# Stony Brook University

# Tracking Multiple Moving Objects in Video

A Thesis Presented
by

**Andrei Todor**

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

**Master of Science**
in
**Computer Science**

Stony Brook University

**August 2009**

The Graduate School

**Andrei Todor**

We, the thesis committee
for the above candidate for the
Master of Science degree, hereby recommend
acceptance of this thesis.

**Dimitris Samaras – Thesis Advisor**
**Associate Professor**
**Computer Science Department**

**Klaus Mueller - Chairperson of Defense**
**Associate Professor**
**Computer Science Department**

**Gregory Zelinsky**
**Associate Professor**
**Psychology Department**

This thesis is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Thesis

# Tracking Multiple Moving Objects in Video

By

**Andrei Todor**

**Master of Science**

in

**Computer Science**

Stony Brook University

**2008**

In this work some basic methods as well as more recent advances in tracking are presented, with the purpose of investigating the problem of multiple target tracking, in the context of tracking people in relatively complex situations. The early methods include the KLT tracker and the Kalman filter. The research has advanced mainly on two fronts: one represented by the kernel density estimation methods, and the other one by the Bayesian tracking and particle filters. The challenges of the problem are presented, as well as the various solutions proposed by the methods of both categories. A lot of challenges related with the proposed problem still remain: large number of targets, long video sequences, occlusions and others. We attempt to address these problems by proposing two new techniques: collaborative human-computer tracking and tracking using occlusion cues.

**Table of contents**

## List of figures

**List of tables**

# 1. Introduction

## 1.1. Problem definition

"Tracking is the problem of generating an inference about the motion of an object given a sequence of images." [33] We will adopt this definition for this work, with two amendments: we will be interested in the motion of more objects at the same time, and these objects will be humans.

In the special case of bearings-only tracking the objects are considered a-dimensional and the interest is just in the position of the objects in the image, or even the 3D position in the scene. We may also be interested in the shape of the object and how it deforms as it moves, and this may be helpful for the tracking task. In a unified approach, this deformation+motion evolution was called "deformotion" [34].

The problem under investigation, multiple target tracking in the context of people tracking, is interesting, considering its diverse applications: surveillance, human-computer interaction, virtual environments, etc. It is also challenging because of the difficulties associated with it: occlusions, cluttered environments, change in appearance and illumination of the targets and possibly a large number of targets. A special attention is paid to the cases where targets are small and body part detection is impractical, which adds to the difficulty of the problem.

## 1.2. Methods overview

Some methods have been successfully applied to single object tracking, but their extension to multiple target tracking arises some problems. Initial methods include the Kanade-Lucas-Tomasi tracker [1] and the Kalman filter [13]. More recent methods are for example the kernel methods, like mean shift. Mean shift is based on kernel density estimation and has been used for many low level vision tasks [4] and for tracking [5]. Another line of development was initiated with the advent of Monte Carlo methods, which include Markov Chain Monte Carlo [8]. This line continues with Bayesian filtering, in particular particle filters. They are similar to the Kalman filter, but they overcome its inability to deal with non-gaussian distributions, in particular with multi-modal distributions. They are able to do this by estimating the PDF of a target by a set of samples of that PDF. The idea was introduced in [2], where it was called bootstrap filtering, and was applied to tracking in the CONDENSATION algorithm [3]. When multiple targets come into the scene one has to make a choice between having one such particle filter for each target and having a single particle filter for the entire scene,

representing the joint pdf for all targets [9]. In one case a difficulty arises when one target occludes another, and the two trackers become confused by each other. Moreover, in such cases the problem of preserving the target identities has to be addressed [11]. In the other case the difficulty is that the complexity of the tracker grows exponentially with the number of targets. Also, as the dimensionality of the state variables increases, the number of particles has to increase again exponentially. An overview and classification of the methods that will be reviewed in this paper is presented in Figure 1.

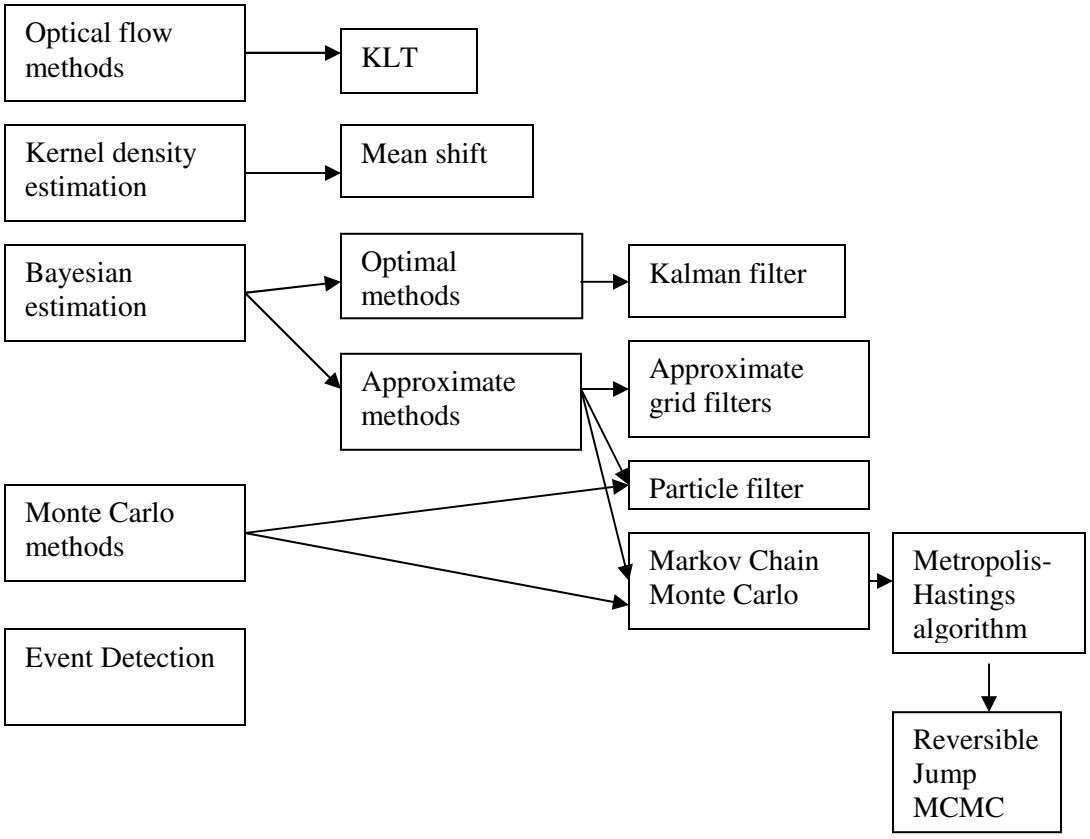



**Figure 1:** Overview of tracking methods

### 1.3. Challenges

- The **large number of targets** is a major difficulty associated with multiple target tracking, many of the methods having been developed, at least initially, with one target in mind, most notably the Kalman filter and mean shift. Extension to multiple targets could in principle be done by having one simple tracker for each target, but this becomes problematic when targets cross each other or just come close to each other, in which cases the so called "kidnapping" phenomenon occurs: the trackers involved end up tracking all the same, most prominent target in the image. Moreover, the number of targets is often **variable**, with targets entering and leaving the scene.
- **Illumination change** is another challenging scenario: one targets goes from shadow into the light, or the opposite, which causes its appearance to change drastically. Many methods, primarily mean shift, rely on the color distribution of the target, and are problematic in such situation. The ones relying on optical flow, like KLT, will also suffer from the illumination change problem, since optical flow assumes brightness constancy between frames, which is no longer satisfied in case of an illumination change. The problem can be addressed by taking into account geometric information, like the contour, but then again this requires some edge detection process, which is itself influenced by the illumination change.
- **Non-rigid deformation**. Other types of changes are due to the fact that the target is moving in an arbitrary way, which leads to non-rigid body deformation in the image, making affine transformation models insufficient.
- **Occlusion**. The major reason for tracking failure in general is occlusion. This could be caused by an object in the environment between the camera and the target, by a camouflaging effect when the target is similar in appearance to the background to the point that segmentation fails to detect the target, or, in our case, another object that is being tracked. In such cases, the methods relying on the contours of the object are the most sensitive.

The images in Figure 2 present typical scenarios we have in mind and illustrate the difficulties just mentioned.

### 1.4. Tracking with an articulated model

A lot of work has been dedicated to human tracking by developing a 3D articulated model of the body. In [21], segmented human hypotheses, identified by taking into account 3D information about the scene, are tracked with a Kalman Filter with explicit handling of occlusions. The hypotheses are verified by walking recognition using an articulated human walking model.

**Figure 2: Typical human tracking scenarios**

In many cases, which are considered in this paper, the articulated model is not a feasible approach, because of lack of accurate images of the body. In such

cases, the body is often far from the camera, resulting in just a few pixels of the entire image. The problem is exacerbated if the camera has low resolution. Moreover, the body can be seen in arbitrary positions and often occluded. In these conditions, reliable identification of body parts is not feasible. With multiple targets, the approach is even more unrealistic taking into account the increase in complexity incurred by having an unnecessarily complex model fit to each target.

Some ideas from the literature based on articulated models can still be applied to solve the various problems encountered when tracking small size targets. Such ideas are learning priors for tracking from small training sets [20] and learning dynamic models [19] by employing Gaussian process latent variable models.

### 1.5. Other approaches

At the other extreme, tracking has been proposed by considering spatio-temporal XT slices of the video, disregarding the vertical component of motion [17].

Other related machine learning techniques that can be applied in tracking are in the category of graphical models. A conditional random field was used in [22], where the observation potentials were learned from data. For fast computation, the CRF was combined with approximate grid filtering. The state space was modeled as a grid, where the state PDF was considered constant in each cell.

A task similar to tracking and with applications to surveillance is detecting events in images and video. In [29], this is achieved by constructing a database of images or videos of "normal" behaviors. Given a new image or video, an explanation is sought for it by trying to recompose it from patches of images or videos from the database in a probabilistic framework. If such an explanation is not found or is considered improbable, the image or video is declared an irregularity.

The relevant literature shows that progress has been made in solving the proposed problem, but the results obtained so far allow for improvement in terms of the length of the tracked sequence, accuracy and efficiency.

## 2. Low level methods

The most basic methods are from low level vision and rely on optical flow in image sequences.

### 2.1. The KLT tracker

The Kanade-Lukas-Tomasi tracker is one of the earliest methods proposed for tracking, although derived from an image registration method. This is in fact also the basic idea behind it: register windows of interest in successive frames of a video.

To present the method formally, we consider the video sequence as a function $I(x, y, t)$ satisfying the property

$$I(x, y, t+\tau) = I(x-\xi, y-\eta, t).$$

The vector $\mathbf{d} = (\xi, \eta)$ is the displacement of the point at $\mathbf{x} = (x, y)$ between time instants $t$ and $t+\tau$. The problem of finding the displacement **d** of a point is that a single pixel cannot be tracked unless it has a very distinctive brightness with respect to its neighbors. Therefore windows of pixels are tracked, with the assumption that all the pixels in the window behave similarly. Any discrepancy that cannot be explained by a translation is considered error and the displacement vector is computed so as to minimize this residue error.

Let $J(\mathbf{x}) = I(x, y, t+\tau)$ and $I(\mathbf{x}-\mathbf{d}) = I(x-\xi, y-\eta, t)$.

Then $J(\mathbf{x}) = I(\mathbf{x}-\mathbf{d}) + n(\mathbf{x})$, where $n$ is noise, and the error to be minimized is

$$\varepsilon = \int_W [I(\mathbf{x}-\mathbf{d}) - J(\mathbf{x})]^2 \, w \, d\mathbf{x},$$

where $W$ is the window under consideration and $w$ is a weighting function.

The solution is given by the equation $G\mathbf{d}=\mathbf{e}$, where $G = \int_W \mathbf{g}\mathbf{g}^T w \, dA$, $\mathbf{e} = \int_W (I - J)\mathbf{g} w \, dA$ and g is the gradient at **x**.

One of the strengths of the KLT tracker is automatic detection of the windows $W$ to be tracked. This is done by following the criterion that a good window is one that can be tracked well. This means that the matrix $G$ of the system must be above the image noise level and well conditioned. In turn, the noise level requirement implies that both eigenvalues of $G$ must be large while the conditioning requirement means that they cannot differ by several orders of

magnitude. As a consequence, if the two eigenvalues of $G$ are $\lambda_1$ and $\lambda_2$, a window is accepted if $\min(\lambda_1, \lambda_2)>\lambda$, where $\lambda$ is a predefined threshold.

The efficiency of the method has only been proved for short sequences and slow motion of the features. The results can be improved by updating the tracked template, with the risk of the template drifting too much from the original one, a problem that was addressed in [14] by considering a compromise between updating the template at each frame and always keeping the initial template.

## 2.2. Other methods

A particular tracking approach that relies on properties of the video acquisition process is presented in [24]. Here a low level technique makes use of the fact that the signal acquired from commercial video cameras scans a number of lines per frame and to reduce flicker two interlaced frames are sent alternatively. This means that each scan line is captured one frame later than its preceding and following line, which leads to artifacts along the edges of moving objects. By measuring these artifacts, the motion of objects can be estimated.

The illumination change problem was solved in an algorithm that computes an illumination invariant optical flow field [27]. The optical flow is computed by minimizing an energy function that penalizes pixels that are similar in the current frame but are different in the next frame. This way the brightness constancy assumption, that requires corresponding pixels in successive frames to be close to the same intensity, is relaxed. The computation of the optical flow is made robust by using a graph cuts formulation [42].

## 3. Kernel density estimation

Kernel density estimation methods pursue the objective of tracking by estimating the underlying distribution of a set of data points.

### 3.1. Feature space analysis using mean shift

Given $n$ data points $\mathbf{x}_i$, $i$=1,…,n in the $d$-dimensional space $R^d$, the multivariate kernel density estimator with kernel $K(\mathbf{x})$ and bandwidth parameter $h$ computed in the point $\mathbf{x}$ is

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

The profile of the kernel $K$ is a function $k(x)$ such that for $x \geq 0$ ,

$$K(\mathbf{x}) = c_{k,d}\, k(\|\mathbf{x}\|^2)$$

The normalization constant $c_{k,d}$ makes $K(\mathbf{x})$ integrate to one. A common profile is

$$k_E(x) = \begin{cases} 1-x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases}$$

which yields the Epanechnikov kernel

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

where $c_d$ is the volume of the unit d-dimensional sphere. Using the Epanechnikov profile, the density estimator can be written as

$$\hat{f}_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right).$$

The first step in the analysis of a feature space with the underlying density $f(\mathbf{x})$ is to find the modes of this density. The modes are located among the zeros of the gradient $\nabla f(\mathbf{x})$. The mean shift procedure is an elegant way to locate these zeros without estimating the density. The gradient of the density estimator is

$$\hat{\nabla} f_{h,K}(\mathbf{x}) \equiv \nabla \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}-\mathbf{x}_i) k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$$

Define the function $g(x) = -k'(x)$. Using $g(x)$ for profile, the kernel $G(x)$ is defined as

$$G(x) = c_{g,d}\, g\left(\|\mathbf{x}\|^2\right)$$

Then

$$\hat{\nabla} f_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}) g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right]$$

The first term is proportional to the density estimate at **x** computed with the kernel $G$, $\hat{f}_{h,G}(\mathbf{x})$, and the second term is the mean shift, $\mathbf{m}_{h,G}(\mathbf{x})$, i.e. the difference between the weighted mean, using the kernel $G$ for weights, and **x**, the center of the window. The expression for the gradient becomes

$$\hat{\nabla} f_{h,K}(\mathbf{x}) = \hat{f}_{h,G}(\mathbf{x}) \frac{2c_{k,d}}{h^2 c_{g,d}} \mathbf{m}_{h,G}(\mathbf{x})$$

yielding

$$\mathbf{m}_{h,G}(\mathbf{x}) = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(\mathbf{x})}{\hat{f}_{h,G}(\mathbf{x})}$$

This expression shows that at location **x** the mean shift vector computed with kernel $G$ is proportional to the normalized density gradient estimate obtained with kernel $K$. The normalization is by the density estimate in **x** computed with kernel $G$. The mean shift vector thus always points toward the direction of maximum increase in the density. The mean shift procedure is then the successive repetition of the following steps:

- computation of the mean shift vector $\mathbf{m}_{h,G}(\mathbf{x})$
- translation of the kernel $G(\mathbf{x})$ by $\mathbf{m}_{h,G}(\mathbf{x})$

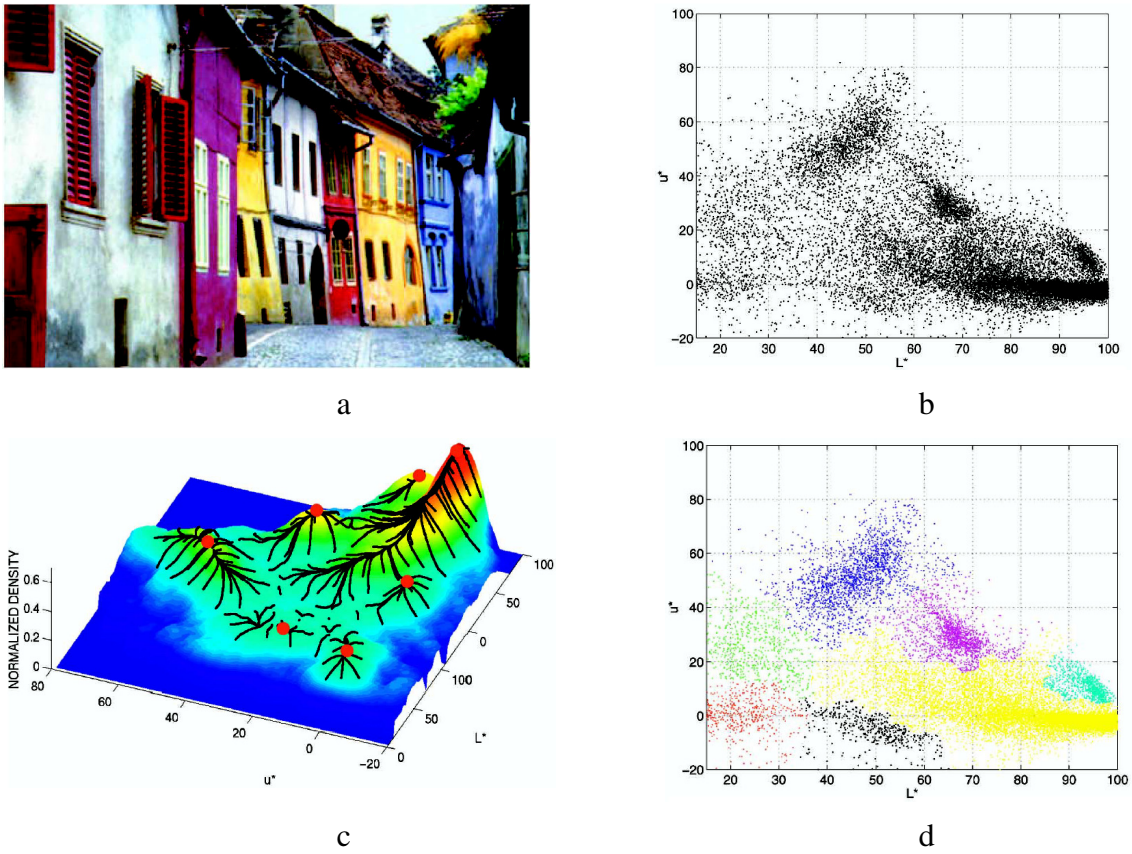



**Figure 3:** 2D feature space analysis. (a) Initial image. (b)Two-dimensional data set of 110,400 points representing the first two components of the L*u*v space of the image (a). (c) Trajectories of the mean shift procedures drawn over the Epanechnikov density estimate computed for the same data set. (d) Decomposition obtained by running 159 mean shift procedures with different initializations.

Mean shift has been successfully used for segmentation and edge preserving smoothing. By considering the data points to be the pixel color in the 3D color space, a mean shift process was initiated for each image pixel. Its color was then replaced by the convergence point.

### 3.2. Tracking with mean shift

To use mean shift for tracking we use the following approach. Let $\{\mathbf{x}^*_i\}_{i=1\ldots n}$ be the pixel locations of the target model, centered at 0. The function $b$ : $R^2 \rightarrow \{1\ldots m\}$ associates to each pixel the index of the histogram bin corresponding to the color of that pixel. The probability of the color $u$ in the target model is derived by employing a kernel profile $k$ which assigns a smaller weight to the locations that are farther from the center of the target. The radius of the kernel profile, $h$, is taken equal to one, by assuming that the generic coordinates $x$ and $y$ are normalized. Hence,

$$\hat{q}_u = C\sum_{i=1}^{n} k\left(\left\|\mathbf{x}_i^*\right\|^2\right)\delta[b(\mathbf{x}_i^*) - u]$$

The normalization constant C is derived by imposing the condition $\sum_{u=1}^{m} \hat{q}_u = 1$.

The target candidates are modeled in a similar way. Let $\{\mathbf{x}_i\}_{i=1\ldots n_h}$ be the pixel locations of the target candidate centered at $\mathbf{y}$ in the current frame. Using the same kernel profile $k$, but with radius $h$, the probability of the color u in the target candidate is given by

$$\hat{p}_u(\mathbf{y}) = C_h\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right)\delta[b(\mathbf{x}_i) - u]$$

The radius of the kernel, $h$, determines the number of pixels, i.e. the scale, of the target candidate.

The most probable location $\mathbf{y}$ of the target in the current frame is obtained by minimizing the distance between the color distributions of the model and target candidate. The Bhattacharyya coefficient is proposed as a measure of the closeness of two distributions. It is given by

$$\rho(\mathbf{y}) = \rho[p(\mathbf{y}), q] = \int \sqrt{p_z(\mathbf{y})q_z}\,dz$$

The search for the new target location in the current frame starts at the estimated target location $\hat{\mathbf{y}}_0$ of the target in the previous frame. Thus, the color probabilities $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1\ldots m}$ of the target candidate at location $\hat{\mathbf{y}}_0$ have to be computed first.

Using Taylor expansion around the values $\hat{\mathbf{y}}_0$, the Bhattacharyya coefficient is approximated as

$$\rho[\hat{p}(\mathbf{y}),\hat{q}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{1}{2}\sum_{u=1}^{m}\hat{p}_u(\hat{\mathbf{y}})\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}}$$

Introducing the formulas for the distributions we obtain

$$\rho[\hat{p}(\mathbf{y}),\hat{q}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)$$

where

$$w_i = \sum_{u=1}^{m}\delta[b(\mathbf{x}_i)-u]\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}}$$

The maximization can be efficiently achieved based on mean shift iterations. The new location of the target is computed at each iteration using the mean shift vector:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h}\mathbf{x}_i w_i g\left(\left\|\frac{\mathbf{y}_0-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\mathbf{y}_0-\mathbf{x}_i}{h}\right\|^2\right)}$$

The method was demonstrated on tasks of tracking people in difficult contexts, like crowded environments (subway scene) or quick motions with occlusions (football game). However, only single targets were tracked.

### 3.3. Improvements

In [6], tracking was extended to multiple targets by combining mean shift with Bayesian filtering. The mean shift was also combined with a sum of squared differences tracker, to improve performance under occlusion [18].

Other improvements were made by more careful consideration of the choice of the bandwidth parameter $h$ [16]. In the initial formulation, $h$ has a fixed value, yielding a radically symmetric kernel. This means that the spatial information is encoded very loosely, and is easy for the tracker to get confused by other objects having the same feature distribution but different spatial configuration of the features. The problem is overcome by dividing the target into more blocks, each block having its own bandwidth parameter $h_j$, which specifies the allowed motion of the pixel. For positions that are expected to move very little, their $h_j$ should be small, penalizing pixels of the same feature that are observed far away from where they originally appeared in the target. In people tracking, for example, this would be the case for the feet. The torso, on the other hand, should have a small $h_j$. The parameters were learned from motion capture data of walking people for this particular task of human tracking.

To overcome the same problems mentioned in the previous paragraph, the kernel density estimate was modified to include both the feature value and the feature location [23]. Asymptotic behavior shows that when the bandwidth of the spatial kernel is 0, the tracker obtained is similar to rigid body tracking approaches such as KLT. When this bandwidth tends to infinity, on the other hand, the method degenerates to histogram tracking as in mean shift.

A similar idea of including spatial information was applied by following the energy minimization formulation, where the energy includes an image energy term based on the closeness of the distributions of intensities and the shape energy, where the shape was encoded using level sets [28].

A method for evaluating multiple feature spaces while tracking uses mean shift as the basic tracking mechanism [39]. The features used for tracking are selected on line based on the hypothesis that the features that best discriminate between object and background are also best for tracking the object. Possible features are ranked based on the two-class variance ratio measure applied to distributions of object and background pixels. The features under consideration are pixel distributions in various color spaces computed by linearly combining the RGB values. Experiments showed improved performance over traditional mean shift in difficult cases of illumination change and camouflage.

# 4. Bayesian tracking

## 4.1. General approach

In the context of Bayesian estimation we define the problem of tracking by considering the evolution of the state sequence $\{\mathbf{x}_k, k \in \mathbf{N}\}$ of a target given by the system model

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})$$

where $\mathbf{f}_k : \mathbf{R}^{n_x} \times \mathbf{R}^{n_v} \to \mathbf{R}^{n_x}$ is a function of the state, $\{\mathbf{v}_{k-1}, k \in \mathbf{N}\}$ is an i.i.d. process noise sequence and $n_x$ and $n_v$ are the dimensions of the state and process noise vectors. At discrete time steps we get measurements of the state according to the measurement model

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k)$$

where $\mathbf{h}_k : \mathbf{R}^{n_x} \times \mathbf{R}^{n_n} \to \mathbf{R}^{n_z}$ is a function of the state, $\{\mathbf{n}_k, k \in \mathbf{N}\}$ is an i.i.d. measurement noise sequence and $n_z$ and $n_n$ are the dimensions of the measurement and measurement noise vectors. The goal is to construct the posterior PDF of the state $p(\mathbf{x}_k | \mathbf{z}_{1:k})$. It is assumed that the prior PDF $p(\mathbf{x}_0 | \mathbf{z}_0) \equiv p(\mathbf{x}_0)$ is available. The posterior can be obtained recursively in two stages, prediction and update. Suppose that the required PDF $p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1})$ at time $k-1$ is available. The prediction stage involves using the system model to obtain the prior PDF of the state at time $k$ via the Chapman-Kolmogorov equation

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}$$

The probabilistic model of the state evolution, $p(\mathbf{x}_k | \mathbf{x}_{k-1})$, which is a Markov model [38], is defined by the system equation and the known statistics of $\mathbf{v}_{k-1}$:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_{k-1}) p(\mathbf{v}_{k-1} | \mathbf{x}_{k-1}) d\mathbf{v}_{k-1}$$

Since by assumption $p(\mathbf{v}_{k-1} | \mathbf{x}_{k-1}) = p(\mathbf{v}_{k-1})$, we have

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \int \delta(\mathbf{x}_k - \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})) p(\mathbf{v}_{k-1}) d\mathbf{v}_{k-1}$$

where is the Dirac function. The Dirac function arises because if $\mathbf{x}_{k-1}$ and $\mathbf{v}_{k-1}$ are known, then $\mathbf{x}_k$ is obtained from a purely deterministic relationship.

At time $k$, a measurement $\mathbf{z}_k$ becomes available and it can be used to update the prior via Bayes' rule:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})}$$

where

$$p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})\, d\mathbf{x}_k$$

The conditional PDF of $\mathbf{y}_k$ given $\mathbf{x}_k$, $p(\mathbf{y}_k \mid \mathbf{x}_k)$, is defined by the measurement model and the known statistics of $\mathbf{n}_k$:

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \int \delta(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k))\, p(\mathbf{n}_k)\, d\mathbf{n}_k$$

The process of estimating posterior densities by the Bayesian approach is illustrated in Fig. 4.

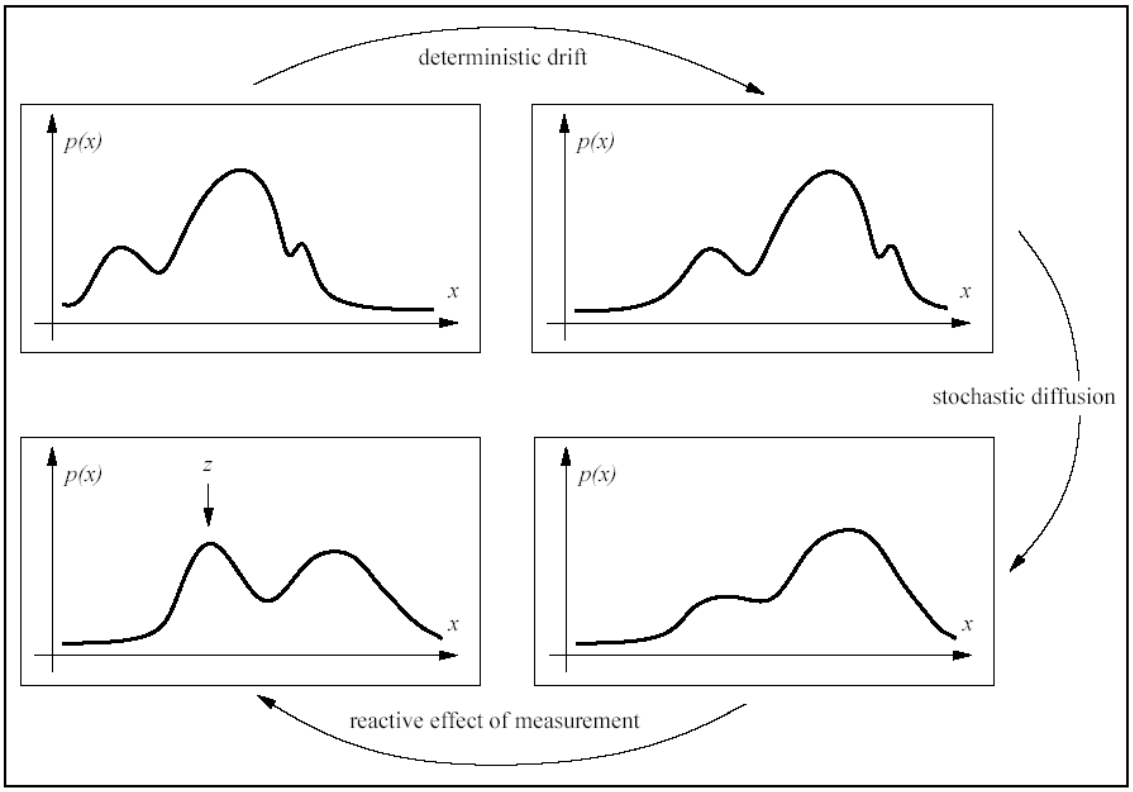



**Figure 4:** Probability density propagation is depicted here as it occurs over a discrete time step. There are three phases: drift due to the deterministic component of object dynamics, diffusion due to random component, reactive reinforcement due to observations.

## 4.2. The Kalman filter

In general, the optimal posterior density cannot be determined analytically. An exact solution can be obtained in a restricted set of cases. The Kalman filter [13] is a solution for such a restricted case. The restrictions are that the noise vectors involved are drawn from Gaussians of known parameters and the functions **f** and **h** are linear. Under these circumstances, if

$p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1})$

is a Gaussian then $p(\mathbf{x}_k | \mathbf{z}_{1:k})$ will also be a Gaussian. Thus, all PDFs involved are Gaussian and the solution can be found in terms of the parameters of these distributions.

We can rewrite the system and measurement equations as

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{v}_{k-1}$$
$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{n}_k$$

where the noise terms are normally distributed:

$$\mathbf{v}_k \sim N(0, \mathbf{Q})$$
$$\mathbf{n}_k \sim N(0, \mathbf{R})$$

The state is estimated by iterating the prediction and update equations:

- prediction

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\mathbf{x}_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$$

- update

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H})\mathbf{P}_k^-$$

$\mathbf{P}_k$ represents the estimate error covariance and an initial value for it is needed as an input to the algorithm.

For the cases where the linearity condition for the standard Kalman filter does not hold, the extended Kalman Filter can be applied. Its equations are obtained by linearizing the system and measurement equations, thus obtaining an approximate solution to the tracking problem.

The Kalman filter provides the optimal solution to the problem of finding the posterior distributions for a restricted case, and the extended Kalman filter is an approximate solution for non-linear dynamic and measurement models. When the other assumptions for the Kalman filter do not hold, one has to resort to other approximate solutions, which will be presented next.

# 5. Particle filters

## 5.1. General method

The particle filter is also an approximate solution to Bayesian estimation, and is an option when the density estimated is multimodal. It is a Monte Carlo method [44], in that it represents the distributions involved by a set of random samples drawn from those distributions.

Suppose we have a set of random samples $\{\mathbf{x}_{k-1}(i):i=1,\ldots, N\}$ from the PDF $p(\mathbf{x}_{k-1}|\ \mathbf{z}_{1:k-1})$. The particle filter is an algorithm for propagating and updating these samples to obtain a set of values $\{\mathbf{x}_k(i):i=1,\ldots, N\}$ which are approximately distributed as $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. Thus the filter is a simulation of the Bayesian prediction and update equations.

- Prediction: each sample is passed through the system model to obtain samples from the prior at time step $k$:

$$\mathbf{x}_k^*(i) = \mathbf{f}_k(\mathbf{x}_{k-1}(i), \mathbf{v}_{k-1}(i))$$

where $\mathbf{v}_{k-1}(i)$ is a sample drawn from the PDF of the system noise $p(\mathbf{v}_{k-1})$

- Update: on receipt of the measurement $\mathbf{z}_k$, evaluate the likelihood of each prior sample and obtain a normalized weight for each sample

$$q_i = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k^*(i))}{\sum_{j=1}^{N} p(\mathbf{z}_k \mid \mathbf{x}_k^*(j))}$$

Thus define a discrete distribution over $\{\mathbf{x}_k^*(i):i=1,\ldots,N\}$, with probability mass $q_i$ associated with element $i$. Now resample $N$ times from the discrete distribution to generate samples $\{\mathbf{x}_k(i):i=1,\ldots,N\}$, so that for any $j$, $p(\mathbf{x}_k(j)=\mathbf{x}_k^*(i))=q_i$.

The particle filter method is illustrated in Figure 5.

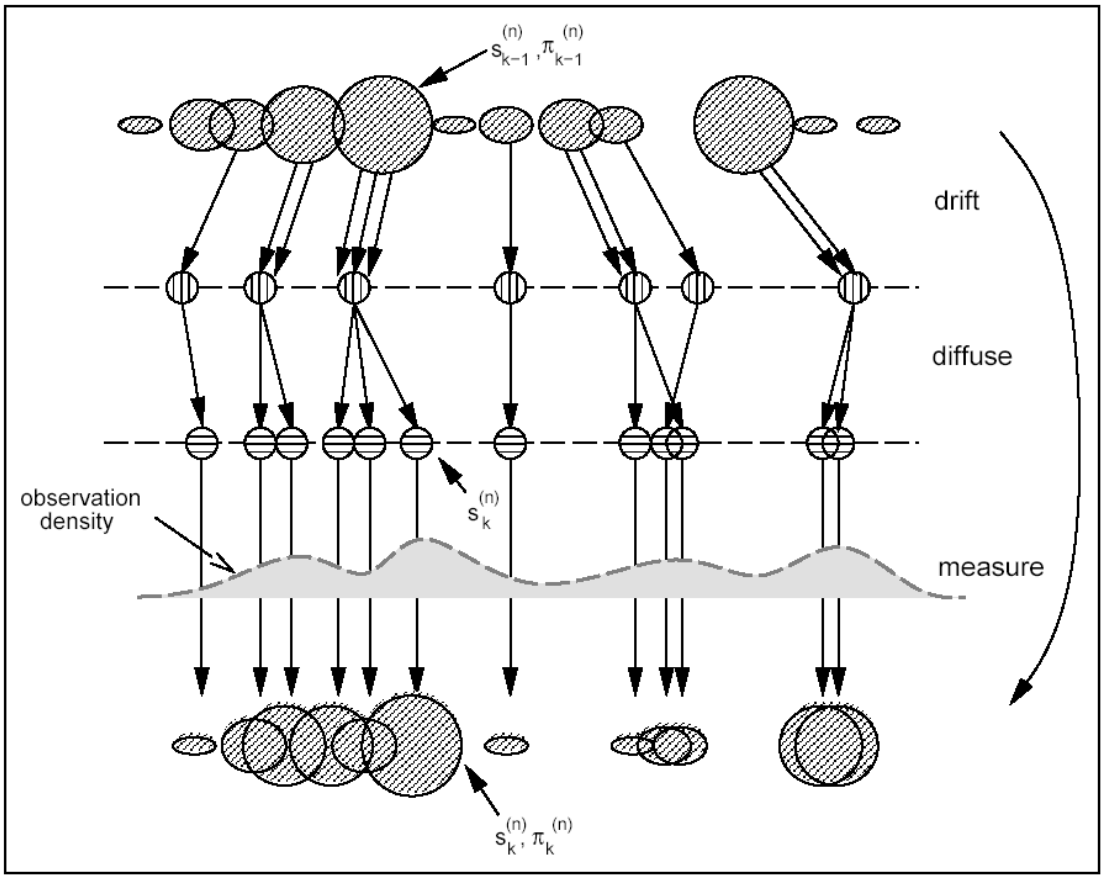



**Figure 5:** One time step in the particle filter algorithm for a one-dimensional density. Each of the three steps, drift, diffuse, measure, of the probabilistic propagation process of Figure 4 is represented by steps in the particle filter algorithm. Each blob represents a particle, and its dimension is proportional to its weight. Multiple arrows starting from a blob mean that the particle is sampled more times.

The justification for the update phase relies on the following theorem [35]. Suppose that samples $\{\mathbf{x}_k^*(i):i=1,\ldots,N\}$ are available from a continuous density function $G(x)$ and that samples are required from the PDF proportional to

$L(x)G(x)$, where $L(x)$ is a known function. Then a sample drawn from the discrete distribution over $\{\mathbf{x}_k^*(i):i=1,\ldots,N\}$ with probability mass $L(\mathbf{x}_k^*(i))/\sum L(\mathbf{x}_k^*(j))$ on $\mathbf{x}_k^*(i)$, tends in distribution to the required density. If $G(x)$ is identified with $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ (the prior) and $L(x)$ with $p(\mathbf{z}_k|\mathbf{x}_k)$ (the likelihood), then this theorem provides a justification for the update procedure.

There are also problems that can arise when using a particle filter. If the region of the state space where the likelihood takes significant values is small in comparison with the region where the prior is significant, many of the samples will receive a very small weighting and will not be selected in the resampling procedure. Thus, samples of the prior remote from the likelihood are effectively wasted, and those nearby are reselected many times. The problem is exacerbated if the likelihood falls in a region of low prior density, where there are relatively few samples. In an extreme case, all of the N samples may collapse to a single value. The problem can be overcome by brute force approach, by employing an enormous number of samples, or by other more subtle approaches. Such an approach is presented in [32], where particle filters were combined with mean shift to move the particles toward the density modes of the posterior, thus avoiding the degeneracy problem.

### 5.2. Tracking with particle filters

Particle filters were applied to tracking in an algorithm called CONDENSATION, for conditional density propagation. Objects are modeled as curves, parameterized in terms of B-splines. The configuration of the spline is restricted to a shape space of vectors $\mathbf{X}$ that allow affine deformations of a template shape. The object dynamics are modeled as a second order linear process:

$$\mathbf{x}_t - \bar{\mathbf{x}} = A(\mathbf{x}_t - \bar{\mathbf{x}}) + B\mathbf{v}_t$$

where $\mathbf{x}_t=(\mathbf{X}_{t-1}\quad\mathbf{X}_t)^{\mathrm{T}}$ and $\bar{\mathbf{x}}$ is the mean value of the state. $A$, $B$ and $\bar{\mathbf{x}}$ are estimated from data using the EM algorithm.

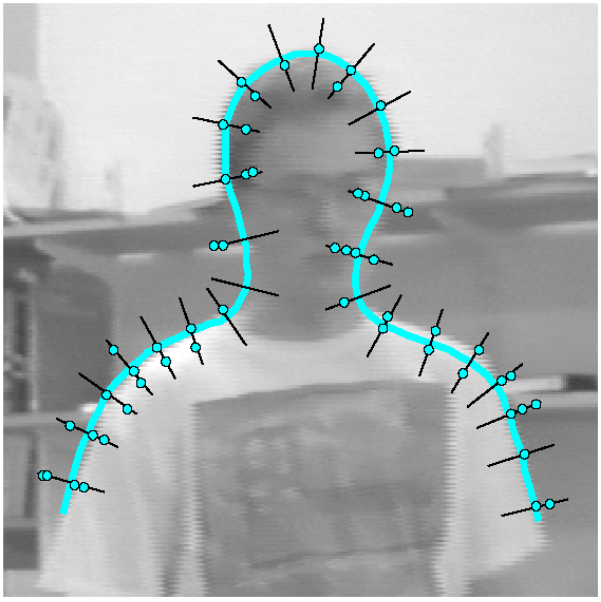


**Figure 6:** Observation process in the CONDENSATION algorithm: The thick line is a hypothesized shape, represented as a parametric spline curve. The spines are curve normals along which high contrast features are sought

The observation model considers the possibility of clutter, so that there can be multiple measurements for one target or no measurements. The measurements are restricted to features detected along a sparse set of lines normal to the tracked curve, as in Figure 6. The observation density proposed, $p(\mathbf{z}|\mathbf{x})$, is the product of one dimensional densities evaluated independently along $M$ curve normals:

$$p(\mathbf{z}\,|\,\mathbf{x}) \propto \exp\left(-\sum_{m=1}^{M}\frac{1}{2rM} f(\mathbf{z}_1(s_m)-\mathbf{r}(s_m);\mu)\right)$$

where:

- $r$ is a variance constant
- $f(\mu; v)=\min(\mu^2, v^2)$
- $s_m=m/M$
- $\mathbf{r}(s)$, $0\le s\le 1$, is the hypothesized curve represented by the shape parameter $\mathbf{X}$
- $\mathbf{z}_1(s)$ is the closest associated feature to $\mathbf{r}(s)$
- $\mu=\sqrt{2}\sigma\log(1/\sqrt{2\pi}\sigma q\lambda)$ is a spatial scale constant
- $\sigma=\sqrt{rM}$
- $q$ is the probability that the target is not visible

- $\lambda$ is the spatial density of the clutter, which is assumed to be a Poisson process with this parameter.

The experiments include tracking a sequence with three people moving around and shows the ability of particle filters to handle multi-modal distributions and multiple targets (Figure 7).

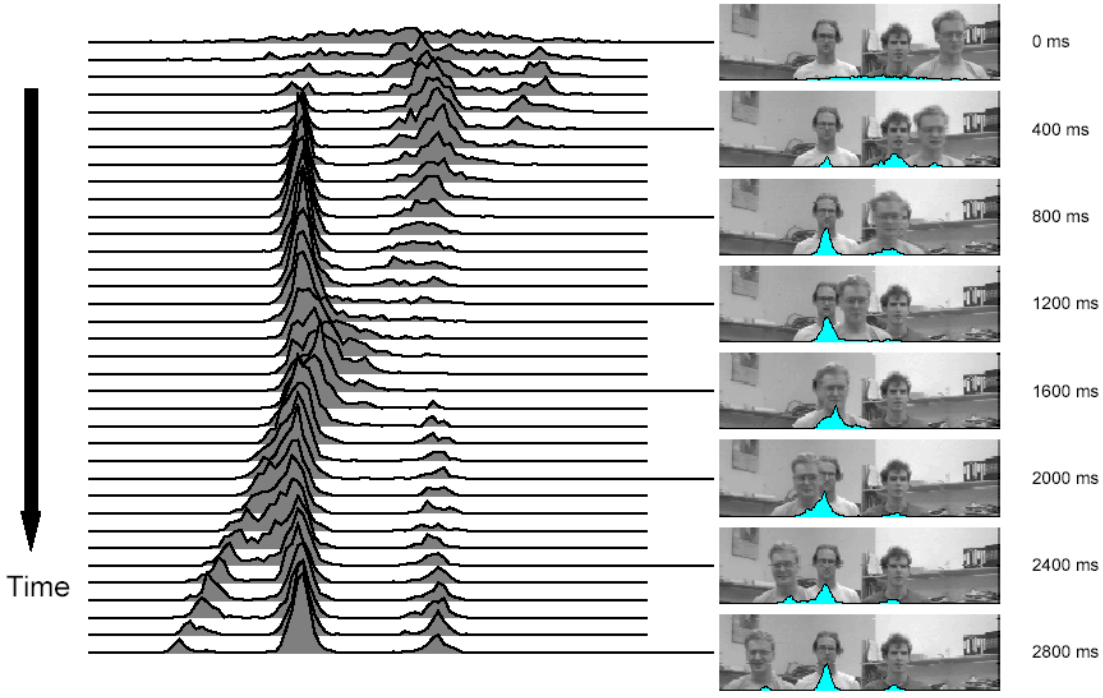



**Figure 7:** An approximate depiction of the state density in the CONDENSATION algorithm. The density is multi-dimensional, but its projection onto the horizontal translation axis is shown here.

### 5.3. Improvements

In another tracking paper [10], particle filters have been combined with active contours based on level sets [45], and the dynamic model was learned from a training set using an autoregressive process.

To improve robustness of tracking, the integration of multiple types of features has been proposed, in a framework that assumes conditional dependence of the features [7]. It is assumed that there are $F$ particle filters, each having its

own state vector $\mathbf{x}_k$ and feature vector $\mathbf{z}_k$, with feature $k$ dependend on features 1,…,$k$-1. The overall posterior can then be computed sequentially as

$P=p(\mathbf{X}|\mathbf{Z})=p(\mathbf{x}_1,\ldots,\mathbf{x}_F|\mathbf{z}_1,\ldots,\mathbf{z}_F)$

$=p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{x}_2|\mathbf{x}_1,\mathbf{z}_1,\mathbf{z}_2)\ldots p(\mathbf{x}_F|\mathbf{x}_1,\ldots,\mathbf{x}_{F-1},\mathbf{z}_1,\ldots,\mathbf{z}_F)=p_1p_2\ldots p_F$

Three types of features are under consideration:

- normals to the Fisher plane, a 2D color model constructed such as to maximize the separation between the color points of the object and background
- color distributions of target and background, each represented as a mixture of Gaussians
- contour of the object, represented by the array of points of the contour in the image.

The color distribution is considered dependent on the Fisher plane, and the contour is dependent on the Fisher plane and the color distribution. The output of the contour particle filter is used as an initialization for a snake fitting procedure, since the dynamic model allows only affine deformations of the contour.

The performance of the method was demonstrated in difficult synthetic and real sequences on which simpler particle filter implementations have failed.

### 5.4. Particle filters for multiple targets

When multiple targets have to be tracked there are two basic approaches. The simplest one is to have independent particle filters for each target. This leads to poor performance in case of targets occluding each other. This happens because the independence assumes the following two hypotheses [9]:

- targets move independently of each other

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \prod_i p(\mathbf{x}_t^i \mid \mathbf{x}_{t-1}^i)$$

- different targets contribute independently to the observation

$$p(\mathbf{z}_t \mid \mathbf{x}_t) \propto \prod_i p(\mathbf{z}_t \mid \mathbf{x}_t^i)$$

Under these assumptions, if the prior is separable, i.e.

$$p(\mathbf{x}_0) = \prod_i p(\mathbf{x}_0^i),$$

then the posterior distribution is separable:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) = \prod_i p(\mathbf{x}_t^i \mid \mathbf{z}_{1:t})$$

The first hypothesis can be accepted, with the caveats that it cannot model interacting targets and it assumes that the objects can compenetrate or they are very small in size. The second hypothesis, however, is unacceptable, because in case of occlusion the contribution of the occluded target to the observation is null.

The other possible approach for multiple targets is a joint particle filter, whose state is composed of the states of all targets. In this case there is an exponential growth in complexity of the particle filter and an exponential growth in the number of particles needed to represent the posterior.

# 6. MCMC and MRF for multiple targets

To keep the complexity of the particle filter manageable a Markov random field can be used, which models just the interaction of nearby targets, considering all others independent. The increase in the number of particles can be avoided by using Markov chain Monte Carlo sampling [8].

## 6.1. Markov Random Fields

An MRF is an undirected graph ($V$, $E$) where the nodes $V$ represent random variables and the edges $E$ represent a neighborhood system [37]. The joint probability over the random variables is factored as the product of local potential functions at each node and interaction potentials $\psi$ defined on neighborhood cliques. We can use a pairwise MRF, where the cliques are restricted to the pairs of nodes that are directly connected in the graph. We obtain the following motion model:

$$P(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \propto \prod_i P(\mathbf{x}_{it} \mid \mathbf{x}_{i(t-1)} \prod_{ij \in E} \psi(\mathbf{x}_{it}, \mathbf{x}_{jt})$$

where $\mathbf{x}_t = \{\mathbf{x}_{it}\}$, $i = 1, \ldots n$ is the state at time step $t$.

The pairwise potentials can be expressed by means of a penalty function $g$ that depends on the number of pixels overlap between the templates associated with the two targets:

$$\psi(\mathbf{x}_{it}, \mathbf{x}_{jt}) \propto \exp(-g(\mathbf{x}_{it}, \mathbf{x}_{jt}))$$

In this way we can penalize targets coming too close to each other.

A MRF with Monte Carlo sampling was also used in [30] to model the interaction between targets close to each other and thus avoiding representing the whole joint state space of all the targets, but there the interactions were not limited to two targets, because the MRF was general, not just pairwise.

### 6.2. Markov Chain Monte Carlo

MCMC [25] is a class of sampling algorithms that work by defining a Markov Chain [38] over the space of configurations $\mathbf{x}$ such that the stationary distribution of the chain is equal to a given target distribution. Metropolis-Hastings is such an algorithm [36]. The target distribution is the approximate posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. At each step we use this algorithm to generate a set of samples from it:

- start with an arbitrary initial configuration $\mathbf{x}_t^{(1)}$, then iterate for $s$=1...$N$-1:
- propose a new assignment $\mathbf{x}_t'$ by sampling from the proposal density $q(\mathbf{x}_t', \mathbf{x}_t^{(s)})$
- calculate the acceptance ratio

$$a = \frac{p(\mathbf{x}_t' \mid \mathbf{z}_{1:t})}{p(\mathbf{x}_t^{(s)} \mid \mathbf{z}_{1:t})} \frac{q(\mathbf{x}_t^{(s)}, \mathbf{x}_t')}{q(\mathbf{x}_t', \mathbf{x}_t^{(s)})}$$

- if $a \geq 1$ then accept $\mathbf{x}_t'$ and set $\mathbf{x}_t^{(s+1)} \leftarrow \mathbf{x}_t'$. Otherwise we accept $\mathbf{x}_t'$ with probability $a$, and reject otherwise, i.e. keep the current state $\mathbf{x}_t^{(s+1)} \leftarrow \mathbf{x}_t^{(s)}$

A simple proposal density is one that selects a single target $m$, chosen from all targets with uniform probability, and updates its state $\mathbf{x}_{tm}$ only by sampling from a single target proposal density $q(\mathbf{x}_{mt}', \mathbf{x}_{mt})$. This density could simply perturb the target state according to a zero mean normal distribution. This one target at a time scheme can be formalized by

$$q(\mathbf{x}'_t, \mathbf{x}_t) = \begin{cases} \frac{1}{n} q(\mathbf{x}'_{mt}, \mathbf{x}_{mt}) & \text{if } \mathbf{x}'_{-mt} = \mathbf{x}_{-mt} \\ 0 & \text{otherwise} \end{cases}$$

where $n$ is the number of targets and $\mathbf{x}_{-mt}$ denotes the joint state of all of the targets excluding target $m$.

### 6.3. Reversible Jump MCMC for varying number of targets

So far it was assumed that the number of targets in the scene is fixed. To model a variable number of targets we introduce a new variable, the set of identifiers $k_t$ of targets currently in view. Each possible set indexes a corresponding joint state space $\mathbf{x}_{k_t}$, its dimensionality determined by the cardinality of the set $k_t$. The motion model becomes now considerably more complex, as it includes the probability of targets entering and leaving the area. The state $(k_t, \mathbf{x}_{k_t})$ can be partitioned into targets $(k_E, \mathbf{x}_{k_E})$ that enter at the current time step and targets $(k_S, \mathbf{x}_{k_S})$ that stay:

$$p(k_t, \mathbf{x}_{k_t} \mid k_{t-1}, \mathbf{x}_{k_{t-1}}) = p(\mathbf{x}_{k_E})\, p(\mathbf{x}_{k_S} \mid \mathbf{x}_{k_{S(t-1)}})\, p(k_t \mid k_{t-1}, \mathbf{x}_{k_{(t-1)}})$$

In reversible jump MCMC [47], the Metropolis-Hastings algorithm is modified to define a Markov Chain over a variable dimension state space. The algorithm starts the chain in an arbitrary configuration. It then selects a move type $m$ from a finite set of moves that either increase the dimensionality of the state (birth), decrease it (death), or leave it unchanged. A move that changes the dimensionality of the state is referred to as a jump. Every jump must have a corresponding reverse jump defined, hence the name reversible jump MCMC.

We assume the existence of a noisy target detector that at each time step returns a set of identifiers $k_d$ of detected targets along with their estimated state $\mathbf{x}_{k_D}$. We can define the following possible moves:

- add: if a detected target is not yet in the identifier set $k_t$, propose adding it
- delete: randomly select an identifier from the set of detected targets that have already been added to $k_t$ and remove it
- stay: if a given target is no longer present in the current state, propose to re-add it and let it stay anyway. From the set of identifiers present at time $t$-1 but no longer present at time $t$, select a target with uniform probability and append it to the state.

- leave: randomly select an identifier from the set $k_t \backslash k_d$ and remove it from the current state of the sampler
- update: leaves the dimensionality of the state unchanged.

### 6.4. Tracking ants

The method was developed in the context of tracking ants in a confined, artificial environment. The ants can enter and leave this environment through a hole cut in the floor on which they move (see Figure 8).

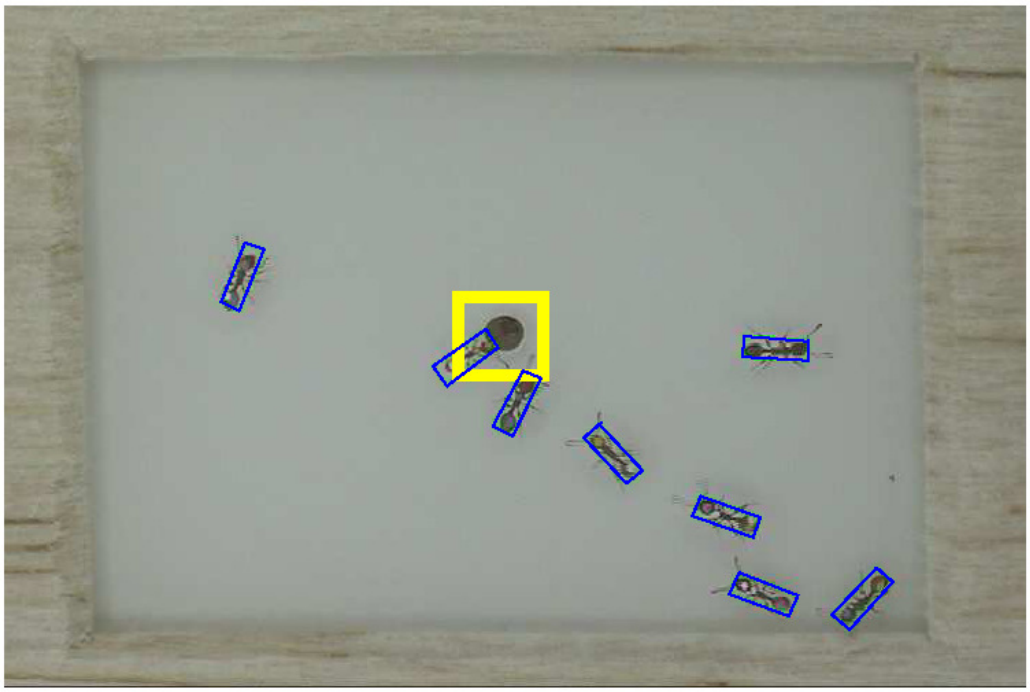


**Figure 8:** Ant tracking with MCMC and MRF

The fact that the environment was artificial and controlled has eliminated the possibility of background clutter. Moreover, for this particular problem one can assume that the objects do not occlude each other, since interactions occur in fact in 2D, but in general this assumption cannot be made. Experiments showed

increase performance of MCMC with MRF over joint and independent particle filters.

### 6.5. Improvements

MCMC was used in association with particle filters for more efficient sampling [26]. A hybrid particle filter starts as a particle filter but then each particle starts a Markov chain that converges to the posterior. Although a single Markov chain will eventually explore the entire state space, it often requires many samples to move between the different modes of the posterior. Multiple, independent chains are used to explore multiple modes more efficiently.

## 7. Linking identities to targets

The problem of linking identities to targets has been addressed with Bayesian network inference in the context of tracking a soccer game with fixed cameras [11, 15]. Four cameras were assembled in order to obtain a wide field of view image of the whole field (see the bottom image of Figure 2).

### 7.1. Building a track graph

The targets (players) in each frame are represented as a set of ellipses, identified by means of a background subtraction technique. Each ellipse may represent one or more targets. Based on these ellipses we construct a target interaction graph. The first task to this end is to put ellipses in frames $t$ and $t$+1 in correspondence. We can define three situation:

- Exact match: there is a one-to-one correspondence between an ellipse at time $t$ and an ellipse at time $t$+1. We consider that we have an exact match if the size and orientation of the ellipses are sufficiently similar and the displacement between their centers is sufficiently small
- An ellipse at time t splits into more ellipses at time $t$+1
- More ellipses at time t merge into one ellipse at time

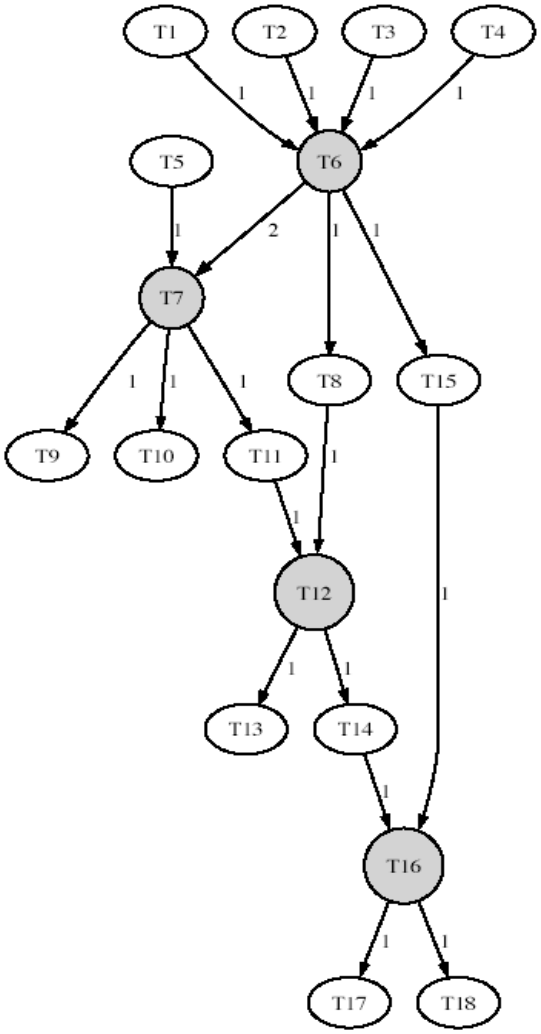



**Figure 9:** A track graph

Therefore at each time step one of five events can occur: *stable*, *merge*, *split*, *appear* or *disappear*. A maximal sequence of stable events is termed a track. During a track there is no change in the identity or number of the subjects represented by the ellipses concerned. If either of the following sequence of

events occurs: *track→split* or *merge→track*, the involved track corresponds to multiple players. However, when the involved track appears in the sequence {*split*, *appear*}→*track*→{*merge*, *disappear*} it may correspond to exactly one player. If the track has long enough duration and the size of the ellipses during the track is on average not too big then the track is considered to be a player track. The application of this analysis results in track graph summarizing the interactions that occur between the targets. Figure 9 shows such a track graph. Each node represents a track and the edges in the graph indicate when the targets from separate tracks merge to begin a new track or the targets in a track split to begin several new tracks.

### 7.2. Defining the state space

Each target's 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. Each split node has a state variable representing how the incoming targets are distributed into the outgoing tracks. 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. Let $N$ be the number of incoming targets for a particular split node and $m$ the number of outgoing tracks. Each track $i$ selects $n_i$ targets from the targets not yet selected. The number of different ways in which each track can select its targets is

$$\binom{N - \sum_{j=1}^{i-1} n_j}{n_i}$$

Hence, the total number of states of the split node is

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

Let each split node $T_i$ have a discrete state variable $S_i$. Moreover let $S=\{S_i;$ $T_i$ is a split node$\}$. Then $S$ can represent all possible solutions of paths given the track graph.

We also have to know how many targets there are in each link. Let $l_{ij}$ be the link count, i.e. the number of targets in the link between tracks $T_i$ and $T_j$. The link counts can be computed with the following procedure:

- Set all link counts to be undefined, $l_{ij}$ =0
- Set link counts of all links connecting two single tracks to one
- For nodes with all links but one defined, set the undefined link so that the number of targets in equals the number of targets out.
- Repeat 3 until no more links are updated.

### 7.3. Bayesian inference

Now we formulate the inference problem of finding the path of each target. It is assumed that the track graph is accompanied by feature vectors measured from each single target track. Let each single track $i$ have a feature vector $A_i$ and let $A=\{A_i;\ T_i$ is a single target track$\}$ be the set of all feature vectors.

We would like to infer the paths given the measurements using MAP,

$$\hat{S} = \arg\max_{S} \ p(S \mid A) .$$

Bayes' formula can be used to maximize the product of the prior and the likelihood function

$$p(S \mid A) \propto p(A \mid S)\, p(S)$$

Noting that every path ends at a tail node, i.e. a node with no outgoing links, we use the tail nodes as representatives for the paths. Let $A_{tails}=\{A_i;\ T_i$ is a tail node$\}$ be the set of tail node features. Further, let path$(A_i,s)=\{\ A_j;\ T_j$ are on the same path as $T_i$ given $S=s\}$ 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

$$p(A \mid S) = \prod_{A_i \in A_{tails}} p(path(A_i, s) \mid S = s)$$

The dependencies between state variables and feature vectors can be viewed in a Bayesian network, for which standard inference algorithms exist [43]. To reduce complexity, in the Bayesian network the dependencies between feature vectors and split nodes that are far away are dropped.

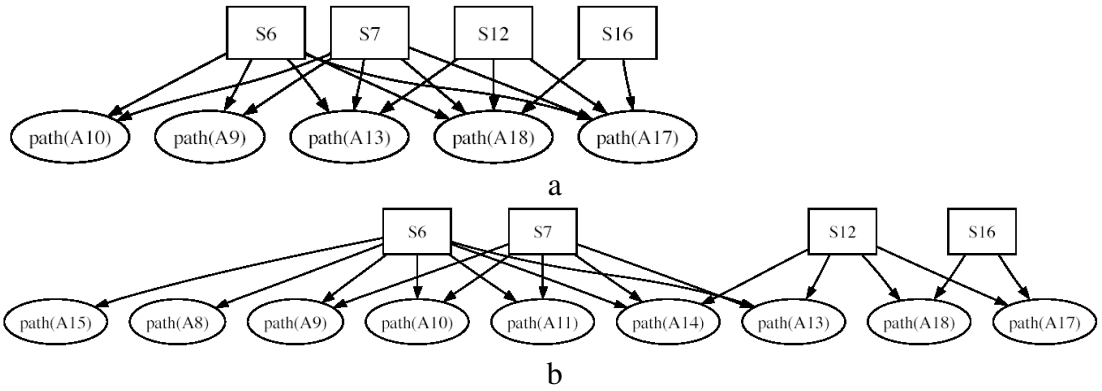



**Figure 10:** The Bayesian network for the track graph in Fig. 9 (a) and the reduced Bayesian network of the same track graph (b).

Figure 10 shows the Bayesian network corresponding to the track graph from Figure 9 and the reduced Bayesian network with some of the dependencies dropped.

The likelihoods can be computed as follows:

$$p(path(A_i, s) \mid S = s) \approx \prod_{A_i \in path(A_i, s) \setminus A_i} p(A_i, A_j)$$

where $p(A_i, A_j)$ is a pairwise measure of how likely it is that two tracks contain the same target.

To define this similarity measure we take into account that in the particular task under consideration, tracking soccer players, the player's identity is frequently revealed by his position relative to his teammates. For each single player track we build a histogram summarizing this information. A 4 dimensional vector records for each player the number of players to the left, right, in front and behind him. To reduce the number of possible values of the vector the range from 0 to 10 of each component is quantized into 6 bins each representing two consecutive values. A histogram is built from the values of this vector for the entire duration of the track.

Let $I_S^{ij}$ denote the similarity score between two tracks based on comparing their relative spatial position histograms. Such tracks are effective for matching tracks of long duration, when we can assume that the team is in a typical formation. However, many of the shorter tracks occur when the team is in transition between typical team formations. We define temporal local measures for these situations.

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 we consider the continuity of appearance and motion. The appearance measure relies on cross correlating the spatio-temporal volumes at the ends involved. This measure is denoted by $I_{\mathrm{i}}^{ij}$. The velocity of the targets at the ends of theses 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 the actual value is then used as the motion measure $I_{\mathrm{d}}^{ij}$.

The combined similarity measure is

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

with $0 \le \alpha \le 1$. We set $p(A_i, A_j) = \exp(-\lambda I^{ij})$, with $\lambda > 0$.

The novelty of this approach was that a long sequence of 10 minutes of a soccer game could be tracked, although with some failures when identities could not be correctly established. Another disadvantage of the method is that it has to be applied off-line, after the entire sequence has been captured. Some results are summarized in Figures 11 - 13.

### 7.4. Other approaches

A similar approach of associating foreground regions in successive frames was used in [31]. Tracking was done at two levels: region level and object level. At region level, tracking is done with the same reasoning about image regions merging and splitting from one frame to the next. At object level is where the two methods depart: in [31] the objects were assigned to regions by explicitly modeling for each object the appearance, spatial distribution and occlusion relationship with other objects. Experiments were presented only on short sequences of under 10 s.

The problem of linking identities to targets and visual tracking of groups was also addressed in [46].

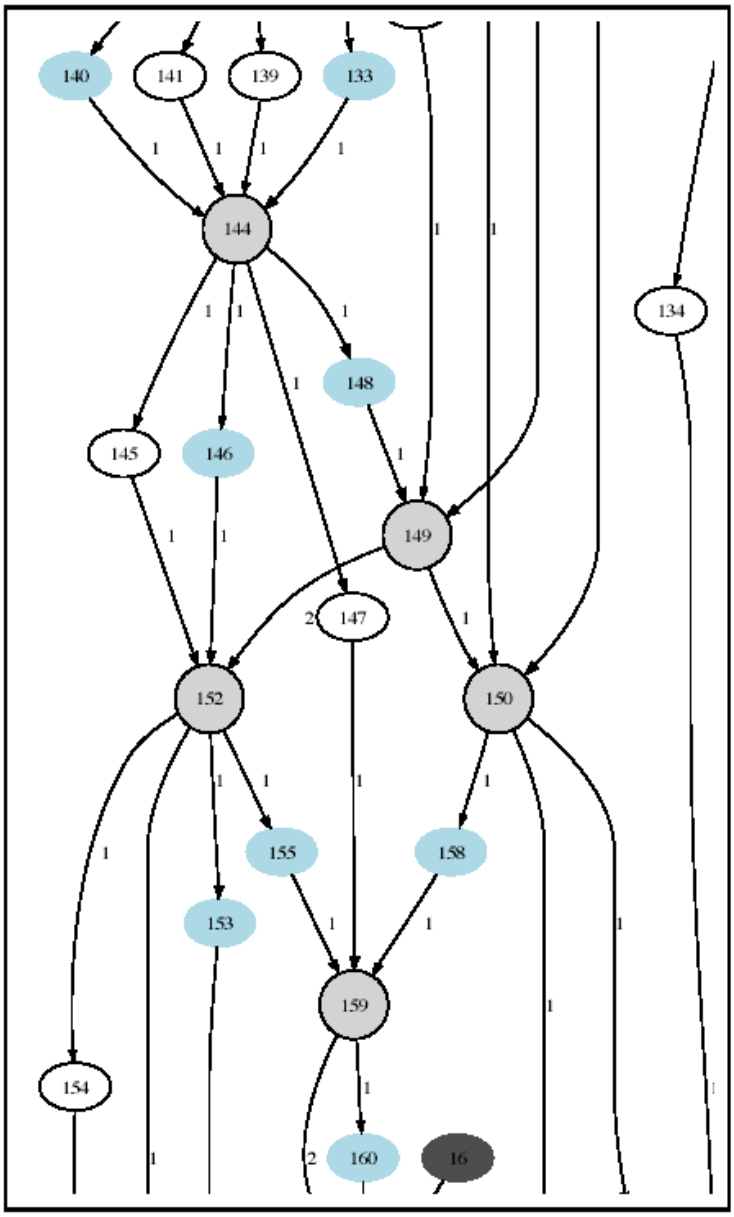



**Figure 11** Track graph obtained from video. The node colors correspond to team A (light blue oval), team B (white), referees (dark gray) and multi-target nodes (black).

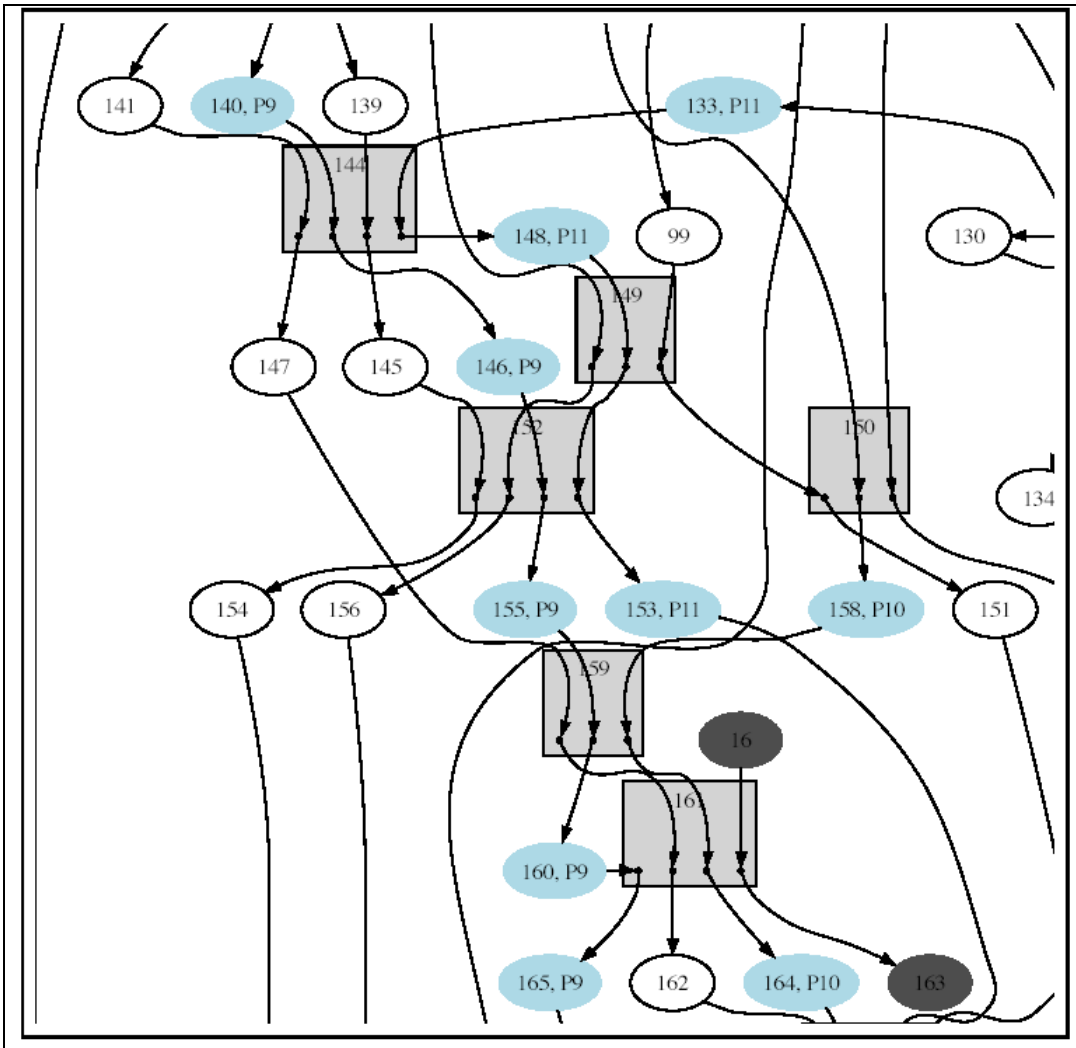



**Figure 12:** The resolved track graph. The square nodes display how the split nodes have been resolved. Ground truth player numbers can be seen for the team A players

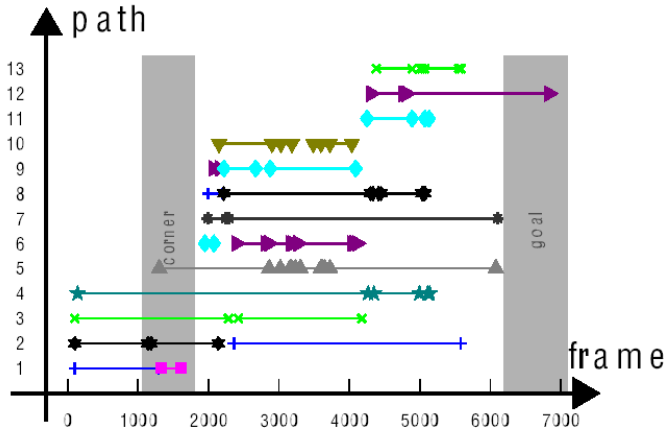



**Figure 13:** Estimated paths for the largest consistent part of the graph. Each line represents a single player track and each row an estimated path. The color and symbol denote the true identity of the track. Ideally there should be one color and symbol per row. On several occasions a target's trajectory is split into several paths. This is caused by links in the track graph having an undetermined number of targets.

### 7.5. Comparison

We presented a number of tracking techniques, starting with standard methods and ending with more sophisticated algorithms specially designed for tracking multiple targets in difficult situations while maintaining their identities. A comparison of the principal algorithms is presented in Table 1.

We can see that all the methods are problematic with occlusions, partial or total. The same discussion could be extended to illumination change and deformation, the performance decreasing in all of these cases. The Monte Carlo methods (particle filters and MCMC) are somewhat better by allowing multiple hypotheses about the object presence, but their likelihood will soon drop because of lack of observations. In the Sullivan et al. paper [11] this is also the main reason for failure: multiple targets on the same track that cannot be correctly resolved, i.e. occlusion. Explicit reasoning about occlusion was made in some approaches, which improved the tracking results [9, 18, 31]. The Monte Carlo methods, being non parametric, can represent multi-modal distributions, and

therefore can track multiple targets, but they cannot maintain their identities. In terms of running time, many of the methods can run in real time, with the observation that the number of particles plays an important role in Monte Carlo methods, a large number of them leading to better performance at the cost of longer running time. MCMC has an advantage here over particle filters. MCMC needs a smaller set of samples because of its mechanism of efficient sampling of the state space.

**Table 1**

| | Efficient exploration of state space | Robustness to occlusion | Ability to track multiple targets | Multi-modal distributions | Non-linear dynamics | Preserving identities | Real time |
|---|---|---|---|---|---|---|---|
| KLT | | No | Yes | | Yes | No | Yes |
| Mean Shift | Yes | No | No | Yes | Yes | No | Yes |
| Kalman filter | Yes | Some | No | No | No | No | Yes |
| Particle filters | No | Some | Yes | Yes | Yes | No | Yes |
| MCMC | Yes | Some | Yes | Yes | Yes | No | Yes |
| Sullivan | | Some | Yes | | Yes | Yes | No |

The methods presented here address one or some of the difficulties encountered in the task of tracking a varying number of human targets in an arbitrary environment, but a system that solves efficiently all the problems has still to be developed.

## 8. Collaborative human-computer tracking

Given the difficulties of multi-target tracking and inspired by the ideas behind the Computer Supported Cooperative Work field [41], we aim to build a joint human-computer system for tracking. This idea is also based on the observation that humans are good at tracking a small number of targets [40] in complex situations, like those described in the introduction, while algorithms that

can track a large number of targets exist, but they often fail in such complex situations. A joint human computer tracker would complement each other's weaknesses: a computer tracker would track a large number of trackers in the presence of a human agent. When a target undergoes occlusions, change of illumination, deformation, or for any other reason the confidence of the tracker in the estimate of the target position falls too low, the computer would emit a signal, thus asking for help from the human agent. Using available hardware for eye tracking, for example, the human could indicate with his or her gaze the true location of the lost target, thus reinforcing the computer tracker.

## 8.1. Human Multiple Object Tracking

We started our quest for the collaborative human computer tracker with an effort to better understand how humans perform the multiple object tracking task. Research in psychology has showed that human tracking performance considerably decreases with the number of targets, and it becomes irrelevant above 4 targets. Previous psychology experiments have been limited to simplistic 2D scenes of moving dots and lines. Therefore we were motivated to perform experiments with more realistic objects moving in 3D, thus leading to inter-target occlusion, illumination changes, etc.. These scenarios are equally important for psychologist and computer scientist, since the ultimate goal is tracking in real videos.

We started by creating 120 animations of 20 seconds each in 3Ds Max, which wee then rendered to videos of 600*800 pixels frame size. The video frame rate was 30 frames per second, thus yelding videos of 600 frames. We created an underwater scene with 9 identical fish, concretely sharks, moving randomly.The entire field of view is completely submerged The underwater scene presents some complex tracking situations:

- The underwater medium has a visibility attenuation effect. As the objects are farther from the viewer, not only they become smaller for the viewer, but harder to distinguish because of this effect
- The one light source, simulating the sun, in conjunction with the waves at the surface, creates parallel rays which cause change in illumination in various areas of the scene
- The ground presents some weeds which add to the complexity of the scene

- Although the model of the objects is identical, their random motion causes significant variation in appearance due to orientation change and scaling caused by distance from the viewer

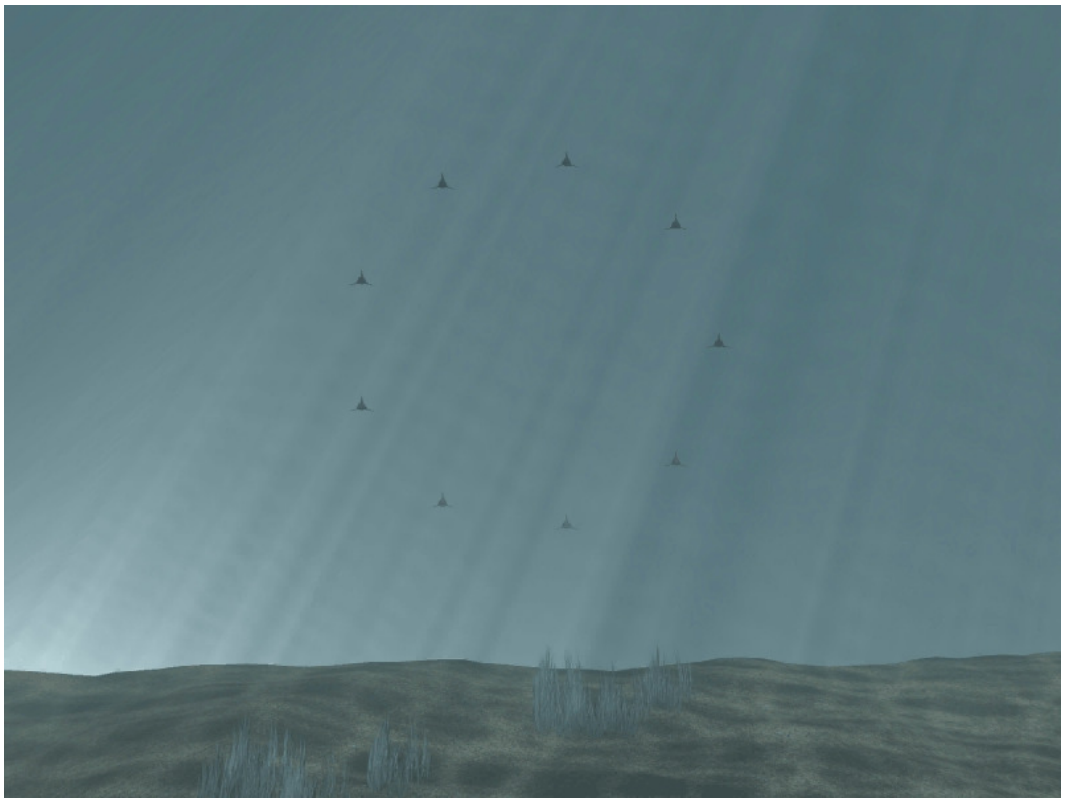


**Figure 14:** Shark scene – first frame

Still, all the scenes had to adhere to some rules of standardization, to make sure that other conditions do not impact the tracking results. The motion of all the objects had to be confined to the field of view, thus here we are not dealing with objects entering or exiting the scene. The initial arrangement of all the objects in the video had to be the same and regular, thus all the fish start arranged in a circle whose center is on the optical axis of the camera, equally distanced among themselves. Their orientation is with the tip of the head pointing towards the camera. This way the appearance of all the objects in the first frame is is the same. After the first frame the sharks start moving independently and randomly, but in a way that imitates the natural motion of fish.

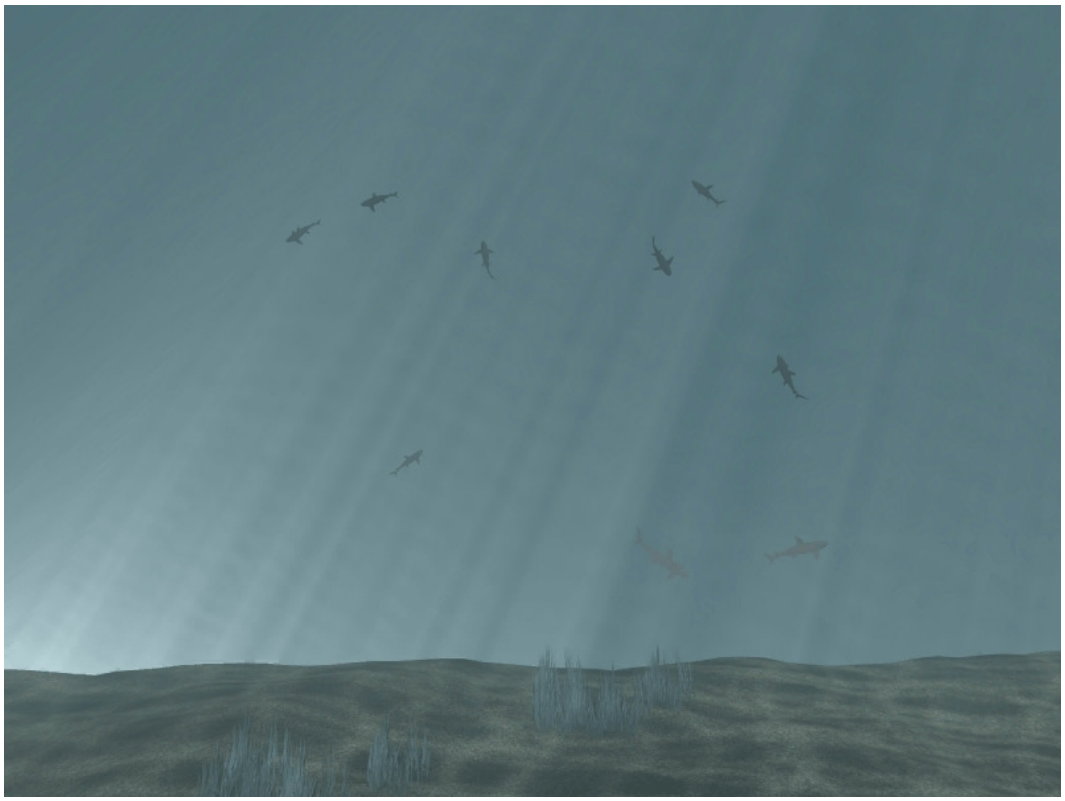


**Figure 15:** Underwater Scene – generic frame

Although 3ds Max presents some built-in automatic motion generators for groups of objects, this motion tends tens to become regular and the object start fo follow each other. For our purposes we had to implement a script function to automatically generate the motion of each shark independently. We give below the main steps for this function:

- Repeat for a fixed number of steps:
  - For each shark
    1. Assuming a constant velocity motion model and given the position in the previous frame, compute an initial proposed 3D position for the object
    2. Add to each 3D component of this position a random number generated uniformly
    3. Perform collision detection against the previously placed objects and walls, by checking for intersections between the 3D bounding boxes. If a collision is detected, go back to the first step

4. Compute the quaternion that represents the rotation of the object, using the constraint that the main axis of the object is aligned with the trajectory line. This constraint ensures that the sharks don't move sideways and they point to the direction where they are going.

The output of this algorithm is a set of control points for the sharks; the final trajectories are computed automatically by 3Ds Max by Bezier interpolation, thus obtaining a position and orientation of the objects in each frame. The number of steps of the algorithm is the number of control points, and thus it has to be set depending on the number of frames, the speed desired for the objects and the level of control for the trajectories.

We found that the resulting motion is close to natural motion, except at certain position where the objects had to make sharp turns to avoid collisions. In these positions the orientation exhibited some sudden, unnatural twists that had to be corrected manually. Given the large number of animations, this was a time consuming task.

After the manual correction we rendered each animation. We also computed and recorded the rendered 2D bounding box of each object for later analysis.

The videos of the animations obtained in this way were used as stimuli presented to subjects in tracking experiments. An EyeLink tracker was used to record the fixation of the subject's gaze during the experiment. The tracker consists of two cameras mounted in front of the subject's eyes on a helmet. The cameras are connected to a computer through an Ethernet connection. While the subject is looking at another computer's screen, where the stimuli are displayed, the eye tracker tracks the pupils and provides readings of the fixation in screen coordinates at 500 Hz.

For the subject experiments, the stimuli were analyzed and two sets of 56 videos each were selected to be presented to the subjects. Four types of experiments were designed. In track-1 experiments, subjects had to track one target. In the same manner there were track-2, track-3 and track-4 experiments. The rest of the targets in each video act as distractors. Each subject was asked to perform one type of experiment. One experiment consisted of two sets of 56 trials, each trial being tracking in one video. At the beginning of each trial the targets are identified by showing the first frame with a red circle drawn around some of the 9 sharks. The subjects are asked to follow all the designated targets. Then the circles disappear and video starts. At the end of the trial the final frame is kept on the screen with a red circle drawn around one of the sharks. The subject is asked if the indicated shark was one of the targets designated in the beginning.

In this manner we can asses to what degree subjects can track a number of targets. Around two subjects performed the experiment for each type of experiment.

Not surprisingly, the tracking performance decreased with the number of targets. More interesting findings concerning the way in which multiple target tracking is performed were discovered by superimposing the gaze fixation on the videos for each subject. For each frame, we computed the average gaze fixation and drew a small circle at that position. We also drew the centroid of each shark and the mass center of the targets. By analyzing visually these trajectories and comparing them for the four types of experiments we concluded that the tracking strategy changes according to the number of targets. Obviously, in the track-1 experiments, the gaze is fixated on the centroid of the target. For the track-2 experiments, most of the time the gaze seems to follow the center of mass of the two targets. For the track-3 and track-4 experiments, the gaze tends to switch among the centroids of the targets, jumping from one to another.

This interesting behavior led us to hypothesize that the jumps are related to problematic events such as occlusions, thus their name, occlusion saccades. We counted how many times such a saccade occurs towards an occluded target. We defined a saccade as the event that the gaze is at a certain distance away from the target before the occlusion and moves to the target during the occlusion. We found that about half of the saccades are related to occlusion.

### 8.2. Collaborative human-computer tracking of synthetic videos

We moved on towards the realization of a collaborative human-computer tracker by addressing the problem of tracking all the sharks in the synthetic videos collaboratively, while preserving their identities.

Our experiments with the basic trackers like KLT, mean shift and independent particle filters revealed that they were too limited for the task. The rate of failure, mostly due to occlusion, was too high for a human assistant to solve all the computer failures in real time. Instead we designed our own computer tracker, customized for the application. It is based on segmentation, particle filters and an extra component for explicit reasoning about occlusions. To it we ad the human interaction part, which is the eye tracker feeding as input to the computer tracker the eye fixation position on the screen. Using eye gaze as input has an important advantage over other input mechanisms due to the very small reaction time, which is important in real time applications.

At each moment the system is operating in one of two modes, referred to as normal mode and failure mode. In normal mode the computer part is in charge of the tracking and the eye tracker information is not taken into account. While in normal mode, at the same time with the actual tracking, the computer tracker evaluates the likelihood that it will experience problems in the future leading to the loss of a target. For the setting we had in mind such a loss can have two causes: either occlusion by other targets or too much similarity between a target and the background (the camouflage effect). In the case of an occlusion, the estimated position for one of the targets involved will probably go with the other target (the so called kidnapping phenomenon). In case of a camouflage, the estimation will stay on the background patch where the target was lost. To prevent these to happen, we prompt for human intervention. A visual flag is showed on the screen indicating the target in danger of being lost.

To predict a loss due to an occlusion, the proximity of other targets is analyzed. If two targets are close, but they have some features that can differentiate them, then there will be no flag, but instead if the occlusion actually happens, the features will be used to differentiate them after the occlusion. If the targets are similar and there are no such distinctive features, the flag will be showed. In our current system we use as a distinguishing feature the size of the targets. The actual occlusion might happen or the targets might just pass closely by each other without occluding, but we cannot know it for sure in advance. Only one of the two targets involved is flagged.

After the flag is put up, the tracker is in failure mode and the human operator is expected to move the gaze and fixate the target that was flagged until the difficulty was overcome. If it was a camouflage, the gaze is used to guide the tracker until the camouflaged target has moved away from the camouflage region. If it was an occlusion, when the two targets split again in the image, it is possible that one target kidnapped the estimation of the other one. If this happens, we reinitialize one of the individual trackers such that each of them follows one target. We do not know which was the kidnapped tracker, so we just pick one and assign it to the other target. If we picked the wrong one, their identities are now switched. But at this time the gaze should be following the target that was flagged at the beginning of the occlusion, so we can use this information to reestablish the correct identities. Because sometimes the gaze position we obtain from the eye tracker is not very accurate as result of improper calibration and the targets are close, we have to be careful in deciding which of the targets the gaze is pointing to. This is why we wait until the targets are relatively far apart and the distance from the gaze to one of them is considerably larger than to the other. At that point the eye fixation will be read and we switch back again the target identities if necessary.

For each frame, the processing starts with background subtraction and segmentation. After the background is subtracted the image is binarized by thresholding. The threshold is set at the intensity value 10. The number of objects is considered to be fixed and known. The tracking proceeds with each object in turn. For each object the corresponding connected component is determined, as well as its bounding box. The location of the objects in the first frames is assumed to be known. After this the initial seed for filling each connected component is given by the mean of a particle filter. A few pixels are checked around the center of the bounding box make sure that a foreground pixel is chosen. The checking proceeds from near to far: first the four pixels above, to the right, below and to the left are checked. Then the positions situated two pixels away in these directions, then three pixels away, and so on until a distance of ten is reached.

Each object has one particle filter. The motion model is a constant velocity model. The state vector is composed by the x-y coordinates of the center of the target and x-y components of the velocity. The tracker works on the intensity image obtained by subtracting the background from the current frame. In normal mode, i. e. no occlusion, the weight of a particle is given by the intensity of the pixel at the location of the particle.

The area of the resulting connected component is checked to make sure it is big enough to be considered one of the objects. The threshold is set at 10. If such a component is found by the process described above, the target will be considered in normal mode (OK). If a component is not found, the target is in color failure mode (FAIL_COLOR), because this happens when the color (intensity) of the foreground is very similar to the color (intensity) of the background. Each individual object tracker will work as a state machine, with the states and transitions described below.

- OK: normal mode. From this state there can be transitions to FAIL_COLOR, SHARK_CLOSE_FLAGGED or SHARK_CLOSE_NOT_FLAGGED. A transition to FAIL_COLOR occurs if a connected component was not found around the estimated position of the target. A transition to one of the other two states occurs if the following conditions are met:
    - in the bounding box of the connected component augmented with 10 pixels on each side there is at least part of another connected component, presumably of another shark which is getting close and might cause an occlusion. The tracker whose object corresponds to the second component is identified
    - the area of the whole second component is bigger than 50

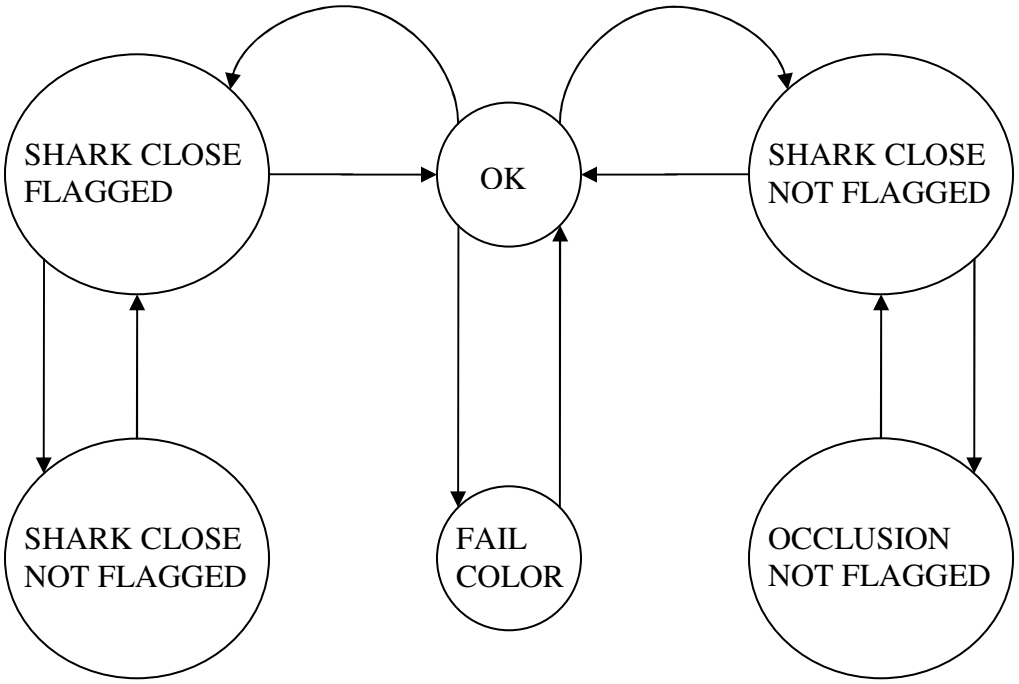



**Figure 16:** State transitions for the computer tracker

- the area of the first component is smaller than the area of the second. This means that during an occlusion the bigger object will stay in OK mode, and the smaller one will change the state
- If the ratio of the area of the first component to the area of the second component is less than 0.8 then the transition is to the SHARK_CLOSE_NOT_FLAGGED state; otherwise to the SHARK_CLOSE_FLAGGED state. The reason is that if the areas of the two components are different enough we can expect that their ordering does not change during the occlusion and we can restore the proper components to the appropriate trackers at the end of the occlusion based on the relative sizes of the components. If their areas are similar we are dealing with a more difficult case and we will require human intervention

- FAIL_COLOR: The target was temporarily lost due to camouflage. The tracker maintains the position from the previous frame. In some cases the tracker is able to recover by itself, but in others it is not. This situation should be dealt with with human intervention. The only transition considered is back to OK state,

although in practice there could be occlusions along with the camouflage, but this would complicate the possible transitions.

- SHARK_CLOSE_NOT_FLAGGED: Another object is close to the tracker's object, but the situation is not severe and can be solved without human intervention. The tracker can go back to OK state if the second object disappears, or it can go to OCCLUSION_NOT_FLAGGED if the two objects start occluding each other. This event is detected by considering the sizes of the first component in two consecutive frames. If there is an occlusion, it means that the connected components corresponding to the two objects have merged, and the size of each of them has largely increased. Because only the smaller object enters the SHARK_CLOSE_NOT_FLAGGED state and the area of the connected component is around the sum of the areas of the two components before the merge, we can require the ratio of the area of the component after the merge to the area before the merge to be larger than 2 for a transition
- OCCLUSION_NOT_FLAGGED: the object is behind another object, but the situation can be solved without human intervention. The only possible transition is to SHARK_CLOSE_NOT_FLAGGED. It happens when the connected component being tracked has split again in two components. This event is detected by considering again the ratio of the areas in two consecutive frames. This time we have to keep in mind that we have a good chance that we obtain the bigger connected component first, so the ratio of the area in the current frame to the one in the previous frame is required to be smaller than 0.9 for a transition. If this condition is met, the other connected component is searched for in the bounding box of the first one augmented with 10 pixels on each side. Once the two connected components have been found, they are attributed to the appropriate trackers. The smaller one is attributed to this individual tracker. However, there is a chance that the other individual tracker involved in the occlusion has already claimed the smaller component, while in fact it should be tracking the larger one. In this case the right connected component, the bigger one is assigned to the other tracker. This works if the other tracker was analyzed before the current one, but it might be that the other tracker will be analyzed after the current one and when its turn comes, it will be assigned the wrong component. This situation was solved by making the other tracker, which is in OK state, go to a special state, called SWITCH, in which the right component is assigned to it.
- SHARK_CLOSE_FLAGGED: Another object is close to the tracker's object, but the situation is severe and requires human intervention. In this case a flag is turned on. The transition to OCCLUSION_FLAGGED occurs in the same conditions as for the SHARK_CLOSE_NOT_FLAGGED state. For the transition to OK state we could rely on the eye gaze information to determine

when the two objects involved in the occlusion are far apart enough and the gaze is close to one of the two. When this happens, the tracker that was flagged will get the connected component which is closer to the eye gaze and the other tracker will get the other one (not implemented yet).

- OCCLUSION_FLAGGED: the object is behind another object, and the situation requires human intervention. The transition to SHARK_CLOSE_FLAGGED happens in a mode similar with the SHARK_CLOSE_NOT_FLAGGED case.

This algorithm does not address the problem of more than two targets being involved in an occlusion, which leads to some failures. Also, the robustness of the algorithm would be improved if it would be made aware of existing failures, like two trackers following the same target or one tracker being stuck in FAIL_COLOR mode.

The user interface of the system consists mainly of the video being displayed as the targets are tracked in real time. In normal mode and in the non-flagged conditions the tracker estimations are not shown to the user. When a target enters in SHARK_CLOSE_FLAGGED mode, a red rectangle is drawn around that target, and the user is instructed to fixate that target and follow it with the gaze. When the target enters the OCCLUSION_FLAGGED state, the rectangle is drawn around the combined foreground object which represents both objects. When the target goes in the SHARK_CLOSE_FLAGGED state again, a single rectangle which contains both targets is drawn, since we do not know which was the flagged one initially. The user's responsibility is to keep fixating the correct target after the occlusion. When the target enters the OK state again, the eye position is read and used to re-identify the targets and the box disappears.

If there are two simultaneous occlusions, i. e. the start of one occlusion is before the end of a previous one, only the first one is flagged for the user.

The system was tested in an experiment where a subject had to assist in tracking the first half of the 112 videos that were presented in the human tracking experiments. Figures 17 and 18 show the same frame after an occlusion between target 4 and target 5. In Figure 17, the computer tracker is running alone, and in Figure 18 the gaze is taken into account and corrects the switching of identities. The gaze position is indicated by the green dot.

The results for all the videos are synthesized in Table 2.

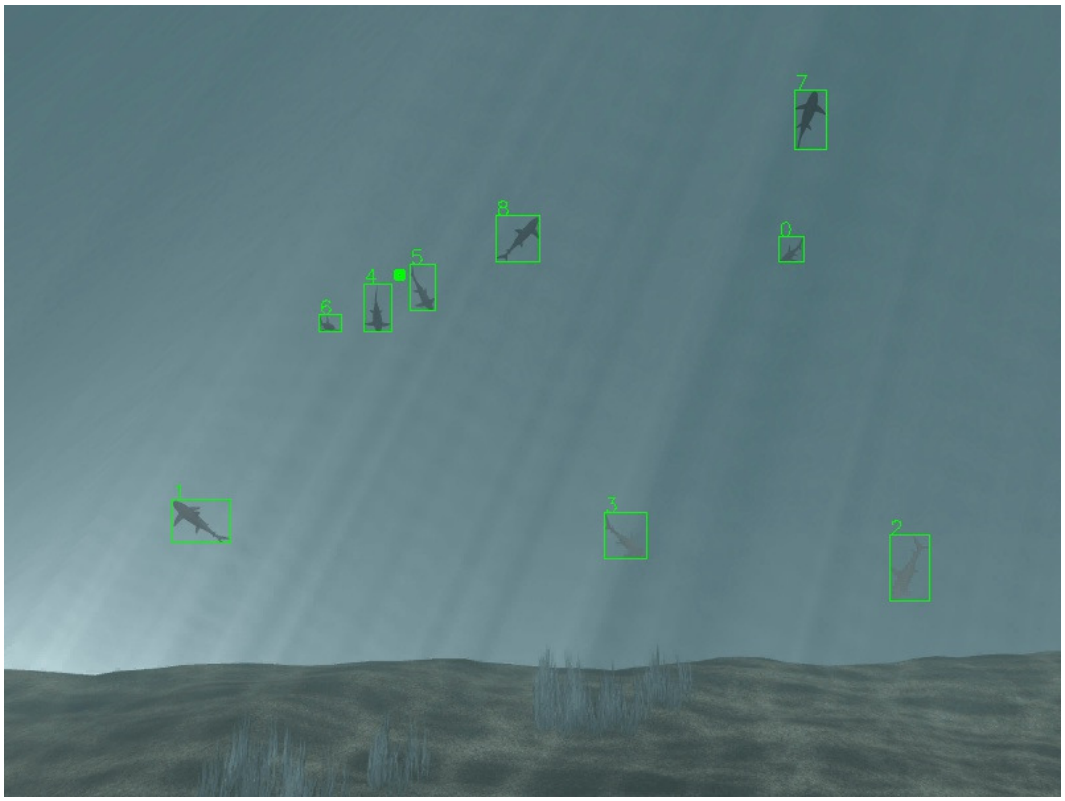



**Figure 17:** Tracking in synthetic sequences, computer only

There were a total number of 53 cases in which the human intervention was required by the system and the situation was tracked correctly as a result. Of these cases the intervention was required because of an occlusion 44 times and because of camouflage 9 times. Of the occlusion cases, the human intervention was essential in 11 situations, and the rest of occlusions could have been tracked through correctly by the computer tracker alone, which means that they were false flags. It is important to recall that not all occlusions require human intervention, but even so not for all the ones that do request it the human intervention is really needed. A better flagging decision in our system could help bring these two numbers closer. This could be achieved by introducing new features for assigning trackers to foreground detections after the occlusion and by playing with the value of the threshold for the size feature, which is currently used.

The total number of tracking failures for the system is 39. By failure we mean that the position estimated by the tracker for a target deviates from the ground truth. After a failure the tracker is not re-initialized. 30 of the failures are due to occlusion and the rest are due to camouflage. Of the occlusion failures, half

are explained by more than two targets being involved in the occlusion, a situation that the current system does not deal with.

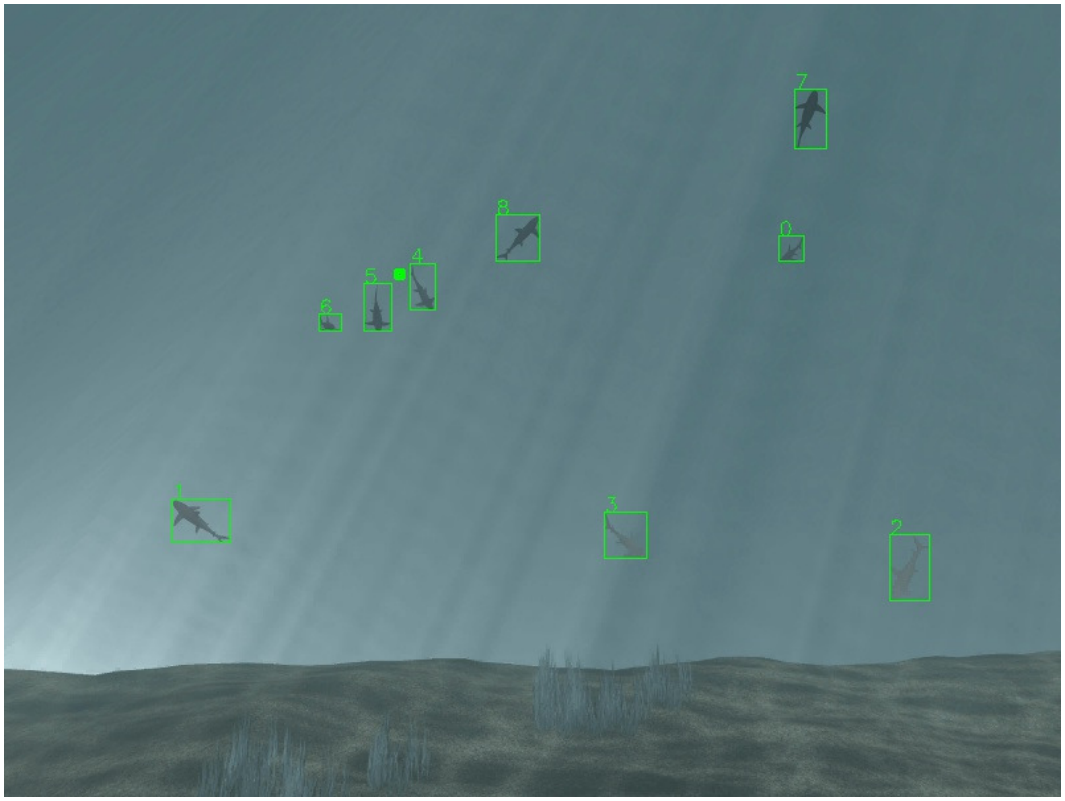



**Figure 18:** Collaborative tracking of synthetic sequences

The two target occlusions are further split into occlusions that were flagged and occlusions that were not flagged. The non-flagged ones are failures of the failure prediction mechanism and can be explained by either a previous failure, which carries on through the end of the video, or by an inversion of the relationship between the sizes of the two targets during the occlusion. Of the flagged failures, 4 are cases in which the computer tracker would have tracked correctly through the occlusion, but the gaze information was wrong because of poor calibration or imprecision. These can also be considered failures of the failure prediction mechanism. There are also 4 flagged cases in which the human input is neutral, that is it could not correct the computer tracker failure.

**Table 2:** Shark tracking results

| | | | | count |
|---|---|---|---|---|
| flagged cases solved by the system | | | | 53 |
| | Occlusion | | | 44 |
| | | human input helps | | 11 |
| | | human input is neutral | | 33 |
| | no occlusion | | | 9 |
| overall failures | | | | 39 |
| | Occlusion | | | 30 |
| | | 2 way | | 15 |
| | | | flagged | 8 |
| | | | | human input damages 4 |
| | | | | human input is neutral 4 |
| | | | not flagged | 7 |
| | | | | change in relative size 4 |
| | | | | previous failure 3 |
| | | 3 way | | 15 |
| | no occlusion | | | 9 |
| | | Flagged | | 6 |
| | | not flagged | | 3 |

The camouflage failures can also be split into failure prediction errors and errors due to bad response of the human assistant.

## 8.3. Collaborative human-computer tracking of real videos

The next step was towards tracking real scenes collaboratively. We identified a set of videos for which such a system would be most adequate. These were videos of a soccer game captured with several fixed cameras and constituted the dataset of the PETS 2003 workshop [48]. The dataset is available online at the address http://www.cvg.cs.rdg.ac.uk/VSPETS/vspets-db.html.

The technique we designed for our shark videos was not adequate because the segmentation we were obtaining for the real videos with the computer tracker was not reliable. The foreground objects were often split into more parts and false positives were introduced, whereas our technique needed crisp foreground/ background segmentation. We still wanted to use independent particle filters as the basic underlying technique with an ingenuous dealing of the interactions among the targets. Such a method was proposed in [49]. We will describe this technique in what follows.

The problem in tracking multiple targets with independent particle filters is the kidnapping phenomenon – the possibility that after an occlusion of a target by another target similar to the first both particle filters follow the same target. The proposed solution is to add an extra re-weighting stage to the particle filter update phase which decreases the weight of the particles that overlap with the particles of other targets.

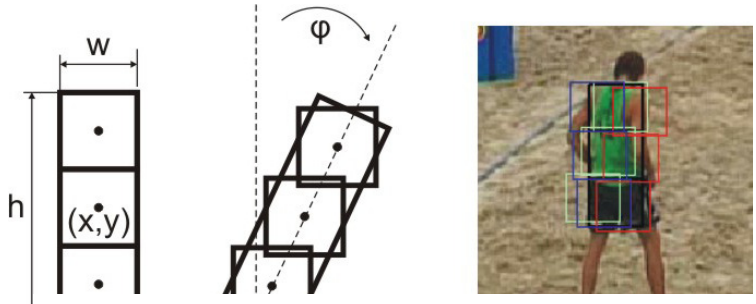



**Figure 19:** Motion model for players and fragmentation into subparts

The camera is assumed to be fixed. This is helpful because using a manual calibration step the homography between image coordinates and real world pitch coordinates can be estimated. A player is annotated as a reference during the initialization phase. Using the homography, the scale changes of every player can be estimated during the entire video based on the screen coordinates. The state model of the particle filter for each player is given by $\mathbf{x}_t = [x_t, y_t, v_x, v_y, \varphi_t]'$, where

($x$, $y$) are the center coordinates of a rectangle window, ($v_x$, $v_y$) are the velocities and φ is the rotation angle. Assuming a constant velocity dynamic model the transition probability $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is given by:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{v}_t$$

In this model the matrix **A**, together with the last position and velocities, defines the drift component of the particle transitions, while $\mathbf{v}_t = \mathrm{N}(0|\Sigma)$ is a normally distributed random component where $\sigma_x$, $\sigma_y$ and $\sigma_\varphi$ are the assumed variances. The possible scale changes of the objects are approximated by the homography.

To describe the measurement model we will use independent $N_B$-bin histograms for the 3 color channels in the RGB color space. In order to preserve spatial information of the appearance, each object is separated into subparts and each subpart is initialized with three reference histograms [$\mathbf{h}^R_{ref}$, $\mathbf{h}^G_{ref}$, $\mathbf{h}^B_{ref}$] for the color channels. In each subsequent frame, a candidate histogram [$\mathbf{h}^R_P$, $\mathbf{h}^G_P$, $\mathbf{h}^B_P$] is computed from each window represented by a particle and compared to the reference histograms. The Bhattacharrya similarity coefficient $D(\mathbf{h}_P, \mathbf{h}_{ref})$ is used as a similarity measure. The color channels are combined and the final likelihood is assumed to be exponential with normalization constant $\lambda$.

The candidate object windows are also divided into subparts, which can be rotated, see Figure 19. Assuming independence of the subparts appearance, the measurement likelihood for a particle with state **x** and consisting of $N_S$ subparts is finally computed by:

$$p(z_C \mid x) = \exp(-\lambda \cdot \sum_{j=1}^{N_S} \sum_{C \in \{H,S,V\}} D^2(\mathbf{h}^C_{j,p}, \mathbf{h}^C_{j,ref}))$$

The object is approximated improperly by the rotated subparts, but the error is compensated by the high number of particles and the weighted contribution of each particle in the final tracker result. In addition the spatial relation integrated into the likelihood computation of the particles by the spatial division leads to more stable tracking results. The number of used subparts and their spatial relation can be changed.

It is inevitable that some background pixels are always included in the tracking window. Some authors apply kernel or mask functions to take that into account. In this case, instead, a background probability $p(\mathbf{z}^B \mid \mathbf{x}^B)$ is included in the formulation of the measurement likelihood of the particles. A background image is computed offline in a preprocessing step, taking advantage of the static camera. With the addition of the background model, the final observation model for a particle is given by:

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}^C \mid \mathbf{x})}{p(\mathbf{z}^C \mid \mathbf{x}) + p(\mathbf{z}^B \mid \mathbf{x})}$$

For every particle with state $\mathbf{x}_t^i$ at time *t*, the background similarity is measured with $D(\mathbf{h}_{j,P}^i, \mathbf{h}_{j,B}^i)$ for every subpart. The histogram $\mathbf{h}_{j,P}^i$ is sampled from the actual frame and $\mathbf{h}_{j,B}^i$ is computed over the same area in the background image.

In order to prevent the kidnapping phenomenon, an extra stage is added to the particle weighting process using an overlap criterion.

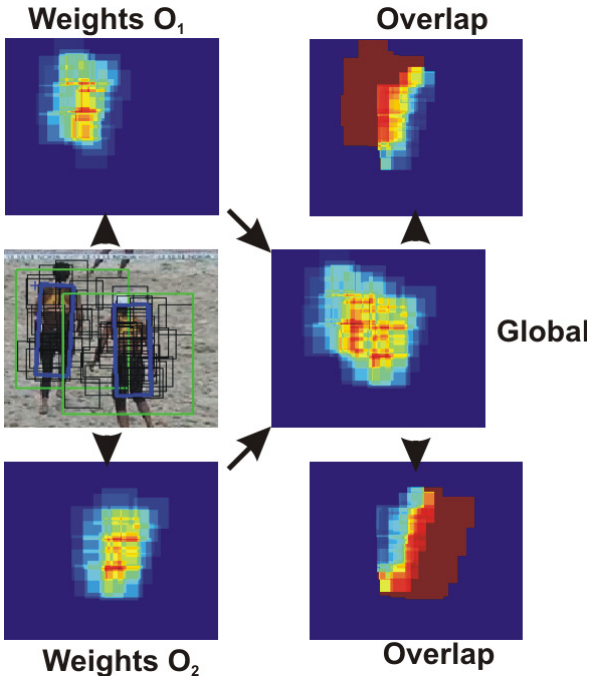



**Figure 20:** Overlap factor

The additional weighting step keeps the particle sets separated and prevents them from drifting to similar objects. This method also encourages separation of particles after total occlusion. In the normal particle filter update stage the particles are treated independently for each object and weighted. Then an individual weight map is constructed for each object. In this map we store at every coordinate the number of particles at that position, see Figure 20. At the same time we create a global weight map, which is the component-wise sum of all

individual weight maps. The ratio between a single object weight and the global weight gives an overlap factor in the range [0...1] for each coordinate and particle respectively. Finally the weight of each particle is multiplied with the overlap factor. If the particles of an object are separated from other objects, their overlap factor is 1 and their weights are not changed. Particles which share their location with particles form other objects get an overlap factor smaller than 1, so they get less weight and are less likely to be resampled. In case of a partial occlusion, the particles will concentrate in the non-overlapping areas. During a complete occlusion between objects, no separation is possible, so the particles will share the same image area. The effect of the reweighting step is illustrated in Figure 21 using an example sequence from the PETS 2003 dataset. Without the additional step particles of both objects drift to the strongest mode and one of the tracks is lost.

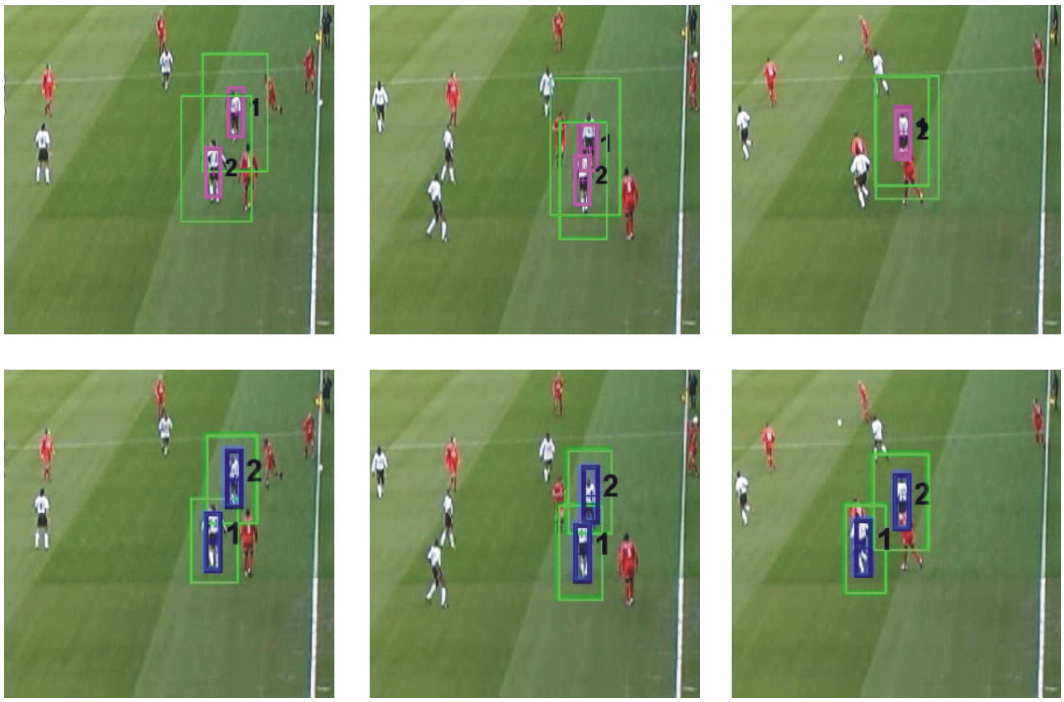
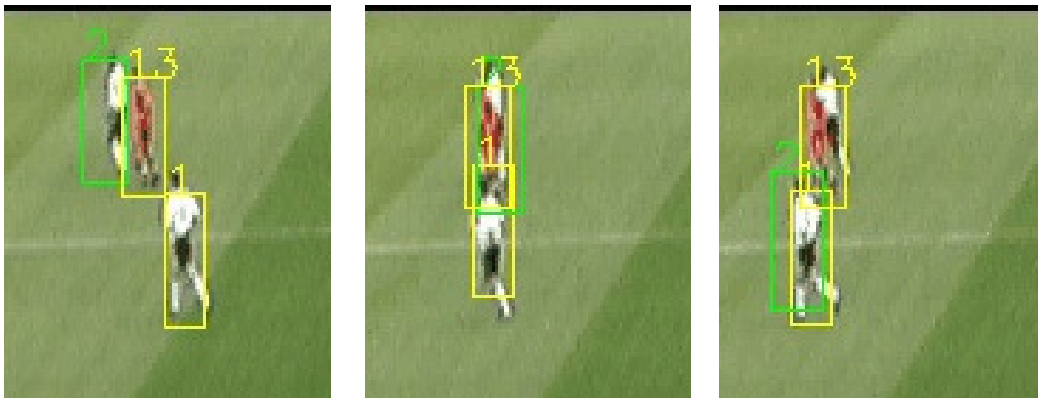


**Figure 21:** Typical results without reweighting (top) and with re-weighting (bottom)

We have decided not to include some features of the original tracker in our final system. The results reported by the authors were limited to a subset of the targets and a portion of the video, but our ambition was to track all the targets

during the entire video and get close to video frame rate processing speed. To increase this processing speed and because the players stay upright most of the time we eliminated the rotation component from the state vector, which allowed us to keep a reduced number of particles. We also eliminated the background modeling from the particle weighting because in our experiments it added a significant computational overhead without a relevant benefit. In exchange, we added a feature that helped some typical failures of the original system. The scenario is presented in Figure 22. Three targets are involved, two of one team (white) and one of the other (red). One white player is occluded by the red player. Because of the reweighting step, the weight of the particles of the occluded player decrease and the particles move to the other white player. Although the reweighting process still takes place, the weight of the particles that move to the wrong player is higher because the correct player is occluded and its appearance similarity is less than the other's.



Frame 50 Frame 66 Frame 70

**Figure 22:** Typical failure scenario

With these observation in mind we decided not to apply the reweighting process if the occluding player and the occluded player are of different teams. To distinguish this situation we use the similarity score given by the Bhattacharyya coefficient. If this score is smaller than a threshold, we decide that the target is occluded by an opposing player. The threshold for each player is computed automatically in the first 20 frames of the video, by averaging the Bhattacharyya coefficient over those frames.

To discourage even more the kidnapping scenario, in the reweighting step we multiply the initial weight by the overlap factor raised to the second power, while in the initial paper the overlap factor is not raised to any power.

We also introduced an overlap factor, denoted by the parameter R, which can control how close the samples have to be in order to consider that they overlap. The original paper corresponds to a factor of 1, where the size of the overlap map is the size of the image, and two particles overlap if they have exactly the same coordinates. When the overlap factor R is an integer number bigger than 1, the overlap map is scaled down by a factor of R, such that two particles will be considered as overlapping if they are less than R pixels apart.

The failure prediction for the soccer videos is more difficult because the motion of the target is often unpredictable and also because there are many occlusion cases which do not lead to failure, and signaling every possible future occlusion would give raise to far too many flags. Instead we tried to use again a threshold on the Bhattacharyya coefficient, but this also proved to be unsatisfactory because the time between the detection and the failure is too small to allow the attention shift of the user and because there are too many false flags. We decided to postpone this problem for the future and proceed to the experiments assuming that we know the failures beforehand. Practically this foreknowledge comes from running the computer tracker alone in a pre-processing step and observing the failures. Again, the trackers are not re-initialized after a failure.

We used 4 videos of the VS-PETS 2003 dataset. The videos are 2500 frames long and have a frame size of 720*576 pixels. The number of players in the field of view of the camera varies largely between 1 and 20. The initial location of the targets and the moment they enter the scene are specified manually. Targets that exit the scene and then enter again are assigned new identities. For each target a template histogram is computed automatically in the first frame. The failures are specified by the target identifier and frames in which they happen. The start frame is specified such that the user that will assist the computer tracker will have enough time to identify the target once it is indicated by the computer tracker as a future failure and to move the gaze to it. This time is around 1 second, but the time expressed in frames will vary depending on the frame rate that can be achieved by the computer tracker.

In all the videos there were 17 failures, but 3 of them were simultaneous with others and so could not be flagged. Because between failures there are long times where the user's attention is not solicited, we extracted from the initial videos short sequences of about 200 frames in which the failures happen.

**Figure 23:** Tracking of real videos, computer only

When a failure is specified in the configuration file, the computer tracker will signal this by showing the bounding box of the target. The user has to fixate that target. Once the indicated target is fixated, the subject has to press the right pad button. While the button is pressed, the fixation position is used as an extra cue for tracking, and the particles undergo an extra stage of reweighting, in which the final weight is influenced in equal proportion by the current weight and the closeness of the particle to the fixation position. The reweighting does not start until after 30 frames from the start frame indicted in the configuration file even if the button is pressed earlier. The reweighting stops if the user releases the button. An exception is if the gaze is closer to another target than to the target designated in the beginning. In this case, mouse input continues until the mouse follows the correct target again.

Because of calibration errors there might be a drift between the true fixated position on the screen and the position output by the eye tracker. The subject has the possibility to correct this drift. The eye tracker estimation is showed in green

on the screen as a circle. By pressing the A, B, X and Y pad buttons the estimation is shifted down, right left or up.

In our experiment, the user was able to correct 12 of the 14 flagged failures using this system. Figures 23 and 24 show the same frame after player 5 is occluded. In figure 23, the computer is tracking alone, and player 5 is kidnapped by player 14. Figure 24 shows the result using the collaborative tracker, where the gaze is used to correct the computer tracker. The estimated centers of the targets are showed with blue dots, and the estimated center of the flagged target is showed with the yellow dot. The larger blue square represents the gaze position.

**Figure 24:** Collaborative tracking of real videos

## 9. Tracking with occlusion cues

In this work we were motivated by the need to extend the system proposed in [7] to multiple targets. In that system, already presented in chapter 5, multiple cues are integrated to track objects robustly through difficult situations. The most important advance is that the conditional dependence between the cues is modeled. The cues are ordered such that a higher order feature is dependent on the lower order feature.

For tracking multiple targets through mutual occlusions we introduce a new type of cue, which we call the occlusion cue, which we will use in addition to the more classical position cue. We also model a bidirectional dependency between these two cues. The way in which this cue can help tracking through occlusion is that by using it we can know that a target is occluded by another target. In this situation we will compare the current window of the occluded target not to its appearance, but to a combination of the appearances of the occluded target and the occluding target, the blending factor being given by the occlusion cue. This is because we know that when a target is occluded we shouldn't expect to see the target, but its occluder. This idea of switching the appearance models depending on the occlusion state of the object is also exploited in [51].

We provide now the theoretical derivation of mutually dependent particle filters. Let us consider $N$ targets being tracked, each one using two particle filters, one with state vector $\mathbf{x}_{A,i}$ and the other one $\mathbf{x}_{B,i}$. The measurement is the same for both state vectors, denoted by $\mathbf{Z}_i$, $i$=1..$N$. As always, the goal is to find the posterior of the state of the overall system given the observation

$$p(\mathbf{x}_{A,1:N}, \mathbf{x}_{B,1:N} : \mathbf{Z}_{1:N}) = p(\mathbf{x}_{A,1:N} \mid \mathbf{x}_{B,1:N}, \mathbf{Z}_{1:N})\, p(\mathbf{x}_{B,1:N} \mid \mathbf{Z}_{1:N})$$

$$= \prod_{i=1}^{N} p(\mathbf{x}_{A,i} \mid \mathbf{x}_{B,1:N}, \mathbf{Z}_{1:N}) p(\mathbf{x}_{B,1:N} \mid \mathbf{Z}_{1:N})$$

$$= \prod_{i=1}^{N} p(\mathbf{x}_{A,i} \mid \mathbf{x}_{B,i}, \mathbf{Z}_i) p(\mathbf{x}_{B,1:N} \mid \mathbf{Z}_{1:N})$$

In the second step we have assumed conditional independence of positions given the occlusion state and the observations. In the third steps we have assumed conditional independence of position on the other targets occlusion state and observations given its own corresponding occlusion state and observation.

*prediction*:

$$p(\mathbf{x}_A^t \mid \mathbf{x}_B^{t-1}, \mathbf{Z}^{t_0:t-1}) = \int_{\mathbf{x}_A^{t-1}} p(\mathbf{x}_A^t, \mathbf{x}_A^{t-1} \mid \mathbf{x}_B^{t-1}, \mathbf{Z}^{t_0:t-1}) d\mathbf{x}_A^{t-1}$$

$$= \int_{\mathbf{x}_A^{t-1}} p(\mathbf{x}_A^t \mid \mathbf{x}_A^{t-1}, \mathbf{x}_B^{t-1}, \mathbf{Z}^{t_0:t-1}) p(\mathbf{x}_A^{t-1} \mid \mathbf{x}_B^{t-1}, \mathbf{Z}^{t_0:t-1}) d\mathbf{x}_A^{t-1}$$

$$= \int_{\mathbf{x}_A^{t-1}} p(\mathbf{x}_A^t \mid \mathbf{x}_A^{t-1}) p(\mathbf{x}_A^{t-1} \mid \mathbf{x}_B^{t-1}, \mathbf{Z}^{t_0:t-1}) d\mathbf{x}_A^{t-1}$$

Here we have applied the Chapman-Kolmogorov equation and the assumption of conditional independence of the current state on the previous state of the other cue and previous measurement, given the previous state.

*update*

$$p(\mathbf{x}_A^t \mid \mathbf{x}_B^t, \mathbf{Z}^{t_0:t}) = p(\mathbf{x}_A^t \mid \mathbf{x}_B^t, \mathbf{Z}^{t_0:t-1}, \mathbf{Z}^t) = \frac{p(\mathbf{x}_A^t, \mathbf{Z}^t \mid \mathbf{x}_B^t, \mathbf{Z}^{t_0:t-1})}{p(\mathbf{Z}^t \mid \mathbf{Z}^{t_0:t-1})}$$

$$\approx p(\mathbf{Z}^t \mid \mathbf{x}_A^t, \mathbf{x}_B^t, \mathbf{Z}^{t_0:t-1}) p(\mathbf{x}_A^t \mid \mathbf{x}_B^t, \mathbf{Z}^{t_0:t-1})$$

$$= p(\mathbf{Z}^t \mid \mathbf{x}_A^t, \mathbf{x}_B^t) p(\mathbf{x}_A^t \mid \mathbf{x}_B^t, \mathbf{Z}^{t_0:t-1})$$

Adopting the notation of [7] and denoting by $\otimes$ the convolution with the state dynamics and by $\times$ the multiplication by the observation density, the algorithm could be schematized as follows:

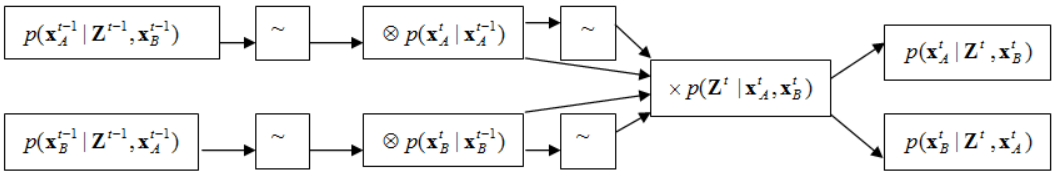



**Figure 25:** Dependent particle filters algorithm

Let us assume that there are $N$ objects to be tracked. Each target has a template represented by a mixture of Gaussians with K components

$$T_i = (N(\boldsymbol{\mu}_{i,1}, \boldsymbol{\Sigma}_{i,1}), p_{i,1}, ..., N(\boldsymbol{\mu}_{i,K}, \boldsymbol{\Sigma}_{i,K}), p_{i,K})$$

The MoGs represent the color appearance in RGB, thus the dimensionality of each component is 3.

The first cue is the position, by which in general we mean a bounding box of fixed or variable size, orientation or shape, but in the simplest case the state vector would encode the coordinates of its center. The second cue is the occlusion

cue, in which we want to record in what degree the target is occluded by other targets. The simple approach to this cue would be a state vector of $N$ components, in which component $i$ is the percentage of the current target occluded by target $i$. This approach has some limitations. First, it assumes that there is a fixed number of targets. Second, it leads to a waste of resources, since the vast majority of its components would be 0 most of the time. A more sophisticated solution would be to consider that there can be a fixed maximum number of targets all interacting with each other at a certain moment, and to have the state vector contain the identity of the occluding targets and the percentages of occlusion for each target. This creates a difficulty in the resampling process of particle filtering, a difficulty that is addressed by the Reversible Jump MCMC algorithm [8].

In any case, there is a constraint that has to be imposed, that the sum of all components of the occlusion state vector should be 1. It is not clear how this constraint should be maintained for a particle filter. For our experiments we considered that there are only 2 targets to be tracked, which occlude each other. In this case, the occlusion state vector has only one component, the percentage by which it is occluded by the other target and the unoccluded part is the difference up to 1.

The state vectors are for each target are position+velocity $\mathbf{x}_A$ and occlusion $\mathbf{x}_B$

$$\mathbf{x}_{A\mathbf{i}} = \begin{pmatrix} x_i & y_i & v_{x,i} & v_{y,i} \end{pmatrix}^T$$

$$\mathbf{x}_{B\mathbf{i}} = \begin{pmatrix} q_i \end{pmatrix}$$

Since the two cues are qualitatively different, we implemented them by two separate particle filters, PF-A and PF-B.

The measurement process follows the method of [7]. When there is no occlusion, the weight attributed to a PF-1 particle would be:

$$\pi_j^t \sim \frac{\sum_{(u,v)\in W_j^t} p(O \mid \mathbf{I}_j^t)}{n_W} - \frac{\sum_{(u,v)\notin W_j^t} p(O \mid \mathbf{I}_j^t)}{n_{\overline{W}}}$$

where $j$ is the particle index, $t$ is the time in frames, $u$ and $v$ are pixel coordinates, $W_j$ is the window represented by particle $j$, $\mathbf{I}$ is the color in the current frame at location $(u,v)$, $n_w$ is the number of pixels inside the window $W$ and $n_{\overline{W}}$ is the number of pixels outside the window. In our implementation, the window $W$ is a tight rectangle around the object, and we have an additional larger window, which

includes the first, of background pixels around the object. $p(O \mid \mathbf{I}_j^t)$ is the probability that a pixel is an object pixel, and is computed using Bayes' theorem:

$$p(O \mid \mathbf{I}) = \frac{p(\mathbf{I} \mid O)\,p(O)}{p(\mathbf{I} \mid O)\,p(O) + p(\mathbf{I} \mid B)\,p(B)}$$

Here $p(\mathbf{I}|O)$ is the probability given by the mixture distribution of the target:

$$p(\mathbf{I} \mid O) = \sum_{k=1}^{m_O} p(\mathbf{I} \mid k)\,p_k$$

where $m_o$ is the number of mixture components and $p_k$ are the component priors. The target mixture distributions are obtained in the first frame by giving the initial bounding rectangle of the targets and fitting a Gaussian mixture model to the pixels inside each rectangle using the expectation-maximization algorithm. Similarly there is a mixture distribution for the background, $p(\mathbf{I}|B)$, which is extracted offline in a similar manner from the background image. The object prior $p(O)$ is set to $n_{\overline{W}} / n_w$.

When the targets interact, however, in evaluating a candidate window for a target we will mix the template MoG of that target with the template MoGs of the other targets in the proportions given by PF-B. (Note that there is no need to make an explicit distinction between a no-occlusion state and an occlusion state; when there is no occlusion the occlusion state vector will be 0 or close to 0 and the tracker will be reduced to an independent tracker).

Then, to weigh a particle of PF-A, we select a particle of PF-B, and compute

$$p(\mathbf{Z}_j \mid \mathbf{x}_{A,j}, \mathbf{x}_{B,j}) = q_j\,p(O_i \mid \mathbf{I}) + (1 - q_j)\,p(O_{i \oplus 1} \mid \mathbf{I})$$

$$p(O \mid \mathbf{I}) \sim \frac{\sum_{(u,v) \in \mathbf{x}_A} p(O \mid \mathbf{I}(u,v))}{n_{\mathbf{x}_A}} - \frac{\sum_{(u,v) \notin x_A} p(O \mid \mathbf{I}(u,v))}{\overline{n}_{\mathbf{x}_A}}$$

Similarly we assign weights for the particles of PF-B: we select a particle of PF-A, which represents a window where we hypothesize the target may be. We compute two weights based on the pixels of that window, one for each target template. In the end we compute a final weight for the PF-B particle, in which each target contributes in the proportion given by the PF-B particle.

What this means is that as a target becomes occluded, the occlusion percentage in the occlusion state vector will increase and the tracker will start tracking the other target. To make sure that the tracker goes back to the right

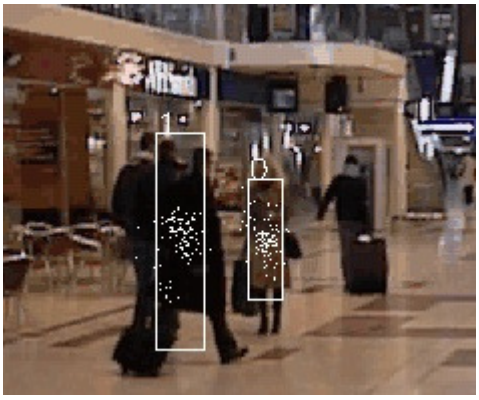
target after the occlusion we added an extra weighting stage to both PF-A and PF-B, in which the weight is reduced for the particles that have large occlusion.

We see that the same measurement process can be used for PF-B. This leads to the conclusion that the two particle filters are dependent, and the dependence is in both ways. In order to evaluate a position hypothesis, we need information about the occlusions it might be involved in, and in order to evaluate the occlusion hypotheses, we need to know in which window to evaluate them.

In the particle filter update stage, we see that the posterior of the object given the pixel is computed several times for the same pixel, for each sample window in which the pixel is contained. This prompts us to the idea of caching these values once computed and accessing them from the cache each time they are needed. Moreover, these values are always used as terms in sums over windows of pixels. The method of integral images presented in [50] provides a very efficient way of computing such sums using only three additive operations once the integral image is computed. In an integral image, instead of storing pixel intensities at each location, the cumulative intensity of all the pixels above and to the left of each location is stored. We use this idea, but instead of intensities we store in our integral matrix the computed, cached probabilities. In this way we achieve a significant speed up for the algorithm.

We tested our algorithm on a video from the PETS-2006 dataset, which presents a train station scenario. We tracked two targets with two particle filters each, one for position and one for occlusion. Each particle filter has 100 particles. Independent particle filters using only the appearance have difficulty tracking through complete occlusions when the occluded target is similar to the background in appearance, but our tracker correctly tracks the occluded target by mixing the appearance templates of the two targets. Figure 26 illustrates this situation. The small white dots represent position particles, and the rectangle is centered at the mean of the particles.

Frame 28



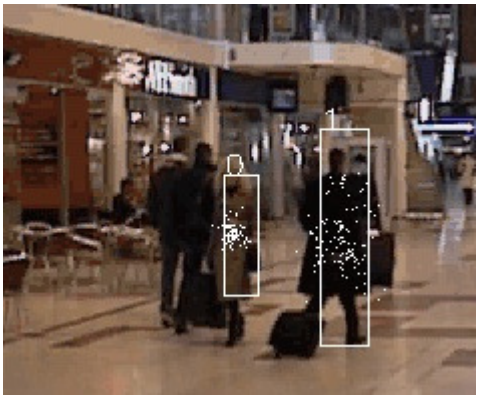
Frame 28

Frame 37

Frame 37

Frame 51

Independent appearance tracker



Frame 51

Occlusion cues tracker

**Figure 26:** Occlusion cues tracker results

## 10. Conclusion

In this work some standard techniques and more recent advances in tracking multiple targets have been presented and compared. The results of the methods based on these techniques still face some challenges. In consequence, some novel ideas for the field have been introduced and their effectiveness has been demonstrated. It was shown that bringing a human observer as an assistant to a computer tracker can improve the tracking performance, in situations in which neither a computer tracker alone nor a person can achieve satisfactory results in real time. Some problems have still to be addressed, such as an adequate failure prediction mechanism. Introducing occlusion cues in tracking can also reduce the number of tracking failures, but the extension of the method to more than two interacting targets has still to be addressed.

## References


1. Carlo Tomasi and Takeo Kanade; Detection and Tracking of Point Features; Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991
2. N.J. Gordon, D.J. Salmond and A.F.M. Smith; "Novel approach to nonlinear / non-Gaussian Bayesian state estimation"; IEE Proceedings-F (Radar and Signal Processing), pp. 140(2):107-- 113, 1993
3. Michael Isard and Andrew Blake; CONDENSATION -- conditional density propagation for visual tracking; Int. J. Computer Vision, 29, 1, 5--28, (1998)
4. Dorin Comaniciu, Peter Meer; Mean Shift: A Robust Approach Toward Feature Space Analysis; PAMI 2002
5. Dorin Comaniciu, Peter Meer, Visvanathan Ramesh; Real Time Tracking of Non-Rigid Objects Using Mean Shift; CVPR 2000
6. B. Han, D. Comaniciu, Y. Zhu, L. Davis; Kernel-Based Bayesian Filtering for Object Tracking; IEEE Conf. Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, 2005
7. F.Moreno-Noguer, A.Sanfeliu, D.Samaras; Integration of Conditionally Dependent Object Features for Robust Figure/Background Segmentation; Proc. International Conference on Computer Vision (ICCV), 2005

8. Z. Khan, T. Balch, F. Delaert; An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets; European Conference in Computer Vision 2004

9. O. Lanz and R. Manduchi; Hybrid Joint-Separable Multibody Tracking; CVPR 2005

10. Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi; Particle Filtering for Geometric Active Contours with Application to Tracking Moving and Deforming Objects; CVPR 2005

11. Peter Nillius, Josephine Sullivan, Stefan Carlsson; Multi-target Tracking – Linking identities using Bayesian Network Inference; CVPR 2006

12. Vasu Parameswaran, Visvanathan Ramesh, Imad Zoghlami; Tunable Kernels for Tracking; CVPR 2006

13. R. E. Kalman; A New Approach To Linear Filtering and Prediction Problems; Transaction of the ASME – Journal of Basic Engineering, March 1960

14. Iain Matthews, Takahiro Ishikawa, Simon Baker; The Template Update Problem; IEEE Transactions on Pattern Analysis and Machine Intelligence, June 2004

15. Josephine Sullivan, Stefan Carlsson; Tracking and Labeling of Interacting Multiple Targets; ECCV 2006

16. Vasu Parameswaran, Visvanathan Ramesh, Imad Zoghlami; Tunable Kernels for Tracking; CVPR 2006

17. Y. Ricquebourg, P. Bouthemy; Real-time tracking of moving persons by exploiting spatio-temporal image slices, IEEE Transactions on Pattern Analysis and Machine Intelligence, Aug. 2000

18. R. V. Babu, P. Perez, P. Bouthemy; Kernel Based Robust Tracking for Objects Undergoing Occlusion, Asian Conference on Computer Vision, 2006

19. R. Urtasun, D. J. Fleet, P. Fua; 3D People Tracking with Gaussian Process Dynamical Models; CVPR 2006

20. R. Urtasun, D. J. Fleet, P. Fua, A. Hertzmann; Priors for People Tracking from Small Training Sets; ICCV 2005

21. R. Nevatia, Tao Zhao; Tracking Multiple People in Complex Situations; PAMI, Sept. 2004

22. L. Taycher, G. Shakhnarovich, D. Demirdjian, Trevor Darrell, Conditional

Random People: Tracking Humans with CRFs and Grid Filters; CVPR 2006

23. A. Elgammal, R. Duraiswami, L. S. Davis, Probabilistic Tracking in Joint Feature-Spatial Spaces; CVPR 2003
24. M. Latzel, E. Darcourt, J. K. Tsotsos, People Tracking Using Robust Motion Detection and Estimation, Computer and Robot Vision 2005
25. R. M. Neal; Probabilistic Inference Using Markov Chain Monte Carlo Methods; Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto
26. K. Choo, D. J. Fleet; People Tracking Using Hybrid Monte Carlo Filtering; ICCV 2001
27. D. Freedman, M. W. Turek; Illumination Invariant Tracking via Graph Cuts; CVPR 2005
28. T. Zhang, D. Freedman; Tracking Objects Using Density Matching and Shape Priors; ICCV 2003
29. O. Boiman, M. Irani; Detecting Irregularities in Images and Video; ICCV 2005
30. T. Yu, Y. Wu; Collaborative Tracking of Multiple Targets; CVPR 2004
31. Y. Huang, I. Essa; Tracking Multiple Objects Through Occlusions; CVPR 2005
32. C. Chang, R. Ansari, A. Khokhar; Multiple Object Tracking with Kernel Particle Filter; CVPR 2005
33. D. A. Forsyth, J. Ponce; Computer Vision A Modern Approach; Prentice Hall, 2003
34. A. J. Yezzi, S. Soatto, Deformotion: Deforming Motion, Shape Average and the Joint Registration and Approximation of Structures in Images; IJCV, July 2003
35. A. F. M. Smith, A. E. Gelfand: Bayesian Statistics without Tears: a Sampling-Resampling Perspective; Amer. Stat., 1992, 46
36. Chib, Siddhartha and Edward Greenberg; Understanding the Metropolis–Hastings Algorithm; American Statistician, 49(4), 327–335, 1995
37. S. Z. Li; Markov Random Field Modeling in Computer Vision; Springer, 1995
38. J. G. Kemeny, J. L. Snell; Finite Markov Chains; Springer-Verlag, 1976

39. R. T. Collins, Y. Liu; On-Line Selection of Disciminative Tracking Features; CMU-RI-TR-03-12, The Robotics Institute; Carnegie Mellon University

40. B. Scholl, Z. Pylyshyn; Tracking Multiple Objects through Occlusion: Clues to Visual Objecthood; Cognitive Psychology, 38, 1999

41. L. Bannon, K. Schmidt; CSCW - Four Characters in Search of a Context. Studies in Computer Supported Cooperative Work - Theory, Practice and Design. J. M. Bowers and S. Benford. Amsterdam, 1991

42. J. Shi, J. Malik; Motion Segmentation and Tracking Using Normalized Cuts, ICCV 1998

43. C. Huang, A. Darwiche; Inference in Belief Networks: A Procedural Guide; International Journal of Approximate Reasoning, 15(3), 1996

44. J. M. Hammersley, D. C. Handscomb; Monte Carlo Methods; Chapman and Hall, London, 1964

45. D. Adalsteinsson, J. Sethian; A Fast Level Set Method for Propagating Interfaces; Journal of Computational Physics, vol. 118, 1995

46. G. Gennari, G. D. Hager; Probabilistic Data Association Methods in Visual Tracking of Groups; CVPR 2004

47. P. Green; Reversible Jump Markov Chain Monte Carlo Computation and Bayesian State Estimation; Biometrika, 82, 1995

48. International Workshop on Performance Evaluation of Tracking and Surveillance, Nice, October 2003

49. T. Mauthner, B. Horst; A Robust Multiple Object Tracking for Sport Applications; Performance Evaluation for Computer Vision, 31st AAPR/OAGM Workshop 2007

50. Paul Viola, Michael Jones; Robust Real-time Object Detection; International Journal of Computer Vision, 2001

51. M. Kaneko, J. Imai; Visual Tracking in Occlusion Environments by Autonomous Switching of Targets; IEICE Transactions on Information and Systems, 2008