A graph-based approach for online multi-object tracking in structured videos with an application to action recognition

Henrique Morimitsu

Thesis presented
to the
Institute of Mathematics and Statistics
of the
University of São Paulo
for
obtaining the title
of
Ph.D. in Sciences

Program: Computer Science

Advisor: Prof. Dr. Roberto Marcondes Cesar Junior

The author received financial support from
FAPESP (grants #2012/09741-0 and #2013/08258-7) and CAPES.

São Paulo, September 2015

A graph-based approach for online multi-object tracking in structured videos with an application to action recognition

This is the original version of the thesis written
by the candidate Henrique Morimitsu, as
submitted to the Judging Committee.

Acknowledgements

I would like to thank my advisor, Prof. Roberto Marcondes Cesar Junior. This work could not have been completed without him. Thank you for your trust, dedication and guidance during all this time. I would also like to give special thanks to Prof. Isabelle Bloch who not only welcomed me during one year at Télécom ParisTech, but also greatly contributed to the conclusion of this project from the beginning until the very end. Her knowledge was invaluable to the development of this thesis as well as my own.

I thank my family, who has been supporting me all the way, sharing all the good moments together and helping me to overcome the hardships along the way. I also thank all the friends I made during this journey, who contributed to this work directly or indirectly. Special thanks to those that I had more contact with: Prof. Roberto Hirata, Eric Keiji, Estephan Wandekoken, Frank Aguilar, Guillaume Tartavel, Qin Qiao, Silvia Cristina and Yuke Li.

I offer my thanks to FAPESP and CAPES for providing financial support to conclude this work. I also thank the Institute of Mathematics and Statistics for receiving me and providing me all the necessary structure for the development of the thesis.

Resumo

MORIMITSU, H. **Uma abordagem baseada em grafos para rastreamento de múltiplos objetos em vídeos estruturados com uma aplicação para o reconhecimento de ações**. 2015. 85 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.

Nesta tese, uma nova abordagem para o rastreamento de múltiplos objetos com o uso de grafos é proposta. Os objetos são rastreados usando uma combinação de filtro de partículas com descrição das imagens por meio de Grafos Relacionais com Atributos (ARGs). O processo é iniciado a partir do aprendizado de um modelo de grafo estrutural utilizando imagens anotadas. Os grafos são usados para avaliar o estado atual do rastreamento e corrigi-lo, se necessário. Esta abordagem também é capaz de usar o modelo aprendido para reencontrar o objeto após situações de oclusão ou movimento abrupto. A principal contribuição desta tese é a exploração do modelo estrutural probabilístico aprendido. Por meio dele, a própria informação estrutural da cena é usada para guiar o processo de detecção em caso de perda do objeto. Tal abordagem difere de trabalhos anteriores, que utilizam informação estrutural apenas para avaliar o estado da cena, mas não a consideram para gerar novas possíveis hipóteses de rastreamento.

Os resultados do rastreamento também são usados como entrada para realizar o reconhecimento das ações de cada jogador presente nos vídeos. Esta etapa é executada classificando uma sequência de entradas por meio de Modelos Ocultos de Markov (HMMs). A viabilidade do método é mostrada realizando o rastreamento de objetos de aparências semelhantes em vídeos sintéticos. A abordagem proposta também é testada em diversos vídeos de jogos de tênis de mesa e na base de dados ACASVA, demonstrando a capacidade do método de lidar com situações de oclusão ou cortes de câmera.

Palavras-chave: Rastreamento de objetos, grafo, informação estrutural, filtro de partículas, reconhecimento de ações.

Abstract

H. MORIMITSU. **A graph-based approach for online multi-object tracking in structured videos with an application to action recognition**. 2015. 85 p. Thesis (Ph.D.) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2015.

In this thesis we propose a novel approach for multiple object tracking in structured videos using graphs. The objects are tracked by combining particle filter and frame description with Attributed Relational Graphs (ARGs). We start by learning a structural model graph from annotated images and then applying it to evaluate the current tracking state and to correct it, if necessary. This approach is also capable of exploring the learned model to find the object after occlusion or abrupt motion. The main contribution of this thesis is the exploration of the learned probabilistic structural model. By doing so, the structural information of the scene itself is used to guide the object detection process in case of tracking loss. This approach differs from previous works, that use structural information only to evaluate the scene, but do not consider it to generate new tracking hypotheses.

The tracking results are also used as an input for recognizing the action of each player in the video. This step is performed by classifying a sequence of inputs using Hidden Markov Models (HMMs). The viability of the approach is shown by tracking objects of similar appearance on synthetic videos. The proposed method is also tested on several videos of table tennis matches and on the ACASVA dataset, showing that the method is able to continue tracking the objects even after occlusion or when there is a camera cut.

Keywords: Object tracking, graph, structural information, particle filter, action recognition.

Contents

List of abbreviations

ARG	Attributed Relational Graphs
CDIS	Center displacement measurement
CERR	Center error measurement
GINT	Global intersection measurement
HITR	Hit ratio measurement
HITT	Hit team ratio measurement
HOG	Histogram of Oriented Gradients
HMM	Hidden Markov Model
HSV	Hue - Saturation - Value color space
OBJI	Object intersection measurement
PCA	Principal Components Analysis
PDF	Probability Density Function
SVD	Singular Value Decomposition

List of symbols

Unless stated otherwise, the following symbol convention will be followed along the text:

- Bold lowercase: represents vectors or sequences.
- Bold uppercase: represents matrices.
- Calligraphic: represents sets.

α	HMM forward variable
β	HMM backward variable
δ	Dirac delta function
γ	HMM variable representing the probability of being at a given state or transition
ζ	Particle cloud confidence score
η	Particle weight
θ	Edge angle function
μ	Mean or average
ξ	HMM variable representing the probability of using one mixture
$\boldsymbol{\pi}$	Vector of HMM starting probabilities
ρ_F	Feature weight constant
ρ_T	Temporal weight constant
σ	Standard deviation
τ_{d_D}	Different object distance threshold
τ_{d_S}	Same object distance threshold
τ_O	Overlapping threshold
τ_S	Score threshold
τ_T	Threshold for changing trackers
ϕ	Appearance score function
ψ	Structural score function

$\boldsymbol{A}$	Matrix of HMM state transitions
$\mathcal{A}_{\mathcal{E}}$	Set of edge attributes
$\mathcal{A}_{\mathcal{V}}$	Set of vertex attributes
$\boldsymbol{b}$	Vector of HMM emission probabilities
$\boldsymbol{C}$	Covariance matrix
d	Distance function
e	Graph edge
f	Graph instantaneous score function
$\mathcal{E}$	Set of graph edges
$\mathcal{F}$	Set of final trackers
G^M	Model graph
G^S	Scene graph
H	Histogram
I	Digital image function
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{K}$	Kernel matrix
$\boldsymbol{M_A}$	Matrix of graph adjacencies
$\boldsymbol{M_C}$	Matrix of target candidates generation
N_C	Number of columns
N_M	Number of mixture models
N_O	Number of objects
N_P	Number of particles
N_R	Number of rows
N_S	Number of states
N_W	Length of the sliding window
$\mathcal{N}$	Normal or Gaussian distribution function
o	One observation
$\boldsymbol{o}$	One sequence of observations
O	Random variable representing one observation
P	One tracker
$\mathcal{P}$	Set of trackers
$\mathbb{P}$	Probability
r	Particle consensus function
s	One HMM state
S	Random variable representing one HMM state
$\mathcal{S}$	Set of HMM states
T	Total time length
v	Graph vertex
$\mathcal{V}$	Set of graph vertices
w	Tracker confidence score
Z	HMM

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Tracking multiple objects simultaneously is a challenging task. Due to the good results demonstrated by single object trackers, using a set of them to track several objects in the same scene could be an interesting option. However, applying this direct approach usually presents some difficulties which are not always found on the single object tracking problem.

Most of the current state-of-the-art single object trackers rely on visual appearance features to describe the object of interest, using texture and shape, among others. These approaches have two main drawbacks. The first one is that, if several objects have a similar appearance, then one or more of the trackers can lose their targets. In fact, even when dealing with only one object, tracking can be lost due to ambiguity with the background. The second, and the main problem when tracking multiple objects, is dealing with occlusion. When an object is partially of fully occluded, its appearance may change significantly, which greatly affects the tracking performance. Therefore, a good method for multi-object tracking must be able to recover tracking after it is lost due to occlusion.

In order to deal with such situations, it is necessary to: (1) realize that a tracking error occurred, and (2) recover the correct target. For this, more data should be extracted from the scene and then used to obtain the required information for making the decision. As it will be further discussed in Chapter 2, many options have been explored before, each one being most suited for a different situation. In this thesis we will be interested in studying the use of spatial relations between objects to recover or correct online tracking. Online tracking, as opposed to offline methods, only use past information to predict the next state, and thus, can be used in real time. We argue that, in some kinds of videos where the scene represents a situation that usually follows a common spatial pattern, it is possible to choose the most likely configuration of the objects by learning some structural properties beforehand. Figure 1.1 shows an example of a table tennis video with a situation where tracking is lost after two players intersect each other. Although the interaction is brief, this

already causes one of the trackers to misjudge its correct target and start to track the other player instead. We intend to solve this kind of problem by exploring the spatial properties of the scene, such as the distance between two objects.

(a) **(b)** **(c)** **(d)**

Figure 1.1: *Example of a multi-object tracking situation. Most single object trackers are able to successfully track the targets when their appearance is clear (a). However, when overlap occurs (b), they are not able to solve the ambiguity problem in appearance and the tracking is lost (c). We propose to recover tracking after such events by using spatial information encoded in graphs (d).*

We shall refer to videos that present discernible spatial patterns as **structured videos**. It is assumed that scenes (or frames) of these videos contain elements that provide a kind of stable spatial structure to be explored. A good example is sport videos. Sports rely on a set of rules that usually constrain the involved objects to follow a set of patterns. These patterns often present some spatial relationships that may be explored. For example, the rules may enforce that the objects must be restricted to a certain area, or that they always keep a certain distance among them. Another case is when evaluating videos of a set of known interactions. In this situation, if a person is always using some object, then there is usually a strong constraint on the distance and relative position between them. Structure may also be found in surveillance, where usually the location is known and controlled, thus the region of interest is constrained by the environment itself. Examples include vision-based monitoring for smart homes since people activities are stable given the environment (e.g. cooking, eating, watching TV, etc.).

Structural relations are explored by using graphs to encode the spatial configuration of the scene. In this thesis, particle filter was chosen as the single object tracker, due to its simplicity and good results demonstrated in previous studies. However, theoretically any other tracker could be used instead to benefit from the added structural information. This makes the proposed framework very flexible and able to be used to potentially improve the results of any single object tracking applied in multi-objects problems. As shall be explained throughout this text, the graphs allow us to numerically evaluate the tracking state at each frame. This value is then used to decide whether the current tracking is correct or if some object was lost and, in this case, the state must be corrected. The graphs also contain the necessary information for recovering the correct position of the target. With this approach it is possible to improve the overall result by recovering tracking after situations of overlapping between objects with similar appearance or when abrupt motion happens.

The use of structural information for recovering tracking is a topic that was not much explored before. Indeed, several of the current state of the art methods based on tracking by detection do use structural information for evaluating the tracking state and solving the data association problem between the frames. However, the detections are usually carried out by off-the-shelf detectors that do not consider scene information. In that sense, one main contribution of this thesis is to explore the learned probabilistic structural model to guide the detection in case of tracking loss. This approach may greatly improve the performance by reducing the search space.

We also show a practical application of tracking, where the results obtained from the tracking step are used to perform action recognition. This is done by accumulating all the tracking results for each object over time to create sequences. Short segments of these sequences are then fed to multiple Hidden Markov Models which classify each of them into one of a given set of possible actions for each object.

The proposed framework is tested using two types of datasets: synthetic and broadcast videos. The former are simple videos built to verify that the proposed approach can successfully recover tracking when the appearance information is ambiguous. The latter consists of videos obtained from Youtube and also from the ACASVA (De Campos *et al.* (2011)) dataset, which present more challenging conditions such as changing of lighting, perspective and camera cuts.

1.2 Objectives

The objective of this thesis is to propose a novel approach to improve tracking of multiple objects using structural information encoded in graphs. Besides, we also intend to show that the tracking results can be applied for performing individual action recognition for each object.

1.3 Contributions

The main contributions of this thesis are the following:

- develop a flexible probabilistic graph model that can be used to evaluate the structure of a scene: several works use object models to perform tracking by detection, but learning a structural model does not seem to have been much explored before;

- propose a method for dealing with abrupt motion by generating candidate object locations based on the model: most of the works rely on visual features to find the object after it is lost, but they fail to explore the structural relations;

- implementation of a novel framework for tracking multiple objects by evaluating multiple hypotheses with scene graphs: the probabilistic structural model is used to both generate new tracking hypotheses and choosing the best configuration for each frame. A framework containing the proposed model is implemented and tested on several videos to analyze its behavior in real situations;

- results showing the viability of using the tracking results for performing action recognition: as a practical application, the tracking results are used for recognizing actions in sport videos.

1.4 Organization of this thesis

In Chapter 2 we present a review of some relevant previous works and how they contributed to the development of our method.

In Chapter 3 the complete framework for tracking multiple objects using graphs is explained. We start by detailing the regular particle filter tracking approach and then explain how graphs are integrated in order to deal with multiple objects.

In Chapter 4 we present how the action recognition is performed. In the beginning, the overall framework is explained, followed by a detailed description of the models and methods used in this step.

In Chapter 5 the experimental results obtained are exposed. We start by showing results obtained in the tracking step. We compare the obtained results for our approach with regular particle filters. Afterwards, some results regarding the action recognition are also presented.

In Chapter 6 we discuss the main conclusions of this work as well as suggestions for future research and some final remarks.

Chapter 2

Related work

2.1 Multi-object tracking

Multi-object tracking has been receiving great attention recently and many approaches have been proposed. In this chapter we will discuss about some current works to give an overview of the state of the art and then present how our approach is inserted in this field.

Predictive, or recursive models, such as the Kalman filter, have been used since the beginning to deal with multi-object tracking (Reid (1979)). More recently, particle filter has become more popular due to its ability to deal with non-linear and non-Gaussian models, which frequently appear in tracking situations.

One strategy when using particle filters is to model all the objects simultaneously as a single state and use a single filter to track all of them. However, by doing so, the state space becomes very large and a huge number of particles may be necessary to obtain good samples for estimation. Widynski *et al.* (2012) circumvent this problem by proposing a ranked partitioned sampling scheme for resampling the particles. This is done by iteratively updating the state vector of the particles for one object at a time. By doing so, new particles do not need to be generated from a joint distribution considering all the parameters at the same time, which in turn creates new particles that are closer to each of the objects. The main drawback is that objects that are processed later are prone to be affected by errors created when updating the state for previous ones. In that sense, the proposed ranking scheme shows its importance, by providing a probabilistic score that is used to process objects that are more likely to be tracked correctly first and thus, decrease the error for the next stages.

Another option is to track each object with a different filter. However, by itself, this approach is not able to deal with objects with similar appearance, as the particles for different objects may end up grouping in only one of them. Okuma *et al.* (2004) deal with this problem by using a mixture particle filter. This method joins the different filters in a single mixture model in order to model the multimodal distribution observed when tracking multiple objects. The joint multimodal distribution assures that all the filters do not end up

representing the same single mode and keep tracking all the objects. Hu *et al.* (2015) also use individual filters for each object, but they tackle the multi-object problem in another way. They propose a more robust single object tracker that uses multiple features (color, edge and Gabor textures) and sparse representation for the appearance model. In a sparse representation framework, the object is represented by a linear combination of a set of object templates and trivial templates. Object templates are obtained from successful detections in previous frames and encode visual features. Trivial templates, on the other hand, are typically a set of $w \times h$ binary vectors having only the i-th value set to one, where w and h are the width and height of the tracked image, respectively. These templates form the basis of the sparse representation and are important to both reduce the comparison overload (by using sparse templates) and deal with partial occlusion (by only considering the visible templates). The authors extend their approach to multi-object tracking by measuring the amount of occlusion (when overlapping occurs) and dealing with it accordingly. If the overlap is low, then the respective templates are used to guide the tracking of each overlapped object. However, when it is high, they ignore the visual features altogether and instead rely only on the motion model of the targets to continue tracking. Our method also uses particle filter for tracking each object individually. We explore the graph information to avoid that all trackers follow a single object. This is done by explicitly penalizing configurations where two or more trackers overlap, and trying to find a better configuration which spreads them on the scene.

Occlusion is one of the main challenges involved in multi-object tracking and some recent works focus on dealing specifically with such situations. Tang *et al.* (2014) propose to extend the well known Deformable Parts Model (DPM), proposed by Felzenszwalb *et al.* (2010), to detect both single and pairs of persons. Although the parts model is somewhat robust to partial occlusion, the authors show that the method already starts to fail in situations with more than 20% of occlusion. They show that their approach can successfully deal with different degrees of occlusion, even very severe ones. The single and pair detectors can also be combined to deal with more crowded scenes. The main drawback is that it is specifically designed for tracking two persons, and a different detector must be trained for other types of objects. Grabner *et al.* (2010) tackle occlusion by embedding the environment context into the model. For this, they detect a set of Harris points (Harris and Stephens (1988)) and analyze their motion over time. The points that present correlated motion with the target are used as supports to cast a vote for the object position at each time. Therefore, when occlusion happens, the object can still be tracked by relying on the supports. Our work follows a similar approach, by using the structural model to keep track or find the most likely position of the target by using the other objects as supports.

Recently, the most popular approach is the one based on tracking by detection. This method relies on using object detectors on every frame to extract target candidates. The re-

sults obtained in this stage are usually noisy (containing both missing and wrong detections). These results are usually first grouped into temporal segments called *tracklets*. Tracklets are usually short segments created by connecting detections that present high appearance and position correlation between frames. The main challenge comes from trying to create longer paths from the tracklets. This corresponds to trying to solve a data correlation problem that must be robust to occlusion, appearance change and appearance of new objects. This has been one of the most active research topic on multiple object tracking, and countless methods have been proposed. We will discuss only a few recent works that represent different popular strategies.

One common choice is to model the situation as an energy minimization problem. In this setup, a cost (energy) function is defined to provide a score for each possible configuration and the state with the lowest cost should be the optimal tracking result. Milan *et al.* (2014) propose an energy model that is composed of six terms which consider: the distance between the estimated location and the detection, the plausibility of the motion model between frames, the distances between multiple objects (to check if occlusion happens), the trajectory persistence (continuity from one border of the image until the other), and a regularizer that constraints the number of objects. All the terms are continuous and easily differentiable, making it possible to use a gradient descent approach for minimizing the energy.

Since most multi-object tracking situations deal with human tracking, another notorious approach consists in adapting the tracking paths according to human interactions. This is usually done by modeling attraction and repulsion forces (e.g. groups tend to walk together, while two persons in opposing directions will change their paths to avoid collision). Yan *et al.* (2014) follow this approach by modeling three kinds of interactions: attraction, repulsion and non-interaction. They are incorporated into a Bayesian framework to update the motion model according to the interaction between each person and those that are in a surrounding area.

Methods that rely on graph representations such as Markov Random Fields (MRF) and Conditional Random Fields (CRF) have been receiving great attention lately. Zhang *et al.* (2015) formalize tracking as an iterative labeling problem. To this aim, the detections are considered vertices of a graph and are connected both spatially (objects of the same frame) and temporally (between frames). Following the MRF framework, unary and binary terms are defined to evaluate the likelihood of each individual label, as well as their relations with their neighbors. The unary function is defined as the probability $\mathbb{P}(y|l)$ that a given object y is labeled with the object ID l. This value is obtained both from the motion model and from a classifier that is trained with instances obtained from previous frames. The binary term is decomposed into two types: spatial and temporal. The spatial term penalizes configurations where neighbor objects receive the same label. The temporal term, on the other hand, encourages neighbor labeling if the objects have similar properties. Wen *et al.* (2014) use

hierarchical hypergraphs to build higher order relations to connect the tracklets. The video is divided into segments which are iteratively connected while moving up in the hierarchy, until obtaining a unique segment for the whole track. Inside each segment, a hypergraph is built whose vertices are the tracklets and the hyperedges connect two or more vertices that are likely to be part of the same path. The attributes of the hyperedges indicate the probability of their vertices being part of the same path. The probability is computed considering three features: appearance similarity, motion model and trajectory smoothness.

Some previous works explored structural relations to improve tracking. Perhaps the closest work to ours is that of Zhang and van der Maaten (2014). They use a model-free approach that learns the appearance and structural relations between the tracked objects online. In the first frame, manual annotations are provided and used to initially train a Histogram of Oriented Gradients (HOG) detector (Dalal and Triggs (2005)) for detecting the object in the next frames, i.e. their approach is also based on tracking by detection. The structural relations are also learned from the first frame by training a Structured Support Vector Machine (SVM), in a very similar fashion as that of the DPM detector (Felzenszwalb *et al.* (2010)). The models are then updated while the tracking is being performed, using a gradient descent approach. The candidate graphs are evaluated using the information obtained from the HOG detectors as well as the distances between any two objects.

Although similar, their work differs from this one in the following aspects: (1) their structural model only computes the difference between the observed distance and an ideal value that comes from the online training. Our model considers both distance and angle information obtained from a probability density function computed by evaluating several training examples. (2) Although they use the structure to improve tracking and to deal with occlusion, it is not used to guide the detection process, which could lead to improved performance by restricting the search space. Our approach uses the structural model to obtain candidates of where the target is likely to be found after tracking loss. (3) Their method of tracking by detection does not consider motion models, while we rely on particle filters.

2.2 Action recognition

The field of action recognition is also vast and it has been the focus of much of the recent research. Since action recognition is not the focus of this thesis, this section does not intend to provide a thorough explanation of the current state of the art. Instead, it will give an overview of some relevant research areas to contextualize our work. A more complete review of action recognition methods can be found in the paper by Aggarwal and Ryoo (2011).

Most of action recognition works focus on *action classification*, which consists in providing a label to a video or image containing one single action being performed. Examples of works in

this field include those by Schüldt *et al.* (2004), Blank *et al.* (2005) and Wang and Schmid (2013). The first work extracts spatio-temporal invariant points from the videos. These points correspond to an extension of Harris points (Harris and Stephens (1988)) to the 3D temporal domain. Each point is described by computing spatio-temporal Gaussian derivatives in its neighborhood, which are then clustered using k-means. The clusters can be interpreted as a codebook, used in the classical bag-of-words approach (Yang *et al.* (2007)). Having the codebook, each sequence can be described by a histogram of clusters and then classified using a SVM. The paper by Blank *et al.* (2005) represents the actions as temporal volumetric shapes. Each action video is represented by a cube containing the volumetric shape described by local and global features, including saliency and orientation. The cubes are then classified by using the Nearest Neighbors procedure. One popular approach that has recently shown promising results is the use of trajectories to classify action sequences. A trajectory is obtained by following a point of interest throughout the video. Wang and Schmid (2013) obtain trajectories from dense optical flow (Farnebäck (2003)). In order to deal with camera motion, a homography is estimated between consecutive frames, and the trajectories that follow the same motion pattern of the camera are removed to avoid clutter. Each trajectory is described by segmenting it into a sequence of spatio-temporal cubes. The area inside each cube is described by histograms of gradients and optical flows. The videos are then classified using the bag of words approach (Yang *et al.* (2007)) with a linear SVM.

In this work, however, we are interested in dealing with another class of problems, where many actions may be performed simultaneously. Furthermore, the actions performed by each person may also vary over time. This kind of situation is related to the *action localization* problem. Action localization consists in finding all the instances of a given action in a longer video. Laptev and Pérez (2007) do so by formulating event detection as an object detection problem. The "objects" consist of small sequences with various configurations of spatial positioning and temporal length extracted from the videos, which are described by spatio-temporal histograms of gradients and optical flows. The sequences are then classified using a set of weak classifiers obtained by Adaboost (Viola and Jones (2004)). In the paper by Lu and Little (2006), the authors propose a joint tracking and action recognition framework where both are treated simultaneously. Thus, the action location is obtained by tracking the target from the first frame. Afterwards, the sequence of tracking results is used to classify the action by means of a set of Hidden Markov Models (HMM). The action recognition module implemented in this thesis is inspired by this work and follows a similar approach.

Chapter 3

Multi-object tracking based on structural information

In this chapter the proposed tracking framework is explained. Figure 3.1 shows a flowchart of the process, which is based on particle filters and graphs.

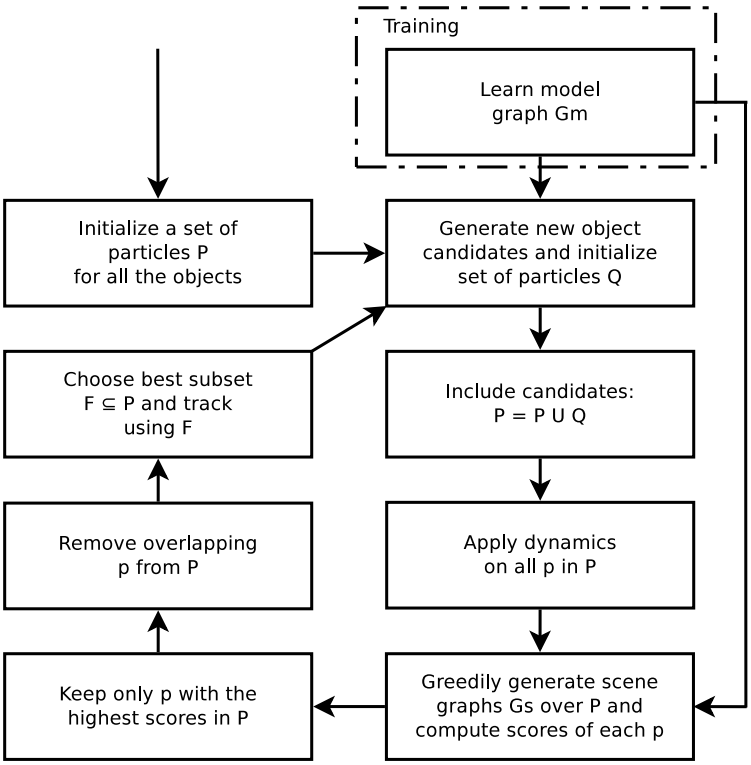


Figure 3.1: *Overview of the proposed framework.*

First, the model graph G^M of the image structure is learned using annotated training images as described in Section 3.2.2. For the tracking step, the position of each of the N_O objects can be either manually annotated in the first frame of the video or obtained automatically using detectors. In this work, we adopt the former option. Multiple hypotheses

about the state of each object i are kept by using a set of trackers

$$\mathcal{P}_i^t = \{(P_j^i, w_j^i) | j = 1, ..., n_i^t\} \tag{3.1}$$

where P_j^i is the j-th tracker of object i, w_j^i is a temporal confidence score and n_i^t represents the number of trackers for object i at instant t.

Abrupt motion is usually caused by a sudden change in speed or direction of the target. It can also happen when the object enters an occluded area, such as behind a wall or outside the field of view of the camera. Another reason is either in case the camera is cut off for a moment, or when the same scene is observed from another point of view. These events will be referred to as *camera cuts*. Abrupt motion is treated by continuously generating new hypotheses about the position of each target. For this, G^M is used to generate candidates on the most likely locations (Section 3.3.1). Each candidate yields a new pair $(P_k^i, 0)$ which is then added to $\mathcal{P}_i^t$. All trackers in $\mathcal{P}_i^t$ are then updated by applying their respective state dynamics.

After including the candidates in the set, a temporal score is computed for every $P_j^i \in \mathcal{P}_i^t$ (Section 3.3.2). This is done by using a greedy approach to generate the scene graphs G_k^S and evaluating them using the model graph G^M (Section 3.3.3). The scores are then used to remove undesired trackers from the set (Section 3.3.4). The final step consists in actually choosing the best trackers from each set to provide the final multi-object tracking result (Section 3.3.5). The next sections detail each step.

3.1 Tracking objects with particle filters

In this section the standard method of tracking with particle filters is presented. Particle filter is an interesting method because it is able to deal with non-Gaussian and non-linear models, as opposed to Kalman filters. For this reason, as evidenced by the works presented in Chapter 2, many recent researches on tracking rely on particle filters.

3.1.1 Bayes recursive filter

Particle filter comes from the Bayes recursive filter. When the Bayes filtering function cannot be solved analytically, particles are used to provide a numerical approximation. Therefore, before presenting the particle filter, the Bayes recursive filter will be explained.

Statistics fundamentals

For completeness, some basic statistical properties that are used for deriving the filtering function will be presented. For more details or proofs, the reader is referred to

Bussab and Morettin (2010). The most important definition is the Bayes' theorem. Given random variables X and Y, their conditional probability can be computed by

$$\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X)\mathbb{P}(X)}{\mathbb{P}(Y)}. \tag{3.2}$$

The left-hand side of this equation is usually called *posterior*. On the right-hand side, the denominator is referred to as *evidence*, while the numerator is composed of the *likelihood* and the *prior*, respectively.

The joint probability of two variables can be expressed as

$$\mathbb{P}(X, Y) = \mathbb{P}(X|Y)\mathbb{P}(Y). \tag{3.3}$$

The same properties are still valid when conditioning the previous variables to another one Z:

$$\mathbb{P}(X|Y, Z) = \frac{\mathbb{P}(Y|X, Z)\mathbb{P}(X|Z)}{\mathbb{P}(Y|Z)}. \tag{3.4}$$

$$\mathbb{P}(X, Y|Z) = \mathbb{P}(X|Y, Z)\mathbb{P}(Y|Z). \tag{3.5}$$

Finally, the marginal distribution of X over Y is given by:

$$\mathbb{P}(X) = \int_{Y \in \mathcal{Y}} \mathbb{P}(X, Y) dY. \tag{3.6}$$

Bayes filter

A classical filtering problem operates over a Hidden Markov Model (Section 4.3). Let $\boldsymbol{x}_t \in \mathcal{X}$ be a hidden state at instant t and $\boldsymbol{o}_t \in \mathcal{O}$ an observation emitted by $\boldsymbol{x}_t$. It is assumed that the model is a Markov process of first order (Section 4.3) and it is conditionally independent of the joint of previous states and observations:

$$\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{o}_{1:t}) = \mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \tag{3.7}$$

where $\boldsymbol{x}_{1:t-1}$ denotes a sequence of states from instant 1 until $t-1$. In the same sense, the observation is also conditionally independent from the joint of previous states and observations:

$$\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_{1:t}, \boldsymbol{o}_{1:t-1}) = \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t). \tag{3.8}$$

The filtering problem consists in estimating the posterior distribution $\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t})$ using the Bayes' theorem:

$$\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t}) = \frac{\mathbb{P}(\boldsymbol{o}_{1:t}|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t)}{\mathbb{P}(\boldsymbol{o}_{1:t})}. \tag{3.9}$$

This can be done by deriving this equation to use the posterior obtained at instant $t-1$ as

a prior at instant t, as follows:

$$\begin{aligned}\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t}) &= \frac{\mathbb{P}(\boldsymbol{o}_{1:t}|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t)}{\mathbb{P}(\boldsymbol{o}_{1:t})} \\ &= \frac{\mathbb{P}(\boldsymbol{o}_t, \boldsymbol{o}_{1:t-1}|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t)}{\mathbb{P}(\boldsymbol{o}_t, \boldsymbol{o}_{1:t-1})}\end{aligned}$$

using the property of Equation 3.5:

$$= \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, \boldsymbol{x}_t)\mathbb{P}(\boldsymbol{o}_{1:t-1}|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t)}{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{o}_{1:t-1})}$$

now, by applying the Bayes' theorem on the second term of the numerator:

$$\begin{aligned}&= \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, \boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{x}_t)}{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{x}_t)} \\ &= \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, \boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})}{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1})}\end{aligned}$$

finally, according to Equation 3.8:

$$= \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})}{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1})}. \tag{3.10}$$

The prior can be further derived in terms of a marginal distribution over the previous state:

$$\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1}) = \int \mathbb{P}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_{t-1}$$

using the property of Equation 3.5:

$$= \int \mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{x}_{t-1}|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_{t-1}$$

by applying the conditional independence of Equation 3.7:

$$= \int \mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\mathbb{P}(\boldsymbol{x}_{t-1}|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_{t-1}. \tag{3.11}$$

By doing so the previous posterior can be directly applied onto the equation to estimate the new value. The evidence of Equation 3.10 can also be expressed in terms of a marginal

distribution:

$$\begin{aligned}\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}) &= \int \mathbb{P}(\boldsymbol{o}_t, \boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_t \\ &= \int \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t, \boldsymbol{o}_{1:t-1})\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_t \\ &= \int \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_t. \end{aligned} \tag{3.12}$$

Replacing Equations 3.11 and 3.12 into Equation 3.10, we obtain the final form of the filter:

$$\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t}) = \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t)\int \mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\mathbb{P}(\boldsymbol{x}_{t-1}|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_{t-1}}{\int \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t-1})d\boldsymbol{x}_t}. \tag{3.13}$$

Except if the system presents some properties such as Gaussian distributions and linear models, Equation 3.10 cannot be solved analytically. When the system is more complex, then the result can only be approximated using, for example, a particle filter.

3.1.2 Particle filter

Let $\boldsymbol{x}_t^i$ be a hypothetical realization of the state $\boldsymbol{x}_t$ and $\delta(\boldsymbol{x}_t - \boldsymbol{x}_t^i)$ be the Dirac delta function centered on $\boldsymbol{x}_t^i$. A particle filter solves the filtering problem by approximating the posterior $\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t})$ by a weighted sum of N_P Dirac masses:

$$\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t}) = \sum_{i=1}^{N_P} \eta_t^i \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^i), \tag{3.14}$$

where each $\boldsymbol{x}_t^i$ is called a particle with associated weight η_t^i.

This work employs particle filter using the ConDensation algorithm, which uses factored sampling (Isard and Blake (1998)) to update the particles. The cloud of particles $P_t = \{(\boldsymbol{x}_t^i, \eta_t^i)|i = 1, ..., N_P\}$ is obtained by resampling them, with repositions, from the past one $P_{t-1} = \{(\boldsymbol{x}_{t-1}^i, \eta_{t-1}^i)|i = 1, ..., N_P\}$. By assuming $\sum_{j=1}^{N_P} \eta_{t-1}^j = 1$, the probability of each particle i being chosen in this step is η_{t-1}^i. Hence, more likely particles can be sampled several times, while others may not be chosen at all.

The particles are then propagated according to a proposal function $\boldsymbol{x}_t^i \sim q(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}^i, \boldsymbol{o}_{1:t})$, which is usually assumed to be the dynamics model $\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, yielding $\boldsymbol{x}_t^i \sim \mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. The propagation phase involves two steps: drift and diffusion. Drift is a deterministic step, which consists in applying the motion dynamics for each particle. Diffusion, on the other hand, is random and it is used to include noise in the model. The new state of a particle i can be expressed as:

$$\boldsymbol{x}_t^i = \boldsymbol{D}\boldsymbol{x}_{t-1}^i + \boldsymbol{U}, \tag{3.15}$$

where $\boldsymbol{D}$ is the motion dynamics and $\boldsymbol{U}$ is the noise matrix.

Finally, the weights of the particles are updated according to the new observations $\boldsymbol{o}_t$ as:

$$\eta_t^i = \frac{\eta_{t-1}^i}{\sum_{j=1}^{N_P} \eta_t^j} \frac{\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i)\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}^i, \boldsymbol{o}_{1:t})}. \tag{3.16}$$

If the proposal function is the dynamics model, the weight update is simplified to:

$$\eta_t^i = \frac{\eta_{t-1}^i}{\sum_{j=1}^{N_P} \eta_t^j} \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i). \tag{3.17}$$

The final estimated state of a cloud of particle P may be computed using several heuristics, but the most widely used is by computing the weighted average:

$$r(P) = \boldsymbol{x}_t = \sum_{i=1}^{N_P} \eta_t^i \boldsymbol{x}_t^i \tag{3.18}$$

In this work, we are also interested in evaluating the overall quality of a cloud P. We propose to do this by computing a confidence score based on the non-normalized weights of the particles:

$$\zeta(P) = 1 - \exp\left(-\sum_{i=1}^{N_P} \eta_{t-1}^i \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i)\right). \tag{3.19}$$

3.1.3 State and dynamics models

In this work, the objects to be tracked are represented by rectangular bounding boxes. Each box is parameterized by its centroid and two measures: height and width. It is assumed that the variation in scale is not significant. Therefore, the state of each particle is given by a column vector $\boldsymbol{x}_t^i = (x_t^i, y_t^i)^T$, which represents one candidate centroid. The motion model is a random walk, i.e. $\boldsymbol{D} = \boldsymbol{I}$ and $\boldsymbol{U} = \boldsymbol{I}\boldsymbol{u}$, yielding:

$$\boldsymbol{x}_t^i = \boldsymbol{x}_{t-1}^i + \boldsymbol{u} \tag{3.20}$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{u} = (u_x, u_y)^T$ such that u_x and u_y are noise terms that follow a Gaussian distribution with zero mean.

More complex states and motion models could be used. For example, the state could also include additional information such as the velocity, acceleration, orientation, scale and so on. The greatest problem when considering more information is that each additional parameter increases the search space in one dimension. Therefore, the number of particles necessary to cover a significant part of this larger space increases exponentially. As when tracking multiple objects the amount of particles necessary to track all of them may increase fast, it was chosen to use the least amount of particles that produced good tracking results.

3.1.4 Color histogram-based tracking

The objects are tracked using color histograms as proposed by Pérez *et al.* (2002). The method works by using color information obtained from the HSV color space. This color model is interesting because it separates the chromatic information (Hue and Saturation) from the shading (Value). However, the authors point out that the chromatic information is only reliable when both the saturation and the value are not too low. Therefore, first an HS histogram with $N_H N_S$ bins is populated using only information obtained from pixels whose Saturation and Value are above a given threshold of 0.1 and 0.2, respectively. The remaining pixels are not discarded, because their information can be useful when dealing with images which are mostly black and white. Those pixels are used to populate a V histogram that is concatenated to the HS one built before. The resulting histogram is composed of $N_H N_S + N_V$ bins. Following the original paper, the variables are set as $N_H = N_S = N_V = 10$.

Each histogram corresponds to one observation $\boldsymbol{o}_t^j$ for object j at instant t for the particle filter. Section 3.3.2 presents more details about how the histograms are compared in order to track each object.

3.2 Building the graphs

3.2.1 Attributed Relational Graph (ARG)

An ARG is a tuple

$$G = (\mathcal{V}, \mathcal{E}, \mathcal{A}_\mathcal{V}, \mathcal{A}_\mathcal{E}), \tag{3.21}$$

where $\mathcal{V} = \{v_i | i = 1, ..., N_O\}$ represents a set of vertices (or nodes), $\mathcal{E} = \{e_{ij} = (v_i, v_j) | i, j = 1, ..., N_O\}$ is a set of directed edges (or arcs), i.e. $e_{ij} \neq e_{ji}$ and $\mathcal{A}_\mathcal{V}$ and $\mathcal{A}_\mathcal{E}$ are sets of attributes of the vertices and the arcs, respectively.

Each frame of the video (also referred to as scene) is represented by one or more ARGs. The vertices of G are the tracked objects, while the edges connect objects whose relations will be analyzed. The desired relations are expressed using a binary adjacency matrix $M_A = (m_{ij})$ where $m_{ij} = 1$ if there is an edge from v_i to v_j. Figure 3.2 shows one possible scene graph generated from the adjacency matrix:

$$M_A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{3.22}$$

Two different kinds of attributes are used: the appearance and the structural attributes.

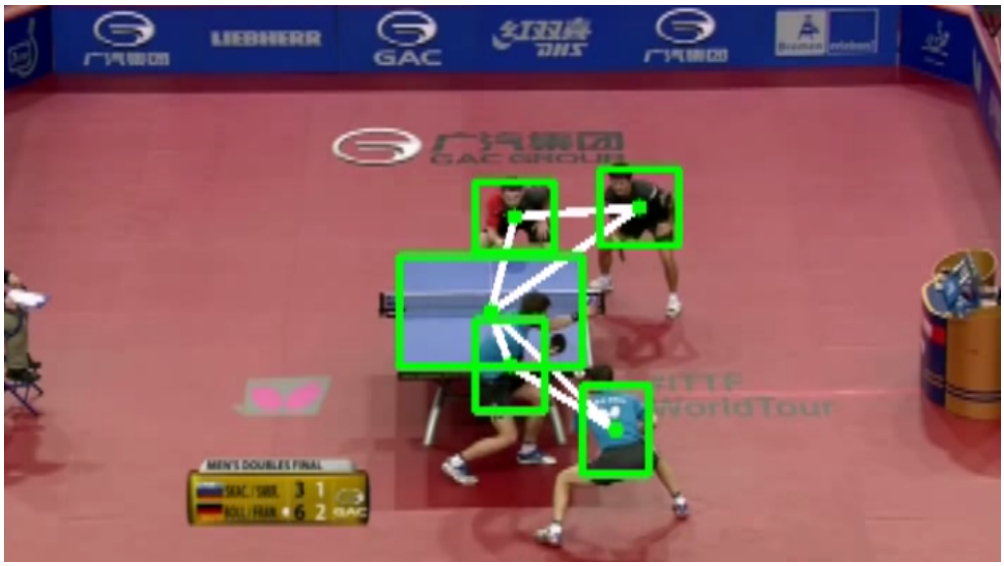


Figure 3.2: *An example of a scene graph.*

Appearance attributes are related to each object and they are stored in $\mathcal{A}_{\mathcal{V}}$. On the other hand, structural attributes represent relations among objects, and thus constitute edge attributes in $\mathcal{A}_{\mathcal{E}}$.

3.2.2 Model graph

The topology of G^M is obtained by means of an adjacency matrix M_A, which must be provided. The set of attributes $\mathcal{A}_{\mathcal{E}}^M$ of G^M is computed from a database of annotated images. Each image is labeled with the state of each relevant object (typically a surrounding bounding box and an object label). Let $\delta_k \in \Delta(i,j)$ be one of the structural attributes to be measured (e.g. the distance between two objects). The annotations are used to estimate the probability density function (PDF) of δ_k. Inspired by Cho *et al.* (2013), the set of attributes chosen was

$$\Delta(i,j) = \{(\theta(e_{ij}), d(v_i, v_j))\} \tag{3.23}$$

and the PDF is estimated by means of histograms H_{δ_k}. The function $\theta(e_{ij})$ represents the clockwise angle between the horizontal axis and the vector $\overrightarrow{v_i v_j}$ and $d(v_i, v_j)$ the distance between the two vertices (Fig. 3.3).

The histograms are built by iterating over all images and collecting the respective observations o, which cast a vote for the histogram bin $H_{\delta_k}(o)$. The PDF is then estimated by normalizing H_{δ_k} to have a sum equal to one. Finally, the normalized histograms are then used as the arc attributes $\mathcal{A}_{\mathcal{E}}^M = \{H_{\delta_k} | \delta_k \in \Delta(i,j)\}$.

The appearance attributes in $\mathcal{A}_{\mathcal{V}}^M$ are not learned from the database. Instead, they are computed from annotations provided in the first frame of the tracking video. For the tests, the appearance was described by using color histograms. However, any other appearance

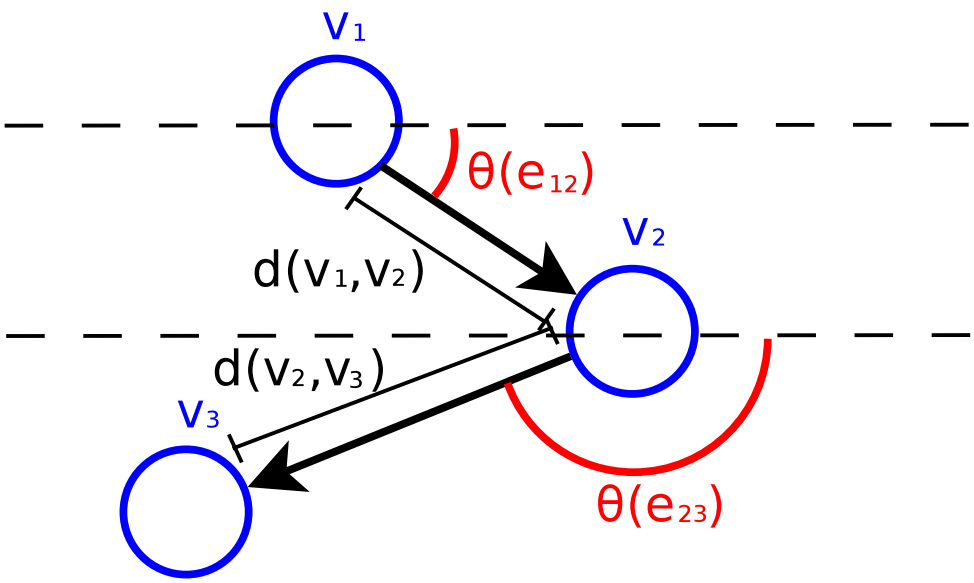


Figure 3.3: *The structural attributes of the edges.*

descriptor could also be considered, like HOG (Dalal and Triggs (2005)) or SIFT (Lowe (2004)). Figure 3.4 shows one graph and the learned histograms for each attribute.

3.2.3 Scene graph

Each graph G_k^S represents a different scene configuration. A vertex $v_i \in \mathcal{V}^S$ of the scene graph G_k^S is associated with one cloud of particles P_j^i for object i. Let $r(P_j^i) = (x_1^{i,j}, x_2^{i,j}, ..., x_{N_S}^{i,j})$ represent the final vector state obtained from Equation 3.18 for the particle cloud P_j^i. Assuming that $x_1^{i,j}$ and $x_2^{i,j}$ represent the 2D coordinates of the object in the image space, the position of v_i is obtained from

$$r_P(P_j^i) = (x_1^{i,j}, x_2^{i,j}), \tag{3.24}$$

i.e. $r_P(P_j^i)$ is a truncated version of $r(P_j^i)$ that only includes the spatial coordinates.

The edges are then produced using the same matrix M_A as in the training. However, recall that each object is tracked by a set of different trackers. Therefore, each scene may be described by multiple graphs, which represent all the possible combinations of different trackers for each object. If all the possible graphs are generated, then each scene will be represented by $\prod_{i=1}^{N_O} |\mathcal{P}_i^t|$ graphs. All these graphs could be evaluated and the best one chosen according to the model to obtain the final tracking result, but this could be computationally expensive due to the large number of combinations. Therefore, in this work a different approach is used, which is explained in Section 3.3.3.

The set of structural attributes $\mathcal{A}^s$ of G^S is not composed of PDFs as in G^M, but of single values for each measurement δ_l extracted from the current frame (i.e. the observations

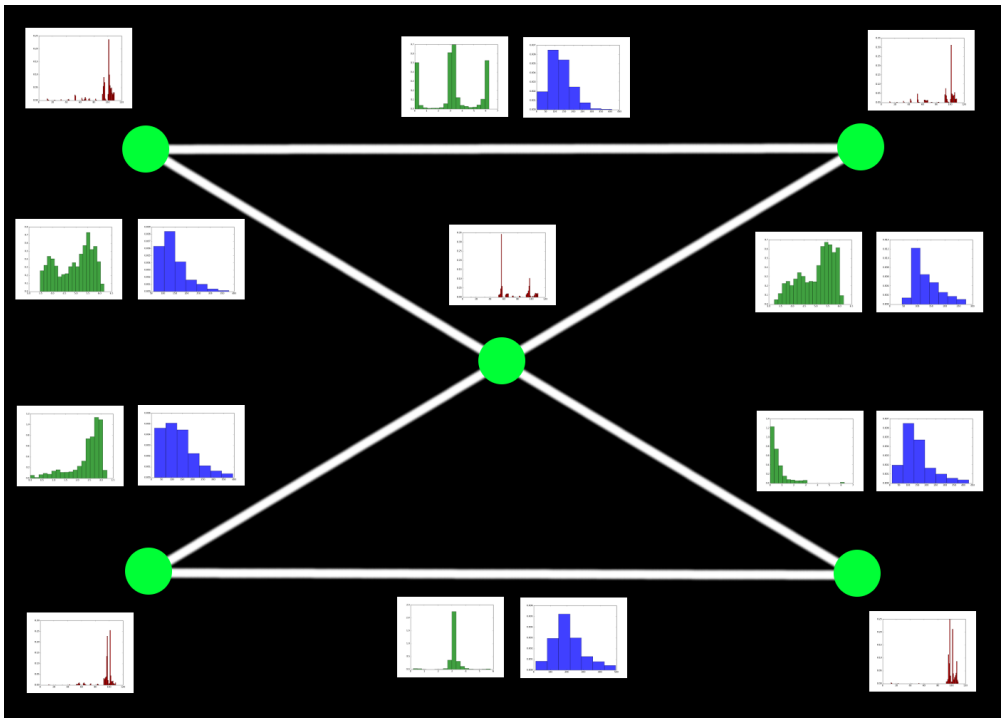


Figure 3.4: *An example of a model graph with the learned attributes. The red histograms represent the vertices attributes (color model), while the green ones represent the angles and the blue ones represent the distances.*

of δ_k). The attributes of the vertices are the associated pairs (P_j^i, w_j^i).

3.3 Tracking using graphs

3.3.1 Generating new candidates

Besides for tracking evaluation, the structural information of G^M is also used to generate new candidate positions for each tracked object. This step is important to deal with abrupt motion. Since the attributes $\mathcal{A}_{\mathcal{E}}^M$ are all relative to the origin of each arc e_{ij}, the position of v_i must be known. Therefore, it is assumed that the trackers for every object will not all fail at the same time. Good candidates can be generated by selecting the positions given by the best trackers as origins. Candidate generation is controlled by using a matrix $M_C = (m_{ij})$, where m_{ij} indicates that, if object i is used as reference, then it generates m_{ij} candidates for object j.

Let $a_{e_{ij}}^M = \{H(\theta(e_{ij})), H(d(v_i, v_j))\}$ be the attribute of an edge e_{ij} from G^M. Candidates are generated for object j as

$$(\hat{\theta}_k = \theta_k + u_\theta, \hat{d}_k = d_k + u_d) \tag{3.25}$$

by simulating according to the distributions given by the histograms $\theta_k \sim H(\theta(e_{ij}))$ and

$d_k \sim H(d(v_i, v_j))$, where $u_\theta \sim \mathcal{N}(0, \sigma_\theta)$ and $u_d \sim \mathcal{N}(0, \sigma_d)$ are Gaussian noises. Each candidate position is then obtained by

$$(v_i(x) + \hat{d}_k \cos(\hat{\theta}_k), v_i(y) + \hat{d}_k \sin(\hat{\theta}_k)). \tag{3.26}$$

Figure 3.5 shows the candidates generated for each object. The candidates are then used to

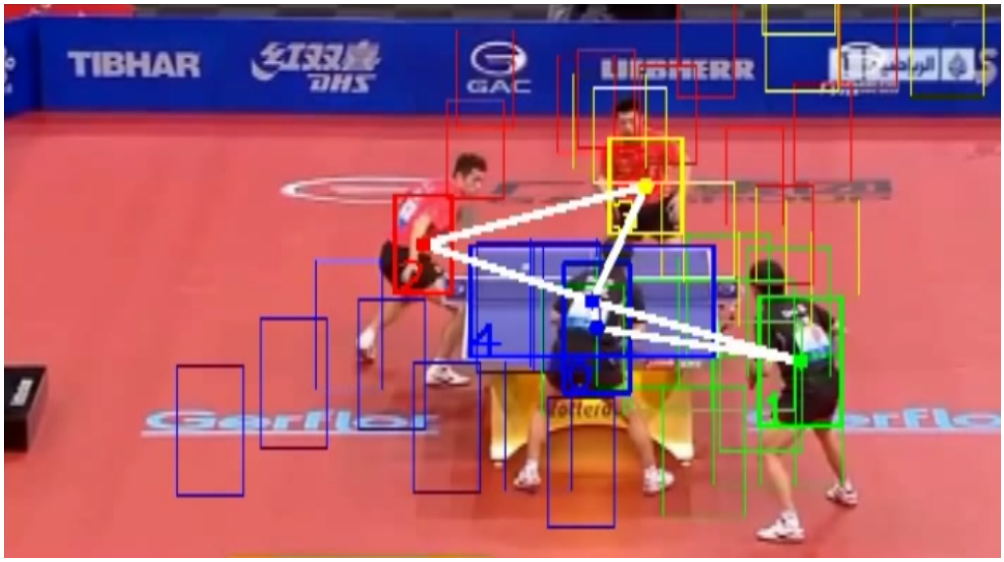


Figure 3.5: *Example of candidates generated for one scene. Rectangles of the same color indicate that they belong to the same object.*

generate new particle clouds P_k^j which are inserted in the set $\mathcal{P}_j^t$. The clouds are initialized by spreading the particles according to a Gaussian distribution centered on the candidate position.

3.3.2 Computing temporal scores of trackers

The temporal score w^i measures the reliability of the associated tracker over time. This is done by computing a weighted accumulation of instantaneous scores:

$$(w^i)^t = \rho_T (w^i)^{t-1} + f(i, G^S, G^M), \tag{3.27}$$

where ρ_T is a given constant and $f(i, G^S, G^M)$ is the instantaneous score function for the vertex v_i^S, which is associated with (P^i, w^i). By doing so, trackers which consistently perform well during longer periods of time have higher scores than those that are only eventually good (usually incorrect trackers).

The instantaneous score is divided into two parts:

$$f(i, G^S, G^M) = \rho_F \phi(i, G^S) + (1 - \rho_F)\psi(i, G^S, G^M), \tag{3.28}$$

where ρ_F is a given weighting factor and $\phi(i, G^S)$ and $\psi(i, G^S, G^M)$ are the appearance and structural score functions of v_i, respectively.

Appearance score

The appearance score is actually the confidence of the particle cloud P_i associated with v_i as shown in Equation 3.19. Hence, it is set as $\phi(i, G^S) = \zeta(P_i)$.

The confidence score depends on the weights of the particles, which are based on the likelihood $\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i)$. The distribution is computed in the same way as in the work by Erdem *et al.* (2012), using the Bhattacharyya distance d_B:

$$\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i) = \exp\left(-\frac{d_B(H^M, H^S)^2}{2\sigma^2}\right), \tag{3.29}$$

where H^M and H^S are histograms of the model and the scene, respectively and

$$d_B(H^M, H^S) = \sqrt{1 - \sum_j \sqrt{H^M(j)H^S(j)}}, \tag{3.30}$$

where $H(j)$ is the j-th bin of histogram H.

Structural score

Let $\boldsymbol{m_i}$ be a vector representing the line i from the adjacency matrix M_A. Let also $\boldsymbol{\theta_i^S} = (H_\theta^M(\theta^S(e_{ij})))_{j=1}^{N_O}$ and $\boldsymbol{d_i^S} = (H_d^M(d^S(v_i, v_j)))_{j=1}^{N_O}$ be the vectors of the values obtained from the bins of the angle and distance model histograms, respectively, i.e. the likelihoods of each structure measurement. The structure score is computed using the dot product:

$$\psi(i, G^S, G^M) = \frac{1}{2\|\boldsymbol{m_i}\|_1}\boldsymbol{m_i} \cdot \boldsymbol{\theta_i^S} + \boldsymbol{m_i} \cdot \boldsymbol{d_i^S}, \tag{3.31}$$

where $\|\boldsymbol{m_i}\|_1$ is the L_1 norm of $\boldsymbol{m_i}$. In other words, this score corresponds to the average of the attributes of the edges originating from v_i.

3.3.3 Generating scene graphs for evaluation

The best trackers are selected by building the scene graphs G_k^S and computing the scores explained before. Figure 3.6 shows some possible graphs that can be generated from some given candidates. Therefore, one option would be to build all possible graphs and to find the one which maximizes the overall score for every tracker. However, this approach was not chosen for two reasons: (1) the number of graph combinations is usually very large and unfeasible to be processed in real time (Figure 3.7), and (2), although the videos are assumed to be structured, it does not necessarily mean that the model graph expresses the

best configuration for every scene. Instead, it was chosen a greedy approach that fixes the vertices for all objects except one and optimize the score for one object at a time.

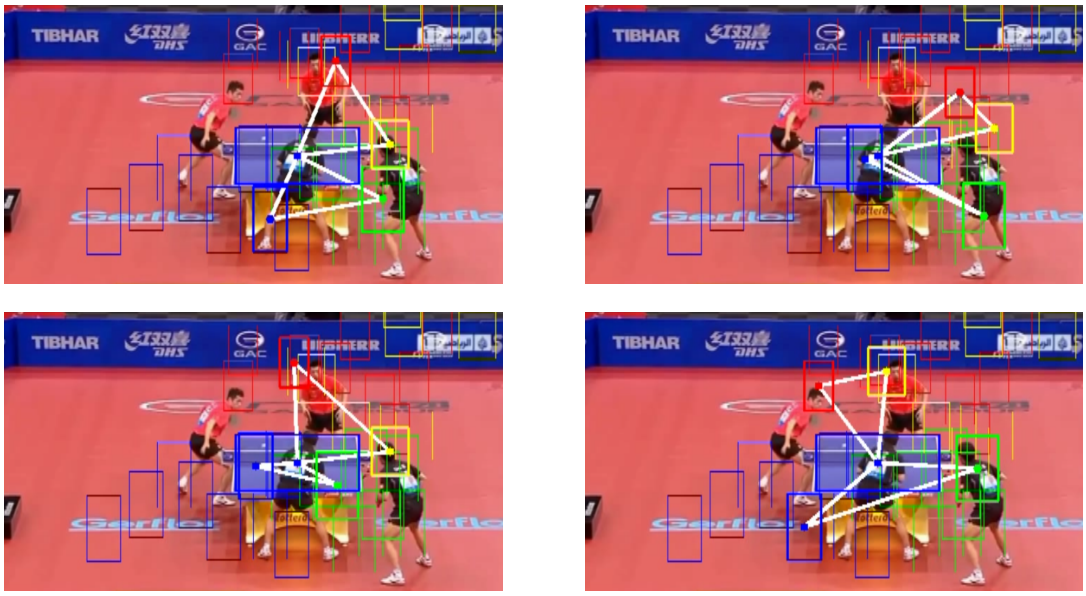


Figure 3.6: *Example of candidate graphs to be analyzed.*

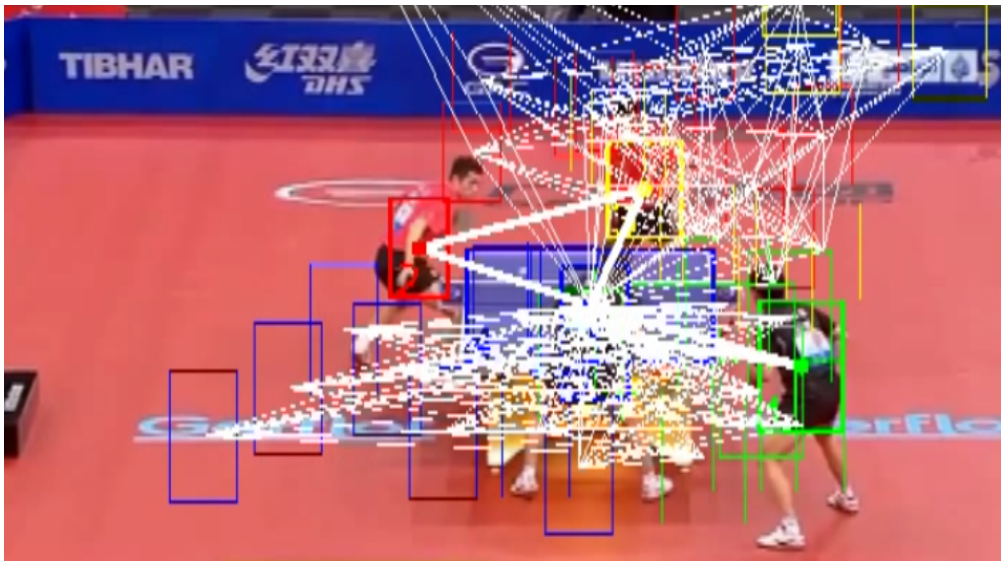


Figure 3.7: *Representation of all possible graphs that can be generated from the candidates.*

When using a greedy approach, the order in which the objects are processed is important. Let $\{(P_*^i, w_*^i)|i = 1, ..., N_O\}$ be the set of the best trackers of each object, i.e.

$$(P_*^i, w_*^i)^t = \underset{(P_j^i, w_j^i) \in \mathcal{P}_i^t}{\arg\max}\ w_j^i. \tag{3.32}$$

A sequence is created by sorting w_*^i in ascending order and processing the objects i one

by one according to this sequence. The rationale is that, since all the other vertices will be fixed, it is better to let the worst tracker vary first in order to have good references for the resulting graph. Let P_l^c be the tracker that is currently being evaluated. This yields a graph G_k^S whose set of vertices is

$$\mathcal{V}_k^S = \{v(P_*^b)|b = 1, ..., N_O\} \cup \{v(P_l^c)\} \setminus \{v(P_*^c)\}, \tag{3.33}$$

where $v(P)$ represents the vertex associated with P. This graph is then used to compute the score w_l^c with $1 \leq c \leq N_O$ and $1 \leq l \leq |\mathcal{P}_c^t|$.

3.3.4 Removing undesired trackers

After computing the score for each tracker, those that are considered as non-significant are removed. This is done by considering two criteria. The first one is thresholding, i.e. removing as many trackers as possible whose scores are too low. More formally, let $\mathcal{Q}_i^{t(1)} = \{(P_j^i, w_j^i)|w_j^i < \tau_S, j = 1, ..., |\mathcal{P}_i^t|\}$, where τ_S is a given score threshold. The thresholded set is obtained by:

$$\mathcal{Q}_i^{t(2)} = \begin{cases} \{(P_*^i, w_*^i)\}, & \text{if } |\mathcal{P}_i^t| = \left|\mathcal{Q}_i^{t(1)}\right| \\ \mathcal{P}_i^t \setminus \mathcal{Q}_i^{t(1)}, & \text{otherwise.} \end{cases} \tag{3.34}$$

The second criterion relies on the fact that one or more trackers will be representing very similar positions (overlapping) for the same object. In such a case, it is not necessary to keep all of them, because they increase the processing burden and do not introduce much information. It is defined that two trackers P_k^i and P_l^i are overlapping when $d(r_P(P_k^i), r_P(P_l^i)) < \tau_{d_S}$, where τ_{d_S} is a given overlapping distance threshold for the same object. The distance function may vary depending on the application, but in this thesis we employ the Euclidean distance between vectors. Let $\boldsymbol{q}^i = ((P_j^i, w_j^i)|(P_j^i, w_j^i) \in \mathcal{Q}_i^{t(2)})$ be a sequencing of $\mathcal{Q}_i^{t(2)}$ sorted in decreasing order of weight. Overlapping trackers are removed by using a greedy approach where the pairs $(P_a^i, w_a^i) \in \boldsymbol{q}^i$ are iteratively taken one by one following the ordering and inserted into a new set $\mathcal{Q}_i^{t(3)}$ whenever they do not overlap with any existing tracker, i.e. :

$$\left(\mathcal{Q}_i^{t(3)}\right)^n = \begin{cases} \left(\mathcal{Q}_i^{t(3)}\right)^{n-1} \cup \{(P_a^i, w_a^i)\}, & \text{if } \forall (P_b^i, w_b^i) \in \left(\mathcal{Q}_i^{t(3)}\right)^{n-1}, \\ & d(r_P(P_a^i), r_P(P_b^i)) \geq \tau_{d_S} \\ \left(\mathcal{Q}_i^{t(3)}\right)^{n-1}, & \text{otherwise,} \end{cases} \tag{3.35}$$

where the exponent n indicates the n-th iteration. At the end of this stage, the final set of trackers $\mathcal{P}_i^{t+1} = \left(\mathcal{Q}_i^{t(3)}\right)^{|\mathcal{R}_i^t|}$ is obtained.

3.3.5 Choosing the final trackers

Considering the temporal consistency of videos, it is interesting to try to avoid changing trackers at each frame. However, in order to recover tracking after abrupt motion or appearance ambiguity, it is also necessary to be able to detect when the tracker should be changed. This is done by considering the temporal score of each tracker. For that, let $\{(\tilde{P}^i, \tilde{w}^i)|i = 0, ..., N_O\}$ be the set of trackers used in the previous frame and $\{(P^i_*, w^i_*)^{t+1}|i = 0, ..., N_O\}$, be the current best ones as defined in Equation 3.32, but for the set $\mathcal{P}^{t+1}_i$. The first candidate trackers for the current frame are given by the set $\mathcal{F}^t_0 = \{(P^i, w^i)|i = 1, ..., N_O\}$, where:

$$(P^i, w^i) = \begin{cases} (\tilde{P}^i, \tilde{w}^i), & \text{if } \tilde{w}^i > \tau_T w^i_* \\ (P^i_*, w^i_*), & \text{otherwise,} \end{cases} \tag{3.36}$$

and τ_T is a given threshold for changing trackers. The final trackers are chosen as every P^j that does not overlap any other P^k by:

$$\mathcal{F}^t_1 = \{(P^j, w^j)|\forall(P^k, w^k) \in \mathcal{F}^t_0, j \neq k, d(r_P(P^j), r_P(P^k)) > \tau_{d_D}\}, \tag{3.37}$$

where τ_{d_D} represents a given distance threshold for different objects.

Overlapping trackers were not included because, when dealing with situations where both appearance and structure are ambiguous (e.g. symmetrical scenes), this method may end up associating multiple trackers with the same object. Therefore, this approach, which tries to find a configuration where each tracker is associated with a different object, is proposed. In other words, configurations where there is less overlapping between trackers of different objects are favored.

Let $\mathcal{M} = \{m\}$ be the set of indices of the objects whose trackers were not included in $\mathcal{F}^t_1$ and $\Omega_P = \{\omega^m = (|\mathcal{P}^{t+1}_m|, w^m_*, \mathcal{P}^{t+1}_m)\}$ be a set of triplets containing the cardinality and best temporal weight of each set of trackers $\mathcal{P}^{t+1}_m$. Let also $\boldsymbol{\Omega_S} = (\omega^m|\omega^m \in \Omega_P)$ be a sequencing of Ω_P where the elements are ordered in increasing order of $|\mathcal{P}^{t+1}_m|$ and secondly by decreasing order of w^M_* (lexicographical order). In other words, the ω^m are ordered by the cardinality of their tracker sets, but any pair (ω^j, ω^k) that has the same cardinality is ordered by its respective weight.

The best tracker for each overlapping object m is chosen by iteratively following the order given by $\boldsymbol{\Omega_S}$. Therefore, the process starts by choosing the best trackers from the objects which have the smallest number of candidate trackers. This is done because, as the trackers are being chosen, the remaining free area (that is not covered by any tracker) is decreased. By processing objects with more trackers later, it is more likely that they will have candidates

in the free area. The set of final trackers for the overlapping objects is obtained by:

$$\mathcal{F}^t_{n+2} = \begin{cases} \mathcal{F}^t_{n+1} \cup (P^m_k, w^m_k), & \text{if } [\exists (P^m_k, w^m_k) \in \mathcal{P}^{t+1}_m | w^m_k \geq \tau_O w^m \text{or} \\ & d(r_P(P^m_k), r_P(P^a_l)) \geq \tau_{d_D}, \forall (P^a_l, w^a_l) \in \mathcal{F}^t_{n+1}] \\ \mathcal{F}^t_{n+1} \cup (P^m, w^m), & \text{otherwise,} \end{cases} \tag{3.38}$$

where $0 \leq n < |\mathcal{M}|$ represents the iteration index, (P^m, w^m) is as defined in Equation 3.36, and τ_O is a given constant which represents a score weight threshold for changing trackers after overlapping. In other words, Equation 3.38 states that a tracker P^i_k is chosen for object m if it passes one of two tests: the first one is that it must not overlap any previously selected tracker for another object. The second test accepts overlapping trackers, but only if they have a high enough score. If no candidate is able to fulfill any of the requirements, then the first candidate obtained previously is chosen.

The final step consists in using the elements $(P^i, w^i) \in \mathcal{F}^t_{|\mathcal{M}|+1}$ to estimate the position of each object at time t, which is obtained by $r_P(P^i)$. The tracking update procedure that is applied at each frame is summarized in Algorithm 3.1.

```
Algorithm update_tracking(G^M, M_A, M_C, P^{t-1}, τ_{d_D}, τ_{d_S}, τ_O, τ_S, τ_T)
    Data: A model graph G^M, adjacency and candidate matrices M_A and M_C,
          respectively, a set of sets of trackers P^{t-1} = {P_i^{t-1}}, where
          P_i^{t-1} = {(P_j^i, w_j^i)} and various thresholds τ
    Result: A set P^t with the updated trackers and another set F^t = {F_i | F_i ∈ P_i^t}
            containing the tracking decision for each object i
    P^t ← P^{t-1};
    for 1 ≤ i ≤ |P^t| do
        P_i^t ← P_i^t ∪ generate_candidates(G^M, M^C, i);
    foreach P_i^t ∈ P^t do
        foreach (P_j^i, w_j^i) ∈ P_i^t do
            apply_dynamics(P_j^i);
    best_weights ← [];
    for 1 ≤ i ≤ |P^t| do
        (P_*^i, w_*^i) ← arg max_{(P_j^i, w_j^i) ∈ P_i^t} w_j^i;
        best_weights.append((P_*^i, w_*^i));
    best_weights.sort(key = w_*^i);
    foreach (P_*^i, w_*^i) ∈ best_weights do
        foreach (P_j^i, w_j^i) ∈ P_i^t do
            V ← P_j^i ∪ {P_*^k | k ≠ i};
            G^S ← build_graph(V, M_A);
            f ← graph_score(G^S);
            w_j^i ← update_score(f);
        remove_low_scores(P_i^t, τ_S);
        remove_overlapping(P_i^t, τ_{d_S});
    F^t ← choose_best_trackers(P^t, τ_{d_D}, τ_O, τ_T);
    return P^t, F^t;
```

Algorithm 3.1: Tracking update algorithm.

Chapter 4

Action recognition

In this chapter, the proposed approach for performing action recognition will be explained. Action classification is applied on each person present in the scene independently. Therefore, in one single scene many actions may be performed at the same time. It is also assumed that the actions of each individual change over time. Figure 4.1 shows an overview of the action framework.

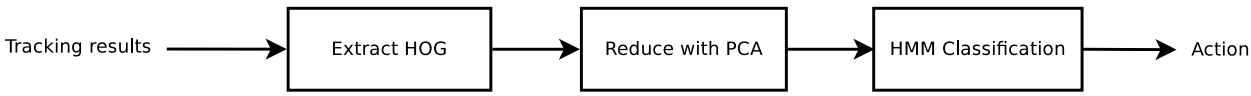


Figure 4.1: *Overview of the action recognition framework.*

The action classification is performed using Hidden Markov Models which operate over sequences of observations. The process starts by accumulating the results obtained from the tracking phase. Each person i at instant t is represented by a sliding window of tracking results $\tilde{\boldsymbol{o}}_t^i = (\tilde{o}^i_{t-\lfloor N_W/2 \rfloor}, \tilde{o}^i_{t-\lfloor N_W/2 \rfloor+1}, ..., \tilde{o}^i_t, ..., \tilde{o}^i_{t+\lfloor N_W/2 \rfloor-1}, \tilde{o}^i_{t+\lfloor N_W/2 \rfloor})$, where N_W is the length of the window. In order to classify the sequence, feature vectors are extracted from each $\tilde{o}^i_t$. This is accomplished by first computing the Histogram of Oriented Gradients (Section 4.1) and then reducing it using Principal Component Analysis (Section 4.2), yielding the new sequence of observations $\boldsymbol{o}^i_t$.

As usual, it is assumed that the set of possible actions is known beforehand. For each action j, a different HMM Z_j is trained using annotated data. Since the training is supervised, each training instance consists of a sequence and an action label. The detailed training steps are explained in Section 4.3.3. After all HMMs are trained, the classification consists in finding which of the Z_j best represents each observation $\boldsymbol{o}^i_t$ (Section 4.3.2). As evidenced by the definition of $\boldsymbol{o}^i_t$, we follow the conventional approach of building the sequence around the center. The main advantage is that in this way it is possible to provide a label while the action is being performed, as opposed to providing it at the end. On the other hand, it also implies that future data is required. Therefore, the action recognition must be delayed by

$\lfloor N_W/2 \rfloor$ frames. However, as N_W is usually small compared to the frame rate of the videos, the delay is not significant in practice.

4.1 Histogram of Oriented Gradients (HOG)

The Histogram of Gradients is a descriptor proposed by Dalal and Triggs (2005). It is used for representing the shape of an object, which is built by computing the image gradients and then populating a histogram in which each bin represents one gradient direction.

Computing the HOG consists in applying a series of standard methods in sequence over the initial image until obtaining the final histogram. In their paper, Dalal and Triggs (2005) provide a detailed coverage of many choices for each step and the observed results. For brevity, in this section only the actual method used in this work, which corresponds to the optimal configuration found by the authors, will be detailed. Figure 4.2 shows the main steps involved in computing the HOG.

Figure 4.2: *Overview of HOG chain. Adapted from (Dalal and Triggs (2005)).*

The first step consists in computing the gradient of the image. This may be done by convolving the image with the simple centered kernels $\boldsymbol{K_H} = [-1, 0, 1]$ and $\boldsymbol{K_V} = [-1, 0, 1]^T$ for obtaining the horizontal and vertical gradients, respectively. The convolution of an image I by a kernel matrix $\boldsymbol{K}$ with N_R rows and N_C columns is represented by $I_C = I * \boldsymbol{K}$ and can be computed by:

$$I_C(x, y) = \sum_{r=-\lfloor N_R/2 \rfloor}^{\lfloor N_R/2 \rfloor} \sum_{c=-\lfloor N_C/2 \rfloor}^{\lfloor N_C/2 \rfloor} I(x + c, y + r)\boldsymbol{K}\left(\lfloor N_C/2 \rfloor + c, \lfloor N_R/2 \rfloor + r\right), \tag{4.1}$$

where the values inside the parenthesis indicate a position inside the image or kernel.

The convolution of I by $\boldsymbol{K_H}$ and $\boldsymbol{K_V}$ yields two gradient images I_{GX} and I_{GY}, respectively. These images are then used to compute the gradient direction at each pixel:

$$I_{GD}(x, y) = \arctan\left(\frac{I_{GY}(x, y)}{I_{GX}(x, y)}\right), \tag{4.2}$$

and the gradient magnitudes:

$$I_{GM}(x, y) = \sqrt{I_{GX}(x, y)^2 + I_{GY}(x, y)^2}. \tag{4.3}$$

The image is then divided in cells using a rectangular grid. Typically, each cell is an 8×8

square. Each pixel inside the cell then casts a vote for its gradient orientation, to create a cell histogram. The votes are weighted by I_{GM}, i.e. stronger gradients contribute more to the histogram. According to the authors, building a histogram with 9 bins using the unsigned gradients values (only considering the interval $[0°, 180°]$) provides the best results.

To account for contrast and illumination changes, the cells are grouped into larger rectangular blocks, which partially intersect each other. The histograms of each cell inside the block are then concatenated to form a larger vector. In this work, each block is a square containing $2 \times 2 = 4$ cells. The blocks share two cells with a neighbor block, i.e. the bottom two cells of a block are shared with its bottom neighbor and so on. Among the normalization choices, the most widely used and also employed here is the $L_2 - Hys$ norm. This norm is computed by first computing the L_2 norm followed by a clipping which limits the maximum values to 0.2. After this step, the resulting vector is once again normalized with the L_2 norm.

After normalizing all the blocks, the HOG descriptor is obtained by concatenating all the block histograms. One important fact to notice is that the HOG descriptor by itself is not invariant to shape and scale changes. Therefore, a fixed window size must be chosen beforehand to represent the object of interest. Since HOG is most widely applied for person detection, the window size is typically chosen as a vertical rectangle of 64×128 pixels. Due to the construction constraints, HOG descriptors are only comparable when extracted from windows of the same size. In order to overcome this restriction, the most commonly adopted approach is to resize the original detection to the chosen size before extracting its HOG.

As mentioned, each cell histogram is represented by a vector of length 9. As each block is a square comprising 4 cells, then its corresponding histogram contains 36 bins. Each block can be viewed as a 16×16 square, where half of it intersects with its horizontal and vertical neighbors. Since the window size is 64×128 pixels, it can be seen that the window contains $7 \times 15 = 105$ blocks. Finally, by multiplying this value by the histogram length of each block, it can be seen that the resulting HOG vector has a high dimensionality of $105 \times 36 = 3780$. As it is usually intractable to classify data of this order of magnitude, we chose to reduce the descriptor length by using Principal Component Analysis.

4.2 Principal Components Analysis (PCA)

Essentially, PCA aims at applying an orthonormal transformation to multivariate data so that it becomes uncorrelated. The main point is that, if the data is uncorrelated, then it is possible to ignore (remove) one variable (axis) without affecting the variance of the others. In other words, by applying PCA it is possible to reduce the dimensionality of the data by discarding bases while minimizing the variance loss. This property is important because, according to Information Theory, the more uncertainty exists, the more information is available. Thus, keeping the data with as much variance as possible also minimizes the

information loss.

The contents of this section are based on the publications of Callioli *et al.* (2007) and Shlens (2014). Let $\boldsymbol{O} = [\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_N]^T$ be a data matrix where each $\boldsymbol{o}_i$ is one observation vector. It will be assumed that all the variables of $\boldsymbol{O}$ have zero mean. Let also $\boldsymbol{C_O} = \frac{1}{N}\boldsymbol{O}\boldsymbol{O}^T$ be the covariance matrix of $\boldsymbol{O}$. The goal of PCA is to find a matrix $\boldsymbol{P}$ to transform the data as $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{O}$ such that the correlation matrix $\boldsymbol{C_Y} = \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^T$ is diagonal.

As shown in the paper by Shlens (2014), this problem corresponds to finding the eigenvectors of $\boldsymbol{C_O}$. In fact, as also explained in the same material, another approach, and actually the most used in practice, is performing the Singular Value Decomposition to obtain the Principal Components. However, explaining the SVD is out of the scope of this thesis, so we will focus on giving an overview of how to obtain the eigenvectors of the covariance matrix.

The eigenvectors of a square matrix, such as $\boldsymbol{C_O}$, can be found by solving:

$$\begin{aligned} \boldsymbol{C_O}\boldsymbol{v} &= \lambda\boldsymbol{v} \\ (\boldsymbol{C_O} - \lambda\boldsymbol{I})\boldsymbol{v} &= 0, \end{aligned} \tag{4.4}$$

where $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{C_O}$ corresponding to the eigenvalue λ and $\boldsymbol{I}$ is the identity matrix. From this formulation, it is clear that the eigenvalues are necessary to compute the eigenvectors. Eigenvalues can be obtained by solving the characteristic polynomial of $\boldsymbol{C_O}$:

$$p(\lambda) = \det(\boldsymbol{C_O} - \lambda\boldsymbol{I}). \tag{4.5}$$

Each eigenvector will represent a base of the space transformation. Data reduction can be performed by simply transforming the data using only the bases with the highest variances, which minimizes the information loss.

4.3 Hidden Markov Model (HMM)

Hidden Markov Models are able to model the statistical properties of signals that present some temporal constraints. More specifically, it can be successfully used for problems that follow the Markov property, i.e. the state at time t only depends on a finite set of states at previous time instants. The most widely studied situation assumes that the state at time t only depends on the state at time $t-1$. In this case it is said that the HMM follows a Markov property of **first order**. In this chapter it is assumed that the HMMs are always of first order. The contents of this section are based on the publications of Duda *et al.* (2001) and Fink (2008).

In a typical Markov problem, the signal is represented by a sequence of observations

at successive time instants. Let $\boldsymbol{o} = (o_1, o_2, ..., o_T)$ be such a sequence of length T. It is assumed that this sequence is generated by a discrete stochastic process, which includes a finite number of states and state transitions. More formally, a basic Markov model (without hidden states) is represented by a finite set of states

$$\mathcal{S} = \{s_1, s_2, ..., s_{N_S}\}, \tag{4.6}$$

which will be referred to only by their indices, i.e. $i = s_i$. It also requires a matrix of state transition probabilities

$$\boldsymbol{A} = \{a_{ij} | a_{ij} = \mathbb{P}(S_t = j | S_{t-1} = i)\} \text{ such that } \sum_j a_{ij} = 1 \text{ for all } i \tag{4.7}$$

and a vector of starting probabilities

$$\boldsymbol{\pi} = (\pi_i | \pi_i = \mathbb{P}(S_1 = i)). \tag{4.8}$$

A Markov model can be interpreted as a complete directed graph where the arcs are labeled with transition probabilities. Figure 4.3 shows an example of a Markov model.

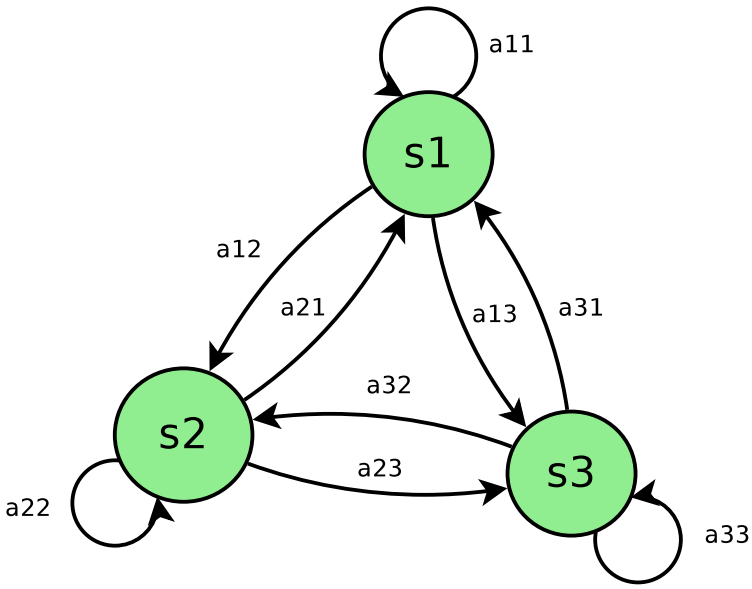


Figure 4.3: *Example of a Markov model with three states. Each circle represents one state, and the directed edges the transitions from one state to another*

In the basic model, it is assumed that the observation corresponds to the states themselves, i.e. each o_i is directly mapped to a state s_i. However, in real situations it is often not possible to observe the states directly. For example, when modeling human speech, one might use each state to represent one phoneme and then follow one sequence of phonemes to produce a word. In practice, though, the phonemes cannot be perceived directly, only

the sound they produce. Therefore, it is necessary to extend the basic model to account for the visible states (sounds) and the invisible ones (phonemes). By doing so, the HMM is obtained. In this approach, the basic Markov model discussed before corresponds to the hidden part, that cannot be directly observed. Each hidden state now emits a visible symbol (observation) o_i. This symbol is generated according to a distribution that can be either discrete (a finite set of possible observations) or continuous (modeled by a probability density function). In this project the emissions are obtained by continuous distributions given by a Gaussian mixture model. Hence, the emission probabilities are modeled by the vector:

$$\boldsymbol{b} = \left(b_j \middle| b_j = \sum_{m=1}^{N_M} c_{jm} \mathcal{N}(\boldsymbol{\mu}_{jm}, \boldsymbol{C}_{jm}) \right) \tag{4.9}$$

where c_{jm} is a normalization constraint such that

$$\sum_{m=1}^{N_M} c_{jm} = 1 \text{ and } 0 \leq c_{jm} \leq 1, \text{for all } (j, m). \tag{4.10}$$

The probability of a hidden state j emitting a specific observation o_k shall be represented by:

$$b_j(o_k) = \mathbb{P}(o_k | S_t = j) = \sum_{m=1}^{N_M} c_{jm} \mathcal{N}(o_k | \boldsymbol{\mu}_{jm}, \boldsymbol{C}_{jm}) = \sum_{m=1}^{N_M} c_{jm} g_{jm}(o_k), \tag{4.11}$$

where $\mathcal{N}(o_k | \boldsymbol{\mu}_m, \boldsymbol{C}_m) = g_{jm}(o_k)$ represents the probability of obtaining o_k from a multivariate Gaussian (normal) distribution with mean vector $\boldsymbol{\mu}_{jm}$ and covariance matrix $\boldsymbol{C}_{jm}$. Figure 4.4 shows an example of a HMM.

When working with HMMs, there are usually three central problems to be considered: evaluation, decoding and learning. However, in the scope of this thesis, the decoding problem is not going to be studied and thus, will not be explained. On the other hand, another problem will be considered, which will be referred as the classification problem. A short description of each problem is provided below:

- Evaluation: This problem consists in, given a HMM, determining the probability that an observed sequence was generated from that model.
- Classification: In this problem, it will be assumed that a set of different HMMs is available and the interest is in finding which of those best represents a particular sequence.
- Learning: This problem is concerned to, given a set of observations and a coarse structure of the model (e.g. number of hidden states), finding the parameters that maximize the probability of the model generating those observations.

In the following sections, the three aforementioned problems will be further explained.

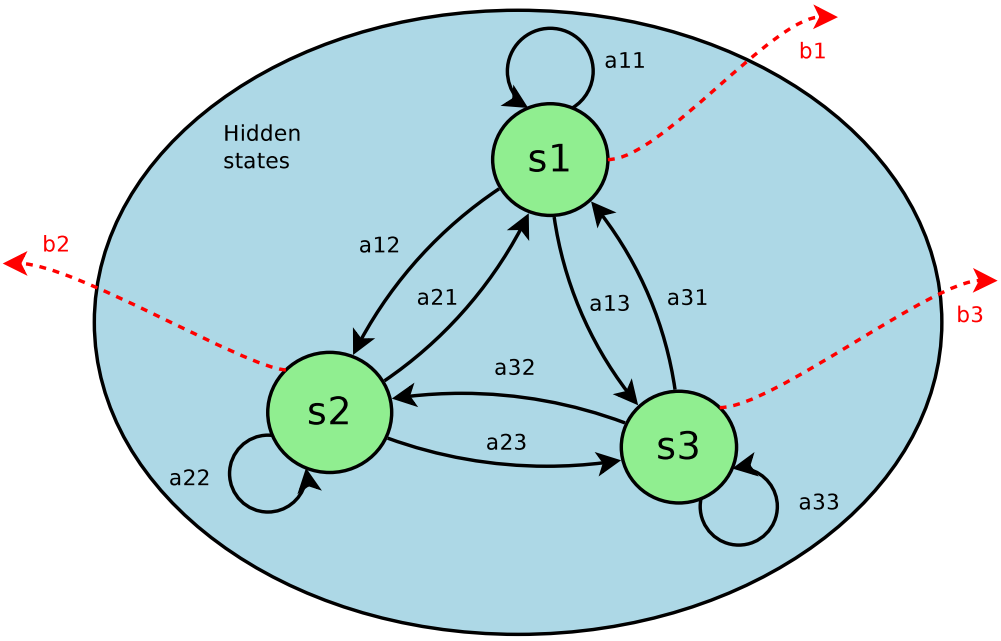


Figure 4.4: *Example of a Hidden Markov model. The three hidden states (inside the ellipse) represent the unobservable properties of the data. Each hidden state has a probability of emitting a certain observable signal (red arrows).*

4.3.1 Evaluation

Suppose that a sequence $\boldsymbol{o} = (o_1, o_2, ..., o_T)$ is given. Since each hidden state can only emit one observation at a time, then this sequence can only be generated by walking through a path of states $\boldsymbol{s} = (s_1, s_2, ..., s_T)$ with the same length T. If the path is known, then the production probability of a HMM Z can be simply obtained by

$$\mathbb{P}(\boldsymbol{o}|\boldsymbol{s}, Z) = \prod_{t=1}^{T} b_t(o_t). \tag{4.12}$$

The probability that such a path $\boldsymbol{s}$ is followed is computed by the product of the respective state transition probabilities. If we define $a_{0i} = \pi_i$ and $s_0 = 0$, we can write:

$$\mathbb{P}(\boldsymbol{s}|Z) = \pi_1 \prod_{t=2}^{T} a_{t-1,t} = \prod_{t=1}^{T} a_{t-1,t}. \tag{4.13}$$

By combining Equations 4.12 and 4.13, it is possible to compute the probability that the sequence $\boldsymbol{o}$ is generated by following the specific path $\boldsymbol{s}$ by:

$$\mathbb{P}(\boldsymbol{o}, \boldsymbol{s}|Z) = \mathbb{P}(\boldsymbol{o}|\boldsymbol{s}, Z)\mathbb{P}(\boldsymbol{s}|Z) = \prod_{t=1}^{T} a_{t-1,t} b_t(o_t). \tag{4.14}$$

In order to compute the total production probability, it is necessary to accumulate over all the possible paths, which include all possible permutations of states in a sequence of

length T. This can be expressed as:

$$\mathbb{P}(\boldsymbol{o}|Z) = \sum_{\boldsymbol{s}} \mathbb{P}(\boldsymbol{o}, \boldsymbol{s}|Z) = \sum_{\boldsymbol{s}} \mathbb{P}(\boldsymbol{o}|\boldsymbol{s}, Z)\mathbb{P}(\boldsymbol{s}|Z). \tag{4.15}$$

As there are ${N_S}^T$ possible paths, if this approach were implemented as an algorithm, its complexity would be $O(T{N_S}^T)$. For most problems, an exponential complexity is not tractable and thus, cannot be used in practice. Fortunately, a much more efficient approach can be derived by exploring the Markov property.

As it can be seen in Equation 4.14, each term only depends on the current and previous states. It is not necessary to know at which time point each state is visited or in which order. Therefore, the computations for each state can be done in parallel throughout the time line. To this aim, a recurrence relation that leads to the well-known **forward algorithm** can be used. The first step is to define the forward variables $\alpha_t(i)$, that correspond to the probability that the first part of the observation until O_t is evaluated and the state i is reached at time t:

$$\alpha_t(i) = \mathbb{P}(O_1, O_2, ..., O_t, S_t = i|Z). \tag{4.16}$$

At the beginning, the probability $\alpha_1(i)$ of generating the observation O_1 at state i is given by the starting probability π_i and the emission probability $b_i(O_1)$:

$$\alpha_1(i) = \pi_i b_i(O_1). \tag{4.17}$$

Afterwards, the new probabilities for the next time instant $t+1$ can be obtained by combining the results computed at the last time instant t. For each state j at time $t+1$, it is necessary to consider all possible paths that come from all other states i at time t. Additionally, the new observation O_{t+1} must be generated from s_j. Therefore, the forward variables can be updated by:

$$\alpha_{t+1}(j) = \left(\sum_i \alpha_t(i) a_{ij} \right) b_j(O_{t+1}). \tag{4.18}$$

By computing until time T, N_S results will be obtained, one for each state. As the complexity of computing each path is $O(TN_S)$, the complexity of the forward algorithm is $O(T{N_S}^2)$. Finally, the probability of model Z producing the sequence $\boldsymbol{o}$ is:

$$\mathbb{P}(\boldsymbol{o}|Z) = \sum_{i=1}^{N} \alpha_T(i). \tag{4.19}$$

The pseudocode for performing the Forward procedure is shown in Algorithm 4.1. The production probability can be directly obtained by summing up the respective values of the

resulting matrix, as presented in Algorithm 4.2.

```
Algorithm forward(o, Z)
    Data: A sequence of observations o = (o_1, o_2, ..., o_T) and a HMM Z = (S, π, A, b)
    Result: A N_S × T matrix α containing all α_t(i) values
    α ← zeros_matrix(N_O, T);
    for 1 ≤ i ≤ N_S do
        α_1(i) ← π_i b_i(o_1);
    for 1 ≤ t ≤ T − 1 do
        for 1 ≤ j ≤ N_S do
            for 1 ≤ i ≤ N_S do
                α_{t+1}(j) ← α_{t+1}(j) + α_t(i) a_{ij} b_j(o_{t+1});
    return α;
```

Algorithm 4.1: Forward algorithm.

```
Algorithm prod_prob(α)
    Data: A matrix of forward variables α
    Result: The production probability P(o|Z)
    prob ← 0;
    for 1 ≤ i ≤ N_S do
        prob ← prob + α_T(i);
    return prob;
```

Algorithm 4.2: Production probability algorithm.

4.3.2 Classification

In some situations, several HMMs Z_i, each representing a different class ω_i are available. In this case, an observation $\boldsymbol{o}$ may be classified by finding the model Z_j that maximizes the posterior probability $Z_j = \arg\max_i \mathbb{P}(Z_i|\boldsymbol{o})$. By using the Bayes' theorem, this expression may be rewritten as:

$$Z_j = \arg\max_i \frac{\mathbb{P}(\boldsymbol{o}|Z_i)\mathbb{P}(Z_i)}{\mathbb{P}(\boldsymbol{o})}. \tag{4.20}$$

Since the denominator is independent of Z_i, it can be ignored for the classification purpose, yielding:

$$Z_j = \arg\max_i \mathbb{P}(\boldsymbol{o}|Z_i)\mathbb{P}(Z_i). \tag{4.21}$$

Notice that the first term is the production probability explained in Section 4.3.1, which implies that the classification problem depends on the evaluation problem. Besides that, it is also necessary to know $\mathbb{P}(Z_i)$. This knowledge may come from an expert or from observation of the data, if available. For simplicity, however, this term is often ignored, and classification is done based solely on the evaluation result. In practice, this corresponds to setting $\mathbb{P}(Z_i) = 1/N$ for all $0 < i \leq N$.

4.3.3 Learning

A good model represents the properties of the data as precisely as possible. In order to do so, the free parameters of the model can be improved by observing the data. In the case of HMMs, the free parameters are the initial probabilities $\boldsymbol{\pi}$, the transition probabilities $\boldsymbol{A}$ and the emission probabilities $\boldsymbol{b}$. The values of each of these parameters provides an answer to the following question: out of the total outputs one state can provide, how many of them are expected to follow this specific path? Using mathematical terms, this corresponds to finding the optimized parameters $\hat{a}_{ij}$ and $\hat{b}_i(o_k)$ defined as:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from } i \text{ to } j}{\text{expected number of transitions out of state } i}, \tag{4.22}$$

$$\hat{b}_i(o_k) = \frac{\text{expected number of emissions } o_k \text{ from state } i}{\text{total number of emissions from state } i}. \tag{4.23}$$

When using continuous emission models, the second step actually corresponds to finding the parameters of the probability density functions that best model the observations.

Two fundamentally different methods exist for training: the Baum-Welch and the Viterbi algorithms (more details in the book of Fink (2008)). The former can be seen as the Expectation Maximization algorithm applied to HMMs. The Baum-Welch algorithm evaluates all possible paths and thus, find parameters that optimize the model for all possible state configurations for the given observations. The Viterbi algorithm, on the other hand, first try to estimate the most likely path each observation will take and then optimize the parameters only for the obtained path. The Viterbi approach may be more efficient but it does not always provide the best global configuration.

In this work, we will use the Baum-Welch algorithm. One of the main reasons for this choice is that the Baum-Welch results can be used to analytically optimize the parameters of the continuous emission models, which is not possible when using the Viterbi approach.

Forward-Backward Algorithm

In order to optimize the parameters of the model, it is necessary to know what is the probability $\mathbb{P}(S_t = i|\boldsymbol{o}, Z)$ of being in a state i at a given instant t. This can be done by using the Forward-Backward algorithm. Very similarly to the forward case, but in the opposing sense, backward variables are defined to represent the probability of generating the remaining sequence $(O_{t+1}, O_{t+2}, ..., O_T)$ starting from state i:

$$\beta_t(i) = \mathbb{P}(O_{t+1}, O_{t+2}, ..., O_T|S_t = i, Z) \tag{4.24}$$

As in the forward approach, $\beta_t(i)$ can be computed by using a recurrence relation of the form:

$$\beta_t(i) = \begin{cases} 1, & \text{if } t = T \\ \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), & \text{otherwise.} \end{cases} \tag{4.25}$$

Having both the forward and backward variables, it is possible to compute the desired probability. First, by applying the rule from Equation 3.5, we have:

$$\mathbb{P}(S_t = i | \boldsymbol{o}, Z) = \frac{\mathbb{P}(S_t = i, \boldsymbol{o} | Z)}{\mathbb{P}(\boldsymbol{o} | Z)} \tag{4.26}$$

The denominator is the production probability obtained by solving the evaluation problem. The numerator can be rewritten using the forward and backward variables:

$$\begin{aligned} \mathbb{P}(S_t = i, \boldsymbol{o} | Z) &= \mathbb{P}(O_1, O_2, ..., O_t, S_t = i | Z) \mathbb{P}(O_{t+1}, O_{t+2}, ..., O_T | S_t = i, Z) \\ &= \alpha_t(i) \beta_t(i) \end{aligned} \tag{4.27}$$

By replacing this in Equation 4.26, we obtain:

$$\gamma_t(i) = \mathbb{P}(S_t = i | \boldsymbol{o}, Z) = \frac{\alpha_t(i) \beta_t(i)}{\mathbb{P}(\boldsymbol{o} | Z)} \tag{4.28}$$

The value $\gamma_t(i)$ represents the probability of being in state i at instant t. Analogously, it also represents the expected number of transitions out of state i at the next time step. A pseudo code of the Backward procedure is provided in Algorithm 4.3. As it can be seen, the algorithm is just a reversed version of the Forward procedure. By combining these results with those obtained from the Forward step (Algorithm 4.1), the values of $\gamma_t(i)$ can be easily computed.

```
Algorithm backward(o, Z)
    Data: A sequence of observations o = (o_1, o_2, ..., o_T) and a HMM Z = (S, π, A, b)
    Result: A N_S × T matrix β containing all β_t(i) values
    β ← zeros_matrix(N_O, T);
    for 1 ≤ i ≤ N_S do
        β_T(i) ← 1;
    for T − 1 ≥ t ≥ 1 do
        for 1 ≤ j ≤ N_S do
            for 1 ≤ i ≤ N_S do
                β_t(j) ← β_t(j) + a_ij b_j(o_{t+1}) β_{t+1}(i);
    return β;
```

Algorithm 4.3: Backward algorithm.

Baum-Welch Algorithm

The Baum-Welch algorithm optimizes the model parameters by using the production probability $\mathbb{P}(\boldsymbol{o}|Z)$. This algorithm guarantees that the new model $\hat{Z}$ produces the observations with equal or higher probability:

$$\mathbb{P}(\boldsymbol{o}|\hat{Z}) \geq \mathbb{P}(\boldsymbol{o}|Z) \tag{4.29}$$

Notice that this method may not provide the best possible fit, as it is bounded to stop at a local maximum. One simple way to overcome this problem is to run several training instances and choose the best one.

The Baum-Welch represents a variant of the Expectation Maximization algorithm. Besides representing the probability $\gamma_t(i) = \mathbb{P}(S_t = i|\boldsymbol{o}, Z)$ for a state in general, it is also interesting to represent the same probability but restricted to a specific transition from state i to j as follows:

$$\begin{aligned}\gamma_t(i,j) &= \mathbb{P}(S_t = i, S_{t+1} = j|\boldsymbol{o}, Z)\\ &= \frac{\mathbb{P}(S_t = i, S_{t+1} = j, \boldsymbol{o}|Z)}{\mathbb{P}(\boldsymbol{o}|Z)}\\ &= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\mathbb{P}(\boldsymbol{o}|Z)}.\end{aligned} \tag{4.30}$$

Notice that this value represents the overall probability of going through transition (i, j) at instant t. By computing $\gamma_t(i, j)$ for each pair (i, j) over all time instants t and averaging by the probability of leaving state i, the new estimates for the transition probabilities are obtained:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \mathbb{P}(S_t = i, S_{t+1} = j|\boldsymbol{o}, Z)}{\sum_{t=1}^{T-1} \mathbb{P}(S_t = i|\boldsymbol{o}, Z)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{4.31}$$

The probability of leaving each state at the first time instant can be used as a starting probability:

$$\hat{\pi}_i = \mathbb{P}(S_1 = i|\boldsymbol{o}, Z). \tag{4.32}$$

When using discrete emission probabilities, the new values can also be estimated analogously:

$$\hat{b}_j(o_k) = \frac{\sum_{t=1}^{T-1} \mathbb{P}(S_t = j, O_t = o_k|\boldsymbol{o}, Z)}{\sum_{t=1}^{T} \mathbb{P}(S_t = j|\boldsymbol{o}, Z)} = \frac{\sum_{t:O_t=o_k} \mathbb{P}(S_t = j|\boldsymbol{o}, Z)}{\sum_{t=1}^{T} \mathbb{P}(S_t = j|\boldsymbol{o}, Z)} = \frac{\sum_{t:O_t=o_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{4.33}$$

Notice that in the numerator of the equation only the instants t where o_k was observed are considered.

However, when using continuous emission models, the parameters of the mixture models

must be computed instead. This can be done by following a similar procedure to that used for discrete models. In order to do so, we observe that the parameters c_{jk} of the mixture models merely represent the expected number of times the k-th mixture will be used when generating the emission for state j. Let $\xi_t(j,k)$ represent the probability of using mixture k in state j at instant t, defined as follows:

$$\begin{aligned}\xi_t(j,k) &= \mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)\\ &= \frac{\mathbb{P}(S_t = j, M_t = k, \boldsymbol{o}|Z)}{\mathbb{P}(\boldsymbol{o}, Z)}\\ &= \frac{\sum_{i=1}^{N}\mathbb{P}(S_{t-1} = i, S_t = j, M_t = k, \boldsymbol{o}|Z)}{\mathbb{P}(\boldsymbol{o}, Z)}\\ &= \frac{\sum_{i=1}^{N}\alpha_{t-1}(i)a_{ij}c_{jk}g_{jk}(o_t)\beta_t(j)}{\mathbb{P}(\boldsymbol{o}, Z)}.\end{aligned} \tag{4.34}$$

Once again, by using the same procedure as in Equation 4.31, the new mixture weights can be obtained by:

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T}\mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)}{\sum_{t=1}^{T}\mathbb{P}(S_t = j|\boldsymbol{o}, Z)} = \frac{\sum_{t=1}^{T}\xi_t(j,k)}{\sum_{t=1}^{T}\gamma_t(j)}. \tag{4.35}$$

By knowing the probability of each mixture being selected, it is now possible to update the parameters of each one of them. Since a Gaussian mixture model is being used, the parameters to be estimated are the mean vector and the covariance matrix. They can both be computed by using the classical formulas for statistical mean and covariance, weighted by the probability of the mixture being chosen as follows:

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T}\mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)o_t}{\sum_{t=1}^{T}\mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)} = \frac{\sum_{t=1}^{T}\xi_t(j,k)o_t}{\sum_{t=1}^{T}\xi_t(j,k)}. \tag{4.36}$$

$$\begin{aligned}\hat{\boldsymbol{C}}_{jk} &= \frac{\sum_{t=1}^{T}\mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)(o_t - \hat{\boldsymbol{\mu}}_{jk})(o_t - \hat{\boldsymbol{\mu}}_{jk})^T}{\sum_{t=1}^{T}\mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)}\\ &= \frac{\sum_{t=1}^{T}\xi_t(j,k)(o_t - \hat{\boldsymbol{\mu}}_{jk})(o_t - \hat{\boldsymbol{\mu}}_{jk})^T}{\sum_{t=1}^{T}\xi_t(j,k)}.\end{aligned} \tag{4.37}$$

By considering that the variance of a set of data $\boldsymbol{X}$ can be rewritten as:

$$\mathrm{Var}[\boldsymbol{X}] = \mathrm{E}[\boldsymbol{X}\boldsymbol{X}^T] - \mathrm{E}[\boldsymbol{X}]\mathrm{E}[\boldsymbol{X}]^T, \tag{4.38}$$

Equation 4.37 can be rewritten as:

$$\begin{aligned}\hat{\boldsymbol{C}}_{jk} &= \frac{\sum_{t=1}^{T} \mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z) o_t {o_t}^T}{\sum_{t=1}^{T} \mathbb{P}(S_t = j, M_t = k|\boldsymbol{o}, Z)} - \hat{\boldsymbol{\mu}}_{jk}\hat{\boldsymbol{\mu}}_{jk}^T \\ &= \frac{\sum_{t=1}^{T} \xi_t(j,k) o_t {o_t}^T}{\sum_{t=1}^{T} \xi_t(j,k)} - \hat{\boldsymbol{\mu}}_{jk}\hat{\boldsymbol{\mu}}_{jk}^T. \end{aligned} \tag{4.39}$$

The advantage of this formulation is that each side of the difference can be computed in parallel in a single pass through the data. A pseudo code of the Baum-Welch procedure can be seen in Algorithm 4.4.

```
Algorithm baum-welch(o, Z, stop_thresh, max_iter)
    Data: A sequence of observations o = (o_1, o_2, ..., o_T), an initial HMM
          Z = (S, π, A, b), a stopping threshold stop_thresh and an iteration
          threshold max_iter
    Result: The updated HMM Ẑ
    count ← 0;
    diff ← ∞;
    α ← forward(o, Z);
    β ← backward(o, Z);
    prev_prob ← prod_prob(α);
    while diff > stop_thresh and count < max_iter do
        π̂ ← zeros_matrix(N_S, 1);
        Â ← zeros_matrix(N_S, N_S);
        b̂ ← zeros_matrix(N_S, 1);
        for 1 ≤ i ≤ N_S do
            γ(i) ← 0;
            for 1 ≤ t ≤ T do
                γ_t(i) ← α_t(i)β_t(i)/prev_prob;
                γ(i) ← γ(i) + γ_t(i);
            for 1 ≤ j ≤ N_S do
                sum_γ_t(i, j) ← 0;
                for 1 ≤ t ≤ T − 1 do
                    γ_t(i, j) ← α_t(i)A_ij b_j(O_{t+1})β_{t+1}(i)/prev_prob;
                    sum_γ_t(i, j) ← sum_γ_t(i, j) + γ_t(i, j)
                Â_ij = sum_γ_t(i, j)/γ(i);
            for 1 ≤ k ≤ N_M do
                sum_ξ_t(i, k) ← 0;
                sum_ξ_t(i, k)x ← zeros_matrix(N_O, 1);
                sum_ξ_t(i, k)xx^T ← 0;
                for 1 ≤ t ≤ T − 1 do
                    ξ_t(i, k) ← 0;
                    for 1 ≤ j ≤ N_S do
                        ξ_t(i, k) ← ξ_t(i, k) + α_t(j)a_ji c_ik g_ik(O_t)β_t(i)/prev_prob;
                    sum_ξ_t(i, k) ← sum_ξ_t(i, k) + ξ_t(i, k);
                    sum_ξ_t(i, k)x ← sum_ξ_t(i, k)x + ξ_t(i, k)o_t;
                    sum_ξ_t(i, k)xx^T ← sum_ξ_t(i, k)xx^T + ξ_t(i, k)o_t o_t^T;
                ĉ_ik ← sum_ξ_t(i, k)/γ(i);
                μ̂_ik ← sum_ξ_t(i, k)x/sum_ξ_t(i, k);
                Ĉ_ik ← sum_ξ_t(i, k)xx^T/sum_ξ_t(i, k) − μ̂_ik μ̂_ik^T;
        b̂ ← (ĉ, μ̂, Ĉ);
        Ẑ ← (S, π̂, Â, b̂);
        α ← forward(o, Ẑ);
        β ← backward(o, Ẑ);
        prob ← prod_prob(α);
        diff ← abs(prev_prob − prob);
        (π, A, b) ← (π̂, Â, b̂);
        prev_prob ← prob;
        count ← count + 1;
    return Ẑ;
```

Chapter 5

Results

In this chapter, the results both for the tracking and the action recognition modules will be presented and discussed. For each module, the datasets used for training and testing are explained. Afterwards, we present how the parameters for each method were computed based on training data. Finally, experimental results are shown.

5.1 Multi-object tracking

The software was developed in Python with the OpenCV library[1]. As explained in Section 3.1, each object is individually tracked using particle filters with color histograms as proposed by Pérez *et al.* (2002).

5.1.1 Evaluation measurements

The tracking results are evaluated using six measurements for each frame. First, let N_o^i be the number of expected objects on frame i and N_f the number of frames of the video. Let also GB_j^i and EB_j^i be the sets of points (pixels) inside the groundtruth and estimated bounding boxes, respectively, of the object j and $c(B)$ represent the centroid of a bounding box B. All the measurements are computed for every frame and then averaged by dividing them by $\sum_{i=1}^{N_f} N_o^i$.

The first measurement is the commonly used center error (CERR), which consists of the Euclidean distance between the centroids of the bounding boxes, defined as

$$\text{CERR} = \sum_j d(c(GB_j^i), c(EB_j^i)). \tag{5.1}$$

The second measurement is the center displacement (CDIS) measure that evaluates the stability of the tracking. This is done by computing the distance between the center of each

[1] http://opencv.org/

corresponding bounding box between two consecutive frames:

$$\text{CDIS} = \sum_j d(c(EB_j^i), c(EB_j^{i-1})) \tag{5.2}$$

The third measurement is the hit detection ratio (HITR). As in (Su *et al.* (2014)), a detection is considered successful when $I(c(GB_j^i), EB_j^i) = 1$, where $I(\cdot)$ is an indicative function that is equal to one when $c(GB_j^i) \in EB_j^i$. Therefore, the measurement is obtained by

$$\text{HITR} = \sum_j I(c(GB_j^i), EB_j^i). \tag{5.3}$$

The fourth measurement is the hit team ratio (HITT). This is a relaxed version of the HITR, where we consider that the tracking is correct even if the bounding boxes for the two players of the same team are swapped. Therefore, this measurement is only used for evaluating tracking for the sports datasets. The reason for using this measurement is that the appearance of players of the same team are very similar, and even for a human observer it is not trivial to identify each player correctly after situations of occlusion or camera cut. More formally, this measurement is computed by the following procedure. Consider the situation for the pair of players 1 and 2 of the same team. First we find the correspondence with the highest intersection: $j^*, k^* = \arg\max_{j,k=1,2}(GB_j^i \cap EB_k^i)$. If $j^* \neq k^*$ then we swap EB_1^i and EB_2^i. The same procedure is applied for the other team. After correcting the bounding boxes, HITT is computed the same way as HITR.

The fifth one is the object intersection ratio (OBJI) which measures the amount of intersection between the bounding boxes of the groundtruth and the result for the same object. More formally,

$$\text{OBJI} = \frac{\sum_j |GB_j^i \cap EB_j^i|}{\max\{|GB_j^i|, |EB_j^i|\}}. \tag{5.4}$$

The final measurement is the global intersection ratio (GINT) which measures the amount of overlapping between all the bounding boxes from the groundtruth with all from the estimation. For this, if we note $\mathcal{GB}^i = \cup_j GB_j^i$ and $\mathcal{EB}^i = \cup_j EB_j^i$, we have

$$\text{GINT} = |\mathcal{GB}^i \cap \mathcal{EB}^i|. \tag{5.5}$$

5.1.2 Datasets

The three datasets used for testing the tracking framework are described below.

Synthetic

This dataset is composed of three synthetic videos created for this work, each one composed of 300 frames. All of them have a similar structure, which consists of nine solid squares with the same color, as shown in Figure 5.1. The challenge is to track the five inner squares while ignoring the four diagonal ones near the corners. In all videos, the central square and the four diagonal ones are fixed, while the other four change according to each video.

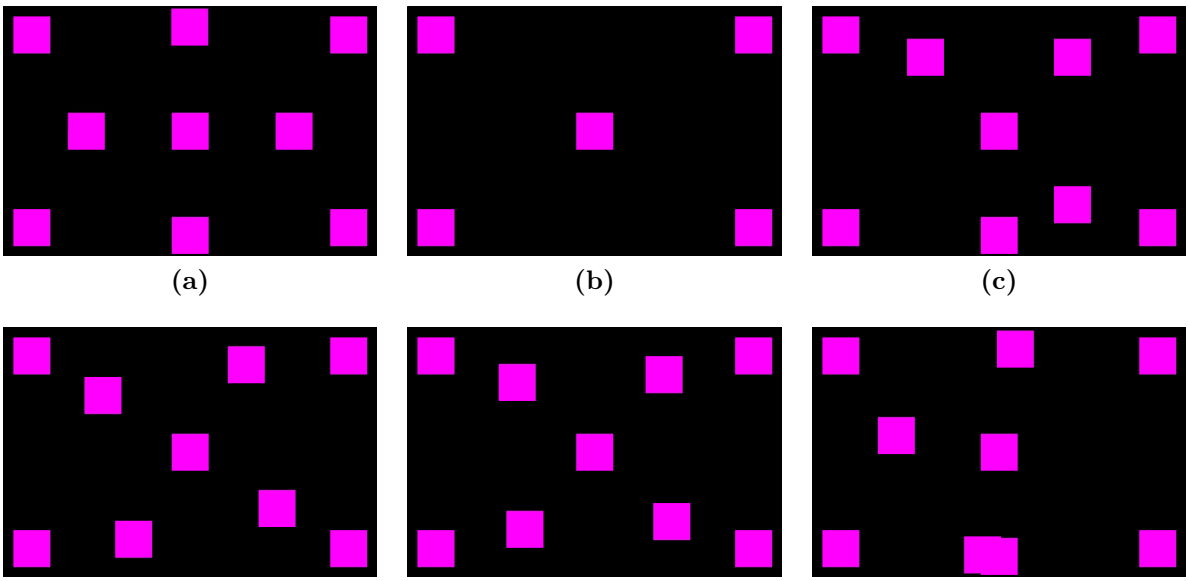


Figure 5.1: *Sample frames from the synthetic dataset. In the ROUND video (column (a)), the four square just move around the central one. In the BLINK video (b), the squares disappear momentarily, and then reappear as before. The OBSTACLE video (c) contains one fixed square at the bottom.*

In the first video (ROUND), the four squares simply turn around the central one in a clockwise orientation with constant velocity. This video is used to test the stability of the method. The second one (BLINK) still shows the same situation as before. However, after 100 frames, the four moving squares disappear for another 100 frames and then reappear. This video is used to test the behavior of the method when abrupt motion happens. The third video (OBSTACLE) shows three of the same squares moving as before, but with the bottom square kept fixed. In this video we want to test the behavior of the method after overlapping of objects with similar appearance.

Youtube table tennis

This dataset is composed of 6 videos containing 6737 frames in total. All the videos are of doubles matches of competitive table tennis collected from Youtube. Figure 5.2 shows some sample frames from this dataset. The videos were edited to remove unrelated scenes (e.g. preparation stage, crowd) and then manually annotated with bounding boxes for training

and groundtruth. The videos are encoded at resolutions varying from 640×360 to 854×480 at 30FPS.

Figure 5.2: *Sample frames from the Youtube table tennis dataset.*

ACASVA

We selected three videos from the ACASVA (De Campos *et al.* (2011)) dataset of badminton doubles matches from the Olympic games in London 2012. As in the table tennis dataset, the videos were edited to remove parts that do not show the game itself and annotations were created manually to be used as groundtruth. The resulting videos were encoded at 854×480 at 30FPS and contained 5766 frames. Figure 5.3 displays some sample frames from this dataset.

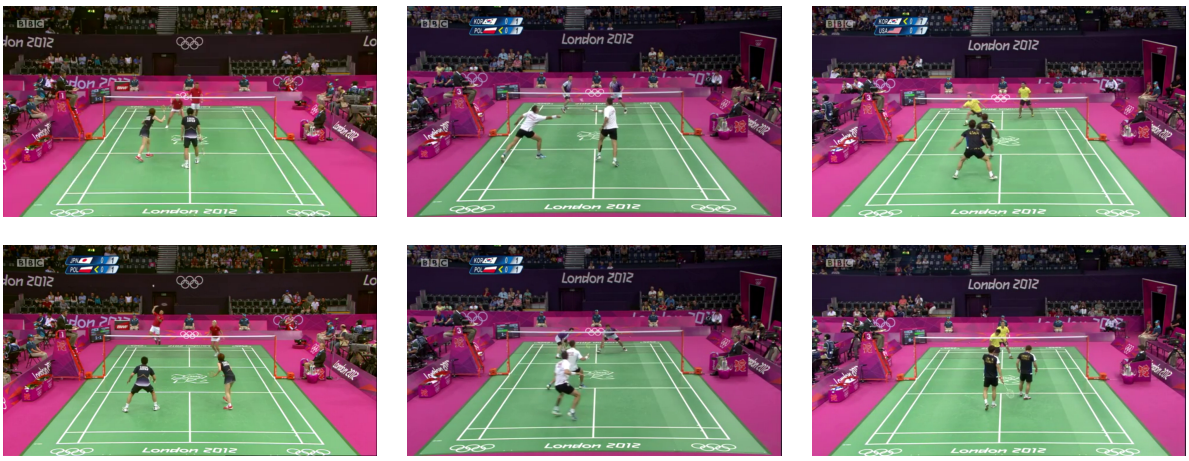


Figure 5.3: *Sample frames from the ACASVA badminton dataset.*

5.1.3 Choosing the parameters

We separated the Youtube videos into two sets, one for parameter estimation and another for evaluation. The set for parameter estimation was composed of one video containing 2461 frames. This video contained a longer table tennis match containing all the expected challenging situation including overlapping between players of the same team and camera cuts. The model graph was learned using all the other 5 videos.

The graph parameters were chosen by running tests on the estimation video. We chose to estimate three parameters: the feature weight ρ_F, the old weight factor ρ_T and the threshold for removing trackers τ_S. The parameters were estimated by keeping all of them fixed, except one. The fixed parameters were empirically chosen beforehand and their values are shown in Table 5.1. We chose to evaluate the results using the CERR measure, which is one of the most commonly adopted in tracking works. For reference, we ran the raw particle filter tracking on this video and the observed error was CERR = 138.

Table 5.1: *Initial fixed parameters used for estimating the new ones.*

feature weight	$\rho_F = 0.5$
old temporal weight factor	$\rho_T = 0.75$
score threshold for removing trackers	$\tau_S = 0.25$

The first parameter tested was the feature weight ρ_F. As can be seen in Figure 5.4, if the color model is totally ignored, the performance is very poor. However, the best results are obtained when the feature weight is kept low, showing that the structure is very important in this kind of video. Following the observed results, we chose to work with $\rho_F = 0.4$.

The analysis of the old weight factor ρ_T in Figure 5.5, shows that changing it does not significantly affect the results, except when its value is high. Indeed, when a high factor is used, the situation is the same as tracking with a single tracker, since no candidate will be able to have a higher score than older trackers. Thus, the observed result is similar to using raw particle filters. However, by analyzing the center displacement measure, it is clear that higher values increase the stability of the results. In that sense, this parameter is set as $\rho_T = 0.75$, which keeps the error low while providing good stability.

Finally, from Figure 5.6 we observe that by using lower threshold values τ_S for removing trackers, the results are better. This is caused because higher values may end up removing correct trackers in some cases, up to the point where only one tracker is kept, which is also similar to the raw particle filter approach. However, it is also important to notice that, by keeping more trackers, the running time is also impacted. As shown in the same Figure 5.6, when using no threshold, the performance is more than two times slower than when using a high value. Therefore, a good choice of parameter must take into consideration both the accuracy and running time. Hence, the value chosen for this parameter was $\tau_S = 0.25$.

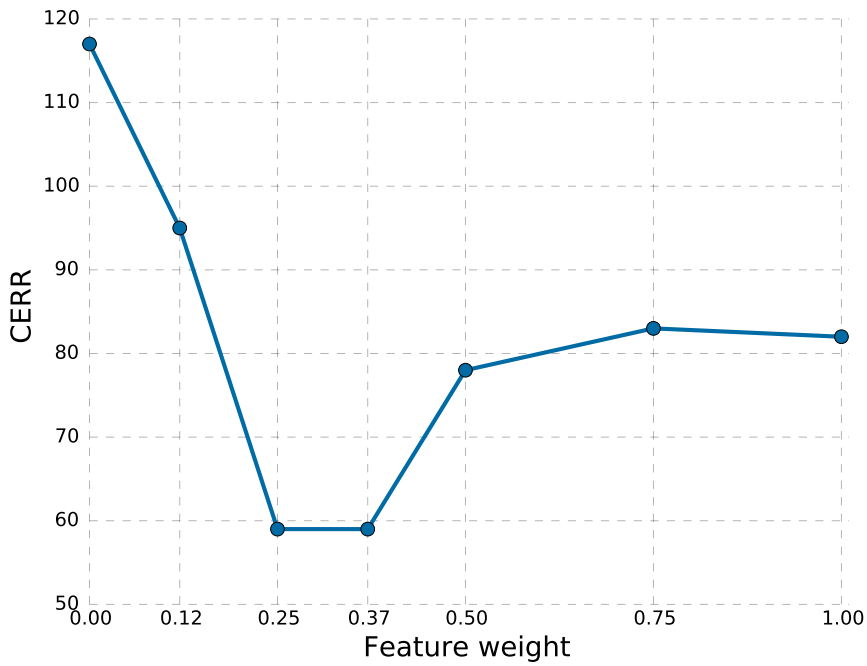


Figure 5.4: *Center error according to feature weight ρ_F.*

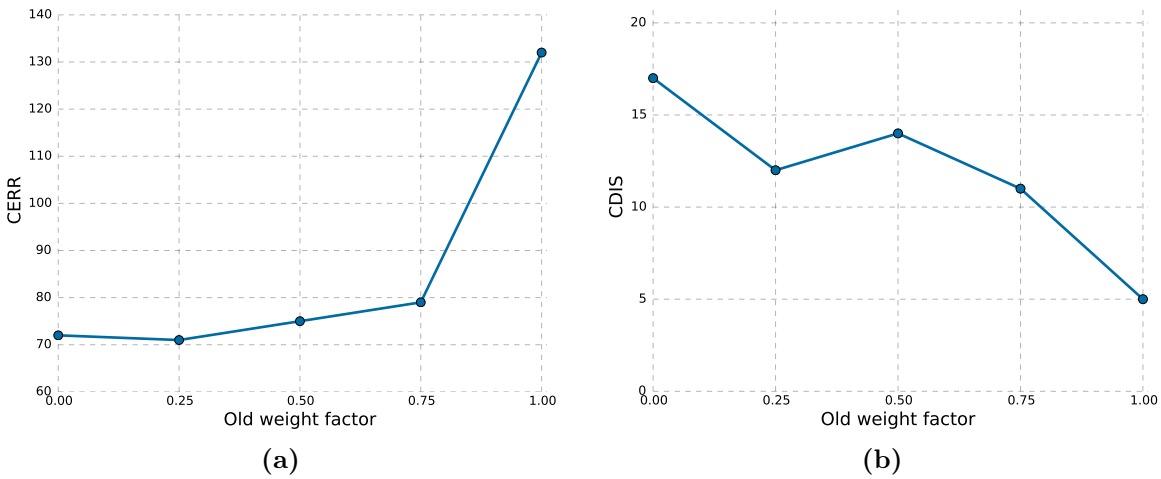


Figure 5.5: *(a) Center error according to old weight factor ρ_T. (b) Variation in center stability caused by changing the parameter.*

Table 5.2 summarizes all the parameters chosen for evaluation. The same values were used for all the experiments, independently of the dataset.

5.1.4 Results on the synthetic dataset

We tracked the five objects in all three videos using both standard particle filters and particles coupled with our structural approach. The chosen adjacency matrix M_A was the complete graph matrix, i.e. every entry is equals to one, except the main diagonal which is zero. The candidate matrix M_C was chosen to use the central square as a reference to

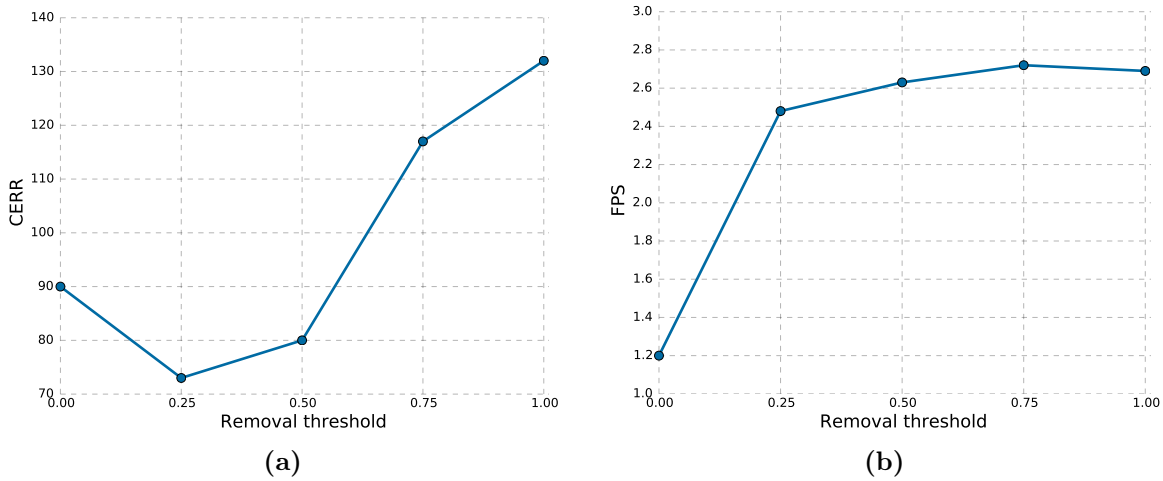


Figure 5.6: *(a) Center error according to threshold for removing trackers τ_S. (b) Variation in performance (frames per second) caused by changing the parameter.*

Table 5.2: *Parameters for the tracking framework.*

number of particles per object	$N_p = 50$
initial particle spread deviation	$\sigma_c = 10$
overlapping distance between same object	$\tau_{d_S} = 50$
overlapping distance between different objects	$\tau_{d_D} = 25$
feature weight	$\rho_F = 0.4$
old temporal weight factor	$\rho_T = 0.75$
score threshold for removing trackers	$\tau_S = 0.25$
threshold for changing tracker	$\tau_T = 1.25$
threshold for changing after overlapping	$\tau_O = 0.75$

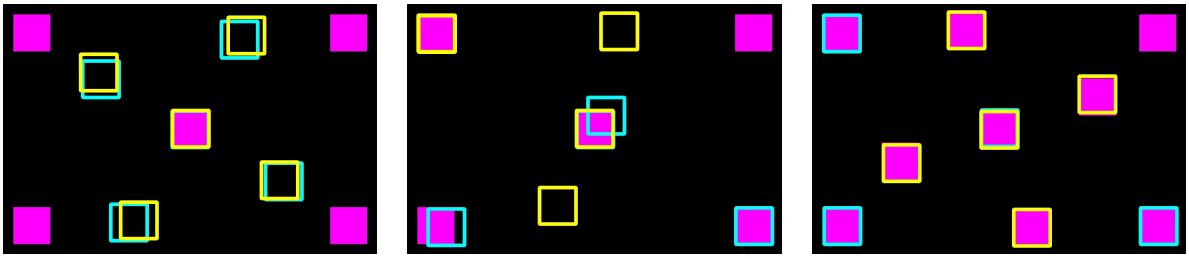
generate 10 candidates for each of the four moving objects. Since the objective of this test was only to check the behavior of the method, the model graphs were trained using the video itself as training. Since usually both structure and appearance are ambiguous, we chose to only evaluate the global intersection rate (GINT) results, which are shown in Table 5.3. Each test was run five times, and the results presented are the averages of all the runs.

Table 5.3: *Observed results for the GINT measurement on the synthetic videos.*

	PF	PF + Graph
ROUND	**0.96**	**0.96**
OBSTACLE	0.52	**0.90**
BLINK $\rho_F = 0.5$	0.71	**0.93**
BLINK $\rho_F = 1.0$	0.71	**0.82**

As can be seen from the results, the use of structure clearly improves the overall results, achieving an intersection ratio of more than 0.9 for all the videos. As is also shown in the table, for the BLINK video, ignoring the structural information and computing the score based only on the appearance (feature weight $\rho_F = 1.0$) clearly affects the results negatively.

This situation is shown in Figure 5.7. When the tracked objects disappear, the trackers tend to choose one of the other squares as the target. However, even after the desired targets reappear, without the structural information the trackers that chose another square cannot decide that the correct targets are better than the current wrong ones. Therefore, most of them remain stationary. However, the results are still better than when using the standard particle filters, because of the restriction that overlapping trackers should be avoided.



(a) **(b)** **(c)**

Figure 5.7: *Example of tracking on the BLINK synthetic videos using graphs with feature weights of 0.5 (yellow bounding boxes) and 1.0 (cyan boxes). When the targets disappear, the trackers remain lost for a while (a), but then regroups on one of the other squares (b). However, without structural information, most of them cannot go back to the correct objects, even after they reappear (c).*

5.1.5 Results on the table tennis dataset

We tested our approach on videos of table tennis doubles matches obtained from Youtube. These videos are interesting because they present challenging real world conditions like appearance change, occlusion and camera cuts. They also present some structure enforced by the rules of the game, which is captured by the graph model.

The task in these videos was to track all the four players and the table. We purposely track using only the torso of the players in order to create more appearance ambiguity and check whether the graph model can deal with this situation. As before, all the tests were performed five times and the average of all of them was taken. The model graph was learned using a leave-one-video-out approach. The candidates matrix is the same as in the synthetic tests, while the chosen adjacency matrix was:

$$M_A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

This matrix considers the relations between players of the same team and all the players and the table. The relationship with the table is important because the players should be close

and on opposite sides of it during the game. On the other hand, exploring the relationship between players of the same team helps to deal with temporary occlusions.

Table 5.4: *Observed results on the table tennis videos.*

Method	CERR	OBJI	HITR	HITT
PF	89	0.65	0.62	0.70
PF + Graph	**49**	**0.69**	**0.70**	**0.78**

The results are presented in Table 5.4. The values correspond to the average of the results obtained from all five videos weighted by their respective number of frames. As pointed by all the measurements, the use of graphs clearly improves the tracking results. Figure 5.8 shows some situations where the graphs are useful. As it can be seen from the first six pictures, when two players overlap, the standard particle filter tracking loses one of them due to the similar appearance. However, the use of the graphs allows our method to recover the track successfully. Camera cuts, causing abrupt motion, can also greatly affect the results if the targets are not found again. But as shown by the last two pictures, the proposed approach is also able to continue tracking even after such events. In Figure 5.9 a plot of the observed center errors and hit team ratios for one challenging video including overlapping and camera cuts is presented. It is clear from the chart that the use of the graphs consistently generates better results along the whole video.

5.1.6 Results on the ACASVA dataset

The tests on this dataset followed the same procedure as in the table tennis videos. The observed quantitative results are presented in Table 5.5.

Table 5.5: *Observed results on the ACASVA badminton videos.*

Method	CERR	OBJI	HITR	HITT
PF	58	**0.67**	0.56	0.65
PF + Graph	**50**	**0.67**	**0.57**	**0.77**

The first three measurements do not point out to any significant improvement, but that is related to the problem briefly mentioned in Section 5.1.1, when introducing the HITT measurement. These measurements only consider that a tracking result is correct if there is a direct association with the groundtruth. Although that assumption is usually reasonable, in team sports videos like these ones, it is often difficult to correctly identify different players from the same team only by the appearance. In situations of ambiguous appearances, multiple particle filter trackers tend to cluster in one of the targets, while losing the other ones. However, even if that happens, the measurements for at least one object will be correct. Our approach tries to separate overlapping trackers, but if the appearance is ambiguous,

Figure 5.8: *Tracking results for the table tennis videos. The green bounding box represents the results using standard particle filters, while the purple ones are the results of the proposed method. The results show that our method is able to recover tracking after occlusion between the objects and also abrupt motion caused by camera cut.*

sometimes the trackers are swapped between the objects. If only two objects have similar appearances and they are swapped, the previous measurements will consider the results as a total mistake.

For this reason, we also evaluate the results using the HITT measurement. As it can be seen, the HITT does present a significant improvement over regular particle filters, indicating

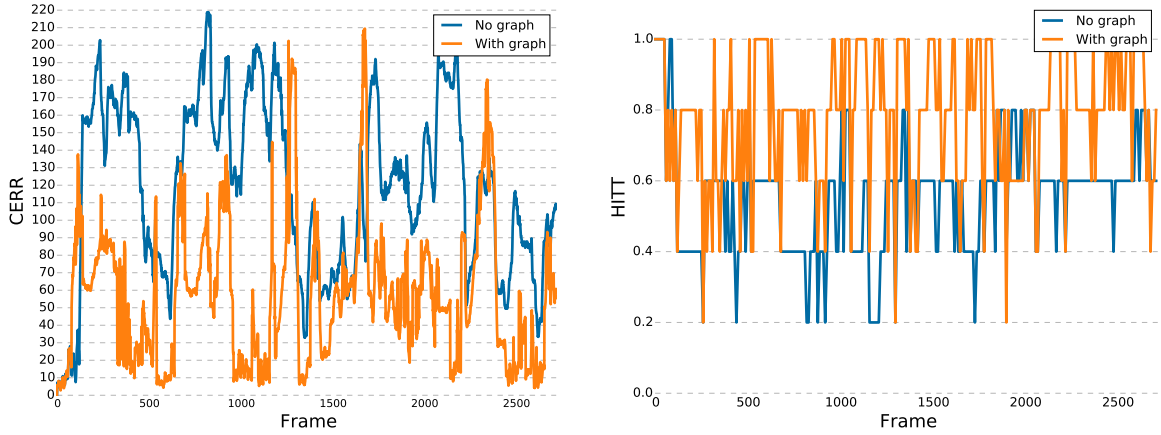


Figure 5.9: *Center error and hit team ratio for one table tennis video including object overlapping and camera cuts. The HITT points were sampled periodically at every 10th observation for more clarity.*

that our method does discriminate more the targets. Besides, even the other measurements point out that the proposed approach is never worse than regular particle filters. Figure 5.10 shows some qualitative results on the testing videos. The results show that our approach successfully recovers tracking after it is lost due to occlusion or abrupt motion in this dataset as well. The CERR and HITT charts for one representative video can be seen in Figure 5.11.

5.2 Action recognition

The action recognition module was inspired by the work of Lu and Little (2006) and was implemented using PCA-HOG descriptors classified by HMMs. The HOG descriptors are extracted using OpenCV, while the PCA and HMM are implemented by the sklearn library[2].

The HOG descriptors are extracted from the bounding boxes of the objects obtained from the tracking module. However, different from Lu and Little (2006), HOGs are extracted as proposed by the original authors (Dalal and Triggs (2005)). The main reason for this choice is due to the lack of customization of the OpenCV HOG module. However, as the observed results using this approach proved to be reasonable, we decided to keep the current implementation. In fact, this should not have a strong impact on the results, as the descriptor proposed by Lu and Little (2006) is just a simpler version of the original.

[2]http://scikit-learn.org/stable/

Figure 5.10: *Tracking results for the ACASVA videos. The white bounding box represents the results using standard particle filters, while the blue ones are the results of the proposed method. The left column shows a situation of camera cut, while the right one shows temporary occlusion.*

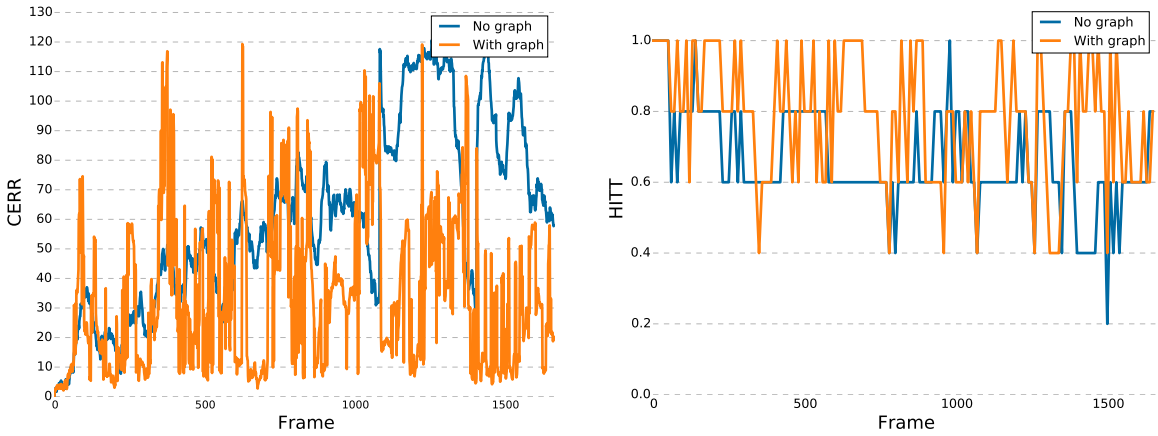


Figure 5.11: *Center error and hit team ratio for one video of the ACASVA dataset. The HITT points were sampled periodically at every 10th observation for more clarity.*

5.2.1 Dataset

The action recognition module was tested using videos of table tennis matches obtained from Youtube. As opposed to the ones using for testing the tracking module, the videos in this dataset were obtained from singles matches (one player at each side of the table). The main reason for this choice was to obtain a clearer view of each player as they performed their actions, decreasing the possibility of occlusion. The dataset was composed of 15 videos containing 9427 frames. All the frames were manually annotated with bounding boxes for each player as well as action labels. Figure 5.12 shows a few example images collected from the dataset used in this test.

Figure 5.12: *Sample frames from the dataset used for action recognition.*

Based on the work of Wang *et al.* (2013), we decided to recognize four actions performed by each player: *serve, backhand, forehand* and *others*, where the last one is merely a wildcard that represents anything else. As the background is very different for each player (e.g. the player in the back is partially occluded by the table), a different set of HMMs was trained for each one, yielding a total of 8 HMMs.

The videos were separated into two sets, one for estimating the parameters of the descriptors and classifiers and another for evaluation. The estimation set was composed of 11 videos with 5672 frames, while the remaining 4 videos with 3755 frames were used for testing. The sets were divided in this way because the videos in the first set were shorter or with missing annotations. Therefore, we preferred to conduct the evaluation tests using the more complete videos.

5.2.2 Parameters estimation

The parameters were estimated by following a 5-fold cross-validation approach on the estimation set and computing the average hit ratio for each configuration. The experiments involved tests for finding the best number of components from PCA, the length of one action sequence and the number of states of the HMMs.

Figure 5.13 shows the performance results for different number of principal components (PCs). The values correspond to the average between the two players. As it can be seen, the hit ratio reaches the top between 25 and 150 PCs. This result shows that, although a higher number of PCs encode more of the original information, it also hinders classification, as the search space increases exponentially. Due to the higher accuracy and considering the computer complexity, we chose to work with 25 PCs.

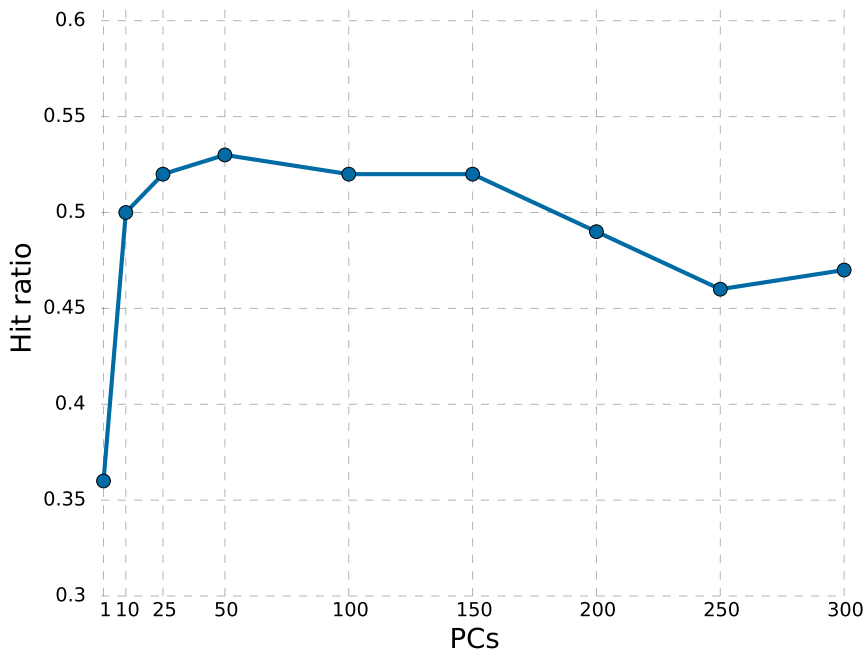


Figure 5.13: *Variation of classification performance according to the numbers of PCs.*

The sequence length and number of HMM states were tested simultaneously in order to find the best combination. The results are shown in Table 5.6. Results in the upper right part of the matrix are not shown due to a limitation in sklearn library that does not allow to train models with more states than the number of available observations. The table shows that changing these parameters does not affect significantly the performance. We have also tried to use different values for each object, but the observed results did not show any expressive changes. Since the results are very similar, we chose to work with the lowest parameters for both variables. The final parameters chosen for the evaluation task are summarized in Table 5.7.

Table 5.6: *Classification hit ratio according to the sequence length and the number of hidden states.*

seq.length / hmm states	5	10	15	20	25
5	0.54	-	-	-	-
10	0.52	0.51	-	-	-
15	0.55	0.56	0.43	-	-
20	0.56	0.48	0.49	0.50	-
25	0.54	0.53	0.54	0.49	0.42

Table 5.7: *Final parameters for the action recognition.*

number of principal components	25
sequence length	5
number of HMM states	5

5.2.3 Evaluation

For this task, the bounding boxes obtained from tracking were collected beforehand and stored. Afterwards, they were segmented into individual action sequences according to the groundtruth. Therefore, the evaluation set was represented by several sequences, where each one should correspond to a complete action. The tests were then conducted in the same fashion as in the estimation phase. They were evaluated in a 5-fold cross-validation approach where 80% of the sequences were used for training using the groundtruth data, while the remaining 20% sequences were provided by the tracking results.

We shall refer to as player 1, the player who is closer to the camera and thus, not occluded by the table, while player 2 is the other one. The results obtained for both players are shown in Table 5.8. As it is apparent, the results for player 2 are slightly worse than for player 1. This can be explained mainly due to the fact that this player may or may not be partially occluded by the table. Therefore, the shape descriptor may be different depending on the position of this player, which causes some classification issues. More detailed results can be seen in the confusion matrix presented in Table 5.9. Figure 5.14 shows some examples of action sequences that were classified.

Table 5.8: *Observed hit ratio results for the action recognition task.*

	Mean	St. deviation
Player 1	0.55	0.05
Player 2	0.47	0.05

The results show that the classification is overall good for player 1, while the player 2 presents more errors. The backhand action for player 1 presents the worse results, but this is not surprising. The reason is that, since the player has his back to the camera, this action is difficult to be perceived. The same can be said for player 2, on the opposite sense, where this action performs better, since his arms and hands are visible during this motion. However, it is clear that the classification for this player suffers for the reason explained before.

Table 5.9: *Confusion matrix of the action classification.*

		Others	Forehand	Backhand	Serve
Player 1	Others	**0.72**	0.12	0.16	0
	Forehand	0.28	**0.52**	0.20	0
	Backhand	0.44	0.16	**0.40**	0
	Serve	0.10	0.20	0.10	**0.60**
		Others	Forehand	Backhand	Serve
Player 2	Others	**0.68**	0.12	0.20	0
	Forehand	0.48	**0.20**	0.32	0
	Backhand	0.40	0.08	**0.52**	0
	Serve	0	0.20	0.40	**0.40**

Figure 5.14: *Examples of action sequences, the first five columns represent the action for player 1, while the others for player 2. Each row correspond to one action: (1) others, (2) forehand, (3) backhand and (4) serve.*

These results are further validated by comparing them with those presented by Wang *et al.* (2013). The confusion matrix obtained by the authors for the badminton dataset with a similar set of actions is presented in Table 5.10. Although the results cannot be directly compared due to a different dataset and methodology, they provide some support that the observed results for our tests are reasonable.

Table 5.10: *Results reported in Wang et al. (2013) for the badminton dataset.*

	Others	Forehand	Backhand	Smash
Others	**0.61**	0.19	0.02	0.18
Forehand	0.1	**0.52**	0.14	0.24
Backhand	0	0.27	**0.66**	0.07
Smash	0.08	0.07	0.07	**0.78**

Chapter 6

Conclusions

6.1 Contributions and discussion

We proposed a graph based approach to explore the structural information of a filmed scene and use it to improve tracking of multiple objects in structured videos. Each object in the scene represents one vertex of the graph, and edges are included to consider their spatial relations. Before the actual tracking, a probabilistic graph model is trained to learn the structure of the scene. The graph is then used for two purposes: (1) evaluate the current tracking state to check if the multiple objects are being correctly tracked and (2) generate new likely target locations to try to recover tracking in case it is lost. During the tracking, each object is individually tracked using particle filters. By merging the current tracking with the candidates generated by the model, multiple graphs are built. They are then evaluated according to the model, and the best one is chosen as the new global tracking state.

One of the advantages of the proposed framework is that, although during this thesis it was built over particle filters, it does not really rely on any information specific from that model. Therefore, the single object tracker could be potentially replaced by any other more suitable choice for other types of objects. That makes this approach very flexible and able to deal with a wider range of applications.

We also used the tracking results to recognize the actions of the players in table tennis matches. This step was done by extracting PCA-HOG features from the image and then classifying them using HMMs. One different HMM was trained for each of the chosen actions (serve, backhand, forehand, others) using annotated data.

The results show that the proposed method successfully increases the tracking precision over regular particle filters. As shown by the results this is mainly due to the fact that the use of the graphs allows us to recover tracking after challenging conditions such as temporary occlusion and camera cuts. The action recognition also showed convincing results, indicating it may be a viable option for further studying.

6.2 Limitations and Future work

One limitation of the proposed framework is that the color-based tracker used for each object is not very robust against appearance or illumination changes. It is also sensible to initialization parameters, i.e. tracking may present poor results if the provided bounding box does not cover the object properly. As the graphs also use the tracker score for evaluating, if the color model is not representative enough, the whole tracking may be affected. This effect can be seen in the resulting videos, where sometimes objects with similar appearance change trackers, which might not happen if the proposed approach of generating new candidates is not used. However, as evidenced by the results, the overall result for all videos is better than when using the classical particle filters.

There are two main paths that we are interested in pursuing in the future to improve this method. The first one consists in making the method more self adaptive. One way to do so is to automatically adjust the number of candidates generated from each reference object, or to use the global structure as a whole to choose the best locations. This could be done by computing a reliability score for each object, combining the hypothesis of all of references into a single set and choosing only the best options.

Another option is, when working with particle filters, to automatically adjust the amount of particles generated for each candidate. This could improve the performance by generating fewer particles in locations that are not very likely to be correct. One direct approach for this is to use the graph score as a base for computing the number of particles. However, this method puts a lot of importance on the structure, and thus it may provide bad results if the trained model is not rich enough. One alternative would be to generate just small clouds of particles at each location and increase them in case the initial evaluation seems good.

Finally, the tracking can be extended by removing the need of training a model beforehand. An interesting research direction would be to adapt the method to train an adaptive model that could be improved as the tracking is happening.

The other path for improving this work concerns the action recognition module. In fact, we originally intended to use the action recognition results to feed the tracking module as well, creating a joint framework where both modules would contribute to the other. But, due to time constraints, this could not be implemented within this thesis. Nonetheless, this remains as a very interesting path to investigate in the future.

The main motivation for this approach is that one of the challenges of tracking is to correctly estimate the size of the bounding box surrounding the object. It is usually assumed that the proportions of the object do not change, but as evidenced by the players in the table tennis matches, it is easy to see that this assumption is not always reasonable. By knowing the action of a player, it would be possible to know whether the width of the bounding box needs to be increased (e.g. if the player spreads his arm) or reduced, for example.

6.3 Scientific production

The work developed during this thesis led the Ph.D. candidate to publish two papers in conferences:

- Henrique Morimitsu, Roberto M Cesar Jr and Isabelle Bloch. A spatio-temporal approach for multiple object detection in videos using graphs and probability maps. In Image Analysis and Recognition, pages 421-428. Springer. 2014.
- Henrique Morimitsu, Roberto M Cesar Jr and Isabelle Bloch. A graph-based approach for object detection and action recognition in videos. In FEAST Workshop of International Conference on Pattern Recognition. IAPR. 2014.

Another paper is submitted to a conference, but the result is still not available.

Bibliography

Aggarwal and Ryoo (2011) Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16. Cited on page 8

Blank *et al.* **(2005)** Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani and Ronen Basri. Actions as space-time shapes. In *IEEE International Conference on Computer Vision*, volume 2, pages 1395–1402. Cited on page 9

Bussab and Morettin (2010) Wilton de O Bussab and Pedro A Morettin. *Estatística básica*. Saraiva. Cited on page 13

Callioli *et al.* **(2007)** Carlos Alberto Callioli, Hygino Hugueros Domingues and Roberto Celso Fabrício Costa. *Álgebra linear e aplicações*. Atual. Cited on page 32

Cho *et al.* **(2013)** Minsu Cho, Karteek Alahari and Jean Ponce. Learning graphs to match. In *IEEE International Conference on Computer Vision*, pages 25–32. Cited on page 18

Dalal and Triggs (2005) Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. Cited on page xiii, 8, 19, 30, 55

De Campos *et al.* **(2011)** Teofilo De Campos, Mark Barnard, Krystian Mikolajczyk, Josef Kittler, Fei Yan, William Christmas and David Windridge. An evaluation of bags-of-words and spatio-temporal shapes for action recognition. In *Workshop on Applications of Computer Vision*, pages 344–351. Cited on page 3, 48

Duda *et al.* **(2001)** Richard O Duda, Peter E Hart and David G Stork. *Pattern classification*. John Wiley & Sons. Cited on page 32

Erdem *et al.* **(2012)** Erkut Erdem, Séverine Dubuisson and Isabelle Bloch. Fragments based tracking with adaptive cue integration. *Computer Vision and Image Understanding*, 116 (7):827–841. Cited on page 22

Farnebäck (2003) Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer. Cited on page 9

Felzenszwalb *et al.* **(2010)** Pedro F Felzenszwalb, Ross B Girshick, David McAllester and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645. Cited on page 6, 8

Fink (2008) Gernot A. Fink. *Markov models for patter recognition: from theory to applications*. Springer. Cited on page 32, 38

Grabner *et al.* **(2010)** Helmut Grabner, Jiri Matas, Luc Van Gool and Philippe Cattin. Tracking the invisible: Learning where the object might be. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1285–1292. Cited on page 6

Harris and Stephens (1988) Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 50. Cited on page 6, 9

Hu *et al.* **(2015)** Weiming Hu, Wei Li, Xiaoqin Zhang and Stephen Maybank. Single and multiple object tracking using a multi-feature joint sparse representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 37(4):816–833. Cited on page 6

Isard and Blake (1998) Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International journal of computer vision*, 29(1): 5–28. Cited on page 15

Laptev and Pérez (2007) Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *IEEE International Conference on Computer Vision*, pages 1–8. Cited on page 9

Lowe (2004) David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. Cited on page 19

Lu and Little (2006) Wei-Lwun Lu and James J Little. Simultaneous tracking and action recognition using the PCA-HOG descriptor. In *Canadian Conference on Computer and Robot Vision*, pages 6–6. Cited on page 9, 55

Milan *et al.* **(2014)** Anton Milan, Stefan Roth and Kaspar Schindler. Continuous energy minimization for multitarget tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 36(1):58–72. Cited on page 7

Okuma *et al.* **(2004)** Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*, pages 28–39. Springer. Cited on page 5

Pérez *et al.* **(2002)** Patrick Pérez, Carine Hue, Jaco Vermaak and Michel Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, pages 661–675. Springer. Cited on page 17, 45

Reid (1979) Donald B Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854. Cited on page 5

Schüldt *et al.* **(2004)** Christian Schüldt, Ivan Laptev and Barbara Caputo. Recognizing human actions: a local SVM approach. In *International Conference on Pattern Recognition*, volume 3, pages 32–36. Cited on page 9

Shlens (2014) Jonathon Shlens. A tutorial on Principal Component Analysis. *arXiv preprint arXiv:1404.1100*. Cited on page 32

Su *et al.* **(2014)** Yingya Su, Qingjie Zhao, Liujun Zhao and Dongbing Gu. Abrupt motion tracking using a visual saliency embedded particle filter. *Pattern Recognition*, 47(5):1826–1834. Cited on page 46

Tang *et al.* **(2014)** Siyu Tang, Mykhaylo Andriluka and Bernt Schiele. Detection and tracking of occluded people. *International Journal of Computer Vision*, 110(1):58–69. Cited on page 6

Viola and Jones (2004) Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154. Cited on page 9

Wang and Schmid (2013) Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, pages 3551–3558. Cited on page 9

Wang *et al.* **(2013)** Zhenhua Wang, Qinfeng Shi, Chunhua Shen and Anton van den Hengel. Bilinear programming for human activity recognition with unknown MRF graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1690–1697. Cited on page xv, 57, 60

Wen *et al.* **(2014)** Longyin Wen, Wenbo Li, Junjie Yan, Zhen Lei, Dong Yi and Stan Z Li. Multiple target tracking based on undirected hierarchical relation hypergraph. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289. Cited on page 7

Widynski *et al.* **(2012)** Nicolas Widynski, Séverine Dubuisson and Isabelle Bloch. Fuzzy spatial constraints and ranked partitioned sampling approach for multiple object tracking. *Computer Vision and Image Understanding*, 116(10):1076–1094. Cited on page 5

Yan *et al.* **(2014)** Xu Yan, Ioannis A Kakadiaris and Shishir K Shah. Modeling local behavior for predicting social interactions towards human tracking. *Pattern Recognition*, 47(4):1626–1641. Cited on page 7

Yang *et al.* **(2007)** Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *International Workshop on Multimedia Information Retrieval*, pages 197–206. Cited on page 9

Zhang and van der Maaten (2014) Lu Zhang and Laurens JP van der Maaten. Preserving structure in model-free tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 36(4):756–769. Cited on page 8

Zhang *et al.* **(2015)** Shun Zhang, Jinjun Wang, Zelun Wang, Yihong Gong and Yuehu Liu. Multi-target tracking by learning local-to-global trajectory models. *Pattern Recognition*, 48(2):580–590. Cited on page 7