Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Exploring structure for long-term tracking of multiple objects in sports videos

Henrique Morimitsu[a,*], Isabelle Bloch[b], Roberto M. Cesar-Jr[a]

[a] *Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil*
[b] *LTCI, CNRS, Télécom ParisTech, Université Paris – Saclay, Paris, France*

ARTICLE INFO

Article history:
Received 15 March 2016
Revised 18 September 2016
Accepted 5 December 2016
Available online 6 December 2016

Keywords:
Multi-object tracking
Structural information
Particle filter
Graph

ABSTRACT

In this paper, we propose a novel approach for exploiting structural relations to track multiple objects that may undergo long-term occlusion and abrupt motion. We use a model-free approach that relies only on annotations given in the first frame of the video to track all the objects online, i.e. without knowledge from future frames. We initialize a probabilistic Attributed Relational Graph (ARG) from the first frame, which is incrementally updated along the video. Instead of using the structural information only to evaluate the scene, the proposed approach considers it to generate new tracking hypotheses. In this way, our method is capable of generating relevant object candidates that are used to improve or recover the track of lost objects. The proposed method is evaluated on several videos of table tennis, volleyball, and on the ACASVA dataset. The results show that our approach is very robust, flexible and able to outperform other state-of-the-art methods in sports videos that present structural patterns.

1. Introduction

Object tracking is a relevant field with several important applications, including surveillance, autonomous navigation, and activity analysis. However, tracking several objects simultaneously is often a very challenging task. Most multi-object trackers first track each object separately by learning an individual appearance model and then consider all the results globally to improve or correct individual mistakes. However, especially in sports videos, the use of appearance models proves to be insufficient because usually many objects (or players) have very similar appearance due to the uniform they wear. This often causes tracking loss after situations of occlusion between players of the same team. Another difficulty is that most trackers rely on the constraint that temporal changes are smooth, i.e. the position of an object does not change significantly in a short period of time. Yet, this is not a reasonable assumption for most sports videos, because they are usually obtained from broadcast television, and thus they are edited and present several camera cuts, i.e. when the scene changes suddenly due to a camera cut off, or change of point of view. Camera cuts often cause problems of abrupt motion, which is regarded as a sudden change in position, speed or direction of the target.

In long-term tracking, the objects are subject to situations of full occlusion and abrupt motion, which may lead to tracking failures. Therefore, the tracker must be able to recover the target after such events. In this paper, we explore the use of spatial relations between objects to recover or correct online tracking. Online tracking, as opposed to batch methods, only uses past information to predict the next state. 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 recover tracking by learning some structural properties. Fig. 1 shows an example of a table tennis video illustrating 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 solve this kind of problem by exploiting the spatial properties of the scene, such as the distance and angle between two objects.

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. A good example is sports 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 exploited. 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.

Structural relations are utilized by using graphs to encode the spatial configuration of the scene. In this paper, color-based

* Corresponding author.
E-mail addresses: henriquem87@vision.ime.usp.br, henriquem87@gmail.com (H. Morimitsu), isabelle.bloch@telecom-paristech.fr (I. Bloch), mcesar@usp.br (R.M. Cesar-Jr).

http://dx.doi.org/10.1016/j.cviu.2016.12.003
1077-3142/© 2016 Elsevier Inc. All rights reserved.

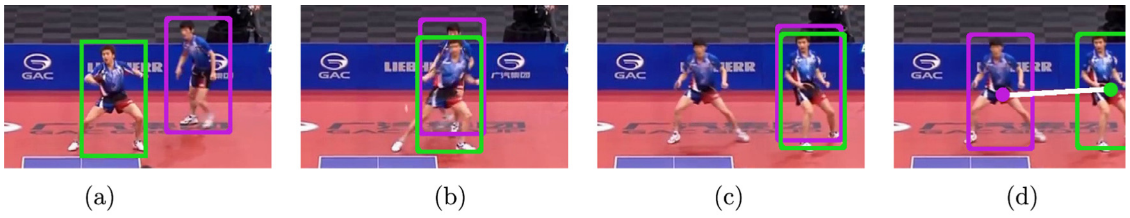

Fig. 1. An example of a multi-object tracking situation. Most single object trackers are able to successfully track the targets when their appearance is clear (a). However, when overlap occurs (b), they are not able to solve the ambiguity problem in appearance and the tracking is lost (c). We propose recovering tracking after such events by using structural spatial information encoded in graphs (d).

particle filter was chosen as the single object tracker due to its simplicity and good results demonstrated in previous studies. However, other trackers could also be employed 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 tracker applied in multi-objects problems. As shall be explained throughout this text, the graphs are utilized to extract structural information from the scene, to generate candidate object positions, and then to evaluate the tracking state at each frame. 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 has not been much explored in the literature before. Indeed, several of the current state-of-the-art methods based on tracking by detection do use structural information at a different level, 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 this sense, one main contribution of this paper is to introduce a new approach that exploits the learned structural model to guide the detection. This approach allows tracking the objects even after challenging events such as long-term occlusions or abrupt motion. The proposed framework is tested on sports videos obtained from Youtube and also from the ACASVA (De Campos et al., 2011) dataset, which present real challenging conditions such as changing of lighting, mutual occlusion, and camera cuts.

The main contributions of this paper are the following: (1) to introduce a structural approach for improving tracking by generating new object candidates, and (2) to formalize tracking as a flexible graph optimization problem that can be adapted to other situations where structural patterns exist.

This paper extends our previous work (Morimitsu et al., 2015). The main novelties of this study are: firstly, the graph model is trained and updated online, without the need of a training dataset, making the method much easier to use in different applications. Secondly, the graph matching function does not rely on heuristics for occlusion and change of tracker anymore. This information is incorporated directly into the scoring function, thus allowing us to formalize the data association as a graph optimization problem. Thirdly, additional results on more cluttered and challenging videos of volleyball matches are included. They are also analyzed more thoroughly using the widely adopted CLEAR-MOT metrics (Bernardin and Stiefelhagen, 2008).

This paper is organized as follows. In Section 2 we present a review of some relevant previous works and how they contributed to the development of our method. In Section 3 we detail the particle filter tracking approach and how it is applied in our problem. In Section 4 the complete framework for tracking multiple objects using graphs is explained. In Section 5 the experimental results obtained are exposed. We compare the obtained results with our approach with other state-of-the-art methods from the literature. In Section 6 we discuss the main conclusions of this work as well as suggestions for future research.

2. Related work

Visual tracking in sports has been receiving great attention over the years and it has been tackled in many different ways. Due to its simplicity and robustness to deal with more complex models, methods based on particle filters became popular (Kristan et al., 2009; Okuma et al., 2004; Xing et al., 2011). Another significant approach relies on the fact that often the background is static and, therefore, tracking may be performed by using background subtraction methods (Figueroa et al., 2006). Other authors explore the use of multiple cameras to obtain more reliable results (Morais et al., 2014).

Recently, the tracking-by-detection framework has become the standard method for tracking multiple objects (Choi, 2015; Milan et al., 2014; Solera et al., 2015; Zhang et al., 2015). This approach consists in obtaining noisy detections at each frame and then connecting them into longer temporal tracks. For this, a data association problem must be solved between all the candidates in order to create stable tracks for each object. The most important part is formulating the association function so that the problem can be solved efficiently, while creating good tracks. Liu et al. (2013) have designed an association function specific for tracking players in team matches. The tracks are associated by assuming a motion model that depends on the game context at the moment. By exploiting local and global properties of the scene, such as relative occupancy or whether one player is chasing another, the authors show that the method is able to successfully track the players during a basketball match. One important challenge in multi-object tracking consists in keeping the correct identities of each object. Shitrit et al. (2014) demonstrate on several sports videos that it is possible to keep the correct identity of the players by relying on appearance cues that are collected sparsely along time. This is done by modeling the problem as flows expressed by Directed Acyclic Graphs, which is solved using linear programming. Lu et al. (2013) show that it is possible to keep correct identities even when using a simple data association strategy. On the other hand, this approach makes use of a Conditional Random Field framework, which assumes both that the tracks for the whole video are available, and that external data is available to train the model parameters. Our proposed method may be interpreted as a tracking-by-detection framework, but instead of using an object detector to generate candidates, we rely on the structural properties encoded in the graph. The identification of each player is handled implicitly, by the graphs. Although this choice is not always optimal, it is efficient, and it only relies on data obtained from the past frames of the video itself.

One challenging condition frequently found in sports scenes is occlusion. Many previous works focused on modeling it explicitly to handle these difficult situations (Tang et al., 2014; Xiang et al., 2015). Zhang et al. (2013) tackle this issue in sports by using a structured sparse model for each person. This approach builds on the robustness of sparse models by assuming that the occlusion model is usually not sparse, but rather a structured connected area. This allows using better models which are able to ignore features from large occlusion areas, e.g. one player occluding another one. In a related topic, Soomro et al. (2015) propose using structural properties encoded in graphs to associate tracks when the videos of football games are cut, which causes occlusion of players as well as abrupt motion. This problem is formulated as a graph matching between different frames. In order to solve the ambiguity problem for the association, the authors use knowledge about the previously learned team formation. Therefore, the model requires some additional external information in order to successfully recover tracking.

The use of structural information for multi-object tracking has also been incorporated into the SPOT tracker (Zhang and van der Maaten, 2014). The authors 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 finding 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). 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 proposal differs from the one presented in this paper 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 more general probability density function. (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, inspired by Grabner et al. (2010), uses some objects as supports for the structural model to generate 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.

Another important issue that must be dealt with during tracking is abrupt motion. Perhaps, the simplest way to deal with it is by generating some additional target location hypotheses around the previous location to try to cover a broader region, as explored by Zhang et al. (2012). Another proposal is to solve the same problem using spatial position information for finding better candidates (Kwon and Lee, 2008; Zhou and Lu, 2010). This is done by dividing the image into several smaller regions and using the information obtained from the density of states of each one to find new locations. More recently, Su et al. (2014) proposed relying on visual saliency information to guide the tracking and restore the target. It is important to note that, although tracking-by-detection methods should be able to deal with abrupt motion, most of them do not behave well in this situation because their association function usually enforces the temporal stability constraint. As previously mentioned, we use a different approach that relies on the structural relations between the objects to find the most likely new locations of a lost target.

3. Tracking objects with particle filters

In this section, the standard method of tracking with particle filters is briefly summarized. A classical filtering problem operates over a Hidden Markov Model (Fink, 2008). Let $\mathcal{X}$ and $\mathcal{O}$ be the sets of states and observations, respectively. 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 and $\boldsymbol{x}_t$ is conditionally independent of the joint of previous states and observations. The filtering problem consists in estimating recursively the posterior distribution $\mathbb{P}(\boldsymbol{x}_t|\boldsymbol{o}_{1:t})$, where $\boldsymbol{o}_{1:t}$ denotes the set of observations from instant 1 to instant t.

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

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

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

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

This work employs particle filter using the ConDensation algorithm, which uses factored sampling (Isard and Blake, 1998) to update the particles. 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 then be expressed as:

$$\boldsymbol{x}_t^i = \boldsymbol{D}\boldsymbol{x}_{t-1}^i + \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{D}$ is the motion dynamics and u is the noise vector.

Finally, the weights of the particles are updated according to the new observations $\boldsymbol{o}_t$ and, if the proposal function is the dynamics model, the weight update is simply:

$$\pi_t^i \propto \pi_{t-1}^i \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i). \tag{3}$$

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

$$r(P) = \boldsymbol{x}_t = \sum_{i=1}^{N_P} \pi_t^i \boldsymbol{x}_t^i \tag{4}$$

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

$$\zeta(P) = 1 - \exp\left(-\sum_{i=1}^{N_P} \pi_{t-1}^i \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i)\right). \tag{5}$$

This increasing function ensures that the score is close to 1 when the sum of the weights is high.

3.1. State and dynamics models

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, yielding:

$$\boldsymbol{x}_t^i = \boldsymbol{x}_{t-1}^i + \boldsymbol{u} \tag{6}$$

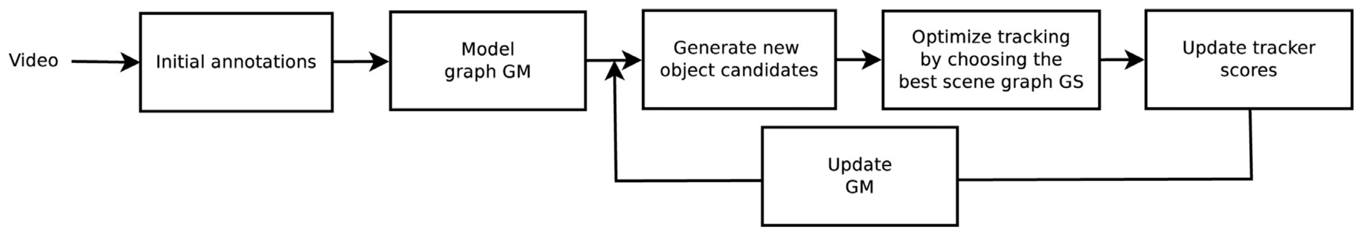
Video
Initial annotations
Model graph GM
Generate new object candidates
Optimize tracking by choosing the best scene graph GS
Update tracker scores
Update GM

Fig. 2. An overview of the proposed framework.

where $\boldsymbol{u} = (u_x, u_y)^T$ is a noise vector whose terms follow a Gaussian distribution $\mathcal{N}(0, \hat{\sigma_u})$. In this work, we adopt an adaptive variance $\hat{\sigma_u}$ that is computed by weighting the fixed variance σ_u according to the quality of the particles. The weight is obtained by:

$$\lambda = \alpha\left(1 - \frac{\sum_{i=1}^{N_P} \mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i)}{\beta}\right), \tag{7}$$

where α is a given spreading factor that controls the impact of the weight on the real variance and β is an upper bound for the maximum weight of the particles. We can choose $\beta = N_P$ or another suitable value according to the data. However, it is not interesting to remove the variance altogether, thus we set a lower bound τ_λ, yielding the final weight:

$$\hat{\lambda} = \max\{\lambda, \tau_\lambda\}. \tag{8}$$

Finally, the adaptive variance is computed as:

$$\hat{\sigma_u} = \hat{\lambda}\sigma_u. \tag{9}$$

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. Since the amount of particles required to track multiple objects increases fast, the smallest number of particles that produced good results was chosen.

3.2. Color histogram-based tracking

The objects are tracked using color histograms (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 H_{HS} histogram with $N_{H_H}N_{H_S}$ bins is populated using only information obtained from pixels whose Saturation and Value are above some given thresholds 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, and they are used to populate an H_V histogram that is concatenated to the H_{HS} one built before. The resulting histogram is composed of $N_{H_H}N_{H_S} + N_{H_V}$ bins. Following the aforementioned paper, the variables are set as $N_{H_H} = N_{H_S} = N_{H_V} = 10$.

Each histogram corresponds to one observation $\boldsymbol{o}_t^j$ for object j at instant t for the particle filter. Section 4.4.2 presents more details about how the histograms are compared in order to track each object.

4. Multi-object tracking based on structural information

In this section, the proposed tracking framework is explained. Fig. 2 shows a flowchart of the process. First, a video with annotations for the first frame is provided as input. Then, the annotations are used to learn the initial structural graph model, which is employed to generate some additional object candidates for recovering tracking in case of loss. The candidates are evaluated according to the model and the final result is used to update the model before processing the next frame.

Tracking is carried out by evaluating two Attributed Relational Graphs (ARGs): a model graph G^M and a scene graph G^S. The definition of both ARGs is presented in Section 4.1. 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. First, the model graph G^M of the image structure is learned using the annotations from the first frame and then updated online at each frame of the video (Section 4.2). Multiple hypotheses about the state of each object i are kept by using a set of trackers

$$\mathcal{P}_i^t = \{(P_j^i, w_j^i)|j = 1, \ldots, n_i^t\}, \tag{10}$$

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. For obtaining the set of trackers, G^M is used to generate candidates on the most likely locations (Section 4.3). Each candidate yields a new pair $(P_k^i, w_k^i = 0)$ 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, the global tracking result is obtained by optimizing a matching function between the model and the scene graphs (Section 4.4). Having found the best trackers, the temporal scores of all of the candidates are updated (Section 4.5). The next sections detail each step.

4.1. Attributed Relational Graph (ARG)

An ARG is a tuple

$$G = (\mathcal{V}, \mathcal{E}, \mathcal{A}_\mathcal{V}, \mathcal{A}_\mathcal{E}), \tag{11}$$

where $\mathcal{V} = \{v_i|i = 1, \ldots, N_O\}$ represents a set of vertices (or nodes), $\mathcal{E} = \{e_{ij} = (v_i, v_j)|i, j \in \{1, \ldots, N_O\}\}$ 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 vertices and edges, 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. Fig. 3 shows one possible scene graph generated from the following adjacency matrix:

$$M_A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{12}$$

Two different kinds of attributes are used: appearance and structural attributes. Appearance attributes are related to each object and they are stored in $\mathcal{A}_\mathcal{V}$. On the other hand, structural

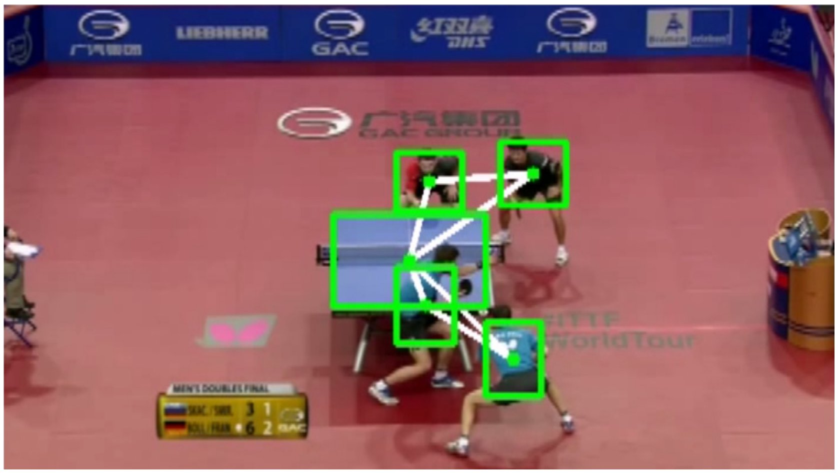

Fig. 3. An example of a scene graph.

attributes represent relations among objects and thus constitute edge attributes in $\mathcal{A}_{\mathcal{E}}$.

4.1.1. Model graph

The model graph G^M is an ARG, whose topology is obtained by means of an adjacency matrix M_A. The choice of the topology depends on the problem, and it may be defined based on common sense or expert knowledge. As a guideline, edges should be added: (1) between objects with similar appearance that tend to cause inter-occlusion (in this way, the graph can be used to handle the ambiguity problem after occlusion), (2) between pairs of objects that present a clear positional relationship (thus allowing the generation of good candidates to search in case of loss), and (3) to exploit the relation between moving objects and some reference (stable) object. The appearance attributes in $\mathcal{A}^M_{\mathcal{V}}$ are computed from annotations provided in the first frame of the video. In our experiments, the appearance is described by using color histograms as presented by Pérez et al. (2002). However, any other appearance descriptor could also be considered, like HOG (Dalal and Triggs, 2005) or SIFT (Lowe, 2004), and the proposed method can be applied directly.

The initial set of attributes $\mathcal{A}^M_{\mathcal{E}}$ of G^M also comes from the structure observed from the annotations of the first frame. However, they are also constantly updated after every frame (Section 4.2). Let δ be one of the structural attributes to be measured (e.g. the distance between two objects). The structural attributes of the model graph consist of the probability density function (PDF) of δ. Inspired by Cho et al. (2013), the chosen set of attributes is

$$\Delta(i, j) = \{(\theta(e_{ij}), d(v_i, v_j))\}, \tag{13}$$

and the PDF is estimated by means of histograms H_δ (H_θ or H_d in this case). The function $\theta(e_{ij})$ represents the clockwise angle between the horizontal axis and the vector $\overrightarrow{v_i v_j}$, and $d(v_i, v_j)$ is the Euclidean distance between the two vertices (Fig. 4). In order to obtain more robustness, the distance is normalized by dividing it by the width of the image. The PDFs are initialized from the annotations obtained from the first frame. For this, we assume that these annotations have the highest confidence value (equals to 1) and we use the convolutional kernels explained in Section 4.2.

Fig. 5 shows one example of a graph and the learned histograms for each attribute.

4.1.2. Scene graph

The scene graph has the same topology as the model graph, and it is built at each frame as follows. Each scene configuration k is represented by a graph G^S_k (some examples of different graph configurations for one scene can be viewed in Fig. 7). A vertex $v_i \in \mathcal{V}^S$ of the scene graph G^S_k is associated with one cloud of particles P^i_j for object i. Let us assume that the state of each object is composed of $|X| \geq 2$ components (in this paper the state is composed of exactly 2 components). Let $r(P^i_j) = (x^{i,j}_1, x^{i,j}_2, \ldots, x^{i,j}_{|X|})$ represent the final vector state obtained from Eq. (4) for the particle cloud P^i_j. In our experiments, $x^{i,j}_1$ and $x^{i,j}_2$ represent the 2D coordinates of the object in the image space, and the position of v_i is given by

$$r_P(P^i_j) = (x^{i,j}_1, x^{i,j}_2), \tag{14}$$

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

The edges are then built using the same matrix M_A as in the training and in the model graph. However, recall that each object is tracked by a set of different trackers. Therefore, each scene may be described by multiple graphs that have the same topology, which represent all the possible combinations of different trackers (Fig. 7 displays candidates that may be added to each tracker set).

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 extracted from the current frame (i.e. the observations of δ). The attributes of the vertices are the associated pairs (P^i_j, w^i_j).

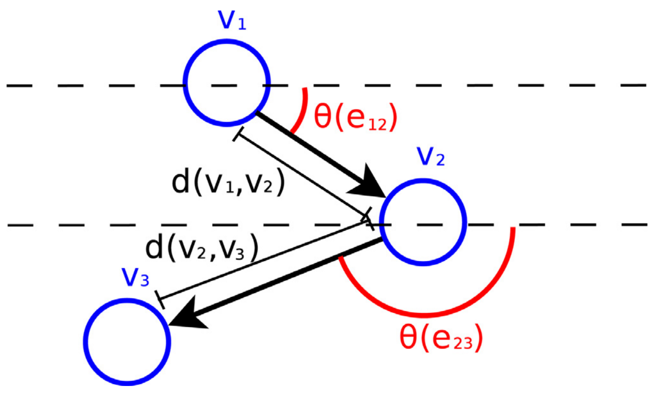

Fig. 4. The structural attributes of the edges.

4.2. Updating the model graph

In the beginning, for each attribute histogram H_δ, the range $range\ (H_\delta) = [min, max]$ and the number of bins $bins(H_\delta)$ must be

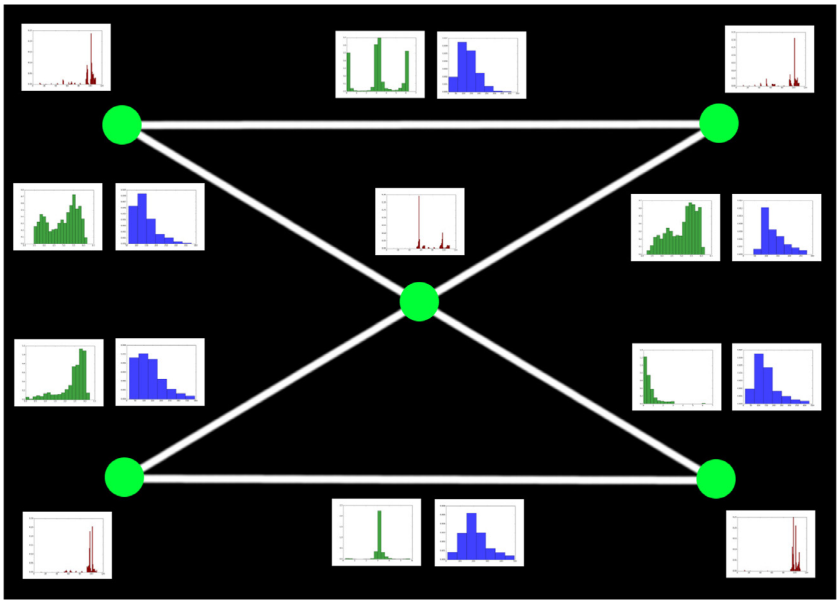

Fig. 5. An example of a model graph with the learned attributes. The red histograms represent the attributes of the vertices (color model), while the green ones represent the angles, and the blue ones represent the distances. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

specified. The tracking results are then used to update the model at each frame. The structural measurements extracted from the scene graph are used to cast a vote and update the model histograms. However, in order to deal with uncertainty and create a more general model, the measurements are first convolved as $\delta * k_C$ with a kernel k_C that depends on the confidence of the tracking result. The confidence is measured according to the likelihood of the particle filter estimation (Eq. 4) as $\mathbb{P}(\boldsymbol{o}_t^i|r(P^i))$.

The convolution kernel is chosen according to the confidence as follows:

$$k_C = \begin{cases} [0.3, 0.4, 0.3] & \text{if } \mathbb{P}(\boldsymbol{o}_t^i|r(P^i)) > 0.7, \\ [0.15, 0.2, 0.3, 0.2, 0.15] & \text{if } 0.3 < \mathbb{P}(\boldsymbol{o}_t^i|r(P^i)) \leq 0.7, \\ [0.1, 0.13, 0.17, 0.2, 0.17, \\ \quad 0.13, 0.1] & \text{otherwise.} \end{cases} \tag{15}$$

Therefore, less confident results are more spread throughout the histogram range to reflect the uncertainty of the observations.

4.3. Generating new candidates

Besides for tracking evaluation, we propose using the structural information of G^M to generate new candidate positions for each tracked object. This step is important to improve overall tracking, particularly to recover the target after a tracking failure occurs.

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

$$(\hat{\theta}_k = \theta_k + u_\theta, \hat{d}_k = d_k + u_d) \tag{16}$$

by simulating according to the distributions given by the histograms $\theta_k \sim H_\theta(\theta(e_{ij}))$ and $d_k \sim H_d(d(v_i, v_j))$, 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

$$(v_i(x) + \hat{d}_k \cos(\hat{\theta}_k), v_i(y) + \hat{d}_k \sin(\hat{\theta}_k)). \tag{17}$$

Fig. 6 shows the candidates generated for each object. The candidates are then used to 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.

In order to avoid generating graphs using unreliable sources, the generated candidates are filtered according to two criteria. Firstly, we check whether a candidate significantly overlaps another older tracker. Therefore, if the overlap ratio is above a given threshold τ_O, the candidate is discarded. Secondly, we also compute the appearance score (see Section 4.4.2) and remove candidates whose score is below τ_S.

4.4. Optimizing tracking using graphs

In this section, the method used for choosing the best trackers for each object is explained. Each tracker receives a score based on a matching function according to its respective scene graph and the model. Later, the best candidates are chosen by optimizing a global function over all the objects.

4.4.1. Computing graph score

The score of a scene graph G^S is obtained by summing over the temporal scores of all of its vertices:

$$s(G^S) = \sum_{i=1}^{N_O} w_{j_G}^i, \tag{18}$$

where $w_{j_G}^i$ corresponds to the temporal weight of the candidate j of object i that is used to create the graph G^S. This score measures the reliability of the associated tracker $P_{j_G}^i$ along time. This is done by computing a weighted accumulation of instantaneous scores:

$$(w_{j_G}^i)^t = \rho_T (w_{j_G}^i)^{t-1} + f(i, G^S, G^M), \tag{19}$$

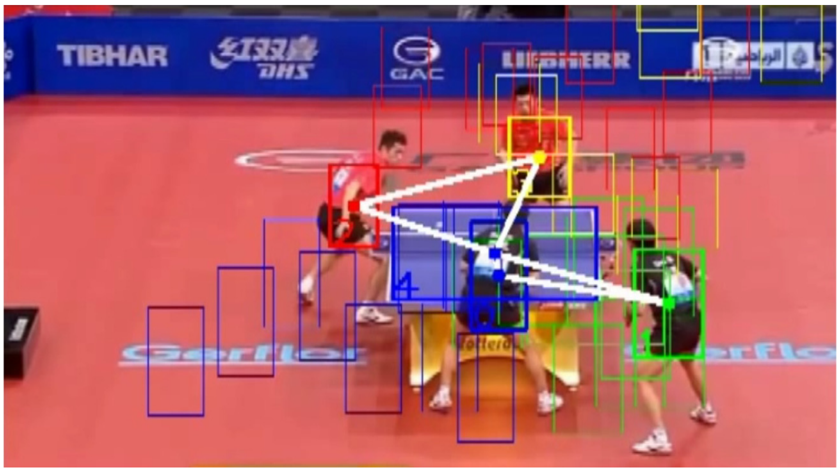

Fig. 6. An example of candidates generated for one scene. Rectangles of the same color indicate that they belong to the same object.

where t indicates the time, 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 $((P^i_{j_G})^t, (w^i_{j_G})^t)$. 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 four parts:

$$f(i, G^S, G^M) = \rho_A\phi_A(i, G^S) + \rho_S\phi_S(i, G^S, G^M) - \rho_O\phi_O(i, G^S) - (1 - \rho_F - \rho_S - \rho_O)\phi_C(i, G^S), \quad (20)$$

where ρ_A, ρ_S and ρ_O are given weighting factors such that $\rho_F + \rho_S + \rho_O \leq 1$. The functions ϕ_A, ϕ_S, ϕ_O and ϕ_C are the appearance, structural, overlapping and changing score functions of v_i, respectively. Notice that the last two functions act as penalty terms and thus, they decrease the score. These functions will be further explained in the following sections.

4.4.2. Appearance score

The appearance score is actually the confidence of the particle cloud P_i associated with object i (vertex v_i) as in Eq. (5). Hence, it is set to

$$\phi_A(i, G^S) = \zeta(P_i). \quad (21)$$

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 Erdem et al. (2012), using the Bhattacharyya distance d_B:

$$\mathbb{P}(\boldsymbol{o}_t|\boldsymbol{x}_t^i) = \exp\left(-\frac{d_B(H^M, H^S)^2}{2\sigma^2}\right), \quad (22)$$

where H^M and H^S are histograms, normalized to sum to one, of the model and the scene, respectively and

$$d_B(H^M, H^S) = \sqrt{1 - \sum_j \sqrt{H^M(j)H^S(j)}}, \quad (23)$$

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

4.4.3. Structural score

Let $\boldsymbol{m_i}$ be a vector representing the line i from the adjacency matrix M_A, i.e. corresponding to object i. 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 likelihoods of each structure measurement coming from the scene according to the model histograms. The structure score is computed using the dot product:

$$\phi_S(i, G^S, G^M) = \frac{1}{2\|\boldsymbol{m_i}\|_1}\boldsymbol{m_i}\cdot\boldsymbol{\theta_i^S} + \boldsymbol{m_i}\cdot\boldsymbol{d_i^S}, \quad (24)$$

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.

4.4.4. Overlapping score

The overlapping score is used to penalize configurations with a high intersection between objects of similar appearance. This is done to facilitate the recovery after one or more trackers with similar models lose the target in case of temporary occlusion. Let $\mathcal{B}^i$ be the set of pixels inside the bounding box of particle cloud P^i and H^i the color histogram of $\mathcal{B}^i$. The overlapping score is obtained by:

$$\phi_O(i, G^S) = \sum_{j=1, j\neq i}^{N_O}\left[(1 - d_B(H^i, H^j))\frac{|\mathcal{B}^i \cap \mathcal{B}^j|}{|\mathcal{B}^i|}\right], \quad (25)$$

where $d_B(H^i, H^j)$ is the Bhattacharyya distance defined in Eq. (23).

4.4.5. Changing score

This function is defined to penalize the change of trackers between consecutive frames. The reason is to reinforce the assumption of smooth movement and, therefore, to keep the same tracker for as long as possible. This score is obtained from a simple indicative function that checks whether the tracker chosen in the last frame is being used in the current frame:

$$\phi_C(i, G^S) = \begin{cases} 0 & \text{if tracker } j \text{ for object } i \text{ from instant } t-1 \text{ is kept,} \\ 1 & \text{if a new candidate } k \text{ replaces the old tracker } j. \end{cases} \quad (26)$$

4.4.6. Choosing the best scene graph

The best trackers are selected by building the scene graphs G_k^S and computing the scores explained before. Fig. 7 shows 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 because the number of graph combinations is usually very large and unfeasible to be processed in real time. Instead, it was chosen an iteratively greedy approach that fixes the vertices for all objects except one and optimizes the score for one object at a time.

Let $\mathcal{V}_*^S = \{v(P_*^i)|i = 1, \ldots, N_O\}$ be the set of vertices with the best scores found at a certain iteration step and whose graph score is $s(G_*^S)$. The initialization procedure of this set will be discussed later. Assume that the optimization is being performed for object

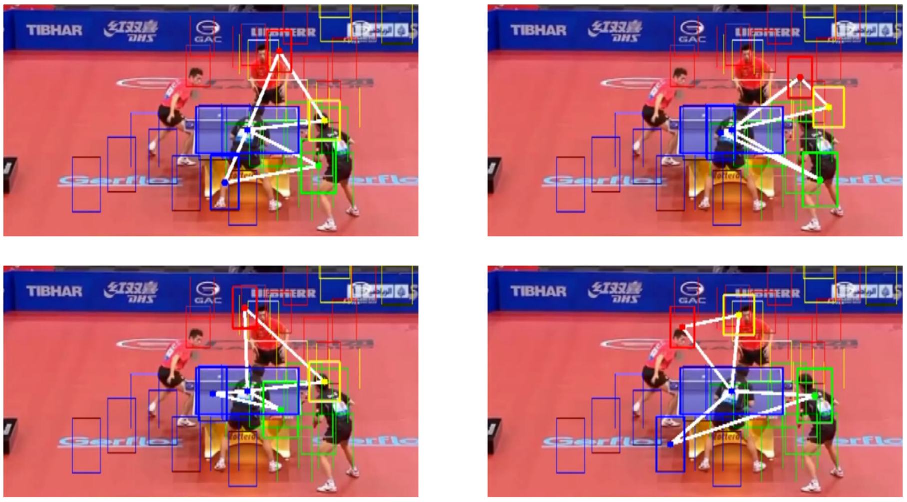

Fig. 7. Some examples of candidate graphs to be analyzed.

j and we are testing its l-th candidate. We fix the set of vertices $\mathcal{V}^S_{j,l} = \mathcal{V}^S_* \cup \{v(P^j_l)\} \setminus \{v(P^j_*)\}$ and compute the score of the associated graph $G^S_{j,l}$. If $s(G^S_{j,l}) > s(G^S_*)$, then we change the set of best vertices as $\mathcal{V}^S_* = \mathcal{V}^S_{j,l}$. This step is repeated for one object at a time and all of its candidates are tested. After all of the objects are evaluated, we repeat the process again until $\mathcal{V}^S_*$ does not change during one whole iteration or if the iteration limit τ_I has been reached. Since the score function is upper bounded, this procedure is guaranteed to converge if enough time is available. However, it does not always find the global maximum, sometimes being stuck at a local peak. In order to try to overcome this, we reinitialize the optimization procedure N_{RI} times and take the best results of all runs. We also employ two different heuristics for choosing the initial vertices and the order of testing the objects.

The first procedure is the score-guided optimization. For this step, the initial set of vertices $\mathcal{V}^S_*$ is chosen as the ones used in the last frame. Let $\{(P^i_*, w^i_*)|i = 1, \ldots, N_O\}$ be the set of the best trackers of each object, i.e.

$$(P^i_*, w^i_*) = \underset{(P^i_j, w^i_j) \in \mathcal{P}^t_i}{\arg\max}\, w^i_j. \tag{27}$$

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. The second heuristic is a completely random run. In this case, both the initial set of vertices and the order of testing objects are chosen randomly. The final set of vertices is chosen by running the score-guided optimization once and repeating the random procedure N_{RI} times to find the set of trackers that yields the best global score.

4.5. Updating trackers

When running the graph optimization, temporal scores are computed but not stored as the final scores of each tracker. The reason is that, since they depend on the whole graph, they should not be updated until the best graph G^S_* has been found. Once the graph is decided, the temporal scores of all candidates can be updated by following the same greedy procedure as explained in Section 4.4.6 using $\mathcal{V}^S_*$ as the initial set of vertices.

We also want to remove trackers whose scores are too low. Therefore, whenever the temporal score of a tracker falls below a threshold τ_R, the tracker is removed. The only exception is when a given object i has only one tracker in its set. In this case, the tracker is kept despite its score.

5. Experimental results

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 in Pérez et al. (2002). In our experiments, we observed that the tracking speed was between 3 and 4 frames per second (FPS) on the table tennis and badminton videos, and around 1.5 FPS on the volleyball sequences, using a machine equipped with an Intel Core i5 processor and 8GB RAM. Notice that the current code is neither parallelized nor utilizes the GPU. Therefore, there is still room for significant improvement in the performance. Specifically, the particle filter and graph evaluations would greatly benefit from parallelization, since all the instances could be processed simultaneously. The source code is publicly available for downloading and testing[2].

5.1. Evaluation metrics

We use the widely adopted CLEAR-MOT metrics (Bernardin and Stiefelhagen, 2008), whose most important results are the MOTP and MOTA. In this section, we will briefly explain each of them.

MOTP is an acronym for Multiple Object Tracking Precision, while MOTA is the Accuracy. The former metric represents the capacity of the tracker to produce results which are close to the ground truth, even if the identity of the objects are swapped. The latter, on the other hand, measures how well the tracker is able to assign tracks to the correct object, without considering how close it actually is from the correct position.

Let d^i_t be the distance between the estimated result and the ground truth for object i at time t and c_t the number of matches found, then we can compute:

$$MOTP = \frac{\sum_{i,t} d^i_t}{\sum_t c_t}. \tag{28}$$

In this paper, the distance d^i_t is actually the overlapping between the estimated bounding box and the ground truth. Therefore, higher values of MOTP indicate better results.

[1] http://opencv.org/.

[2] https://github.com/henriquem87/structured-graph-tracker.

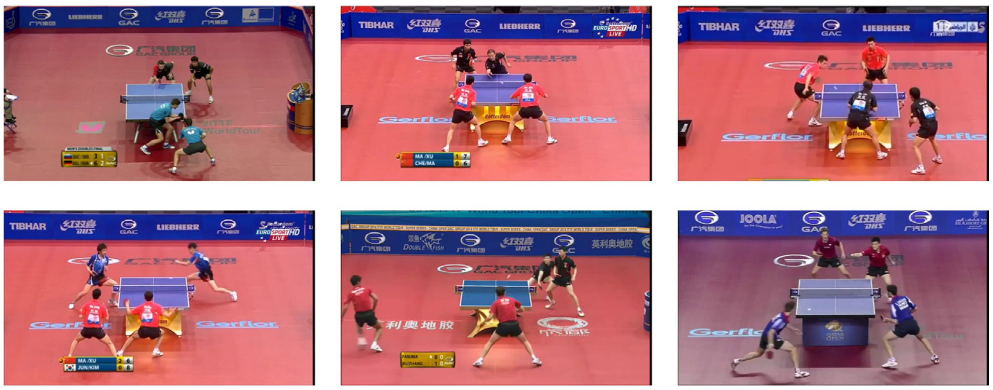

Fig. 8. Sample frames from the Youtube table tennis dataset.

For the MOTA, let g_t be the number of objects that exist at instant t. Let also m_t, fp_t and mme_t be the number of misses, false positives, and mismatches, respectively. Then, the metric can be obtained by:

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}. \tag{29}$$

We shall call MOT General or *MOTG*, a metric that corresponds to the average of *MOTP* and *MOTA*:

$$MOTG = \frac{MOTP + MOTA}{2}. \tag{30}$$

We also evaluate the results using the following additional metrics obtained from CLEAR-MOT:

- *IDSW*: number of ID switches, i.e. number of times the current association contradicts the one from the previous frame;
- *TPrate*: rate of true positives, i.e. total number of true positives divided by $\Sigma_t g_t$;
- *FPrate*: rate of false positives.

5.2. Datasets

The three datasets used for testing the tracking framework are described below. The proposed method is designed for handling videos where some structural properties can be observed. Among the many existing broadcast sports videos, we found out that sports like volleyball, table tennis, and badminton exhibit a clear structure that could be exploited. These videos also present some challenging conditions, such as similar appearance, occlusion, and rapid motion, which may cause even state-of-the-art methods to struggle. To the best of our knowledge, there is no public sports dataset that presents the desired characteristics for evaluating this method (Dubuisson and Gonzales, 2016). Therefore, we decided to collect appropriate videos to verify if the proposed approach could contribute for handling tracking problems in this class.

5.2.1. 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. Fig. 8 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 ground truth. The videos are encoded at resolutions varying from 640 × 360 to 854 × 480 pixels in each frame and at 30 FPS.

5.2.2. ACASVA

We selected three videos from the ACASVA (De Campos et al., 2011) dataset[3] 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 ground truth. The resulting videos were encoded at 1280 × 720 pixels per frame at 30 FPS and they contained 5766 frames. Fig. 9 displays some sample frames from this dataset.

5.2.3. Youtube volleyball

Following the same guidelines as before, videos of volleyball matches were chosen and edited. This dataset is composed of 3 videos recorded at 30 FPS at a resolution of 854 × 480, containing 5080 frames. The videos in this dataset do not contain cuts, which allows us to analyze the behavior of the tested trackers in this configuration. Videos using two different camera angles were collected, and those captured from a side view present some fast motion when the camera follows the ball from one side of the court to the other. This dataset also contains a significantly greater number of players to be tracked (6 on each team). Fig. 10 displays some sample frames from this dataset.

5.3. Choosing the parameters

We separated the Youtube table tennis 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 with all the expected challenging situation including overlapping between players of the same team and camera cuts.

We performed a test to find the best weights ρ_A, ρ_S and ρ_O for the scoring function (Eq. 20). Each configuration was evaluated five times and the presented results correspond to the average between all the observations.

For the test, we tried all combinations of values in the interval [0, 1] with an increment of 0.2, i.e. each parameter could be 1 out of 6 values. Ignoring the invalid configurations (where $\rho_A + \rho_S + \rho_O > 1$), this produces 56 possibilities. We evaluated all of them with the *MOTG* metric and found out that the best configuration consisted of $\rho_A = 0.4$, $\rho_S = 0$ and $\rho_O = 0.6$ with $MOTG = 0.61$. Besides finding the best parameters, we were also interested in verifying how each parameter affected the results. In order to do so, for each parameter value, we computed the average *MOTG* of all configurations that used that parameter and evaluated how

[3] Access to the dataset was provided as a courtesy by the original authors (De Campos et al., 2011).

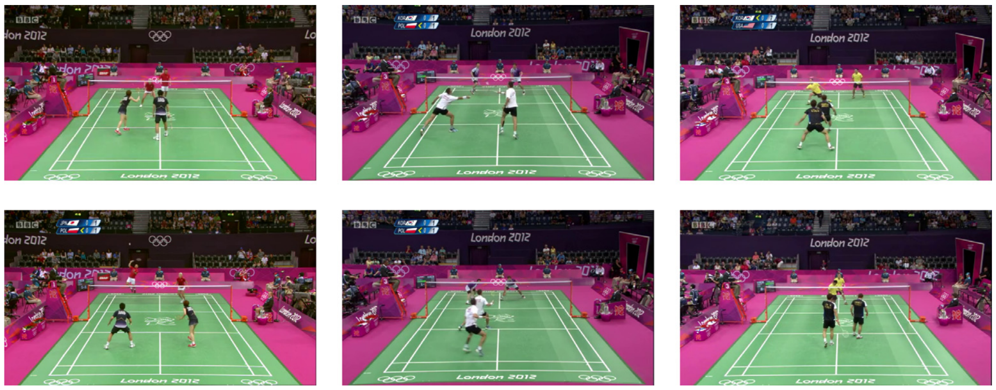
Fig. 9. Sample frames from the ACASVA badminton dataset.

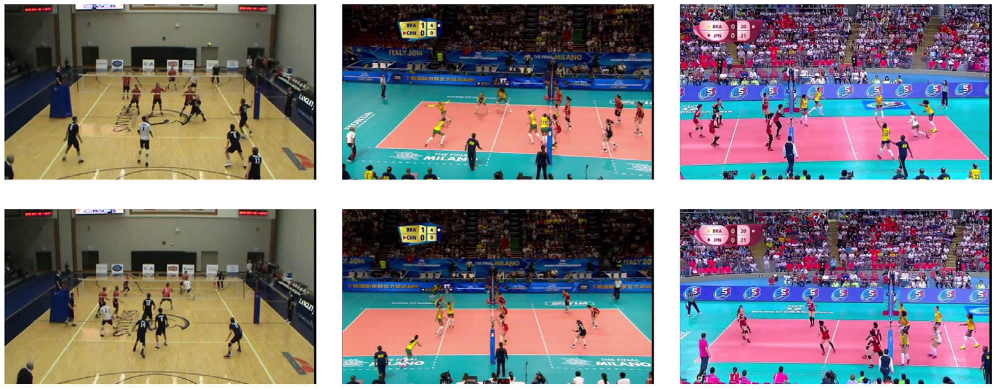
Fig. 10. Sample frames from the Youtube volleyball dataset.

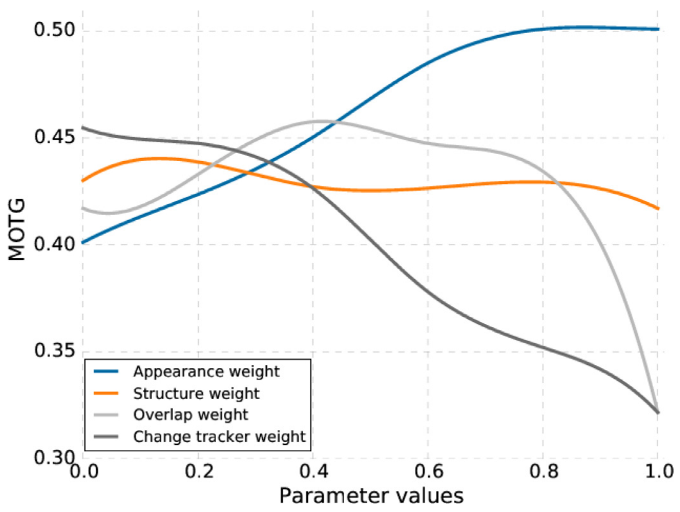

Fig. 11. *MOTG* according to graph parameter selection. Change tracker weight represents $(1 - \rho_A - \rho_S - \rho_O)$.

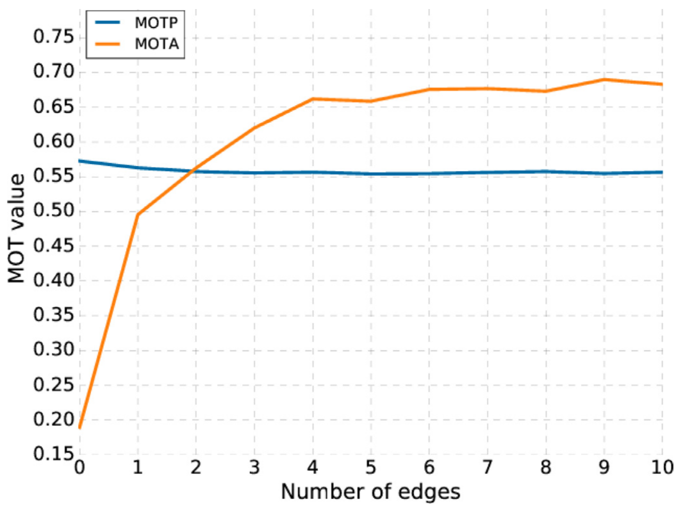

Fig. 12. *MOTA* and *MOTP* according to the number of edges in M_A.

the scored changed according to each one of them. Note that lower values have more samples, since more valid combinations of the remaining parameters exist. Fig. 11 shows the MOTG curves for each parameter value. It is worth noting that the method performance varies smoothly with the parameters, thus showing robustness to small parametric changes.

It is clear that the appearance is an important feature, showing a direct relation to the score. The change tracker, on the other hand, proved to hinder the performance and thus, should not be used. The overlap term presents the best performance at near the middle of the interval, while the structural part does not seem to affect much the results. Although the structural score is not used in the best configuration found for this test, some other configurations which did employ structure also appeared near the top results. Besides, the chart in Fig. 11 shows that the use of structure does not affect negatively the overall scores, thus indicating that the structure could be useful in some other applications. Although not used for evaluation in this test, the inclusion of structure to generate candidates for tracking significantly improves the results, as it will be shown in the next section.

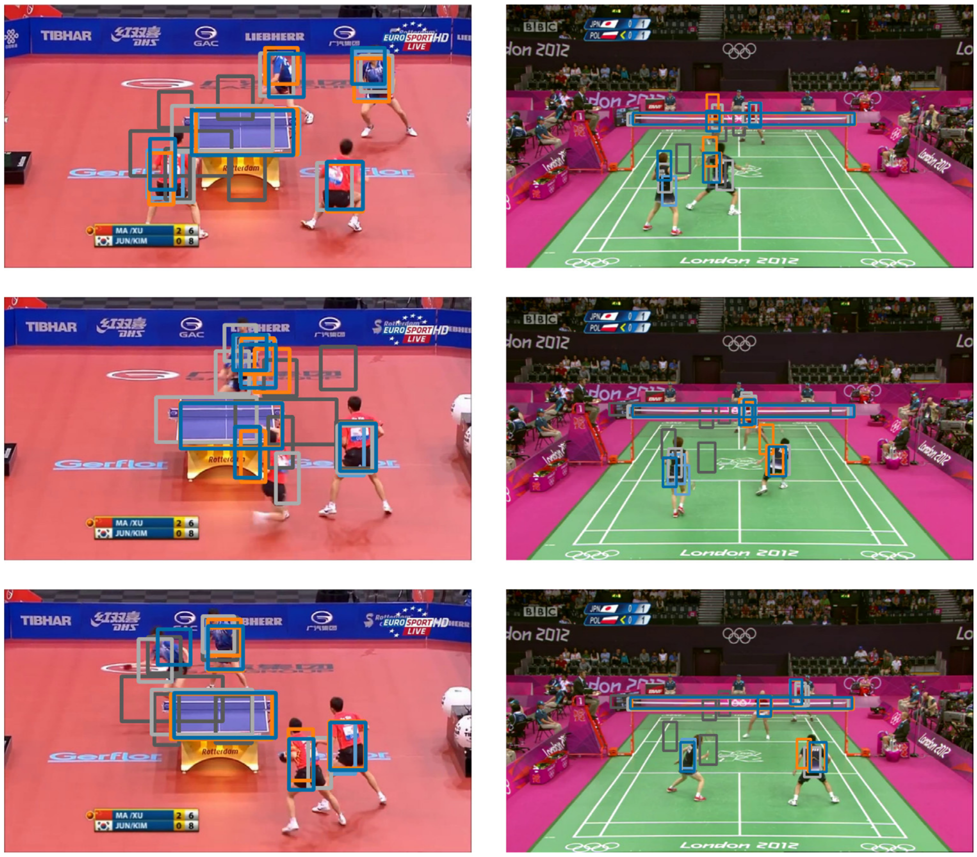

Fig. 13. Tracking results after occlusion. Each method is represented by a different color. Dark blue: Online Graph (ours), orange: PF, light gray: STRUCK, dark gray: SPOT, light blue: Offline Graph. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Parameters for the tracking framework.

distance histogram range	*range* $(H_d) = [0, 1]$
angle histogram range	*range* $(H_\theta) = [0, 2\pi]$
number of distance histogram bins	$bins(H_d) = 25$
number of angle histogram bins	$bins(H_\theta) = 18$
number of particles per object	$N_p = 50$
number of random optimization runs	$N_{RI} = 10$
initial particle spread deviation	$\sigma_c = 10$
particle spreading factor	$\alpha = 5$
maximum particle sum of weights	$\beta = 25$
appearance weight	$\rho_A = 0.4$
structure weight	$\rho_S = 0$
overlapping weight	$\rho_O = 0.6$
old temporal weight factor	$\rho_T = 0.8$
score threshold for removing candidates	$\tau_S = 0.4$
score threshold for removing old trackers	$\tau_R = 0.2$
overlap threshold for removing candidates	$\tau_O = 0.25$
graph optimization iteration threshold	$\tau_I = 10$
conv. kernel for confidence in (0.7, 1.0]	$k_C = [0.3, 0.4, 0.3]$
conv. kernel for confidence in (0.3, 0.7]	$k_C = [0.15, 0.2, 0.3, 0.2, 0.15]$
conv. kernel for confidence in [0.0, 0.3]	$k_C = [0.1, 0.13, 0.17, 0.2, 0.17, 0.13, 0.1]$

We also experimented with varying the old temporal weight factor ρ_T, the score threshold for removing candidates τ_S and the score threshold for removing old trackers τ_R. However, the results did not show any clear behavior, as demonstrated by the previous parameters. Therefore, we just chose one of the top performing configurations.

Another parameter that is worth investigating is the graph topology represented by the adjacency matrix M_A. We conducted some tests by varying the number of edges of the graph to see how it affected the performance. As the graph in the table tennis video is composed of 5 vertices, we tested all configurations until reaching a complete graph (10 edges). For each configuration with k edges, we randomly generated 5 adjacency matrices to be tested. Notice that the suppression of edges in M_A affects the information available for generating the candidates using the matrix M_C. In order to decrease the impact of the candidates generation on the evaluation of the topology, we defined $M_C = (m_{ij} = 5, \text{if } i \neq j, \text{otherwise } 0)$, i.e. each vertex generates 5 candidates for all the others. All the other parameters were fixed according to the tests reported above. The result is presented in Fig. 12.

The results evidence that using more edges has a clear impact on the accuracy, which significantly increases until around 4 edges. One point worth of note is that the use of additional edges does not negatively impact the results. Based on these results, and on knowledge about the configuration of the game, we chose to use the same adjacency matrix defined in Eq. (12), where the first four lines and columns represent the players, while the last one represents the net or the table (from now on referred to as middle line), on the badminton and table tennis videos. The adjacency matrix for the volleyball videos followed a similar design, where the net was connected to every player and the players of the same team were fully connected. This matrix considers the relations between players of the same team and all the players and the middle line. The relationship with the middle line is important because the players should be close and on opposite sides of it

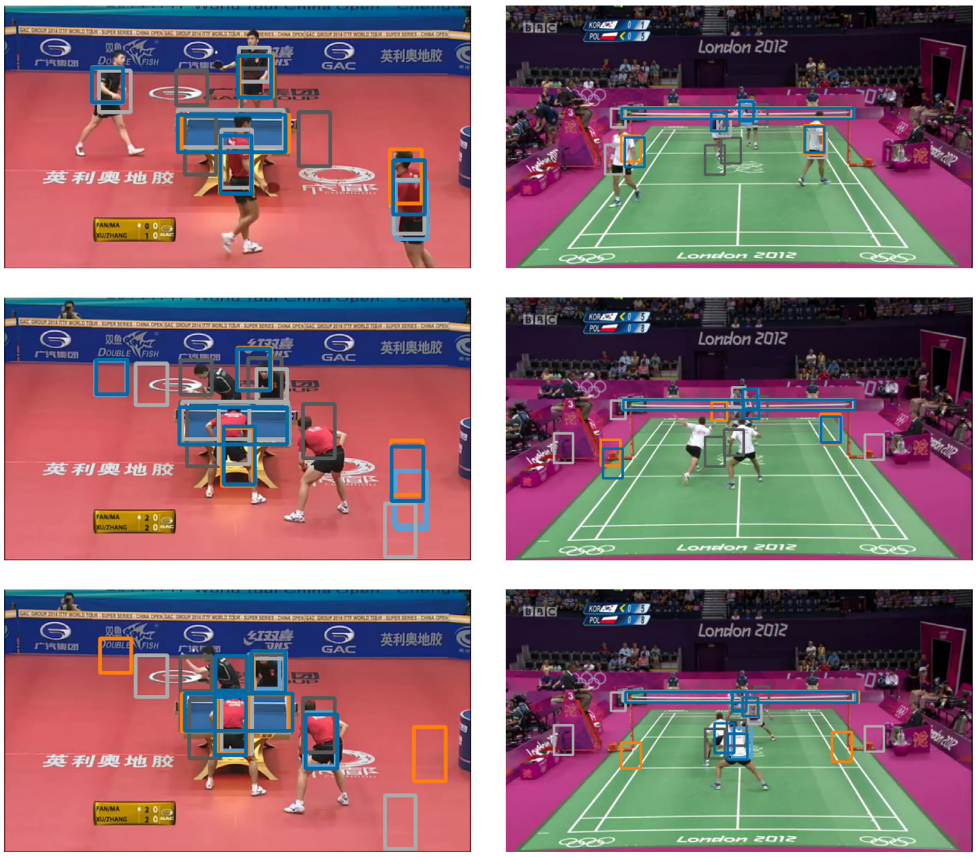

Fig. 14. Tracking results after a camera cut. Each method is represented by a different color. Dark blue: Online Graph (ours), orange: PF, light gray: STRUCK, dark gray: SPOT, light blue: Offline Graph. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

during the game. On the other hand, exploiting the relationships between players of the same team helps us to handle temporary occlusions. For the remaining experiments, the candidates matrix M_C was chosen to use the middle line as a reference to generate 10 candidates for each player.

Table 1 summarizes all the parameters chosen for the evaluation. The same values were used for all the experiments, independently of the dataset.

5.4. Experimental setup

We tested our approach on all the datasets previously presented. For the table tennis dataset, the video used for estimating the parameters was not included in this evaluation step. The task in these videos was to track all the players and the middle line. 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. All the videos are captured from a single camera, which moves in two of the volleyball videos, but it is fixed in all the others. As before, all the tests were performed five times and the average of all of the results was taken.

We compare our approach (Online Graph) with other methods from the literature. The first one was the same particle filter color tracker we used (PF), but without the graph information. In this way, we could verify whether the addition of graphs brings any significant improvement to the classical approach. The second is a previous version of this method (Offline Graph) as proposed by Morimitsu et al. (2015). This method requires an annotated dataset for training the model offline as well as uses a somewhat simpler score function. This test allows us to demonstrate the contributions of this method over the previous one. The third one was SPOT (Zhang and van der Maaten, 2014). This tracker also uses a structural graph as ours, but only uses distance information for structure. The tracking procedure consists in classifying the multiple graphs generated from HOG detectors using a structured SVM. The last tracker is STRUCK (Hare et al., 2011), a single object tracker that, according to a recent benchmark (Wu et al., 2015), was the best performing method in several datasets. This method employs kernelized structured SVM to handle tracking by classifying the input directly into the spatial position domain. In this way, the intermediate step of dividing the input templates into positive and negative training samples is skipped. Both PF and STRUCK are single object trackers and, thus, they track each object independently. STRUCK was included in the comparison to verify whether the use of a set of highly discriminative trackers alone would be able to solve the proposed problem.

5.5. Evaluation on the datasets

The results are presented in Table 2. The values correspond to the average of the results obtained from all videos. Although the precision of the proposed method is a bit lower than the ones obtained by other approaches, there is a significant increase in accuracy. This result is further evidenced by the best true and false positive rates. Even on the volleyball dataset, which does not contain camera cuts, our graph approach is better than STRUCK, showing that the structural constraints are a valuable aid in improving tracking in more cluttered scenes. Also, the online method showed a significant improvement over the offline one in the number of ID switches. This shows that the proposed approach is much more stable and do not cause many tracking failures. One point to note

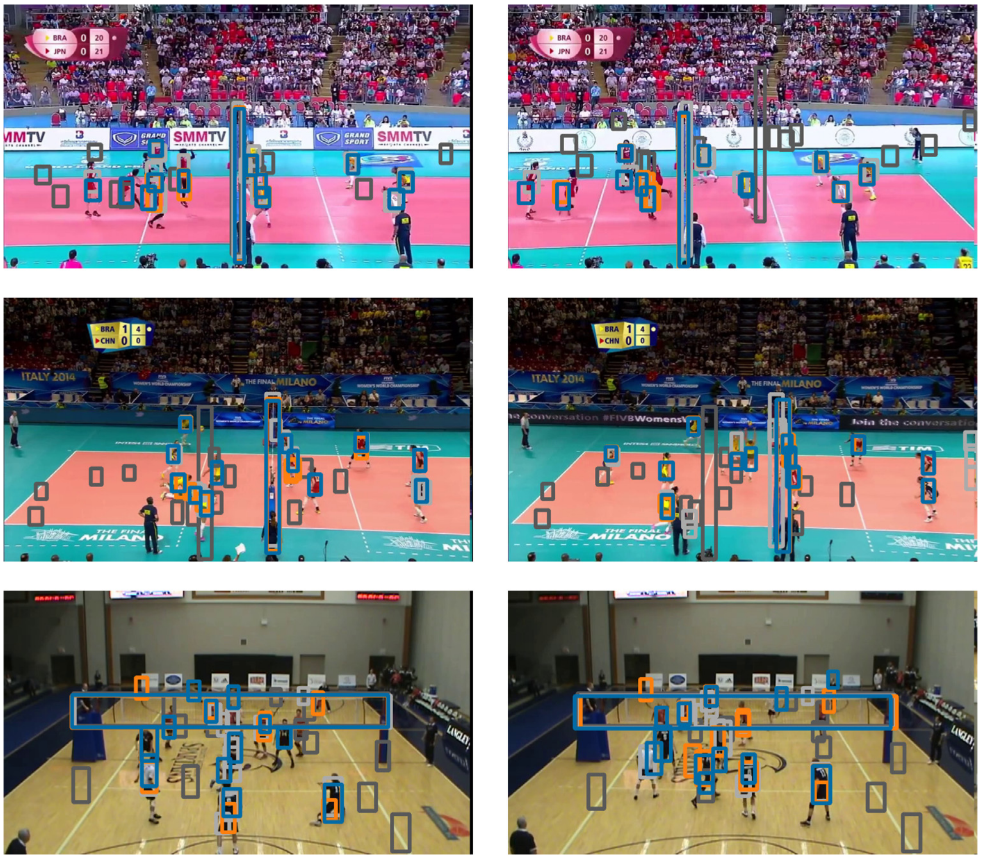

Fig. 15. Tracking results with many objects and a more cluttered scene. Each method is represented by a different color. Dark blue: Online Graph (ours), orange: PF, light gray: STRUCK, dark gray: SPOT. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Observed results on the datasets. The arrows indicate whether lower or higher values are better.

Dataset	Method	*MOTP* ↑	*MOTA* ↑	*IDSW* ↓	*TPrate* ↑	*FPrate* ↓
Youtube table tennis & ACASVA badminton	Online graph	0.589	**0.796**	85	**0.893**	**0.096**
	Offline graph	**0.622**	0.515	207	0.743	0.229
	PF	0.608	0.46	89	0.724	0.264
	SPOT	0.539	−0.008	**51**	0.492	0.5
	STRUCK	0.619	0.486	126	0.734	0.229
Youtube volleyball	Online graph	0.495	**0.367**	**624**	**0.667**	**0.302**
	PF	0.503	0.189	850	0.575	0.386
	SPOT	0.447	−0.608	836	0.179	0.786
	STRUCK	**0.552**	0.294	675	0.631	0.338

is that STRUCK performed similarly or worse than the particle filter approach in the badminton and table tennis sequences. This is explained because the videos in these datasets often contain many situations of camera cut. When this happens, both PF and STRUCK can only recover tracking when the target gets close to the point where it was lost. In that sense, the particle filters usually are able to recover the target more often because the particles are spread in a broader area than the STRUCK search radius. Since STRUCK conducts a dense neighbor search, as opposed to the sampled spread of PF, its search area must be kept smaller, and thus it is unable to detect the target in many situations. Another reason is that STRUCK updates the model along the video. In this case, if the target is tracked incorrectly, the model tends to deteriorate if the target is lost. It can also be observed that SPOT did not show good results on these datasets. According to the observed results, the main reason seems to be that the structural model used by SPOT is sometimes too rigid and not well suited for a situation where the structural properties between the objects are subject to large changes in a short amount of time.

Figs. 13–15 show some results observed on the videos. As it can be seen, our approach successfully recovers tracking after occlusion (Fig. 13) or camera cuts (Fig. 14), while PF and STRUCK are not able to re-detect the target after such situations. The videos from the volleyball dataset also present scenes captured from two different angles and with some camera motion. Even in these more challenging scenes, with many more objects, the graphs help to keep more correct tracks (Fig. 15). It is interesting to note that sometimes even the more robust STRUCK tracker is not able to deal with temporary occlusion, losing one of the targets, as shown in the last picture of Fig. 13. SPOT, on the other hand, does not suffer significantly from abrupt motion. However, as it is evident from the pictures, sometimes the more rigid model ends up causing many tracking misses at the same time. These results further evidence the flexibility of the proposed method, that is able to accept a wide range of spatial configurations.

We also evaluated the behavior of each tracker during the video. For that, we computed the instantaneous *MOTA* and *MOTP* values in a single frame. Fig. 16 shows how they vary along time in a video.

One of the first things we can notice is that STRUCK and sometimes PF present very high MOTP in the Youtube table tennis video. However, analyzing only this metric may be misleading. This happens because, after a camera cut, most PF and STRUCK trackers are lost. This causes the MOTP to be evaluated only on the few remaining trackers, which usually shows a higher precision than when evaluating many matchings. As can be observed from the MOTA, the accuracy is very low when such events occur, demonstrating that the actual results are not very good. From the charts,

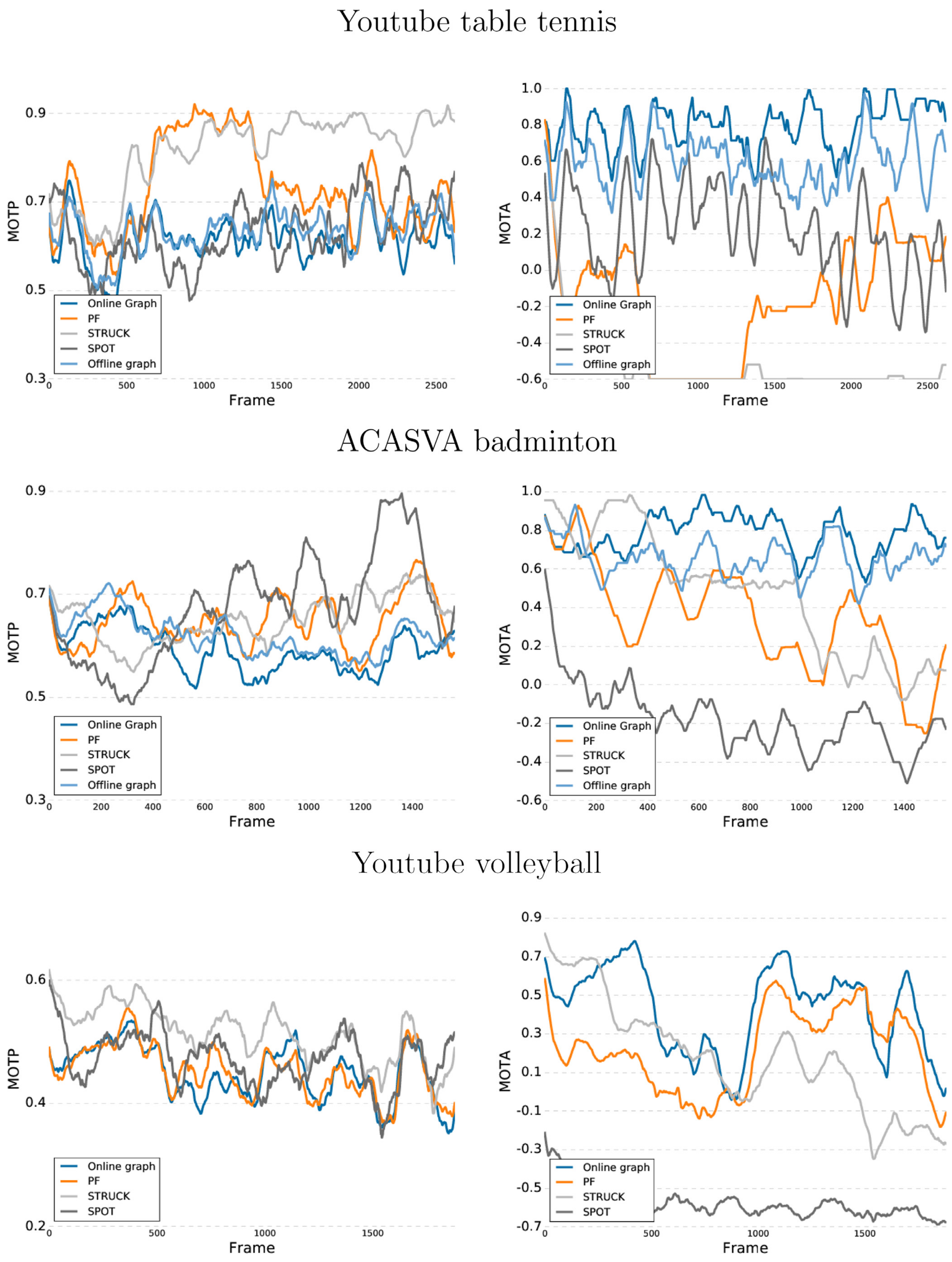

Fig. 16. Instantaneous MOTP and MOTA for one video from each dataset including object overlapping and camera cuts. The charts were smoothed using a moving average window of 100 frames.

the better accuracy of our method is evidenced, as it presents the best overall performance in all the videos. By evaluating the accuracy curves, we can also observe that the other trackers, especially PF and STRUCK, suffer from the drifting problem, i.e. performance deteriorates as the video goes on. The use of graphs fixes this problem, creating much more stable results even on longer sequences.

6. Conclusion

We proposed a graph-based approach to exploit the structural information of a filmed scene and to 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. The graph is then used to generate new likely target locations to try to improve tracking during longer periods of occlusion and camera cut. 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. The source code of the developed framework is publicly available for testing.

One of the advantages of the proposed framework is that it does not really rely on any information specific to a particular tracker. Therefore, the single object tracker could be potentially replaced by any other more suitable choice for other types of objects. This makes this approach very flexible and able to deal with a wider range of applications.

The results show that the proposed method successfully increases the tracking precision over other state-of-the-art methods for sports structured videos. As shown by the results, the candidates generated by the structural properties are successfully able to recover tracking in case of loss, while keeping their identities correct. This, in turn, greatly contributes to decrease drifting, which is also a challenging condition to deal with in longer videos. These experiments showed the robustness of the method to handle inter-object occlusion and video cuts from a single camera. Situations depicting the use of moving cameras are also supported, as long as the camera movement is limited (as evidenced by the volleyball tests), or smooth enough to provide sufficient time for the graphs to adapt to the new distributions. Video cuts between different cameras can also be accepted to some extent. If the cameras maintain roughly the same scene structure, e.g. slightly different angles or a change from a rear/side view to a top one without changing orientation, the graph configuration would not change so abruptly and the graph would be able to incorporate these changes. On the other hand, if the change is large, such as changing from one side of the court to the other, the method most likely would not be able to adapt well to the new situation.

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 sensitive 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 as a vertex attribute for evaluating, if the color model is not representative enough, the whole tracking may be affected. Therefore, one future extension of this work consists in replacing the color-based particle filter single tracker by a more robust one, such as STRUCK (Hare et al., 2011). The graphs could also be enriched by including other types of edge attributes. For example, we could encode more complex appearance and structural relations (Bloch et al., 2006).

As another future development, we want to improve the method by making it 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.

Acknowledgements

The authors are grateful to São Paulo Research Foundation (FAPESP) (grants #2011/50761-2, #2012/09741-0, #2013/08258-7 and #2014/50135-2), CNPq, CAPES, NAP eScience - PRP - USP, and PICS CNRS/FAPESP 227332 (2014-2015).

References

Bernardin, K., Stiefelhagen, R., 2008. Evaluating multiple object tracking performance: the clear mot metrics. J. Image Video Process. 2008, 1.

Bloch, I., Colliot, O., Cesar, R., 2006. On the Ternary Spatial Relation Between. IEEE Trans. Syst. Man Cybernetics SMC-B 36 (2), 312–327.

Cho, M., Alahari, K., Ponce, J., 2013. Learning graphs to match. In: IEEE International Conference on Computer Vision, pp. 25–32.

Choi, W., 2015. Near-online multi-target tracking with aggregated local flow descriptor. In: IEEE International Conference on Computer Vision, pp. 3029–3037.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, pp. 886–893.

De Campos, T., Barnard, M., Mikolajczyk, K., Kittler, J., Yan, F., Christmas, W., Windridge, D., 2011. An evaluation of bags-of-words and spatio-temporal shapes for action recognition. In: Workshop on Applications of Computer Vision, pp. 344–351.

Dubuisson, S., Gonzales, C., 2016. A survey of datasets for visual tracking. Mach. Vision Appl. 27 (1), 23–52.

Erdem, E., Dubuisson, S., Bloch, I., 2012. Fragments based tracking with adaptive cue integration. Comput. Vision Image Understanding 116 (7), 827–841.

Figueroa, P.J., Leite, N.J., Barros, R.M., 2006. Background recovering in outdoor image sequences: an example of soccer players segmentation. Image Vision Comput. 24 (4), 363–374.

Fink, G.A., 2008. Markov Models for Pattern Recognition: From Theory to Applications. Springer.

Grabner, H., Matas, J., Van Gool, L., Cattin, P., 2010. Tracking the invisible: learning where the object might be. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1285–1292.

Hare, S., Saffari, A., Torr, P.H., 2011. Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision, pp. 263–270.

Isard, M., Blake, A., 1998. CONDENSATION - conditional density propagation for visual tracking. Int. J. Comput. Vision 29 (1), 5–28.

Kristan, M., Perš, J., Perše, M., Kovačič, S., 2009. Closed-world tracking of multiple interacting targets for indoor-sports applications. Comput. Vision Image Understanding 113 (5), 598–611.

Kwon, J., Lee, K.M., 2008. Tracking of Abrupt Motion Using Wang-Landau Monte Carlo Estimation. In: European Conference on Computer Vision. Springer, pp. 387–400.

Liu, J., Carr, P., Collins, R.T., Liu, Y., 2013. Tracking sports players with context-conditioned motion models. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1830–1837.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60 (2), 91–110.

Lu, W.-L., Ting, J.-A., Little, J.J., Murphy, K.P., 2013. Learning to track and identify players from broadcast sports videos. IEEE Trans. Pattern Ana. Mach. Intell. 35 (7), 1704–1716.

Milan, A., Roth, S., Schindler, K., 2014. Continuous energy minimization for multitarget tracking. IEEE Trans. Pattern Anal. Mach. Intell. 36 (1), 58–72.

Morais, E., Ferreira, A., Cunha, S.A., Barros, R.M., Rocha, A., Goldenstein, S., 2014. A multiple camera methodology for automatic localization and tracking of futsal players. Pattern Recognit. Lett. 39, 21–30.

Morimitsu, H., Cesar Jr, R.M., Bloch, I., 2015. Attributed graphs for tracking multiple objects in structured sports videos. In: IEEE International Conference on Computer Vision Workshops, pp. 34–42.

Okuma, K., Taleghani, A., De Freitas, N., Little, J.J., Lowe, D.G., 2004. A Boosted Particle Filter: Multitarget Detection and Tracking. In: European Conference on Computer Vision. Springer, pp. 28–39.

Pérez, P., Hue, C., Vermaak, J., Gangnet, M., 2002. Color-based probabilistic tracking. In: European Conference on Computer Vision. Springer, pp. 661–675.

Shitrit, H.B., Berclaz, J., Fleuret, F., Fua, P., 2014. Multi-commodity network flow for tracking multiple people. IEEE Trans. Pattern Anal. Mach. Intell. 36 (8), 1614–1627.

Solera, F., Calderara, S., Cucchiara, R., 2015. Learning to divide and conquer for online multi-target tracking. In: IEEE International Conference on Computer Vision, pp. 4373–4381.

Soomro, K., Khokhar, S., Shah, M., 2015. Tracking when the camera looks away. In: IEEE International Conference on Computer Vision Workshops, pp. 25–33.

Su, Y., Zhao, Q., Zhao, L., Gu, D., 2014. Abrupt motion tracking using a visual saliency embedded particle filter. Pattern Recognit. 47 (5), 1826–1834.

Tang, S., Andriluka, M., Schiele, B., 2014. Detection and tracking of occluded people. Int. J. Comput. Vision 110 (1), 58–69.

Wu, Y., Lim, J., Yang, M.-H., 2015. Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. 37 (9), 1834–1848.

Xiang, Y., Alahi, A., Savarese, S., 2015. Learning to track: online multi-object tracking by decision making. In: IEEE International Conference on Computer Vision, pp. 4705–4713.

Xing, J., Ai, H., Liu, L., Lao, S., 2011. Multiple player tracking in sports video: a dual–mode two-way Bayesian inference approach with progressive observation modeling. IEEE Trans. Image Process. 20 (6), 1652–1667.

Zhang, K., Zhang, L., Yang, M.-H., 2012. Real-time compressive tracking. In: European Conference on Computer Vision. Springer, pp. 864–877.

Zhang, L., van der Maaten, L.J., 2014. Preserving structure in model-free tracking. IEEE Trans. Pattern Anal. Mach. Intell. 36 (4), 756–769.

Zhang, S., Wang, J., Wang, Z., Gong, Y., Liu, Y., 2015. Multi-target tracking by learning local-to-global trajectory models. Pattern Recognit. 48 (2), 580–590.

Zhang, T., Ghanem, B., Xu, C., Ahuja, N., 2013. Object tracking by occlusion detection via structured sparse learning. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1033–1040.

Zhou, X., Lu, Y., 2010. Abrupt motion tracking via adaptive stochastic approximation Monte Carlo sampling. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1847–1854.