Attributed Graphs for Tracking Multiple Objects in Structured Sports Videos

Henrique Morimitsu
University of São Paulo
São Paulo, Brazil
henriquem87@vision.ime.usp.br

Roberto M. Cesar-Jr.
University of São Paulo
São Paulo, Brazil
roberto.cesar@vision.ime.usp.br

Isabelle Bloch
LTCI, CNRS, Télécom ParisTech,
Université Paris - Saclay
Paris, France
isabelle.bloch@telecom-paristech.fr

Abstract

In this paper we propose a novel approach for tracking multiple object in structured sports videos using graphs. The objects are tracked by combining particle filter and frame description with Attributed Relational Graphs. We start by learning a probabilistic structural model graph from annotated images and then use it to evaluate and correct the current tracking state. Different from previous studies, our approach is also capable of using the learned model to generate new hypotheses of where the object is likely to be found after situations of occlusion or abrupt motion. We test the proposed method on two datasets: videos of table tennis matches extracted from YouTube and badminton matches from the ACASVA dataset. We show that all the players are successfully tracked even after they occlude each other or when there is a camera cut.

1. Introduction

Object tracking is a very important task for several applications like activity analysis, surveillance and automated navigation. Most trackers rely on visual information to follow the object throughout the video. However, this approach presents some significant drawbacks that may cause it to produce poor results. A challenging situation is when a tracked object is lost due to temporary occlusion or appearance ambiguity. In this case it is important to be able to notice that the object is lost and to find its location again. Good trackers may be able to acknowledge that the target is missing, by evaluating their score. However, since they usually rely on smooth movement assumptions, they are not able to recover the tracking if the target is far away from the position where it was lost. Figure 1 shows a typical example. Two persons with similar appearance are being tracked, but when they overlap and separate, the tracking for one of them is lost.

We argue that in some situations this problem can be solved by reasoning about the spatial relationships between the objects. If the scene represents a situation that usually follows a common spatial structure, then it is possible to choose the most likely configuration of the objects. In structured video scenes, some elements provide a kind of stable spatial structure to be explored. 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 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.

In this work we explore such structural relations by using graphs to encode the spatial configuration of the scene. We propose to use particle filter coupled with a structural model represented by graphs in order to track multiple objects in videos. However, the proposed framework is not limited to particle filters. In fact, any other single object tracker could also be used instead and benefit from the structural information. This makes the proposed framework very flexible and able to be used to potentially improve the results of any single object tracking used for multi-objects problems.

The use of structural information allows us both to evaluate the quality of the tracker of each object, and to deal with situations of abrupt motion. The contributions of this paper

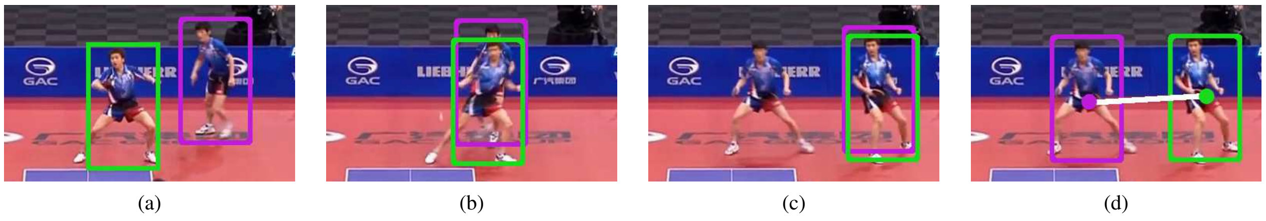

Figure 1. Example of a multi-object tracking situation. Most trackers are able to successfully track the targets when their appearance is clear (a). However, when overlap occurs (b), traditional trackers are often not able to solve the ambiguity problem in appearance and the tracking is lost (c). We propose to use structural information to recover the target position (d) after such events.

are the following: (1) a flexible probabilistic graph model that can be used to evaluate the structure of a scene, (2) a method for dealing with abrupt motion by generating candidate object locations based on the model, and (3) a novel framework for tracking multiple objects by evaluating multiple hypotheses with scene graphs.

The remainder of this paper is organized as follows. We present an overview of the general framework in Sec. 3. In Sec. 4 we review the concepts about tracking with particle filter. Section 5 specifies the graphs we use in this work. We show in Sec. 6 how the graphs help in the multi-object tracking process. In Sec. 7 we discuss the implementations details and present some experimental results. Finally, we present our conclusions in Sec. 8

2. Related work

Tracking by detection methods have been one of the focus of current efforts to deal with complex tracking situations [1, 18, 20]. This method consists in producing a motion track for each object by solving a data association problem between several noisy detections obtained from each frame. Although recent research has shown promising results, the data association function used in more robust models is usually very complex and difficult to optimize in real time.

Another approach is to initialize the tracking with annotations provided at the beginning of the video [9, 21]. This approach usually relies on first learning a discriminative appearance model for each object. Afterwards, the objects are tracked by conducting a local search guided by some predetermined motion model.

Multi-object tracking has been applied to sports videos, with methods based on particle filters being popular [11, 16, 22]. Other authors have tried to improve tracking by exploring more complex motion models. Liu *et al.* [13] used game context features designed specifically to model the motion behavior of players during a team match. Another challenging condition frequently found in sport scenes is mutual occlusion. Zhang *et al.* [25] tackle this issue by using a structured sparse model for each person. This approach builds on the robustness of sparse models [15] by reasoning that the occlusion model is usually not sparse, but rather a structured connected area.

Besides being used for occlusion reasoning, the use of structural information for improving multi-object tracking does not seem to have been much explored before. There are works like that of Grabner *et al.* [7], that consider the spatial relations between the object and a set of correlated points to deal with situations of occlusion. However, this method is designed for single object tracking and the structure is somewhat rigid. Perhaps the closest works to ours is that of Zhang and van der Maaten [24]. They propose a model-free tracking approach that includes the structural relations between objects into the data association function to perform tracking by detection. However, their work differs from the present 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 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. Our approach uses the structural model to obtain candidates of where the target is likely to be found after tracking loss.

Another important issue that must be dealt with during tracking is abrupt motion. Perhaps, the simplest way to overcome it is by generating some additional target location hypotheses around the previous location to try to cover a broader region [23]. Some authors have proposed to divide the image into several regions and use the information obtained from the density of states of each one to handle abrupt motion [12, 26]. Doulamis [5] also divides the image into regions and then assigns each of them to one of the tracked objects. These regions are used to re-initialize the tracker when the likelihood of the current target is low. In [19], Su *et al.* propose to rely on visual saliency information to guide the tracking and restore the target. As mentioned before, one of our main contributions is that the information obtained from the structure of the scene itself is used to find the most likely new object locations.

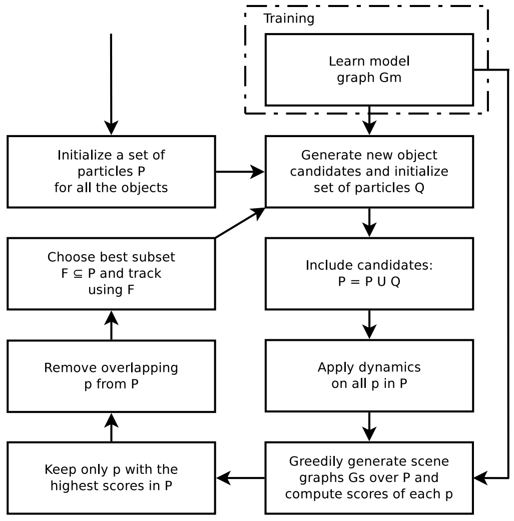
Training
Learn model graph Gm
Initialize a set of particles P for all the objects
Generate new object candidates and initialize set of particles Q
Choose best subset F ⊆ P and track using F
Include candidates: P = P U Q
Remove overlapping p from P
Apply dynamics on all p in P
Keep only p with the highest scores in P
Greedily generate scene graphs Gs over P and compute scores of each p

Figure 2. Overview of the proposed framework.

3. Overview of the framework

In this section we explain the proposed framework to provide an overview of the whole process. Figure 2 shows a chart representation of the framework.

First, the model graph G^m of the image structure is learned using annotated training images as described in Sec. 5.1. For the tracking step, the position of each of the N_o objects o_i is manually annotated in the first frame of the video. We keep multiple hypotheses about the state of o_i by using a set of trackers $\mathcal{P}_i^t = \{(P_j^i, w_j^i)\}_{j=1}^{N_{o_i}^t}$, where P_j^i is the j-th tracker of object i, w_j^i is a temporal confidence score and $N_{o_i}^t$ is the number of tracker candidates at instant t.

We cope with abrupt motion by continuously generating new hypotheses about the position of each target. For this, we use G^m to generate candidate positions c_k^i from the most likely locations (Sec. 6.1). Each c_k^i 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, we compute a temporal score for every $P_j^i \in \mathcal{P}_i^t$ (Sec. 6.2). 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 (Sec. 6.3). The scores are then used to remove undesired trackers from the set (Sec. 6.4). The final step consists in actually choosing the best trackers from each set to provide the final multi-object tracking result (Sec. 6.5). The next sections detail each step.

4. Object tracking with particle filter

This work employs particle filter using the ConDensation algorithm. ConDensation uses factored sampling [10] on particle filter models in order to track objects. The particle filter tracking consists in estimating the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ of a cloud P of weighted particles $\{(\mathbf{x}_t^i, \pi_t^i)\}_{i=1}^{N_p}$, where $\mathbf{x}_t^i$ is the state of particle i and π_t^i its weight, computed from the observed measurements $\mathbf{z}_{1:t}$ until the instant t. The ConDensation approach computes this distribution by generating a new cloud of n particles by sampling them, with repositions, from the old ones. By assuming $\sum_{j=1}^{N_p} \pi_{t-1}^j = 1$, the probability of each particle i being chosen in this step is π_{t-1}^i. Hence, more likely particles can be sampled several times, while others may not be chosen at all. Then, for each particle, a new state is predicted. The prediction 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:

$$\mathbf{x}_t^i = A\mathbf{x}_{t-1}^i + B\mathbf{u}, \tag{1}$$

where A is the motion dynamics matrix and $B\mathbf{u}$ is the noise term.

The weight of each particle i, is computed by:

$$\pi_t^i = \frac{p(\mathbf{x}_t^i|\mathbf{z}_t)}{\sum_{j=1}^{N_p} p(\mathbf{x}_t^j|\mathbf{z}_t)}. \tag{2}$$

Finally, the decision about the current state of the tracked object given by the cloud of particles P is obtained by the weighted average of the particles:

$$s(P) = \sum_{i=1}^{N_p} \pi_t^i \mathbf{x}_t^i. \tag{3}$$

It is also important to be able to evaluate the overall quality of P. We propose to do this by computing a confidence score based on the non-normalized weights of the particles:

$$c(P) = 1 - \exp\left(-\sum_{i=1}^{N_p} p(\mathbf{x}_t^i|\mathbf{z}_t)\right). \tag{4}$$

5. Attributed Relational Graph (ARG)

An ARG is a tuple $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}_\mathcal{V}, \mathcal{A}_\mathcal{E})$, 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 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.

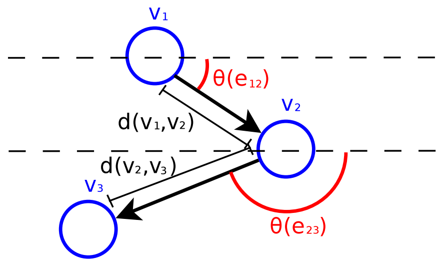

Figure 3. The structural attributes of the edges.

We chose to use two different kinds of attributes: the appearance and the structural attributes. Appearance attributes are related to each object and they are, therefore, stored in $\mathcal{A}_{\mathcal{V}}$. On the other hand, structural attributes represent relations among objects, being thus treated as edge attributes in $\mathcal{A}_{\mathcal{E}}$.

5.1. Learning the model graph

The topology of G^m is obtained by means of a provided adjacency matrix M_a. We obtain the set of attributes $\mathcal{A}^m_{\mathcal{E}}$ of G^m 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 $\lambda_k \in \Lambda$ be one of the structural attributes we are interested in (*e.g.* the distance between two objects). We want to use the annotations to estimate the probability density function (PDF) of λ_k. Inspired by [2] we chose $\Lambda = \{\theta(e_{ij}), d(v_i, v_j))\}$ and we estimate the PDF by means of histograms H_{λ_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).

The histograms are built by iterating over all images and collecting the respective observations I, which cast a vote for the histogram bin $H_{\lambda_k}(I)$. At the end, the PDF is estimated by normalizing H_{λ_k} to have a sum equal to one. Finally, the normalized histograms are then used as the edge attributes $\mathcal{A}^m_{\mathcal{E}} = \{H_{\lambda_k}\}_{\lambda_k \in \Lambda}$.

The appearance attributes in $\mathcal{A}^m_{\mathcal{V}}$ are not learned from the database. Instead, they are computed from annotations provided in the first frame of the tracking video. For our tests, we chose to describe the appearance by using color histograms. However, any other appearance descriptor could also be considered, like HOG [3] or SIFT [14].

5.2. Building the scene graphs

Each graph G^S_k represents one different scene configuration. 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. The position of v_i is obtained from $s_p(P^i_j)$, where $s_p(\cdot)$ is a truncated representation of Eq. 3 to include only 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.

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

6. Improved tracking with ARGs

6.1. Generating new candidates

Besides 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 $\mathcal{A}^m_{\mathcal{E}}$ are all relative to the origin of each edge e_{ij}, the position of v_i must be known. For this, we assume that the trackers for every object will not all fail at the same time. We control candidate generation 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 $a^m_{e_{ij}} = \{H(\theta(e_{ij})), H(d(v_i, v_j))\}$ be the attribute of an edge e_{ij} from G^m. We generate m_{ij} candidates for object j as $(\hat{\theta}_k = \theta_k + u_\theta, \hat{d}_k = d_k + u_d)$ 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 $u \sim \mathcal{N}(0, \sigma)$ is a Gaussian noise. Each candidate position is then obtained by $\mathbf{o_k} = (v_i(x) + \hat{d}_k \cos(\hat{\theta}_k), v_i(y) + \hat{d}_k \sin(\hat{\theta}_k))$. The candidates are then used to generate new particle clouds P^j_k which are inserted in the set $\mathcal{P}^t_j$. The clouds are initialized by spreading the particles according to a Gaussian distribution $\mathcal{N}(\mathbf{o_k}, \mathbf{\sigma_c})$.

6.2. 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:

$$(w^i)^t = \alpha(w^i)^{t-1} + g(i, G^s, G^m), \qquad (5)$$

where α is a given constant and $g(i, G^s, G^m)$ is the instantaneous score for the vertex v^s_i, 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:

$$g(i, G^s, G^m) = \beta\phi(i, G^s) + (1 - \beta)\psi(i, G^s, G^m), \qquad (6)$$

where β is a given weighting factor and $\phi(i, G^s)$ and $\psi(i, G^s, G^m)$ are the appearance and structural scores of v_i, respectively.

6.2.1 Appearance score

The appearance score is actually the confidence of the particle cloud P_i associated with v_i as shown in Eq. 4. Therefore we set $\phi(i, G^s) = c(P_i)$.

The confidence score depends on the weights of the particles, which are based on the likelihood $p(\mathbf{z}_t|\mathbf{x}_t^i)$. We compute this distribution in the same way as in [6], using the Bhattacharyya distance d_B:

$$p(\mathbf{z}_t|\mathbf{x}_t^i) = \exp\left(-\frac{d_B(H^m, H^s)^2}{2\sigma^2}\right), \tag{7}$$

where H^m and H^s are histograms of the model and the scene, respectively and $d_B(H^m, H^s) = \sqrt{1 - \sum_I \sqrt{H^m(I)H^s(I)}}$, where $H(I)$ is the bin I of histogram H.

6.2.2 Structural score

Let $\mathbf{m_i}$ 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 $\mathbf{d_i^s} = (H_d^m(d^s(v_i, v_j)))_{j=1}^{N_o}$ 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 measurements. We compute the structure score using the dot product:

$$\psi(i, G^s, G^m) = \frac{1}{2\|\mathbf{m_i}\|_1}\mathbf{m_i} \cdot \boldsymbol{\theta_i^s} + \mathbf{m_i} \cdot \mathbf{d_i^s}, \tag{8}$$

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

6.3. Generating scene graphs for evaluation

We select the best trackers by building the scene graphs G_k^s and computing the scores as explained before. Therefore, this could be formulated as an optimization problem of finding the graph with the highest score. However, we did not choose this approach because the main purpose of this step is to compute a good score for all possible candidates, not to find the best global configuration based on them. Another option would be to find the highest scores for each vertex individually. However, this may also present problems, as the resulting scores might come from many different subgraphs that, when merged together, do not represent a good global configuration. We balance the complexity of scoring all the vertices while keeping a reasonable global configuration by using a greedy approach. For this, we fix the vertices of all objects except one and optimize the score for one object at a time.

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.* $(P_*^i, w_*^i) = \arg\max_{(P_j^i, w_j^i) \in \mathcal{P}_i^t} w_j^i$. We create a sequence 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 $\mathcal{V}_k^s = \{v(P_*^b)\}_{b=1}^{N_o} \cup \{v(P_l^c)\} \setminus \{v(P_*^c)\}$, where $v(P)$ represents the vertex associated to P. This graph is then used to compute the score w_l^c with $1 \leq c \leq N_o$ and $1 \leq l \leq |\mathcal{P}_c^t|$.

6.4. Removing undesired trackers

After computing the score for each tracker, we remove those that are considered non-significant. 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 $\mathcal{Q}_i^t = \{(P_j^i, w_j^i) : w_j^i < \tau_s\}_{j=1}^{|\mathcal{P}_i^t|}$ and $(P_*^i, w_*^i) = \arg\max_{(P_k^i, w_k^i)} w_k^i$, where τ_s is a given score threshold. The thresholded set is obtained by:

$$\mathcal{R}_i^t = \begin{cases} \{(P_*^i, w_*^i)\}, & \text{if } |\mathcal{P}_i^t| = |\mathcal{Q}_i^t| \\ \mathcal{P}_i^t \setminus \mathcal{Q}_i^t, & \text{otherwise.} \end{cases} \tag{9}$$

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. We consider that two trackers P_k^i and P_l^i are overlapping when $d(s_p(P_k^i), s_p(P_l^i)) < \tau_{d_s}$, where τ_{d_s} is a given overlapping distance threshold for the same object. Let $\mathcal{S}^i = ((P_j^i, w_j^i) : (P_j^i, w_j^i) \in \mathcal{R}_i^t)$ be a sequencing of $\mathcal{R}_i^t$ sorted in decreasing order of weight. We remove overlapping trackers by using a greedy approach where the pairs $(P_a^i, w_a^i) \in \mathcal{S}^i$ are iteratively taken one by one following the ordering and inserted into a new set $\mathcal{T}_i^t$ whenever they do not overlap with any existing tracker, *i.e.*:

$$(\mathcal{T}_i^t)^n = \begin{cases} (\mathcal{T}_i^t)^{n-1} \cup \{(P_a^i, w_a^i)\}, & \text{if } d(s_p(P_a^i), s_p(P_b^i)) \geq \tau_{d_s}, \\ & \forall (P_b^i, w_b^i) \in (\mathcal{T}_i^t)^{n-1} \\ (\mathcal{T}_i^t)^{n-1}, & \text{otherwise,} \end{cases} \tag{10}$$

where the exponent n indicate the n-th iteration. The final set of trackers is obtained by $\mathcal{P}_i^{t+1} = (\mathcal{T}_i^t)^{|\mathcal{R}_i^t|}$.

6.5. Choosing the final trackers

Considering the temporal consistency of videos, we try to avoid changing trackers at each frame. However, in order to recover tracking after abrupt motion or appearance ambiguity, it is necessary to be able to detect when the tracker should be changed. We do so by considering the temporal score of each tracker. For that, let $\{(\hat{P}^i, \hat{w}^i)\}_{i=0}^{N_o}$ be the set of trackers used in the previous frame and $\{(P_*^i, w_*^i)\}_{i=0}^{N_o}$, $(P_*^i, w_*^i) = \arg\max_{(P_j^i, w_j^i) \in \mathcal{P}_i^{t+1}} w_j^i$ be the current best ones. The first candidate trackers for the current frame are given by the set $\mathcal{F}_0^t = \{(P^i, w^i)\}$, where:

$$(P^i, w^i) = \begin{cases} (P_*^i, w_*^i), & \text{if } w_*^i > \tau_b \hat{w}^i \\ (\hat{P}^i, \hat{w}^i), & \text{otherwise,} \end{cases} \tag{11}$$

and τ_b is a given threshold for changing trackers. We now choose as the final trackers every P^j that does not overlap any other P^k by: $\mathcal{F}_1^t = \{(P^j, w^j) : d(s_p(P^j), s_p(P^k)) > \tau_{d_d}, \forall (P^k, w^k) \in \mathcal{F}_0^t, j \neq k\}$, where τ_{d_d} represents a given distance threshold for different objects.

We chose to not include overlapping trackers 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, we propose another approach which tries to find a configuration where each tracker is associated with a different object.

Let $\mathcal{M} = \{m\}$ be the set of indices of the objects whose trackers were not included in $\mathcal{F}_1^t$. Let also $\Omega_P = \{\omega^m = (|\mathcal{P}_m^{t+1}|, w_*^m, \mathcal{P}_m^{t+1})\}$ be a set of triplets containing the cardinality and best temporal weight of each set of trackers $\mathcal{P}_m^{t+1}$. Assume that $\Omega_S = (\omega^m : \omega^m \in \Omega_P)$ is a sequencing of Ω_P where the elements are ordered first in increasing order of $|\mathcal{P}_m^{t+1}|$ 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 their respective weights.

We choose the best tracker for each overlapping object m by iteratively following the order given by Ω_S. Therefore, we start by choosing the best trackers from the objects that 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:

$$\mathcal{F}_{n+2}^t = \begin{cases} \text{if } \exists (P_k^m, w_k^m) \in \mathcal{P}_m^{t+1} : w_k^m \geq \tau_o w^m \text{ or} \\ \quad d(s_p(P_k^m), s_p(P_l^a)) \geq \tau_{d_d}, \forall (P_l^a, w_l^a) \in \mathcal{F}_{n+1}^t \\ \text{then } \mathcal{F}_{n+1}^t \cup (P_k^m, w_k^m) \\ \text{otherwise, } \mathcal{F}_{n+1}^t \cup (P^m, w^m) \end{cases} \tag{12}$$

where $0 \leq n < |\mathcal{M}|$ represents the iteration index, (P^m, w^m) is as defined in Eq. 11 and τ_o is a given constant which represents a score weight threshold for changing trackers after overlapping. In other words, Eq. 12 states that a tracker P_k^i 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 $(P^i, w^i) \in \mathcal{F}_{|\mathcal{M}|+1}^t$ to estimate the position of each object at time t, which is obtained by $s_p(P^i)$.

Table 1. 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	$\tau_{d_d} = 25$
remove candidate score threshold	$\tau_s = 0.18$
threshold for changing to a better tracker	$\tau_b = 1.25$
threshold for changing after overlapping	$\tau_o = 0.75$
old temporal weight factor	$\alpha = 0.75$
feature weight	$\beta = 0.4$

7. Experimental results

7.1. Implementation details

The software was developed in Python with the OpenCV library[1]. We track the objects using color histograms as proposed by Pérez *et al.* [17]. The dynamics of the particles were simulated by a random walk based on a Gaussian distribution $\mathcal{N}(0, 10)$. We show in Table 1 the empirically chosen parameters used for all the experiments.

The performance of our approach was verified by performing tests on two datasets. The first one was composed of table tennis doubles matches obtained from Youtube. This dataset contains seven videos with 6756 frames in total. The second one was built from videos from the ACASVA dataset [4]. We selected three videos of badminton matches from the Olympic games in London, 2012, which contained 5766 frames in total. All the videos were encoded in 854×480 at 30 FPS.

The tests were performed on a computer equipped with an Intel® Core™ i5 3.6GHz processor and 8GB RAM. The observed runtime of our method during tests was around 2 and 3 FPS. Notice that this value corresponds to the code without any parallelization. Both the particle filter and the evaluations of the graphs are highly parallelizable and thus, could be greatly optimized if implemented on GPU.

7.2. Evaluation measurements

We evaluate the tracking results using three 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

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

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 $\sum_i^{N_f} N_o^i$.

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 CERR $= \sum_j d(c(GB_j^i), c(EB_j^i))$.

The second measurement is the hit detection ratio (HITR). As in [19], we consider that a detection is 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 we have HITR $= \sum_j I(c(GB_j^i), EB_j^i)$.

The third one 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. The reason for using this measurement is that the appearances 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.

7.3. Performance evaluation

We tested our approach on table tennis doubles matches videos obtained from Youtube and badminton matches from the ACASVA [4] dataset. 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.

We compare our method with two state of the art trackers. The first one is SPOT [24], which tracks multiple objects using HOG detectors and structural models. The second one is STRUCK [8], a single object tracker that uses a structured output SVM for estimating the object position at the next step. We used the original implementations as provided by the respective authors. The parameters were also kept the same, except that, for STRUCK, we used histogram features, that provided better results.

The task in these videos was to track all the four players and the table or the net. 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 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 M_c was chosen to use the table or the net as a reference to generate 10 candidates for each of the four players. The adjacency matrix considered the relations between players of the same team, as well as their relation to the table or net.

Table 2. Observed results on both datasets.

Dataset	Method	CERR	HITR	HITT
Table tennis	PF + Graph	**63**	**0.62**	**0.75**
	PF	107	0.55	0.60
	SPOT	83	0.39	0.44
	STRUCK	159	0.41	0.42
Badminton	PF + Graph	**50**	**0.57**	**0.77**
	PF	58	0.56	0.65
	SPOT	59	0.32	0.37
	STRUCK	74	0.55	0.60

The results presented in Table 2 correspond to the average of all the videos weighted by their respective number of frames. As pointed by all the measurements, the use of graphs clearly improves the tracking results in both datasets. The CERR and HITR for the badminton videos do not show significant improvement. However, this is mainly due to the fact that sometimes the bounding boxes for players of the same team are swapped, as previously mentioned in Section 7.2. In such a situation, the results for CERR and HITR will be worse than tracking a single player with two different trackers, which usually happens when using individual trackers. As it is further evidenced by the observed HITT, the use of graphs successfully tracks both players more consistently than the other trackers. Besides, even considering the problem in CERR and HITR, the results of our approach are still better nonetheless.

Figure 4 shows some situations illustrating where the graphs are essential to avoid tracking errors. As it can be seen from the pictures, when two players overlap, individual trackers may lose the target due to the appearance ambiguity. However, the use of the graphs allows our method to recover the track successfully. Camera cut, causing abrupt motion can also greatly affect the results if the targets are not found again. But as shown by the results, the proposed approach is also able to continue tracking even after such events. In Fig. 5 a plot of the observed CERR and HITT for one representative video including overlapping and camera cuts from the ACASVA dataset is presented. As it is evidenced by the charts, the proposed approach using graphs present the best overall results. When the objects are clear in the scene and there are no situations of abrupt motion, STRUCK usually presents the best results, as pointed by the beginning of the CERR chart. However, after the tracker is lost, it is not able to recover the target anymore, resulting in a high error afterwards. SPOT tracker is not so affected by

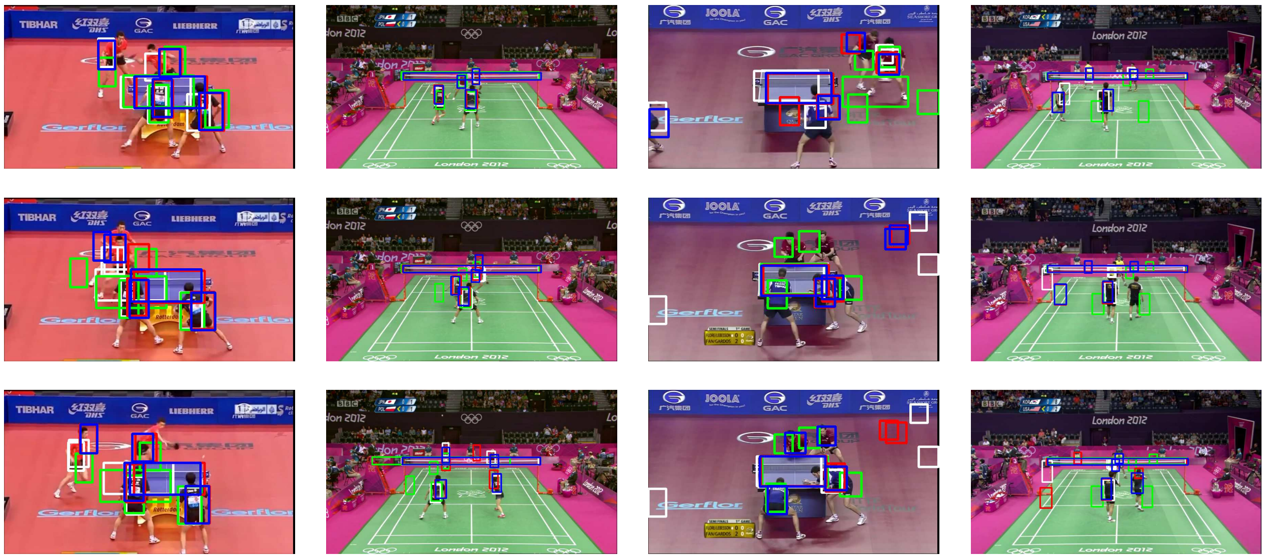
(a) (b) (c) (d)

Figure 4. Tracking results for both dataset. Each method is represented by a different color. Blue: ours, red: particle filter, green: SPOT, white: STRUCK. Columns (a) and (b) show results after a momentary occlusion, while columns (c) and (d) show situations of camera cut. The first row represents the initial situation, in the second row, the camera cut or occlusion happens and then, the third row shows the results for both approaches.

abrupt motion, as it uses a sliding window detector at every frame. However, it did not present very good results in our tests. One possible reason is that the initial annotations do not provide good HOG descriptors and thus, the learned object models may not be very discriminative.

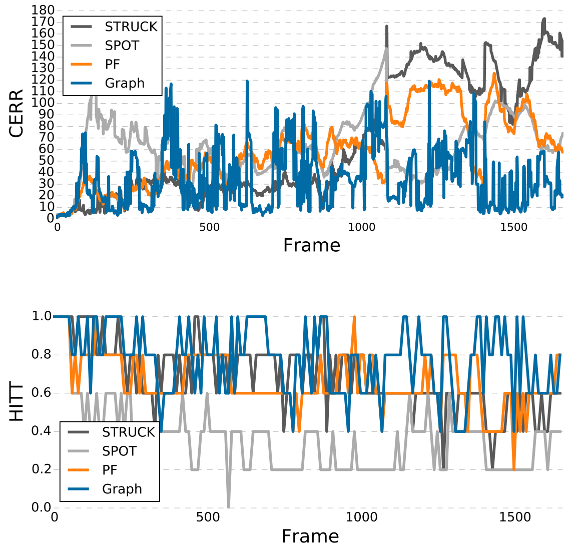

Figure 5. Center error and hit team ratio for one video of the ACASVA dataset including object overlapping and camera cuts. The HITT points were sampled periodically at every 10th observation for more clarity.

8. Conclusion

We proposed a graph based approach to explore the structural information of the scene and use it to improve tracking of multiple objects in videos. Our method is able to both use the information to generate new likely target locations and also to evaluate each tracker. This allows us to improve tracking by recovering the targets after they are lost. As evidenced by the experimental results, this approach coupled with particle filter successfully improves the results of tracking on structured videos. Besides, the structural information does not depend on the particle filter, and could be coupled with any other single object tracking method to potentially improve its results.

As future work we plan to make the method more self adaptive by automatically choosing some parameters. For example, the number of generated candidates can be automatically chosen based on the confidence of both the reference and the object itself. Another possibility is, when using particles, to automatically choose the best spread by adapting the size of the cloud at each location.

Acknowledgements

We would like to thank São Paulo Research Foundation (FAPESP) (grants #2011/50761-2, #2012/09741-0 and #2013/08258-7), CNRS/FAPESP 227332, CAPES, CNPq and NAP eScience - PRP - USP.

References

[1] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1820–1833, 2011.

[2] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *IEEE International Conference on Computer Vision*, pages 25–32, 2013.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

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

[5] A. Doulamis. Dynamic tracking re-adjustment: a method for automatic tracking recovery in complex visual environments. *Multimedia Tools and Applications*, 50(1):49–73, 2010.

[6] E. Erdem, S. Dubuisson, and I. Bloch. Fragments based tracking with adaptive cue integration. *Computer Vision and Image Understanding*, 116(7):827–841, 2012.

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

[8] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270, 2011.

[9] W. Hu, W. Li, X. Zhang, and S. 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, 2015.

[10] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

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

[12] J. Kwon and K. M. Lee. Tracking of abrupt motion using Wang-Landau Monte Carlo estimation. In *European Conference on Computer Vision*, pages 387–400. Springer, 2008.

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

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[15] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2259–2272, 2011.

[16] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*, pages 28–39. Springer, 2004.

[17] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, pages 661–675. Springer, 2002.

[18] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1815–1821, 2012.

[19] Y. Su, Q. Zhao, L. Zhao, and D. Gu. Abrupt motion tracking using a visual saliency embedded particle filter. *Pattern Recognition*, 47(5):1826–1834, 2014.

[20] S. Tang, M. Andriluka, and B. Schiele. Detection and tracking of occluded people. *International Journal of Computer Vision*, 110(1):58–69, 2014.

[21] N. Widynski, S. Dubuisson, and I. Bloch. Fuzzy spatial constraints and ranked partitioned sampling approach for multiple object tracking. *Computer Vision and Image Understanding*, 116(10):1076–1094, 2012.

[22] J. Xing, H. Ai, L. Liu, and S. Lao. Multiple player tracking in sports video: A dual-mode two-way bayesian inference approach with progressive observation modeling. *Image Processing, IEEE Transactions on*, 20(6):1652–1667, 2011.

[23] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *European Conference on Computer Vision*, pages 864–877. Springer, 2012.

[24] L. Zhang and L. J. van der Maaten. Preserving structure in model-free tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 36(4):756–769, 2014.

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

[26] X. Zhou and Y. Lu. Abrupt motion tracking via adaptive stochastic approximation monte carlo sampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1847–1854, 2010.