Multiple interacting targets tracking with application to team sports*

Matej Kristan, Janez Perš,
Matej Perše, Stanislav Kovačič
Faculty of Electrical Engineering
University of Ljubljana, Slovenia
matej.kristan@fe.uni-lj.si

Marta Bon
Faculty of Sports
University of Ljubljana, Slovenija

Abstract

The interest in the field of computer aided analysis of sport events is ever growing and the ability of tracking objects during a sport event has become an elementary task for nearly every sport analysis system. We present in this paper a color based probabilistic tracker that is suitable for tracking players on the playground during a sport game. Since the players are being tracked in their natural environment, and this environment is subjected to certain rules of the game, we use the concept of closed worlds, to model the scene context and thus improve the reliability of tracking.

1 Introduction

Computer aided analysis of human motion is becoming an appealing tool for enhancing the training effect in the domain of multiple player sports. The potential of the computer integration with coaching is vast, since the problems that demand a mass of computation can relatively easily be dealt with using the computer. Tracking all players on the playground during a game and obtaining their trajectories, is thus the cornerstone for nearly every type of analysis that is interesting for the sport experts.

In this paper we address the problem of multiple interacting targets tracking in indoor sport games such as basketball and handball, and present a system for tracking players using a static top-view plan. Since the sports domain can be considered as a semi-controlled environment, we use the closed world terminology presented in [4], and apply it to the case of color based particle filtering. We show how the concept of the closed world assumptions can be used to combine multiple separate trackers to improve the tracking performance.

The remainder of the paper is organized as follows: Section 2 defines the assumptions of the closed world, and Section 3 the engine of the tracker. In Sections 4 to 7, a tracker using closed world assumptions for a single player is presented. Section 8 augments the set of closed world assumptions by a new assumption, and extends the tracker to the case of multiple players. Section 9 describes the experiments for evaluation of the tracker, and in Section 10 we draw some conclusions.

2 The closed world

The *closed world* concept has been introduced to the vision community by Intille and Bobick [4] in 1995. The main premise is that for a given region in space and time, a specific context is adequate to explain that region (i.e. determine all objects within the region). This region is called the closed world and the context is a boundary in space of knowledge, outside of which knowledge is not helpful in solving the tracking problem. Thus, by considering the sport match as a closed world, we define the context as a set of closed world assumptions: *(i) The camera overlooking the playground is static, and positioned such that it's optical axis is approximately perpendicular to the floor. (ii) The playground is bounded, and its model can be calculated. (iii) The illumination is nonuniform in space and time. (iv) The players' textures are known in the beginning, and vary during the game.* An additional closed world assumption will be defined later in the paper.

3 Particle filtering

The use of particle filters in computer vision is a relatively new and popular method for tackling the problem of tracking. In the domain of visual tracking, this approach has been pioneered by the work of Isard and Blake [5] in 1998, and numerous variations and extensions of the method have been presented since. We present here only the basic concept and notations, and refer the interested reader to [1, 3, 5] for more details.

Let $\mathbf{x}_{t-1}$ denote the state of a tracked object at time-step $(t-1)$, let $\mathbf{y}_{t-1}$ be an observation at that state, and let $\mathbf{y}_{1:t-1}$ denote a set of all observations up to a time-step $(t-1)$. From a Bayesian point of view, all of the interesting information about the target's state $\mathbf{x}_{t-1}$ is embodied by its posterior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. The aim of tracking is then to recursively estimate this posterior as the new observations

*The paper appeared in the proceedings of the 4th International Symposium on Image and Signal Processing and Analysis ISPA,September 2005, pp.322-327.

$\mathbf{y}_t$ become available. This process is characterized by two steps: prediction (Eq. 1) and update (Eq. 2).

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \quad (2)$$

The recursion for the posterior thus requires a specification of a dynamical model describing the state evolution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and a model that evaluates the likelihood of any state given the observation $p(\mathbf{y}_t|\mathbf{x}_t)$.

In our implementation, we use the well known CONDENSATION algorithm, where the posterior at time-step $t-1$ is represented by a finite particle set with normalized weights $w_{t-1}^{(i)}$

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \approx \left\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N}, \quad (3)$$

such that all the weights sum to one. At time-step t, the particles are resampled according to their weights in order to obtain an unweighted representation of the posterior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \approx \left\{\tilde{\mathbf{x}}_{t-1}^{(i)}, \frac{1}{N}\right\}_{i=1}^{N}$ and are then simulated according to the dynamical model $p(\mathbf{x}_t^{(i)}|\tilde{\mathbf{x}}_{t-1}^{(i)})$, to obtain a representation of the prediction $p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \approx \left\{\mathbf{x}_t^{(i)}, \frac{1}{N}\right\}_{i=1}^{N}$. Finally, a weight is assigned to each particle according to the likelihood function $w_t^{(i)} = p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$, all weights are normalized, and the posterior at time-step t is approximated by a new particle set $p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$.

4 The Player model

Our aim is to track players on the playground during a sport event, which requires modelling the players themselves, or the regions that contain them. We chose an ellipse to model these regions, since it provides a low-dimensional presentation and is still robust enough to capture different appearances of a player. A state of the tracked player $\mathbf{x}_t \in \mathcal{X}$, with $\mathcal{X}$ denoting the state shape-space, is thus parameterized by an ellipse $\mathbf{x}_t = (x_t, y_t, a_t, b_t)$ with a center at (x_t, y_t), and with parameters a_t and b_t denoting the width and height, respectively.

It is usually argued that the player's aim is to act in an unpredictable fashion to confuse the opponent, which implies that the dynamics should be modelled by a Brownian motion. The player's motion, however, is restricted by his *task* and *laws of physics*; i.e. a player's task is to travel from region *A* to region *B*, and during that traversal the position cannot be changed instantly and arbitrarily due to the effects of the inertia. We therefore define the state evolution model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t-1} + \mathbf{d}_t, \Lambda_t), \quad (4)$$

where $\mathcal{N}(\cdot; \mu, \mathbf{\Sigma})$ is a normal distribution with mean μ and covariance matrix $\mathbf{\Sigma}$ and $\mathbf{d}_t$ is a drift constant at time t for all particles.

The covariance matrix in Eq. (4) is a diagonal matrix

$$\Lambda_t = diag(\sigma_{xy}^2, \sigma_{xy}^2, \sigma_{ab}^2, \sigma_{ab}^2) \cdot H_t^2, \quad (5)$$

where $H_t = \sqrt{a_t^2 + b_t^2}$ is the size of the ellipse belonging to the state $\mathbf{x}_t$.

The time-step constant drift $\mathbf{d}_t$ is estimated on a set of smoothed past estimates of the average states, which are calculated as follows: Let $\mathbf{o}_{t-T:t-1} = \{\mathbf{o}_k\}_{k=t-T}^{t-1}$ denote a set of T past smoothed states $\mathbf{o}_k = (o_{x_k}, o_{y_k}, o_{a_k}, o_{b_k})$ of a tracked target and let $\pi_{t-T:t-1} = \{\pi_{\mathbf{o}_k}\}_{k=t-T}^{t-1}$ denote the set of their weights. Let $\tilde{\mathbf{o}}_t = \mathbf{o}_{t-1} + \mathbf{d}_t$ denote the current prediction on these sets. At time-step t, when the approximation to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ becomes available, an average state $\hat{\mathbf{x}}_t = (\hat{x}_t, \hat{y}_t, \hat{a}_t, \hat{b}_t)$ is calculated as

$$\hat{\mathbf{x}}_t = \sum_{i=1}^{N} w_t^{(i)} \mathbf{x}_t^{(i)}, \quad (6)$$

and the weights $w_{\hat{\mathbf{x}}_t} = p(\mathbf{y}_t|\hat{\mathbf{x}}_t)$ and $w_{\tilde{\mathbf{o}}_t} = p(\mathbf{y}_t|\tilde{\mathbf{o}}_t)$ are obtained via the likelihood function. The new smoothed state $\mathbf{o}_t$ is then calculated as a weighted sum

$$\mathbf{o}_t = \frac{\tilde{\mathbf{o}}_t \cdot w_{\tilde{\mathbf{o}}_t} + \hat{\mathbf{x}}_t \cdot w_{\hat{\mathbf{x}}_t}}{w_{\tilde{\mathbf{o}}_t} + w_{\hat{\mathbf{x}}_t}}, \quad (7)$$

and the corresponding weight is calculated as $\pi_{\mathbf{o}_t} = p(\mathbf{y}_t|\mathbf{o}_t)$.

The new time-constant drift $\mathbf{d}_{t+1}$ is then estimated as a weighted sum of successive past differences between the smoothed average states

$$\mathbf{d}_{t+1} = c_o^{-1} \sum_{k=t-T+1}^{t} (\mathbf{o}_k - \mathbf{o}_{k-1}) G_k(t), \quad (8)$$

where the weights are defined as

$$G_k(t) = \pi_k \pi_{k-1} e^{-\frac{1}{2}\frac{(k-t)^2}{\sigma_o^2}}, \quad (9)$$

and where $c_o = \sum_{k=t-T+1}^{t} G_k(t)$ is the normalization factor. The first two terms in Eq. (9) are the weights of the corresponding smoothed states and the last term is a Gaussian, which is used to assign higher a priori weights to the more recent states. The number of smoothed states in Eq. (8) is for practical applications set to $T = 3\sigma_o$, since the a priori weights of all older smoothed states are negligible .

Assuming that a player can not radically change his or hers direction and velocity within one half of a second, a value for the parameter σ_o was chosen to comply with this time frame. Since all our test sequences are recorded at a frame rate of 25 frames per second, we have chosen this parameter to be $\sigma_o = 4.3$, which means, that in our application only $T = 13$ past smoothed states are considered.

5 The Color cue

Usually, a player is depicted on an image only by a small number of pixels, typically about 10x10, and its shape

changes rapidly. We therefore model the player by a color histogram within an elliptical region. Such an approach has proven on many occasions [2, 6] to be a powerful and robust characterization of an object appearance. To increase robustness to clutter, a weighting kernel as in [2] is employed, which assigns higher weights to the pixels that are closer to the center of an ellipse, and lower weights to those farther away. Furthermore, if some a priori knowledge of which pixels are not likely to belong to the player is available, it should be used in the construction of the histogram; i.e. these pixels should be ignored. An elegant way of discarding the pixels that do not belong to the player is the use of a mask function, which assigns an a priori zero weight to those pixels that are not likely to have been generated by the players and a value one to all the other pixels. A construction of such a mask function will be explained in the following sections.

Let $E = (x, y, a, b)$ be an elliptical region at some state $\mathbf{x}_t = (x, y, a, b)$. The RGB color histogram with B bins $\mathbf{h}_\mathbf{x} = \{h_{i\mathbf{x}}\}_{i=1}^{B}$, sampled within the ellipse E, is then defined as

$$h_{i\mathbf{x}} = f_h \sum_{\mathbf{u}\in E} K(\mathbf{u})M(\mathbf{u})\delta_i(b(\mathbf{u})), \tag{10}$$

where $\mathbf{u} = (x, y)$ denotes a pixel within the elliptical region E, $\delta_i(\cdot)$ is a Kronecker delta function positioned at histogram bin i, $b(\mathbf{u}) \in \{1...B\}$ denotes a histogram bin index associated with the color of a pixel at location $\mathbf{u}$, $K(\cdot)$ is a weighting kernel as in [2] positioned at the center of an ellipse, $M(\mathbf{u})$ is some a priori binary mask function, and f_h is a normalizing constant such that $\sum_{i=1}^{B} h_{i\mathbf{x}} = 1$.

The presence of a player in a given state is evaluated by comparing some candidate histograms to a reference histogram of the tracked player. As a measure of distance between two histograms, say $\mathbf{h}_1$ and $\mathbf{h}_2$, a Bhattacharyya distance was chosen due to its optimality when considering a frequency coded data, and is defined as

$$\varrho(\mathbf{h}_1, \mathbf{h}_2) = 2(1 - \sum\nolimits_i \sqrt{\mathbf{h}_{1i}\mathbf{h}_{2i}}). \tag{11}$$

Since the model of the background can be obtained, we define a measure that also considers some information from the background. Let $\hat{\mathbf{h}}_t$ be a reference histogram of the target, let $\mathbf{x}_t$ be some hypothesized state of the target and let $\mathbf{h}_A$ and $\mathbf{h}_B$ be the histograms at that state, sampled on the current and the background image, respectively. Furthermore, let β denote a portion of all pixels within the elliptical region, that are not assigned to the background by the mask function $M(\cdot)$. The proposed measure is then defined as

$$D(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B) = \beta^{-1} D_n(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B), \tag{12}$$

where $D_n(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B)$ is the normalized distance between $\mathbf{h}_A$ and $\hat{\mathbf{h}}_t$ given the background histogram $\mathbf{h}_B$ and is defined as

$$D_n(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B) = \frac{\varrho(\mathbf{h}_A, \hat{\mathbf{h}}_t)}{\sqrt{\varrho(\mathbf{h}_B, \hat{\mathbf{h}}_t)^2 + \varrho(\mathbf{h}_A, \hat{\mathbf{h}}_t)^2}}. \tag{13}$$

We have observed by empirical evaluation that the distance in Eq. (12) follows a Gamma like function and we define the likelihood function of state $\mathbf{x}_t$ as

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto D(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B)^{\gamma_1 - 1} e^{\frac{-D(\mathbf{h}_A, \hat{\mathbf{h}}_t; \mathbf{h}_B)}{\gamma_2}}, \tag{14}$$

where the parameters a and b were estimated on a large number of successfully tracked objects and are given by

$$\hat{\gamma}_1 = 2.242, \quad \hat{\gamma}_2 = 0.055. \tag{15}$$

6 Generating mask

While histograms are powerful color cues for tracking textured objects, they can fail severely when the object is moving on a similarly textured background. This is usually due to their lack of the ability to capture the spatial relations in texture, and the fact that they are always sub-sampled, in order to increase their robustness. There is, however, still some useful information left in the current and the background image; the difference between the two. By thresholding this difference image with some appropriate threshold, we can construct a mask image, which filters out the pixels that are likely to belong to the background. Since the lighting conditions and the background color vary heavily across the playground, the threshold has to be estimated dynamically.

We propose a simple solution to estimating this threshold. At time t, a smoothed state $\mathbf{o}_t$ is calculated via Eq. (7) and histograms $\mathbf{h}_A$ and $\mathbf{h}_B$ are sampled at that state from the current image $A(\cdot)$ and the background image $B(\cdot)$, respectively. If a distance $\varrho(\mathbf{h}_A, \mathbf{h}_B)$ exceeds some prescribed threshold value ϱ_{thresh} then a mask function

$$M_D(\mathbf{u}) = \begin{cases} 1 & ; \quad \|A(\mathbf{u}) - B(\mathbf{u})\| \geq T_{mask} \\ 0 & ; \quad otherwise \end{cases} \tag{16}$$

is generated with a threshold T_{mask} calculated such that at least 25 percent of pixels in the mask function $M_D(\cdot)$ within the state's $\mathbf{o}_t$ ellipse are assigned to the background. If the threshold ϱ_{thresh} is not exceeded, the mask is set to unity, i.e. $M_D(\mathbf{u}) = 1$ for all pixels $\mathbf{u}$.

7 Adaptation

As the player moves across the playground, its texture varies due to nonuniform lighting conditions, influences of the background and variations in the player's pose. To improve tracking under these conditions, we adapt the reference histogram $\hat{\mathbf{h}}_t$ at time-step t with the histogram $\mathbf{h}_{\mathbf{o}_t}$, sampled at the current smoothed state $\mathbf{o}_t$ (Eq. 7) of the target. The extent by which the model adapts to the target is determined by the normalized distance function defined in Eq. (13); that is, if the smoothed state is likely to have been falsely estimated, then the model histogram should be updated by a very small amount, or not at all, and it should be

adapted by some larger amount otherwise. The adaptation equation follows the form of

$$\hat{\mathbf{h}}_t = \alpha \mathbf{h}_{\mathbf{o}_t} + (1-\alpha)\hat{\mathbf{h}}_t^-, \tag{17}$$

where the superscript in $\hat{\mathbf{h}}_t^-$ denotes the reference histogram prior to adaptation. Let $\mathbf{h}_A$ be a histogram sampled at a state $\mathbf{o}_t$ on the current image and let $\mathbf{h}_B$ be a histogram sampled at the same state but on the background image. The adaptation parameter α is then defined as

$$\alpha = \Omega_{max} \cdot (1.0 - D_n(\mathbf{h}_A, \hat{\mathbf{h}}_t^-; \mathbf{h}_B)), \tag{18}$$

where Ω_{max} denotes the maximal adaptation.

8 Tracking multiple players

The closed world assumption about the camera position imposes a restriction, that players are viewed from above. Since it is usually not a valid situation when one player ends up on top of another, we can assume that it is highly unlikely to observe a complete occlusion during the match.

This forms a new closed world assumption: *(v)At a given time-step, two players can not occupy the same position.*

With this in mind, we devise a framework for combining separate particle filters by incorporating a spatial information of the neighboring players. For the sake of the argument, let's presume that at a given time-step, the states of all players are known. Let ${}^{(j)}\mathbf{x}$ denote the state of the j^{th} player and let there be N_p players present on the playground. The player with an index j is presented in the current image by some set of pixels ${}^{(j)}\mathbf{K}$. Assume that for all pixels from the set ${}^{(j)}\mathbf{K}$, the closest center among all players is the one of the j^{th} player. Hence, if the true states $\{{}^{(j)}\mathbf{x}\}_{j=1}^{N_p}$ at a given time-step were in fact known, it would be possible to partition the image into pairwise-disjoint regions, such that each region would contain exactly one of the sets ${}^{(j)}\mathbf{K}$.

A partitioning of the space which achieves this kind of topology is the Voronoi partitioning [7], mainly used by the surface triangulation and obstacle avoidance algorithms. A Voronoi partitioning among N_p points, called seeds in the triangulation terminology, is completely defined by a set of points $\mathbf{S} = \{s_j\}_{j=1}^{N_p}$ and generates a set of N_p pairwise-disjoint convex partitions $\mathbf{P} = \{P_j\}_{j=1}^{N_p}$, where each partition contains exactly one seed and for every point in a particular partition, the closest seed is the one included by that partition. For illustrative purposes an example of a Voronoi two-dimensional space partitioning by four seeds is shown in Fig. (1).

In reality, at time-step $t+1$, prior to the tracking iteration, the true states of the players are not known, however, the predictions of type

$${}^{(j)}\tilde{\mathbf{x}}_{t+1} = {}^{(j)}\mathbf{o}_t + {}^{(j)}\mathbf{d}_{t+1} \tag{19}$$

are available, where $\mathbf{o}_t$ is the smoothed estimate of the j^{th} player's state from the previous time-step (Eq. 7) and

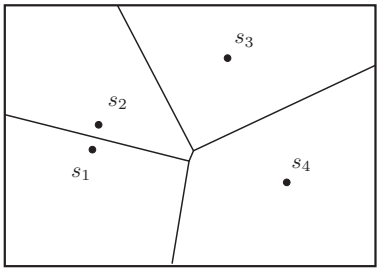

Figure 1. Four seeds divide the space into four pairwise-disjoint convex Voronoi partitions.

${}^{(j)}\mathbf{d}_{t+1}$ is the j^{th} player's drift at the current time-step (Eq. 8).

Prior to a tracking iteration, therefore, an image can be partitioned into N_p Voronoi regions based on the predictions of all N_p players. The idea is then to track each player only within a region that contains its prediction. Restriction of a tracker for the j^{th} player to its partition P_j can easily be achieved by use of an additional mask function ${}^{(j)}M_V(\mathbf{u})$ defined as

$${}^{(j)}M_V(\mathbf{u}) = \begin{cases} 1 & ; \quad \mathbf{u} \in P_j \\ 0 & ; \quad otherwise \end{cases}, \tag{20}$$

where $\mathbf{u}$ is a pixel contained by the region P_j. The mask function ${}^{(j)}M(\mathbf{u})$ in Eq. (10) is then defined as an intersection of masks ${}^{(j)}M_D(\cdot)$ (Eq. 16) and ${}^{(j)}M_V(\cdot)$

$${}^{(j)}M(\mathbf{u}) = {}^{(j)}M_D(\mathbf{u}) \cap {}^{(j)}M_V(\mathbf{u}), \tag{21}$$

where by superscript $(\cdot)^{(j)}$ we emphasize that all masks are player-dependent.

The closed world tracking scheme for multiple players with separate particle filters is described in the following section.

8.1 The Algorithm

Let there be a set of N_p separate particle filter trackers, each for one player, defined by

$$\left\{ {}^{(j)}\hat{\mathbf{h}}_t, {}^{(j)}p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}), {}^{(j)}p(\mathbf{x}_t|\mathbf{x}_{t-1}) \right\}_{j=1}^{N_p}, \tag{22}$$

where the j^{th} tracker is defined by its posterior from the previous time-step ${}^{(j)}p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, its transition distribution ${}^{(j)}p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the model histogram ${}^{(j)}\hat{\mathbf{h}}_t$. This is the initial input data for the tracking iterations at the current time-step t.

A one time-step iteration of the multiple-player tracking algorithm can be summarized as follows: Firstly, a set of seeds $\mathbf{S} = \{s_j\}$ is initialized at the predictions of all players $s_j = {}^{(j)}\tilde{\mathbf{x}}_t$ (Eq. 19) and the corresponding Voronoi partitions $\mathbf{P} = \{P_j\}$ are calculated. A tracker with index $j = 1$

is selected, its mask function ${}^{(1)}M(\mathbf{u})$ (Eq. 21) is constructed and its CONDENSATION iteration is processed. After completing this iteration, a smoothed estimate of the first player's average state ${}^{(1)}\mathbf{o}_t$ becomes available (Eq. 7) and its corresponding weight is calculated as $\pi_{\mathbf{o}_t} = p(\mathbf{y}_t|\mathbf{o}_t)$. A histogram is sampled at that state on the current image and the model ${}^{(1)}\hat{\mathbf{h}}_t$ is adapted according to Eq. (17). As described in the Sect. (6) a treshold for the mask function ${}^{(1)}M_D(\cdot)$ is calculated within the ellipse of the state ${}^{(1)}\mathbf{o}_t$. Finally, the seed s_1 in the set $\mathbf{S}$ is replaced by the smoothed average state ${}^{(1)}\mathbf{o}_t$ and the Voronoi partitioning is recalculated. This procedure is repeated for the tracker with index $j = 2$ and continued in the same manner, until all of the trackers are processed. The algorithm is depicted by the Alg. (8.1).

Algorithm 1. *The multiple player tracking algorithm.*

- For all j set: $\mathbf{S} = \{s_j\}_{j=1}^{N_p}; s_j = {}^{(j)}\tilde{\mathbf{x}}_t$
- For $j = 1 : N_p$
  1. Construct a set of Voronoi regions $\mathbf{P} = \{p_j\}_{j=1}^{N_p}$ on a set of seeds $\mathbf{S}$.
  2. Construct the Voronoi mask ${}^{(j)}M_V(\mathbf{u})$ via Eq. (20).
  3. Construct the difference image mask ${}^{(j)}M_D(\mathbf{u})$ as described in Sect. (6).
  4. Run the classical CONDENSATION iteration on the j^{th} tracker (Sect. 3).
  5. Calculate the smoothed average state ${}^{(j)}\mathbf{o}_t$ (Eq. 7).
  6. Sample a histogram at ${}^{(j)}\mathbf{o}_t$ and adapt the model to that histogram as in Sect. (7).
  7. If needed, calculate the threshold for the mask $M_D(\mathbf{u})$ (Sect. 6).
  8. Update the j^{th} Voronoi seed by the smoothed average state: $s_j = {}^{(j)}\mathbf{o}_t$.
- End For j

---

9 Experiments

Tests were conducted on a sequence of 948 images with size of 384x288 pixels, from a recorded handball match where the players were approximately 9x9 pixels large. The frame rate of the sequence was 25 frames per second, which was equivalent to approximately 38 seconds of the match. All images comprised twelve players, where six of them wore white dresses, and the other six wore dark dresses. The playground was mainly yellow and blue, with a few advertisements, and it influenced the colors of the players severely: white players looked yellow on the yellow part of the playground and became blue when they moved on the blue part. Since the handball is a sport where contacts and near-occlusions happen frequently and some players of the same team often move close to one another, we feel it is a good testing ground for multiple-player trackers. In Fig. (2), an image from the entire test sequence is presented, along with some of the contacts that happened during that sequence and two examples of players moving over the advertisement.

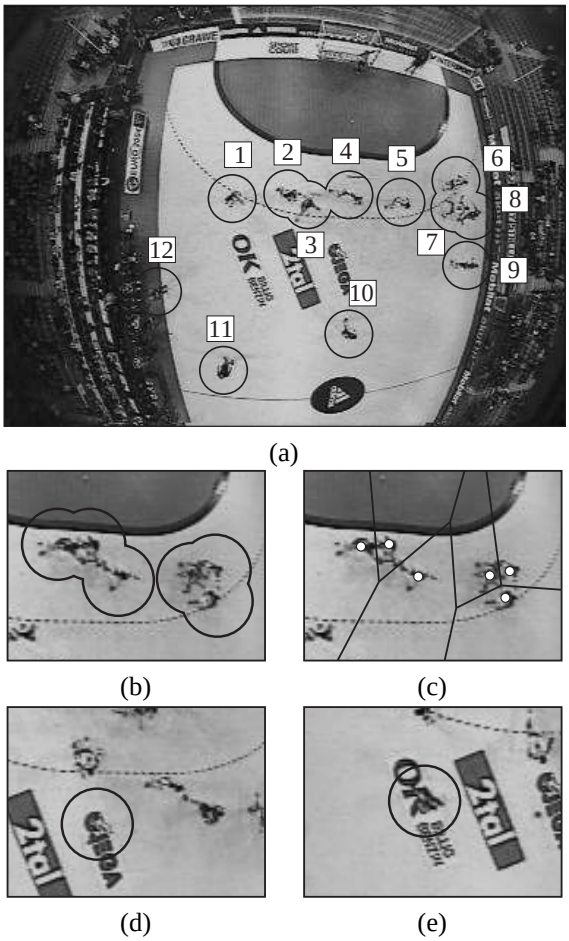

Figure 2. A typical image with 12 players from the test sequence for evaluation of tracker (a), an example of colliding players (b) and corresponding Voronoi partitions (c), and two occasions of player moving over the advertisement (d),(e).

Two tests were conducted to compare the Alg. (8.1) to the one that considers each player separately and independently of one another. The parameters for both trackers are listed in Tab. (1). In the first experiment, the tracker was manually initialized on a single player and the tracking pro-

Table 1. Tracker parameters used in experiments.

ellipse size parameter space	$H_x, H_y \in [6, 10]$
standard deviations of dynamics	$\sigma_{xy} = 0.3$, $\sigma_{ab} = 0.05$
standard deviation for the state smoothing	$\sigma_v = 4.3 frames$ or $\sigma_v = 0.172s$
color likelihood parameters	$\hat{\gamma}_1 = 2.242$, $\hat{\gamma}_2 = 0.055$
number of cells in a color RGB histogram	$B = 8x8x8$
maximal histogram adaptation	$\Omega_{max} = 0.04$
number of samples in a single particle filter	$N = 20$
threshold for masking	$\varrho_{thres} = 1.56$
the state smoothing parameter	$\sigma_o = 4.3$

Table 2. The average failure rates for tracking *twelve* players with a single-player and a multiple-player tracker. The symbols $F_r/(38s)$ and $F_r/(1min)$ denote the expected failure rates for 38 seconds of the game and recalculated to a 1 min of the game, respectively.

tracker type	$F_r/(38s)$	$F_r/(1min)$
single-player	21.6	34.1
multiple-player	3.8	6.0

ceeded over the entire sequence of the match; this experiment was conducted five times for each of the twelve players, and the number of instances when the tracker failed, and had to be reinitialized was recorded. The tracking failure was regarded as an occasion, when the tracker obviously lost the player, and did not seem to be able to properly continue with tracking. As a result, the expected failure rate was computed for the given sequence and is reported in Tab. (2).

In the second experiment, the tracker was initialized manually on all of the twelve players, and all players were tracked simultaneously over the entire sequence. When a particular player was lost, it was reinitialized manually and the tracking proceeded. Again, this experiment was conducted five times, and the expected failure rate was calculated (Tab. 2).

From the Tab. (2) it is obvious that the failure-rate decreased dramatically when the information about the neighboring players was taken into account, thus rendering the multiple-player tracker superior to the single-player tracker.

10 Conclusion

An algorithm for tracking multiple interacting players is presented in this article. The problem of tracking is formulated as a set of closed world assumptions, and the tracker based on particle filtering is built such, to satisfy each assumption. A multiple player tracker is proposed as a set of separate trackers, one for each player, which are combined by inferring a Voronoi partitioning among all players at each time-step, and tracking each player within its Voronoi cell. The multiple-player tracker was compared on a very demanding data-set with a single-player tracker and came out as being superior. The sequence on which the trackers were tested is indeed an arduous one, and one must not be misled to think that the single-player tracker performed poorly. The only difference between the single- and multiple-player tracker is that the latter considered the spatial relations between all players, while still tracking each player separately. Thus, the decrease in failure-rate of a multi-player tracker is mainly due to the ability of taking into account the clashes and near-occlusions among the players. Conversely, if a tracker performed poorly on this sequence it would not necessarily mean that it has poor tracking capabilities, whereas good performance certainly implies good tracking capabilities.

In the future research, we plan to conduct more tests and compare our trackers to some reference trackers presented in the literature. We are also considering a possibility for a tracker to automatically reinitialize when the tracking fails.

11 Acknowledgement

The research described in this paper has been sponsored by InspireWorks.

References

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–151, 2000.

[3] A. Doucet, S.Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, (10):197–208, 2000.

[4] S. S. Intille and A. Bobick. Closed-world tracking. In *Int. Conf. of Computer Vision*, pages 672–678, 1995.

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

[6] P. Perez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of IEEE*, 292(3):495–513, January 2004.

[7] J. R. Sack and J. Urrutia. *Handbook of Computational Geometry*. Elsevier Science Pub Co, 2000.