Proceedings of the IEEE ICRA 2009
Workshop on People Detection and Tracking
Kobe, Japan, May 2009

Multi-target Tracking on a Large Scale: Experiences from Football Player Tracking

J. Sullivan, P. Nillius and Stefan Carlsson,
Royal Institute of Technology,
Stockholm, Sweden.

Abstract

Multi-target tracking requires locating *the targets and* labeling *their identities. The latter is a challenge when many targets, with indistinct appearances, frequently occlude one another, as in football and surveillance tracking. We present an approach to solving this labeling problem.*

When isolated, a target can be tracked and its identity maintained. While, if targets interact this is not always the case. We build a track graph which denotes when targets are isolated and describes how they interact. Measures of similarity between isolated tracks are defined. The goal is to associate the identities of the isolated tracks, by exploiting the graph constraints and similarity measures.

We formulate this as a Bayesian network inference problem, allowing us to use standard message propagation to find the most probable set of paths in an efficient way. The high complexity inevitable in large problems is gracefully reduced by removing dependency links between tracks. We apply the method to a 10 min sequence of an international football game and compare results to ground truth.

1. Introduction

A multi-target tracking system capable of analyzing hours of footage reliably and robustly could potentially help automate many useful applications. There are numerous situations involving people/objects moving and interacting in a particular domain where the tracks of the targets over time provide a rich source of information for analysis of behavior. Such domains include - traffic-pedestrian junctions, travelers at airports, insect/animal tracking and team games.

However, automatic visual multi-target tracking in such domains with frequent interactions is a challenging problem (even when only considering instances with favorable viewing conditions). Given long enough sequences, situations will arise where it is not possible to reliably maintain a target's identity, when it occludes and/or is occluded by other targets, using continuity of appearance or motion alone. Some form of identity re-initialization is required when the interacting targets separate. This re-initialization can take the form of linking tracks before and after the interaction based on matching certain properties of the tracks involved. This in essence is the approach taken in this paper.

We see multi-target tracking as a two-stage process, when there are no real-time constraints. Initially targets are detected and tracked using background subtraction and continuity of motion constraints. When two or more targets meet and cannot be disambiguated a new track is formed and follows this target group. The process is repeated for all targets throughout the sequence. The result is a *track graph* with the different tracks as nodes and edges denoting how the tracks split and merge into new tracks.

In the second stage we try to find each target's path through the graph. This is achieved by exploiting the constraints imposed by the graph structure and by the feature vectors extracted to describe the appearance (e.g. image intensity, gait patterns) of each track. We view this as an inference problem where we want to find the most likely set of paths for the targets given the appearance of the tracks. This can be solved efficiently using Bayesian network inference.

For long sequences, with many targets, finding the global optimum of the resulting posterior becomes intractable due to the combinatorial explosion that occurs with the numerous split and merge situations. We solve this by reducing the dependencies between the tracks. In effect it means that similarities between tracks are only used for tracks within a certain time window. The size of this time window can be set dynamically to meet set criteria for complexity and memory use.

Over the last couple of years, many algorithms and results have been presented [7, 4] with regard to the problem of multiple object tracking. Prevalent are algorithms based on kalman filtering [12, 6], advanced techniques of particle filtering [11, 10, 9, 3] and multiple-hypothesis trackers [4]. The quality of the results presented though improving have yet to be shown working robustly on long sequences (>30 secs). Therefore, one of our major contributions is that we evaluate the performance of our method on a continuous 10

minute clip of an international football match and demonstrate its viability in solving large scale problems. The results obtained are promising.

1.1. Paper Overview

The paper is organized as follows. Section 2.1 provides a more detailed review of the *track graph*, mentioned in the introduction, the assumed starting point of our target linking algorithm. Section 2.2 describes the problem we wish to solve, of linking the identities of the nodes in our track graph. In Section 2.3 the solution space, imposed by the track graph, is defined and parameterized. Section 2.4 states the problem as an inference problem and shows how Bayesian network inference can be used to find the solution. For large problems containing thousands of nodes it is necessary to find an approximate solution. Section 2.4.1 discusses how this can be done by assuming independence between nodes distant in time. Section 3 reports on applying our method to football tracking. The experimental setup is described, as well as a brief review of how the track graph is constructed. The results of the path finding are then presented. To finish conclusions are made focusing on the quality of the results obtained and upon the scalability and generic nature of the solution put forward in the paper. Also discussed are possible improvements and future avenues of research involving combining unsupervised clustering and our path finding algorithm to provide a complete solution to the labeling problem.

2. Linking Identities in the Track Graph

2.1. Preliminaries

The theory in this paper assumes we have access to a track graph summarizing the interactions that occur between the targets in the sequence being analyzed. Therefore, before proceeding further we must introduce more formally the concept of the *track graph*. Each node in the graph represents a track. A track is a temporal sequence of image regions, one per frame (see figure 1). Each region corresponds to the spatial extent of one or more targets. During a track neither the number of targets it represents changes nor do the identities of these targets. The edges in the graph indicate when

- the targets from separate tracks merge (due to partial occlusion) to begin a new track or
- the targets in a track separate/split to begin several new tracks, each with fewer targets than the parent one.

Figure 2 displays a small example of such a *track graph*. The white nodes indicate tracks of a single target and grey those representing multiple targets.

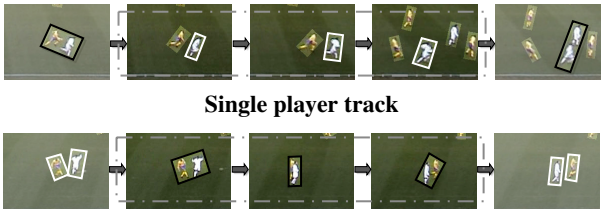


Figure 1. **Single and multiple target tracks** from a football game. The top row shows a single target track, shown in white. The bottom row is a multiple target track, shown in black. Tracks are sandwiched between interactions with other tracks. During a track the number of targets involved and their identities remain fixed.

There are, of course, numerous possible ways to obtain this graph [1]. For now though this issue is set aside and assumed to have been solved, however, we revisit it in section 3 while reviewing the methods put forward in [1].

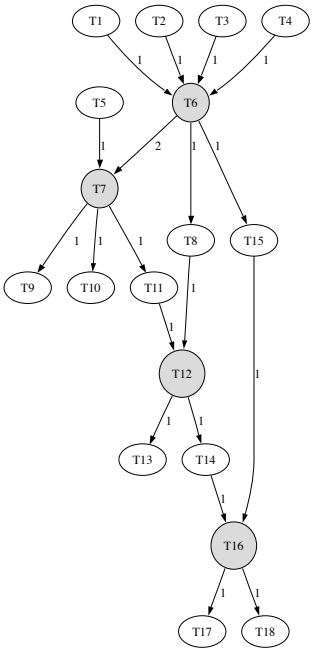


Figure 2. **An example of a simple track graph.** Each node corresponds to a track: white - an individual target, grey - multiple targets. The edges of the graph are directed corresponding to the temporal constraints and indicate when tracks merge or separate.

2.2. General Approach

On top of the track graph it is assumed that there are feature vectors measured from each single target track. These feature vectors can consist of elements such as color, shape, position and velocity. Using the feature vectors to compare tracks and the constraints imposed by the track graph we find the most likely configuration of paths. To do this we parameterize the solution space imposed by the track graph

so that we have a state vector that can represent all possible configuration of paths through the track graph. The paths are then found by inferring the state given the features of the tracks.

2.3. The solution space

Each targets path through the graph is known when it is known exactly how the incoming targets are distributed into the outgoing tracks when a track is split up. Therefore, we will represent the solution space by viewing the splits as track switches. Each split/switch has a state variable representing how the targets are distributed into the outgoing tracks.

When defining the state variables of the split nodes care must be taken so that the state space becomes as compact as possible. Each set of values of the state variables should correspond to one unique solution. This can be done in the following way.

Let N be the number of incoming targets for a particular split node. It doesn't matter how many incoming tracks the node has, so we can assume it has one single incoming track, as in Figure 3. Moreover, let the node have m outgoing tracks, each having $n_j, j = 1, ..., m$ targets also summing up to a total of N targets.

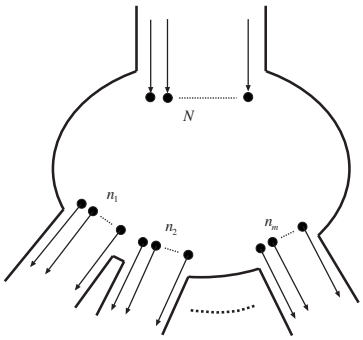


Figure 3. The size of the state variable of a split node is determined by the number of outgoing tracks and how many targets, n_i are in each track i.

The number of ways to distribute the targets into the outgoing tracks can be found through a process of iteratively selecting the targets to go into a track. Each track i selects n_i targets from the targets not yet selected. In this way each track can select its targets in $\binom{N-\sum_{j=1}^{i-1} n_j}{n_i}$ different ways. Hence, the total number of states of the split node is

$$\prod_{i=1}^{m} \binom{N - \sum_{j=1}^{i-1} n_j}{n_i}. \tag{1}$$

Note that the incoming targets is considered an ordered set while the selection in the process above is unordered. To define the ordering in the outgoing tracks we let them keep their relative ordering within each outgoing track, as illustrated with the example in Figure 4. This avoids getting redundant states, which would happen when two targets in the same track are switched in one split node and then switched back when entering a subsequent split node, which is equivalent to the targets not getting switched in either node.

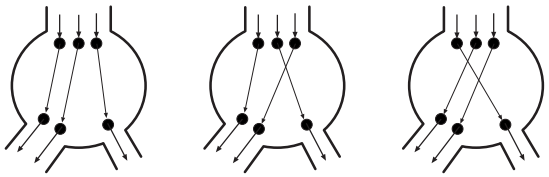


Figure 4. All three states of a node splitting three incoming targets into one double and one single target track.

The variable representing the split state of the node is a product of the "selection states" of the outgoing tracks of the node. Getting the selection states from the node state is a matter of using integer divisions and the modulo operator.

Let each split node, T_i, have a discrete state variable S_i which represents exactly how the targets are split into the outgoing tracks. The number of values S_i can take is given by (1). Moreover, let

$$\mathcal{S} = \{S_i; T_i \text{ is a split node}\} \tag{2}$$

be the set of state variables for all the split nodes. Then $\mathcal{S}$ can represent all possible solutions of paths given the track graph. There is also a one-to-one mapping between the values of the state variables and the solution space.

2.3.1 Computing the number of targets in the tracks

This section described how the number of targets in each link is computed. Let l_{ij} be the link count, i.e. the number of targets in the link between tracks T_i and T_j.

1. Let all link counts be undefined, $l_{ij} = 0$.
2. Set link counts to all single tracks to one,
   $l_{ij} = 1, \quad T_i$ connected to T_j and (T_i or T_j are single track)
3. For nodes with all links but one defined, set the undefined link so that number of targets in equals number of targets out.
4. . Repeat 3 until no more links are updated.

The above procedure will propagate the number of targets through the track graph. In practice there will be inconsistencies and some links will be left undefined. These parts of the graph are left unresolved at this point, but there are several possibilities how they could be handled in the future, e.g. by better modeling or by merging nodes.

2.4. The Inference Problem

Let each single track i have feature vector A_i and let

$$\mathcal{A} = \{A_i; T_i \text{ is a single target track}\} \tag{3}$$

be the set of all feature vectors.

We would like to infer the paths given the measurements using the max posterior estimate,

$$\hat{\mathcal{S}} = \underset{\mathcal{S}}{\operatorname{argmax}}\, P(\mathcal{S}|\mathcal{A}). \tag{4}$$

As usual, Bayes formula can be used to instead maximize the product of the prior and the likelihood function.

$$P(\mathcal{S}|\mathcal{A}) \propto P(\mathcal{A}|\mathcal{S})P(\mathcal{S}) \tag{5}$$

The split node state variables $\mathcal{S}$ are local and causally independent. The measurements on the other hand, depend on the state variables in the sense that the values of the state variables define the targets' paths. Tracks on the same path contain the same target, hence their measurements are dependent. Measurements from different targets are assumed to be independent.

We note that every path ends at a tail node, i.e. a node with no outgoing links. The tail nodes are used as representatives for the paths. Let

$$\mathcal{A}_{tails} = \{A_i; T_i \text{ is a tail node}\} \tag{6}$$

be the set of tail node features. Further, let

$$\text{path}(A_i, s) = \{A_j; T_j \text{ are on the same path as } T_i \text{ given } \mathcal{S} = s\} \tag{7}$$

be the feature vectors of all tracks on the path defined by the state s and leading to the track T_i. Then the likelihood function can be factorized as

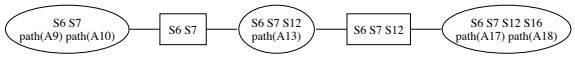
$$P(\mathcal{A}|\mathcal{S}) = \prod_{A_i \in \mathcal{A}_{tails}} P(\text{path}(A_i, s)|\mathcal{S} = s). \tag{8}$$

The dependencies between state variables and feature vectors can be viewed in a Bayesian network showing the causal dependencies between the nodes. The track graph in Figure 2 has the Bayesian network in Figure 5.

Inference on a Bayesian network can be done efficiently using message propagation. We use the junction tree algorithm, [2, 5]. This algorithm creates a secondary structure, the junction tree, consisting of cliques and sepsets. The cliques are the smallest sets of variables on which the inference can be solved using local computations and message propagation. The sepsets show the common variables between neighboring cliques which are the margins that is computed when performing the message propagation.

The most probable configuration of a Bayes net can be found by using max-marginalization in the message propagation. We have used Kevin Murphy's implementation in the Bayes Net Toolbox for Matlab [8]. All we have to provide are the likelihoods for the cliques and the priors.

(a)



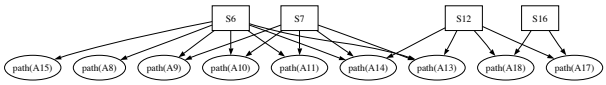
(b)

Figure 5. Bayesian network (a) and junction tree (b) for the track graph in Figure 2

2.4.1 Reducing complexity

Message propagation will solve the inference problem efficiently and it will give a globally optimal solution (under the assumptions). For large problems though, there will be a combinatorial explosion.

To apply this approach to large scale problems it is necessary to reduce the complexity. We do this by dropping the dependencies between feature vectors and split nodes that are more than a certain number of links away. The effect is that we optimize shorter but overlapping paths in the graph. A Bayes net for our track graph can look like the Bayes net in Figure 6. As can be seen, paths to all single target tracks are taken into account, but the levels of dependencies have been reduced. In this case the paths to T_{17} and T_{18} have dropped the dependencies to the split nodes T_6 and T_7. In effect this means that the tracks T_{17} and T_{18} will not be compared with tracks above T_6 and T_7. It also mean that the complexity have been reduced and for larger problems this is crucial.



(a)

(b)

Figure 6. To reduce the complexity we remove dependencies between tracks that are distant in time. The result is that the algorithm optimizes shorter local paths that are overlapping. The Bayesian network (a) now has paths to all single track nodes with ancestors and A_{17} and A_{18} does not depend on S_6 and S_7 anymore. (b) shows the resulting junction tree.

2.4.2 Building the Bayes Net

The Bayes net is built through the following procedure.

1. For each split node T_i in the track graph, add the state variable S_i.
2. For each single track node T_i with ancestors:

(a) Add an observed node representing all paths leading to A_i.

(b) In a breadth-first fashion collect split nodes that are ancestors to T_i until the product of the split nodes' state sizes (the clique size) have reached a set limit.

(c) Connect the collected split nodes' state variables to the new observed node.

2.4.3 Computing the Conditional Probability Tables

For the inference algorithm we need to provide the probability distributions for each clique in the junction tree. The probability distribution is the product of the prior and the likelihood. In this paper we use a flat prior.

The conditional probability tables are computed as described below.

1. For each observed node in the BN representing paths leading to A_i:

   (a) Get the parents $\mathcal{S}_{pa}$

   (b) For each combined state s_{pa} of the variables in $\mathcal{S}_{pa}$ (the number of combined states is the product of the size of each state variable $S \in \mathcal{S}_{pa}$):

      i. Compute the likelihood $P(\text{path}(A_i, s_{pa})|\mathcal{S}_{pa} = s_{pa})$

Basically, at this step we go through all paths in the local graph around the split nodes associated with parents, $\mathcal{S}_{pa}$, of the observed node. In our example with the Bayes net in Figure 6, when computing the likelihoods of for paths leading to T_{17} we go through all possible paths from T_{11}, T_8 and T_{15} to T_{17}.

2.4.4 Computing the Likelihoods

The likelihoods in this case are the probability density for the measurements given that they all are from the same model. This model is considered unknown, but if there is a set of models, $\mathcal{M}$, that will make the measurements independent we can compute the likelihood in the following way.

$$\begin{aligned} &P(\text{path}(A_i, s_{pa})|\mathcal{S}_{pa} = s_{pa}) \\ &\quad = \int_{M \in \mathcal{M}} P(\text{path}(A_i, s_{pa})|\mathcal{S}_{pa} = s_{pa}, M)P(M) \\ &\quad = \int_{M \in \mathcal{M}} \prod_{A_j \in \text{path}(A_i, s_{pa})} P(A_j|M)P(M) \end{aligned} \tag{9}$$

Sometimes it is easier to find a pairwise measure how likely two tracks contain the same target. These can be used in such a way that all pairwise similarity measure are used exactly once.

$$\begin{aligned} &P(\text{path}(A_i, s_{pa})|\mathcal{S}_{pa} = s_{pa}) \\ &\quad \approx \prod_{A_j \in \text{path}(A_i, s_{pa}) \setminus A_i} P(A_i, A_j) \end{aligned} \tag{10}$$

Since we have overlapping paths, the similarity measure between the other members in the path will be used in the cliques for paths originating from the those members.

3. Football Tracking

At this stage we focus on applying our method to the problem of tracking football players in a competitive professional game. Football occurs in a structured closed environment where it is relatively easy to perform reliable effective image processing, but on the other hand provides many complicated and challenging motions and interactions between players. It is a happy compromise between analyzing generic video sequences and those engineered in the lab.

Here we review an approach to constructing the track graph and to defining a measure of similarity between single player tracks. This takes us to the assumed starting point of our Bayesian inference problem.

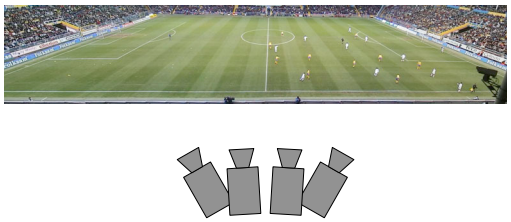
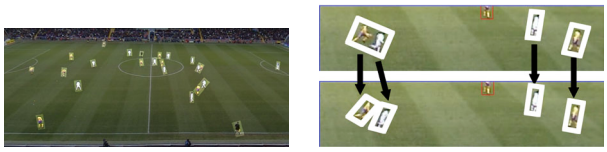


Figure 7. Multi-camera system used to capture a stationary, high-resolution video covering a large area.

3.1. Extracting the Track Graph

Figure 7 displays the multi-camera system used to provide a high resolution, wide-field of view video of the football game. The resulting video allows all the players to be seen at all times. As the cameras are stationary it is possible to perform reliable and accurate background subtraction to highlight the positions of the targets in each image (see figure 8 (a)). Temporal analysis of the foreground regions found at each frame, matching the regions in one frame to those in the next (figure 8 (b)), allows the identification of the single and multiple player tracks and the interactions between them. There are two teams wearing two distinctly colored uniforms, as well as the officials. It is possible to assign each single track to one of these three categories based on simple matching of exemplar rgb histograms. Figure 12 displays a portion of the track graph obtained from examining our football clip in this manner. For the interested



(a) foreground regions (b) matched regions

Figure 8. **(a)** The foreground regions found by background subtraction. **(b)** An example of matching the found regions between one frame and the next. Note that one of the examples is a split, marking the end of one track and the beginning of two more.

reader, the complete graph is included as part of the supplementary material. We manually obtained the ground truth for the identity of the team A single target tracks. The temporal extent of each player's single tracks are displayed in figure 9. We would like to obtain this figure automatically.

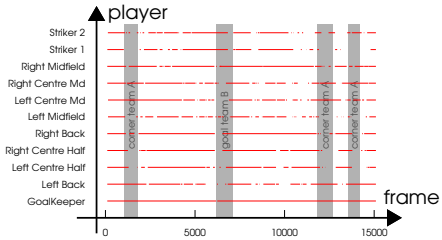


Figure 9. The temporal extent of the team A single player tracks for the ten minute clip examined. Each line corresponds to a single player track. The shaded areas display when the major congestion events occur.

3.2. Similarity Measure Between Player Tracks

We now define a measure of similarity between every pair of single player tracks from the same team. In football a player's identity is frequently revealed by his position relative to his teammates. Most obviously the goal-keeper is always behind all his teammates. We exploit this simple idea. For each single player track we build a histogram summarizing the player's position relative to his teammates for the duration of the track. Each bin of the histogram corresponds to a particular configuration of teammates to the left, right, behind and in front of the player. There a fixed number of such configurations as there are eleven players on a team, see [1] for details. Let I_{s}^{ij} denote the similarity score between two tracks based on comparing their relative spatial position histograms. This measure is particularly effective for matching tracks of long duration. Such tracks, generally, occur when the team is in typical formations and the players are in set positions within these formations. However, many of the shorter tracks occur when the team is in transition between typical team formations rendering the relative spatial position information less effective. To compensate for this deficiency we define temporally local measures.

Two tracks T_i and T_j are temporally close if the end of T_i occurs before and within t frames of the start of T_j. If t is small enough and T_i and T_j represent the same player, it is reasonable to assume continuity of appearance and motion. On this basis we construct appearance and motion based measures between temporally close track pairs. The appearance measure relies on cross-correlating the appropriate spatio-temporal volumes at the ends involved. This measure is denoted by I_{i}^{ij}. The velocity of the targets at the ends of these tracks is also estimated. Given these velocities and the final position of T_i, an estimate of the start position of T_j is obtained. The difference between this estimate and actual value is then used as our motion measure - I_{d}^{ij}. After appropriate rescaling of the different I's a combined similarity matrix is produced:

$$I^{ij} = \begin{cases} (1-2\alpha)I_{\mathrm{s}}^{ij} + \alpha I_{\mathrm{i}}^{ij} + \alpha I_{\mathrm{d}}^{ij} & \text{if } T_i, T_j \text{ temporally close.} \\ I_{\mathrm{s}}^{ij} & \text{otherwise} \end{cases} \tag{11}$$

with $0 \leq \alpha \leq 1$, see figure 10. The similarity scores are then converted into the appropriate form, (eqn 10), by setting $P(A_i, A_j) = \exp(-\lambda I^{ij})$, with $\lambda > 0$.

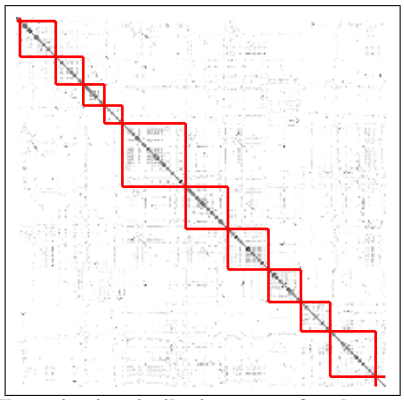


Figure 10. **The pairwise similarity scores for the team A tracks**. Black indicates high similarity and white low similarity. The rows of the matrix have been re-ordered to group tracks of the same identity together and to reveal the structure within the matrix. The red lines denote the sub-blocks of constant identity.

3.3. Results

We are now almost ready to present the identity linking results. Before running the inference procedure, the number of targets in each link is computed. In many parts of our football clip graph, it is not possible to determine the number of targets in each link and sometimes there are inconsistencies (the number of input and output targets at an interaction are unequal). Presently our theory cannot handle such situations, thus these parts of the graph are left unsolved. Accordingly, the Bayes net is divided into parts, which are analyzed separately. For our football graph 14 separate parts are isolated. In figure 12(b) we can see an explicit demonstration of solutions found for some split nodes.

For a clearer picture of the quality of the results obtained the found paths (for team A players) within largest parts are shown in figure 11. As can be seen, the majority of the tracks are correctly linked.

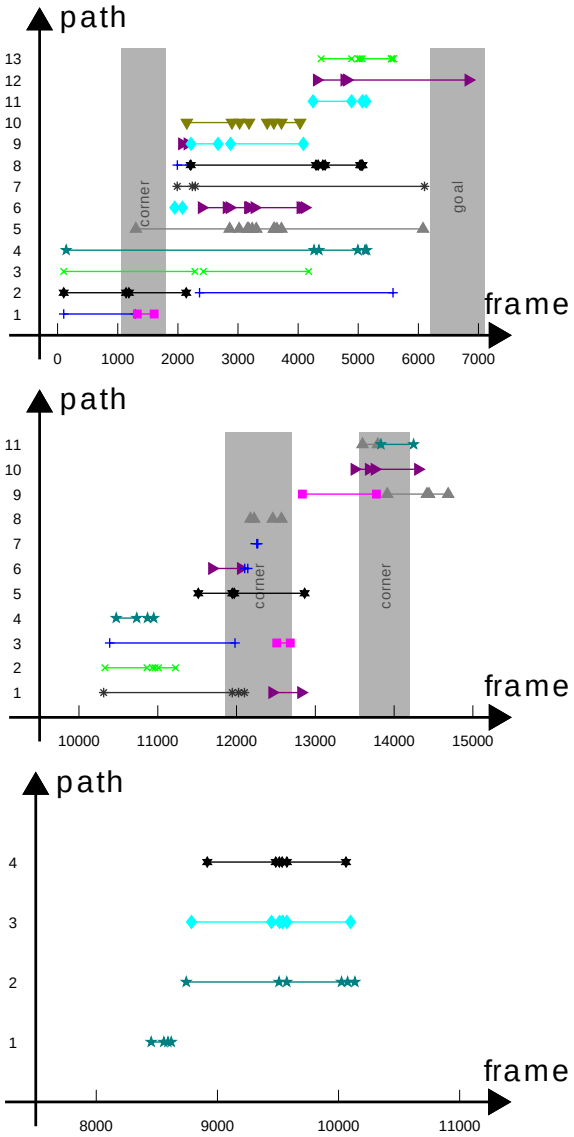


Figure 11. **Estimated paths for the three largest consistent parts of the graph.** Each line represents a single player track and row an estimated path. The color and symbol denote the true identity of the track. Ideally there should only be one color and symbol per row. On several occasions a target's trajectory is split into several paths. This is caused by links in a part having an undetermined number of targets. The rest of the parts contain a similar number of tracks and quality of results to the bottom graph.

To summarize the overall results - 85% (out of 73) of the connections considered are correctly resolved. However, not all decisions made at each split node have the same degree of confidence associated with them. Fortunately, as we are working with probabilities and within a Bayesian framework we can compute the absolute probability for each possible resolution of a particular split node. Comparing the relative value of the most probable to the next most probable resolution provides a confidence level of our estimate. By only including estimates that are certain, we can eliminate some of the connection errors. Figure 13 shows the percentage of correct connections as we remove the less certain split estimates. When 25% of the connections remain we have 100% correct connections.

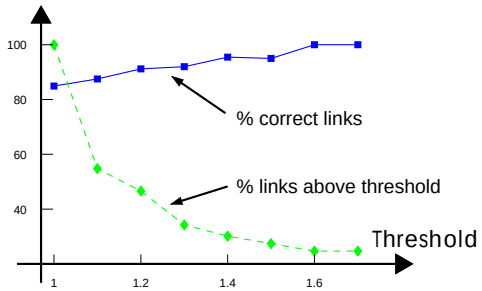
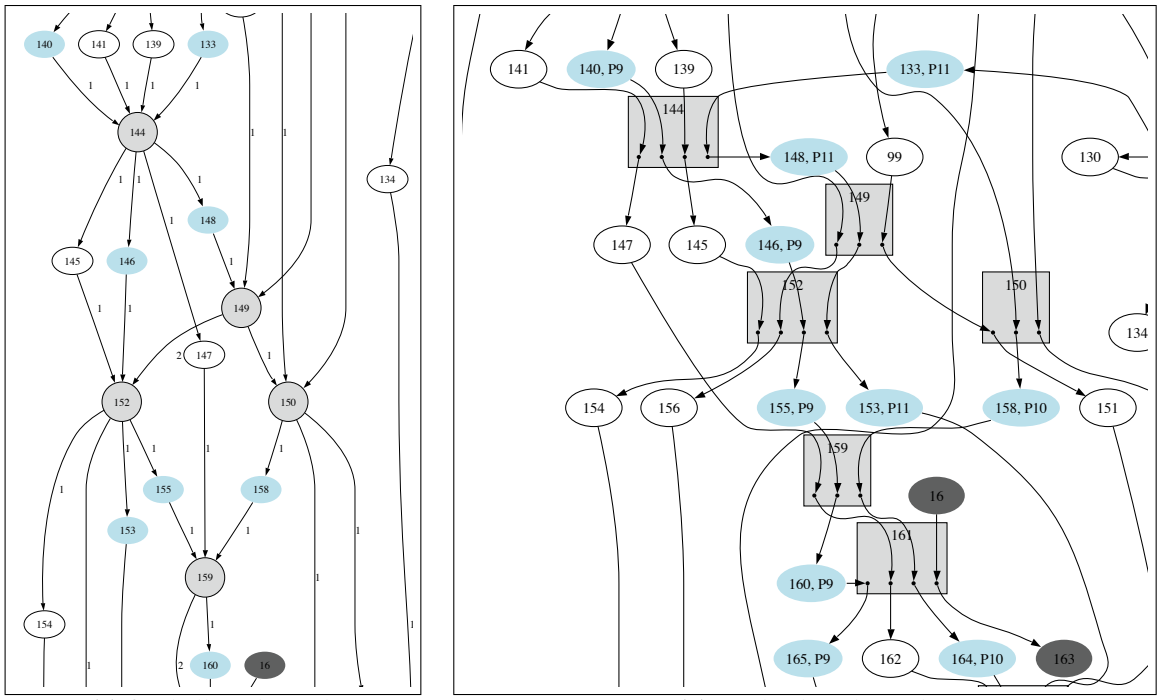


Figure 13. **The percentage of correct connections.** Using the marginal probabilities we can remove uncertain estimates. We threshold on the ratio between the most and second most probable state of a split node. This increases the percentage of correct links, but reduces the number of connections made. At a threshold of 1.6 we make no wrong connections, but only 25% of the connections are left (out of 73).

4. Conclusions and Future Research

When tracking multiple targets over a long period, it is inevitable that inter-target occlusions will occur where it is not possible to immediately link the identities of the targets entering and those exiting the interaction. It is therefore necessary to compare targets over extended periods of time in the attempt to link their identities. In this paper we achieve this by considering a two-stage solution. The first stage involves the construction of a track graph describing the interactions between targets. The second stage, the focus of this paper, exploits the track graph and similarity measurements between the tracks to infer the most likely configuration of paths for all targets. This is achieved by parameterizing the solution space imposed by the track graph and inferring the parameters given the measurements. To make large scale problems computationally feasible we only consider tracks within a certain window of interactions to be explicitly dependent.

Promising results on a challenging (and relatively lengthy) data set are presented. The main limitation to improving the results, presently, are the assumptions concerning the track graph. For problems with many targets interacting frequently it is not realistic to expect that the number of targets in each node can be counted explicitly. Relaxing



(a) football track graph **(b)** resolved track graph

Figure 12. **(a)** This is a small part of the football clip track graph. The node colors correspond to team A (light blue oval), team B (white), referees (dark grey) and multi-target nodes (black). **(b)** The corresponding resolved track graph. The square nodes display how the split nodes have been resolved. Ground truth player numbers can be seen for the team A players.

this assumption, with more sophisticated modeling, would increase the size of the parts of real-world track graphs we could examine. Of course, as in most tracking algorithms, there is room for improvement in the modeling of the appearance of the targets.

In a complementary approach, the identities of single player tracks can be linked by un-supervised clustering using the similarity matrices shown in figure 10. Clustering can be performed without reference to the *track graph*, thus by-passing the computational bottlenecks and inconsistencies in the graph. Reliable results have been obtained when long tracks are included in the clustering process [1]. A fruitful avenue of future research would be to investigate how to optimally combine clustering and the path finding algorithm presented here to obtain a more complete labeling of the player identities.

References

[1] Author. Tracking and Labelling of Interacting Multiple Targets. ECCV06 submission ID 1027. Supplied as additional material eccv06.pdf. 2, 5, 6, 8

[2] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Syst.* Springer, 1999. 4

[3] P. Figueroa, N. Leite, R. Barros, I. Cohen, and G. Medioni. Tracking soccer players using the graph representation. In *ICPR*, pages 787–790, 2004. 1

[4] M. Gelgon, P. Bouthemy, and J. Le Cadre. Recovery of the trajectories of multiple moving objects in an image sequence with a pmht approach. *J. Image & Vision Computing*, 23(1):19–31, 2005. 1

[5] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996. 4

[6] S. Iwase and H. Saito. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *ICPR*, pages 751–754, 2004. 1

[7] Z. Khan, T. Balch, and F. Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *European Conference on Computer Vision*, 2004. 1

[8] K. Murphy. The bayes net toolbox for matlab. In *Computing Science and Statistics*, volume 33, 2001. 4

[9] C. Needham and R. Boyle. Tracking multiple sports players through occlusion, congestion and scale. In *BMVC*, 2001. 1

[10] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004. 1

[11] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *International Conference on Computer Vision*, 2003. 1

[12] M. Xu, J. Orwell, and G. Jones. Tracking football players with multiple cameras. In *IEEE International Conference on Image Processing*, 2004. 1