Afterwards, an efficient solution to the labeling problem has been proposed by decomposing the big problem into small node-wise problems that can be solved locally. The decomposition not only reduces the computational complexity, but also supports an incremental and scalable graph construction. Moreover, it opens the possibility for a parallel implementation.

6.2 Future works

In this section, we first suggest possible improvements to each contribution of the thesis. Afterwards, we propose some interesting paths for further research.

We start with the iterative hypothesis testing (IHT) strategy, presented in Chapter 3. Currently, the inference of the tracklet appearance is done by weighted average of the appearances of individual detections within the tracklet (refer to Equation 3.6). This is based on the assumption that the appearance features are affected by Gaussian noise. We would like to stress that this inference should be adapted to the characteristics of the feature observation process. For example, when the appearance features of the detections are affected

by outliers, a RANSAC-like approach should be used. Moreover, the appearance dissimilarity of a node with respect to the key-node is computed as the ℓ_2-distance between them (refer to Equation 3.7). Obviously, this definition of the distance does not generalize to all kinds of features and it needs to be defined according to the feature type. For example, when region covariance [104] is considered to describe the appearance, the distance between two region covariance matrices cannot be effectively described by ℓ_2-distance. Based on these two observations, we suggest to adapt the tracklet appearance inference step and appearance dissimilarity definition with the tracking scenario at hand.

The discriminative label propagation (DLP) framework, presented in Chapter 5, can be improved in several ways. Some of them are:

- **Embedding higher order motion models.** By embedding the higher order motion models during graph construction step, false detections can be handled more effectively as they are generally incoherent with motion models.

- **Using confidence levels in a continuous manner.** Currently, the graph construction step turns the variable reliability of the features into sporadic measurements by hard thresholding. By exploiting the feature confidence values, the label propagation can be improved.

- **Exploiting exclusivity based on appearance information.** Currently, the exclusion graph is constructed based on the spatio-temporal information only. It does not exploit the fact that two detections cannot be assigned to the same target when they differ in appearance. Incorporating exclusivity from appearance information is relevant to certain applications such as sports. In sports, the jersey color can be often extracted reliably and the color dissimilarity can be used to assert the fact that two detections with different jersey colors cannot be labeled similarly.

In the remainder of the section, we discuss about the possible directions for further research.

Deployment of our tracking solutions. Our proposed solutions can be deployed in several scenarios such as biological cell tracking, evolution of cracks in metals under tension, sport player tracker, etc. Each of these scenarios have some specificities to be addressed. For example, merging and splitting of cells are interesting to the biologists for lineage construction, cell evolution event detection (birth, division, death, etc.). Since our IHT works with hypotheses

and is computationally efficient, we believe that it can be adapted to handle these specificities easily and track numerous cells over a long period of time.

Learning of target appearances for long-term tracking. To track targets reliably over long time, a tracker should learn the target appearances discriminatively. Starting with a pre-defined pool of features, a subset of features and cut-points that discriminate the targets are selected by using, for example, AdaBoost [98] or random fern classifier [105]. Random ferns are shown to be more robust than AdaBoost subject to wrong training samples [106]. The training samples are defined automatically, by exploiting structural constraints, like 'detections that occur at the same time-instant must be different', or 'unambiguously matched detections should be same', etc. as in [96, 97]. Such discriminatively learned appearance features can be directly integrated into IHT framework to compute the appearance (dis)similarity of a node with respect to the key-node appearance. Moreover, the appearance graph construction step in DLP can benefit from such appearance features too.

Learning of graphs. Most of the graph-based MOT solutions are design-based, *i.e.*, the spatial and appearance distances are combined in some ad-hoc way. Due to recent availability of many (diverse) MOT datasets [1,2], it is timely and relevant to learn the graph(s) and to infer the tracks simultaneously. We can envision to extend both IHT and DLP to learn the parameters.

[1] www.motchallenge.net

[2] http://www.codesolorzano.com/celltrackingchallenge/Cell_Tracking_Challenge/Welcome.html

Derivations

In this appendix, we provide detailed derivation/explanation of

- solution of the DC program using majorization-minimization technique (Section A.1),
- node-wise decomposition of the global objective function (Section A.2), and
- non-interference criterion of the parallelization (Section A.3).

These concepts have been presented in Chapter 5.

A.1 Majorization-minimization

In this appendix, we explain how we can minimize a DC problem using majorization minimization technique. We write our objective function as

$$\begin{aligned} g(\boldsymbol{Y}) &= \sum_{l=0}^{K} \alpha_l \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}_l^{(+)} \boldsymbol{Y}) - \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}^{(-)} \boldsymbol{Y}) \\ &:= f(\boldsymbol{Y}) - h(\boldsymbol{Y}) \end{aligned}$$

where $f(\boldsymbol{Y}) := \sum_{l=0}^{K} \alpha_l \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}_l^{(+)} \boldsymbol{Y})$ and $h(\boldsymbol{Y}) := \mathbf{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}^{(-)} \boldsymbol{Y})$ and both convex in $\boldsymbol{Y}$. We can 'majorize' $g(\boldsymbol{Y})$ using convexity of $h(\boldsymbol{Y})$ as

$$\begin{aligned} g(\boldsymbol{Y}) &= f(\boldsymbol{Y}) - h(\boldsymbol{Y}) \\ &\leq f(\boldsymbol{Y}) - \left[h(\boldsymbol{Y}^{(k)}) + \nabla h^\top(\boldsymbol{Y}^{(k)})(\boldsymbol{Y} - \boldsymbol{Y}^{(k)}) \right] \\ &= f(\boldsymbol{Y}) - \nabla h^\top(\boldsymbol{Y}^{(k)}) \boldsymbol{Y} - h(\boldsymbol{Y}^{(k)}) \\ &\quad + \nabla h^\top(\boldsymbol{Y}^{(k)}) \boldsymbol{Y}^{(k)} \\ &:= \hat{g}(\boldsymbol{Y}, \boldsymbol{Y}^{(k)}) \end{aligned} \tag{A.1}$$

where the inequality in Equation (A.1) is due to the convexity of $h(\boldsymbol{Y})$. We can see that $g(\boldsymbol{Y}) \leq \hat{g}(\boldsymbol{Y}, \boldsymbol{Y}^{(k)}), \forall \boldsymbol{Y} \in \mathcal{P}_{nm}$ with the equality holding when $\boldsymbol{Y} = \boldsymbol{Y}^{(k)}$. In the literature, $\hat{g}(\boldsymbol{Y}, \boldsymbol{Y}^{(k)})$ is called the *majorization* of $g(\boldsymbol{Y})$ [91]. Since we have $\nabla \hat{g}(\boldsymbol{Y}) = \nabla f(\boldsymbol{Y}) - \nabla h(\boldsymbol{Y}^{(k)})$, the Lipschitz constant of $\nabla \hat{g}$, written in short as $\mathrm{LC}(\nabla \hat{g})$, is same as that of $\nabla \hat{f}$, *i.e.*, $\mathrm{LC}(\nabla \hat{g}) = \mathrm{LC}(\nabla f)$.

From Equation A.1, the solution

$$\boldsymbol{Y}^{(k+1)} = \underset{\boldsymbol{Y} \in \mathcal{P}_{nm}}{\operatorname{argmin}} \, \hat{g}(\boldsymbol{Y}, \boldsymbol{Y}^{(k)}) \tag{A.2}$$

follows the inequality

$$g(\boldsymbol{Y}^{(k+1)}) \leq \hat{g}(\boldsymbol{Y}^{(k+1)}, \boldsymbol{Y}^{(k)}) \leq \hat{g}(\boldsymbol{Y}^{(k)}, \boldsymbol{Y}^{(k)}) = g(\boldsymbol{Y}^{(k)}), \tag{A.3}$$

where the first inequality and the last equality follow from Equation (A.1) and the second inequality follows from Equation (A.2). Therefore, above iterate in Equation (A.2) monotonically decreases $g(\boldsymbol{Y})$. In order to solve the convex problem in Equation (A.2), we use the projected gradient method [92], which is a special forward-backward splitting method [107], as

$$\boldsymbol{Y}^{(l+1)} = \mathbf{Proj}_{\mathcal{P}} \left(\boldsymbol{Y}^{(l)} - \beta_l \nabla \hat{g}(\boldsymbol{Y}^{(l)}, \boldsymbol{Y}^{(k)}) \right), \tag{A.4}$$

where **Proj** is the projection onto the probability simplex $\mathcal{P}$, and β_l is the step size that can be fixed or determined by the line search. We can set $\beta_l \leq 1/\mathrm{LC}(\nabla \hat{g}) = 1/\mathrm{LC}(\nabla f) = 1/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{L}_{\text{eff}}^{(+)} := \sum_{l=0}^{K} \alpha_l \boldsymbol{L}_l^{(+)}$.

Alternatively, a DC program can also be minimized by using proximal methods [108]. We refer the readers to [108] for details.

A.2 Node-wise decomposition of the global objective function

In this section, we derive our node-wise decomposition of the objective function

$$g(\boldsymbol{Y}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left[\sum_{l=0}^{K} \alpha_l w_{ij}^{(l)} - w_{ij}^{(-)} \right] \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2. \tag{A.5}$$

Formally, replacing the ℓ_2-norm by a convex loss function ϕ in Equation (A.5), we write the objective function as

$$g(\boldsymbol{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left[\sum_{l=0}^{K}\alpha_l w_{ij}^{(l)} - w_{ij}^{(-)}\right]\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \equiv \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j), \tag{A.6}$$

where we define $w_{ij}^{(\mathrm{eff})} := \sum_{l=0}^{K}\alpha_l w_{ij}^{(l)} - w_{ij}^{(-)}$.

We would like to isolate the contribution of the p-th node as

$$\begin{aligned} g(\boldsymbol{Y}) &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \\ &= \sum_{j=1}^{n} w_{pj}^{(\mathrm{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p}\sum_{j=1}^{n} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) && \text{(A.7)} \\ &= \sum_{j=1}^{n} w_{pj}^{(\mathrm{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p}\left[w_{ip}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_p) + \sum_{j\neq p} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j)\right] && \text{(A.8)} \\ &= \underbrace{\sum_{j=1}^{n} w_{pj}^{(\mathrm{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p} w_{ip}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_p)}_{g_p(\boldsymbol{Y})} + \sum_{i\neq p}\sum_{j\neq p} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) && \text{(A.9)} \end{aligned}$$

In Equation (A.9), the second term is independent of $\boldsymbol{y}_p$ and we define $g_p(\boldsymbol{Y})$ as

$$g_p(\boldsymbol{Y}) := \sum_{j=1}^{n} w_{pj}^{(\mathrm{eff})}\phi(\boldsymbol{y}_p, \boldsymbol{y}_j) + \sum_{i\neq p} w_{ip}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i, \boldsymbol{y}_p) \tag{A.10}$$

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

$$\boldsymbol{y}_i^{(k+1)} \begin{cases} = \boldsymbol{y}_i^{(k)} & \text{if } i \neq p, \\ \in \operatorname*{argmin}_{\boldsymbol{y}\in\Delta_m} g_i(\boldsymbol{y}_1^{(k)}, \cdots, \boldsymbol{y}, \cdots, \boldsymbol{y}_n^{(k)}) & \text{if } i = p. \end{cases} \tag{A.11}$$

Then, by construction,

$$\begin{aligned} g(\boldsymbol{Y}^{(k+1)}) &= g_p(\boldsymbol{Y}^{(k+1)}) + \sum_{i\neq p}\sum_{j\neq p} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) && \text{(A.12)} \\ &\leq g_p(\boldsymbol{Y}^{(k)}) + \sum_{i\neq p}\sum_{j\neq p} w_{ij}^{(\mathrm{eff})}\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) && \text{(A.13)} \\ &= g(\boldsymbol{Y}^{(k)}). && \text{(A.14)} \end{aligned}$$

That means, the node-wise minimization will monotonically decrease the objective function. This approach is similar to the co-ordinate descent method.

It should be noted that we have not assumed anything in Equation (A.10), except the convexity of ϕ. To proceed, we make following assumptions:

- The loss function ϕ is coincident, *i.e.*, $\phi(\boldsymbol{y}_i, \boldsymbol{y}_i) = 0$.
- The loss function is symmetric with respect to its arguments, *i.e.*, $\phi(\boldsymbol{y}_i, \boldsymbol{y}_j) = \phi(\boldsymbol{y}_j, \boldsymbol{y}_i)$.

Then, we can simplify Equation (A.10) as

$$\begin{aligned} g_p(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n) &= \sum_{j=1}^{n} (w_{pj}^{(\text{eff})} + w_{jp}^{(\text{eff})}) \phi(\boldsymbol{y}_p, \boldsymbol{y}_j) \\ &= \sum_{j=1}^{n} \widetilde{w}_{pj}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_j), \end{aligned} \tag{A.15}$$

where $\widetilde{w}_{pj}^{(\text{eff})} := w_{pj}^{(\text{eff})} + w_{jp}^{(\text{eff})}$. It should be noted that $g_p(\boldsymbol{Y})$ is a difference of convex (DC) function which can be minimized by using majorization minimization [91] or proximal methods [108].

A.3 Parallel implementation

In this section, we derive the non-interference criterion for the parallelization.

Recalling Equation (A.6), we write our objective function as

$$g(\boldsymbol{Y}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j)$$

As discussed earlier, we can decompose the objective function around node p as

$$g(\boldsymbol{Y}) = g_p(\boldsymbol{Y}) + \sum_{i \neq p} \sum_{j \neq p} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.16}$$

where $g_p(\boldsymbol{Y}) := \sum_j \widetilde{w}_{pj}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_j)$ and $\widetilde{w}_{pj}^{(\text{eff})} := w_{pj}^{(\text{eff})} + w_{jp}^{(\text{eff})}$.

Let us decompose the second term around node $q \neq p$. Then,

$$\sum_{i \neq p} \sum_{j \neq p} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.17}$$

$$= \sum_{j \neq p} w_{qj}^{(\text{eff})} \phi(\boldsymbol{y}_q, \boldsymbol{y}_j) + \sum_{i \notin \{p,q\}} \sum_{j \neq p} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.18}$$

$$= \sum_{j \neq p} w_{qj}^{(\text{eff})} \phi(\boldsymbol{y}_q, \boldsymbol{y}_j) + \sum_{i \notin \{p,q\}} w_{iq}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_q) + \sum_{i \notin \{p,q\}} \sum_{j \notin \{p,q\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.19}$$

$$= \sum_{j \neq p} (w_{qj}^{(\text{eff})} + w_{jq}^{(\text{eff})}) \phi(\boldsymbol{y}_q, \boldsymbol{y}_j) + \sum_{i \notin \{p,q\}} \sum_{j \notin \{p,q\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.20}$$

$$= g_q(Y) - \widetilde{w}_{qp}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_q) + \sum_{i \notin \{p,q\}} \sum_{j \notin \{p,q\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.21}$$

where we used the fact that $\phi(\boldsymbol{y}_q, \boldsymbol{y}_q) = 0$ in the second term of Equation (A.19) to proceed to Equation (A.20) and we use the fact that $g_q(\boldsymbol{Y}) := \sum_j \widetilde{w}_{qj}^{(\text{eff})} \phi(\boldsymbol{y}_q, \boldsymbol{y}_j)$ in Equation (A.21).

Consider the third term in Equation (A.21). We decompose it at node r as

$$\sum_{i \notin \{p,q\}} \sum_{j \notin \{p,q\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.22}$$

$$= g_r(\boldsymbol{Y}) - \widetilde{w}_{rp}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_r) - \widetilde{w}_{rq}^{(\text{eff})} \phi(\boldsymbol{y}_q, \boldsymbol{y}_r) + \sum_{i \notin \{p,q,r\}} \sum_{j \notin \{p,q,r\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.23}$$

Combining all the terms, we can write the objective function as

$$\begin{aligned} g(\boldsymbol{Y}) = g_p(\boldsymbol{Y}) + g_q(\boldsymbol{Y}) + g_r(\boldsymbol{Y}) - \big[\widetilde{w}_{pq}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_q) + \widetilde{w}_{rp}^{(\text{eff})} \phi(\boldsymbol{y}_p, \boldsymbol{y}_r) \\ + \widetilde{w}_{qr}^{(\text{eff})} \phi(\boldsymbol{y}_q, \boldsymbol{y}_r)\big] + \sum_{i \notin \{p,q,r\}} \sum_{j \notin \{p,q,r\}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \end{aligned} \tag{A.24}$$

We can generalize the decomposition of Equation (A.24) as

$$g(\boldsymbol{Y}) = \sum_{i \in \mathcal{J}} g_i(\boldsymbol{Y}) - \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}} \widetilde{w}_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j), \tag{A.25}$$

where $\mathcal{J}$ is the set of nodes at which the objective function has been decomposed, and $\bar{\mathcal{J}}$ is its complement.

The negative terms in Equation (A.25) are called *interference* terms. To nullify these terms, we should pickup the nodes in J such that there are no edges between the nodes in $\mathcal{J}$, *i.e.*, $\forall (p,q) \in \mathcal{J} \times \mathcal{J}, w_{pq}^{(\text{eff})} = w_{qp}^{(\text{eff})} = 0$. Then, Equation (A.25) becomes

$$g(\boldsymbol{Y}) \quad = \quad \sum_{i \in \mathcal{J}} g_i(\boldsymbol{Y}) + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \mathcal{J}} w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{A.26}$$

At each node $i \in \mathcal{J}$, we solve a local optimization problem as

$$\boldsymbol{y}_i^{(k+1)} \in \underset{\boldsymbol{y}_i \in \Delta_m}{\operatorname{argmin}}\, g_i(\boldsymbol{y}_1^{(k)}, ..., \boldsymbol{y}_{i-1}^{(k)}, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^{(k)}, ..., \boldsymbol{y}_n^{(k)}) \tag{A.27}$$

We denote by $\boldsymbol{Y}^{(k+1)} := [\boldsymbol{y}_1^{(k)}, ..., \boldsymbol{y}_p^{(k+1)}, ..., \boldsymbol{y}_q^{(k+1)}, ..., \boldsymbol{y}_r^{(k+1)}, ..., \boldsymbol{y}_n^{(k)}]^\top$ the label assignment matrix after having its $|\mathcal{J}|$ coordinate(s) updated in parallel. Then,

$$
\begin{aligned}
g(\boldsymbol{Y}^{(k)}) - g(\boldsymbol{Y}^{(k+1)}) &= \sum_{i \in \mathcal{J}} \left[g_i(\boldsymbol{Y}^{(k)}) - g_i(\boldsymbol{Y}^{(k+1)}) \right] \\
&\quad + \sum_{i \in \bar{\mathcal{J}}} \sum_{j \in \bar{\mathcal{J}}} \left[w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - w_{ij}^{(\text{eff})} \phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) \right] \\
&= \sum_{i \in \mathcal{J}} \left[g_i(\boldsymbol{Y}^{(k)}) - g_i(\boldsymbol{Y}^{(k+1)}) \right] \qquad \text{(A.28)}
\end{aligned}
$$

Consider

$$
\begin{aligned}
& g_i(\boldsymbol{Y}^{(k)}) - g_i(\boldsymbol{Y}^{(k+1)}) \\
&= \sum_{j} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k+1)}) \right] \\
&= \sum_{j \in \mathcal{J}} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k+1)}) \right] \\
&\quad + \sum_{j \in \bar{\mathcal{J}}} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k)}) \right] \qquad \text{(A.29)}
\end{aligned}
$$

In Equation (A.29), the first term is zero because we have chosen nodes i and j such that there are no edges between i and j, *i.e.*, $\widetilde{w}_{ij}^{(\text{eff})} = 0$. By the same argument, we can add a term $\sum_{j \in \mathcal{J}} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k)}) \right]$ in Equation (A.29). Thus, we have

$$
\begin{aligned}
g_i(\boldsymbol{Y}^{(k)}) - g_i(\boldsymbol{Y}^{(k+1)}) &= \sum_{j \in \mathcal{J}} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k)}) \right] \\
&\quad + \sum_{j \in \bar{\mathcal{J}}} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k)}) \right] \\
&= \sum_{j} \widetilde{w}_{ij}^{(\text{eff})} \left[\phi(\boldsymbol{y}_i^{(k)}, \boldsymbol{y}_j^{(k)}) - \phi(\boldsymbol{y}_i^{(k+1)}, \boldsymbol{y}_j^{(k)}) \right] \\
&\geq 0. \qquad \text{(A.30)}
\end{aligned}
$$

where the inequality follows because of the definition of $\boldsymbol{y}_i^{(k+1)}$ in Equation (A.27). Combining Equations (A.28) and (A.30), we conclude that $g(\boldsymbol{Y}^{(k)}) \geq g(\boldsymbol{Y}^{(k+1)})$. Therefore, the objective function decreases at each iteration and the overall decrement is sum of individual decrements.

Publications

B

This thesis has culminated into following publications:

1. **Amit Kumar K.C.**, Laurent Jacques and Christophe De Vleeschouwer, "*Discriminative and Efficient Label Propagation on Complementary Graphs for Multi-Object Tracking*", under revision in the Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2015. Available at http://arxiv.org/abs/1504.01124.

2. Liza Lecarme, Eric Maire, **Amit Kumar K.C.**, Christophe De Vleeschouwer, Laurent Jacques, Aude Simar and Thomas Pardoen,"*Heterogeneous void growth revealed by in situ 3-D X-ray mocrotomography using automatic cavity tracking*". Acta Materialia, Vol. 63, p. 130-139, 2014.

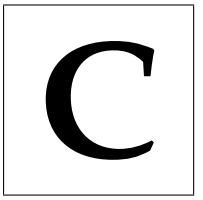
3. **Amit Kumar K.C.** and Christophe De Vleeschouwer, "*Discriminative Label Propagation for Multi-Object Tracking with Sporadic Appearance Features*", International Conference on Computer Vision (ICCV), 2013.

4. **Amit Kumar K.C.**, Damien Delannay, Laurent Jacques and Christophe De Vleeschouwer, "*Iterative hypothesis testing for multi-object tracking with noisy/missing appearance features*", Detection and Tracking in Challenging Environments (DTCE) workshop on 11th Asian Conference on Computer Vision (ACCV), 2012.

5. **Amit Kumar K.C.** and Christophe De Vleeschouwer, "*Prioritizing the propagation of identity beliefs for multi-object tracking*", British Machine Vision Conference (BMVC), 2012.

6. **Amit Kumar K.C.**, Pascaline Parisot and Christophe De Vleeschouwer, "*Demo: Spatio-temporal template matching for ball detection*", International Conference on Distributed Smart Cameras (ICDSC), 2011.

Datasets



The proposed algorithms have been evaluated on the following well-known and challenging datasets: APIDIS [6], PETS-2009 S2/L1 [82] and TUD Stadtmitte [103]. APIDIS is a multi-camera sequence acquired during a basketball game, whereas the other two are monocular sequences. In the following, we refer to PETS-2009 S2/L1 and TUD Stadtmitte by PETS and TUD respectively.

APIDIS is a 15 minute video dataset is generated by 7 cameras distributed around a basketball court. The frame rate is 25 fps. The candidate detections are computed independently at each time instant based on a ground occupancy map, as described in [18]. 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. 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 digit feature is obtained by running a digit-recognition algorithm [87] 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. These features are computed for all available cameras. Finally, the confidence of the feature is computed for each feature individually as

- for color feature, it is computed the ratio of number of cameras, for which features are computed, to the total number of available cameras.
- for digit feature, it is set to either zero or one. It is set to one if the digit-recognition algorithm decides that the digit is present inside at least one of the cameras. Otherwise, it is set to zero. Note that this might not correspond to the actual confidence in the estimated/recognized digit.

For example, an actual jersey number is 10 might be recognized by the digit-recognition algorithm as digit 0 but the confidence is set to 1.

The ground-truth is obtained from [6]. Some authors use a subset of this video. For example, Russel *et al.* [73] uses only 500 frames, whereas Ben Shitrit *et al.* [27] uses only 1 minute (=1500 frames) of this dataset. The main challenge in the APIDIS dataset comes from the inherent sporadic nature of the digit feature. Moreover, the jersey color is not measured with the same reliability level, since it becomes unreliable in case of occlusions between players.

A sample of such a measurement is shown in Figure C.1 for a specific player (digit=8, color=0.24). It can be easily observed that the appearance characteristics are noisy and sometimes missing. Indeed, these features cannot always be reliably measured for each frame because of occlusions, illumination change, unfavorable shirt orientation with respect to the camera, etc. Despite the fact that these features are noisy and sporadic, they are very discriminant and relevant because they allow the identification of the players which, in turn, supports the interpretation of the game directly.

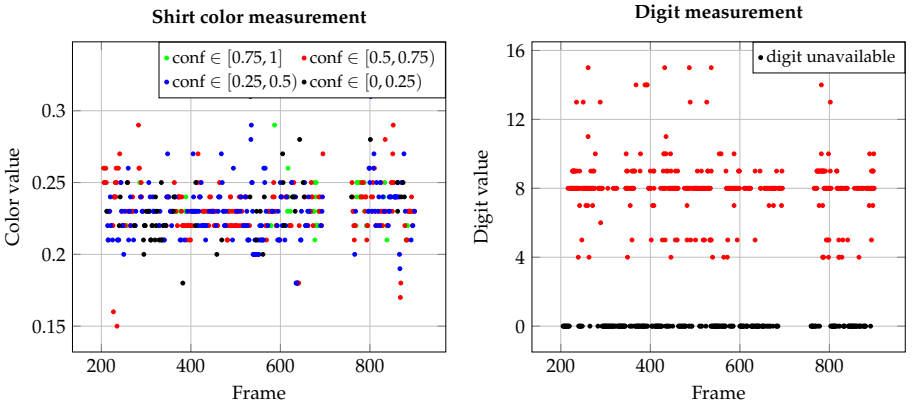


Figure C.1: **Shirt color and digit measurement along time for a player (digit=8, color=0.24)**. The shirt color feature is more frequently available than the digit feature. However, the color measurement is not reliable all the time. The digit feature is highly sporadic. *Best viewed in color.*

Some sample frames of the APIDIS dataset are depicted in Figure C.2.

PETS and **TUD** are publicly available monocular 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 is 179 frames long but the targets frequently occlude each other

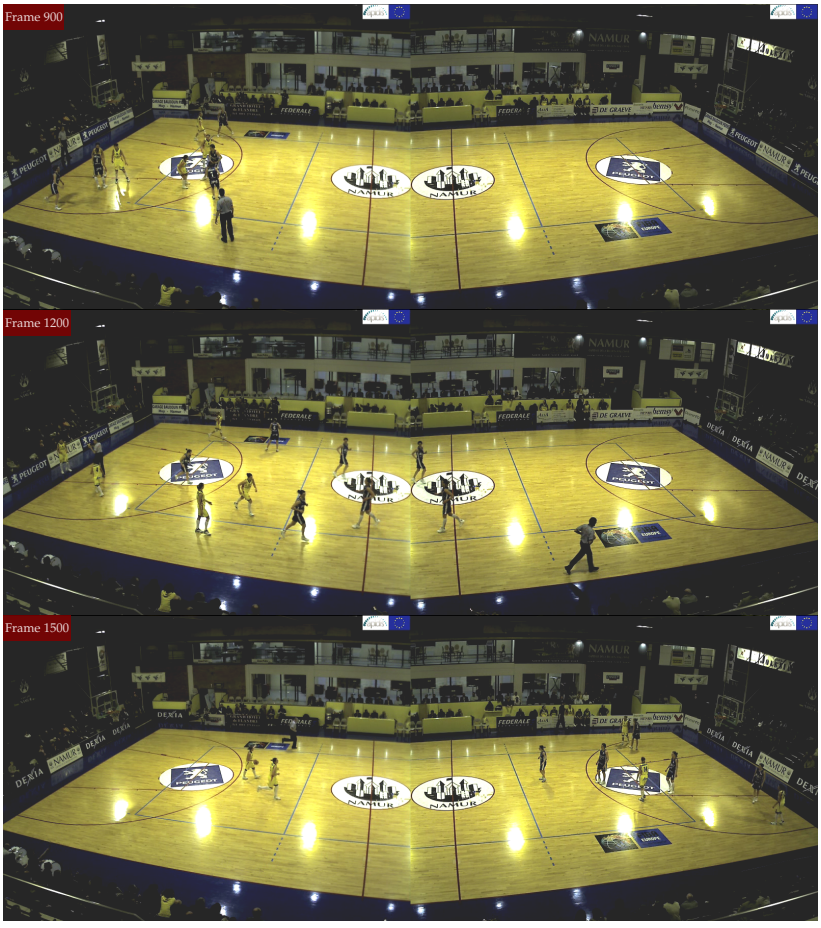


Figure C.2: **Sample frames of APIDIS dataset**. Only views 1 and 6 are shown.

because of the low view-point. Detection results and the ground-truth are obtained from [109]. 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 makes the features sporadic over time.

Some sample frames from PETS and TUD datasets are depicted in Figure C.3.

(a) PETS dataset

(b) TUD dataset

Figure C.3: **Sample frames from PETS and TUD datasets.** *Best viewed in color.*

Bibliography

[1] M. Ocakli and M. Dermirekler, "Video tracker system for traffic monitoring and analysis," in *IEEE Signal Processing and Communication Applications*, 2007.

[2] A. Alahi, Y. Boursier, L. Jacques, and P. Vandergheynst, "Sports players detection and tracking with a mixed network of planar and omnidirectional cameras," in *ICDSC, Como, Italy*, 2009.

[3] O. Debeir, P. Ham, R. Kiss, and C. Decaestecker, "Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes," *IEEE Transaction on Medical Imaging*, vol. 24, no. 6, June 2005.

[4] R. Jiang, D. Crookes, N. Luo, and M. Davidson, "Live-cell tracking using sift features in dic microscopic videos," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 9, September 2010.

[5] L. Lecarme, E. Maire, A. K. K.C., C. De Vleeschouwer, L. Jacques, A. Simar, and T. Pardoen, "Heterogenous void growth revealed by in situ 3-d x-ray microtomography using automatic cavity tracking," *Acta Materialia*, vol. 63, pp. 130–139, 2014.

[6] [Online]. Available: http://sites.uclouvain.be/ispgroup/index.php/Softwares/APIDIS

[7] [Online]. Available: http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html

[8] [Online]. Available: http://fcl.uncc.edu/nhnguye1/anttrackingmcmc.html

[9] D. H. Rapoport, T. Becker, A. M. Mamlouk, S. Schicktanz, and C. Kruse, "A novel validation algorithm allows for automated cell tracking and

the extraction of biologically meaningful parameters," *PloS one*, vol. 6, no. 11, 2011.

[10] B. Yang and R. Nevatia, "Online learned discriminative part-based appearance models for multi-human tracking," in *ECCV*, 2012.

[11] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *ICCV*, 2009.

[12] M. Taj and A. Cavallaro, "Multi-camera track-before-detect," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, Aug 2009, pp. 1–6.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel based object tracking," *PAMI*, vol. 25, no. 5, pp. 564–577, May 2003.

[14] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: a unifying framework," *IJCV*, 2004.

[15] J. Czyz, B. Ristic, and B. Macq, "A particle filter for joint detection and tracking of color objects," *Image and vision computing*, vol. 25, no. 8, pp. 1271–1281, 2007.

[16] M. Arulampamm, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Transactions on Signal Processing*, 2002.

[17] M. Bregonzio, M. Taj, and A. Cavallaro, "Multi-modal particle filtering tracking using appearance, motion and audio likelihoods," in *ICIP*, 2007.

[18] D. Delannay, N. Danhier, and C. De Vleeschouwer, "Detection and recognition of sports(wo)men from multiple views," in *ICDSC, Como, Italy*, 2009.

[19] A. Alahi, Y. Boursier, L. Jacques, and P. Vandergheynst, "Sparsity driven people localization with a heterogeneous network of cameras," *Journal of Mathematical Imaging*, 2011.

[20] D. Anoraganingrum, "Cell segmentation with median filter and mathematical morphology operation," in *Image Analysis and Processing, 1999. Proceedings. International Conference on*, 1999.

[21] A. K. K.C., P. Parisot, and C. De Vleeschouwer, "Demo: Spatio-temporal template matching for ball detection," in *ICDSC*, 2011.

[22] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[23] A. K. K.C., 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*. Springer, 2013, vol. 7729, pp. 412–426.

[24] A. K. K.C. and C. De Vleeschouwer, "Prioritizing the propagation of identity beliefs for multi-object tracking," in *BMVC*, 2012.

[25] ——, "Discriminative label propagation for multi-object tracking with sporadic appearance features," in *ICCV*, 2013.

[26] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *PAMI*, vol. 33, pp. 1806–1819, September 2011. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2011.21

[27] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Tracking multiple people under global appearance constraints," in *ICCV*, 2011.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[29] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003.

[30] L. Ford and D. Fulkerson, "Maximal flow through a networks," *Canadian Journal of Mathematics*, vol. 8, no. 399, 1956.

[31] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, 1988. [Online]. Available: http://dx.doi.org/10.1145%2F48014.61051

[32] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *Journal of Algorithms*, vol. 22, no. 1, 1997.

[33] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

[34] M. Szummer, P. Kohli, and D. Hoiem, "Learning CRFs using graph cuts," in *ECCV*, 2008.

[35] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *PAMI*, vol. 26, no. 2, pp. 147–159, 2004.

[36] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *PAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.

[37] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of Uncertainty in AI*, 1999, pp. 467–475.

[38] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 736–744, 2001.

[39] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *IJCV, 70(1):41–54*, 2006.

[40] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *CVPR*, 1999.

[41] L. Sun, Q. De Neyer, and C. De Vleeschouwer, "Multimode spatiotemporal background modeling for complex scenes," in *EUSIPCO*, 2012.

[42] P. Parisot and C. De Vleeschouwer, "Graph-based filtering of ballistic trajectory," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2011.

[43] P. Carr, Y. Sheikh, and I. Matthews, "Monocular object detection using 3D geometric primitives," in *ECCV*, 2012.

[44] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multi-camera people tracking with a probabilistic occupancy map," *PAMI*, 2008.

[45] S. M. Khan and M. Shah, "Tracking multiple occluding people by localizing on multiple scene planes," *PAMI*, vol. 9, no. 3, March 2009.

[46] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[47] A. Bosch and A. Zisserman, "Image classification using random forests and ferns," in *ICCV*, 2007.

[48] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[49] S. Tang, M. Andriluka, and B. Schiele, "Detection and tracking of occluded people," in *BMVC*, 2012.

[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[51] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012.

[52] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[53] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. AC-24, no. 6, pp. 843–854, December 1979.

[54] lngemar J. Cox and S. L. Hingorani, "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *PAMI*, vol. 18, no. 2, pp. 138–150, February 1996.

[55] K. Murty, "An algorithm for ranking all the aassignments in order of increasing cost," *Operations Research*, vol. 16, pp. 682–687, 1968.

[56] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *ECCV*, 2008.

[57] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[58] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid boosted multi-target tracker for crowded scene," in *CVPR*, 2009.

[59] W. Brendel, M. Amer, and S. Todorovic, "Multiobject tracking as maximum weight independent set," in *CVPR*, june 2011, pp. 1273 –1280.

[60] A. R. Zamir, A. Dehghan, and M. Shah, "GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs," in *ECCV*, 2012.

[61] B. Yang and R. Nevatia, "An online learned CRF model for multi-target tracking," in *CVPR*, 2012.

[62] ——, "Multi-target tracking by online learning a CRF model of appearance and motion patterns," *IJCV*, 2014.

[63] W.-L. Lu, J.-A. Ting, K. P. Murphy, and J. J. Little, "Identifying players in broadcast sports videos using conditional random fields," in *CVPR*, 2011.

[64] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy, "Learning to track and identify players from broadcast sports videos," *PAMI*, 2012.

[65] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[66] A. Milan, K. Schindler, and S. Roth, "Detection- and trajectory-level exclusion in multiple object tracking," in *CVPR*, 2013.

[67] H. Jiang, S. Fels, and J. Little, "A linear programming approach for multiple object tracking," in *CVPR*, june 2007, pp. 1 –8.

[68] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *CVPR*, 2008.

[69] H. Pirsiavash, D. Ramanan, and C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR*, 2011.

[70] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *CVPR*, 2013.

[71] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-commodity network flow for tracking multiple people," *PAMI*, 2013.

[72] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-target tracking - linking identities using bayesian network inference," in *CVPR*, 2006.

[73] C. Russeel, F. Setti, and L. Agapito, "Efficient second order multi-target tracking with exclusion constraints," in *BMVC*, 2011. [Online]. Available: http://www.bmva.org/bmvc/2011/proceedings/paper13/paper13.pdf

[74] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury, "A stochastic graph evolution framework for robust multi-target tracking," in *ECCV*, 2010.

[75] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *J. Image Video Process.*, vol. 2008, pp. 1:1–1:10, January 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/246309

[76] B. Ristic, B.-N. Vo, D. Clark, and B.-T. Vo, "A metric for performance evaluation of multi-target tracking algorithms," *Transactions on Signal Processing (TSP)*, vol. 59, no. 7, pp. 3452–3457, 2011.

[77] B. Wu and R. Nevatia, "Tracking of multiple, partially occluded humans based on static body part detection," in *CVPR*, 2006.

[78] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 319–336, 2009.

[79] T. Nawaz, F. Poiesi, and A. Cavallaro, "Measures of effective video tracking," *Transactions on Image Processing (TIP)*, vol. 23, no. 1, January 2014.

[80] C. Piciarelli, C. Micheloni, and G. Foresti, "Trajectory based anomalous event detection," *IEEE Transactions on Circuits and Systems for Video Technology*, November 2008.

[81] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[82] [Online]. Available: http://www.cvg.rdg.ac.uk/PETS2009/

[83] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958. [Online]. Available: http://dx.doi.org/10.1287/opre.6.6.791

[84] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *CVPR*, 2012.

[85] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *CVPR*, 2011.

[86] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *PAMI*, vol. 33, no. 9, 2011.

[87] C. Verleysen and C. De Vleeschouwer, "Recognition of sport players' numbers using fast color segmentation," in *Proc. of the SPIE-IS&T Electronic Imaging*, 2012.

[88] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *Image Processing, IEEE Transactions on*, vol. 16, no. 11, pp. 2649 –2661, nov. 2007.

[89] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[90] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, 2005.

[91] B. K. Sriperumbudur and G. R. G. Lankriet, "On the convergence of the concave-convex procedure," in *NIPS*, 2009.

[92] P. H. Calamai and J. J. Moré, "Projected gradient methods for linearly constrained problems," *Mathematical programming,39:93–116*, 1987.

[93] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proceedings of the IEEE, 100(9)*, September 2012.

[94] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *ICML*, 2006.

[95] W. Tong and R. Jin, "Semi-supervised learning by mixed label propagation," in *AAAI, pages 651–656*, 2007.

[96] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by online learned discriminative appearance models," in *CVPR*, 2010.

[97] C.-H. Kuo and R. Nevatia, "How does person identity recognition help multi-person tracking?" in *CVPR*, 2011.

[98] 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.

[99] A. I. Chen and A. Ozdaglar, "A fast distributed proximal-gradient method," in *Communication, Control, and Computing (Ann. Allerton Conf.)*, 2012, pp. 601–608.

[100] Y. Nesterov, *Introductory Lectures on Convex Optimization. A Basic Course*. Springer, 2004.

[101] M. Á. Carreira-perpiñán and Z. Lu, "The Laplacian Eigenmaps Latent Variable Model," *Journal of Machine Learning Research-Proceedings Track, 2:59–66*, 2007.

[102] Z. Lu, C. Sminchisescu, and M. Á. Carreira-perpiñán, "People tracking with the laplacian eigenmaps latent variable model," in *NIPS*, 2007.

[103] [Online]. Available: http://www.d2.mpi-inf.mpg.de/node/428

[104] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *ECCV*. Springer, 2006, pp. 589–600.

[105] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *PAMI*, vol. 32, no. 3, pp. 448–461, 2010.

[106] P. Parisot, B. Sevilmis, and C. De Vleeschouwer, "Training with corrupted labels to reinforce a probably correct teamsport player detector," in *Advanced Concepts for Intelligent Vision Systems*, 2013.

[107] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[108] W. y. Sun, R. J. Sampaio, and M. Candido, "Proximal point algorithm for minimization of DC function," *Journal of Computational Mathematics-International Edition*, vol. 21, no. 4, pp. 451–462, 2003.

[109] [Online]. Available: http://www.milanton.de/data/