# Robust Data Association

Vincent Lepetit Ali Shahrokni Pascal Fua
Computer Vision Laboratory
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
Email: {Vincent.Lepetit, Ali.Shahrokni, Pascal.Fua}@epfl.ch

## Abstract

*We present a novel method for performing data association that handles complex motion models and also increases the robustness of tracking. Instead of using a motion model defined recursively, we prefer to robustly fit the motion model over multiple frames simultaneously. This allows us to elegantly handle arbitrarily complex motion models, track initiation, occlusion and false alarms. This also improves the robustness to motion model weakness and to frequent false-negatives and false-positives, and the autonomy of the tracker. Our algorithm is easy to implement and we show its capabilities on two real examples of complex motion tracking.*

## 1. Introduction

All the numerous approaches to tracking are intrinsically recursive: at time $t$, a state containing the objects parameters is estimated, according to the observations in the coming frame and the state estimated at the previous step. This state contains the object position as well as object dynamics estimated from the sequence of images observed so far and can be regarded as a summary of observation in the images. The successive states are considered forming a Markov chain, and using a predictive motion model, the state at time $t$ is predicted from state estimated at time $t-1$ only.

The first drawback we see in the recursive expression of tracking is that the previous states are not updated according to the observations in new frames. If one such state does not reflect all the information present in the related frame, this could make the tracker fail. Of course, some solutions have been given. In Multiple Hypothesis Tracking, the several hypotheses of assignments between targets and measurements are simultaneously tracked, and by pruning the tree after a while, the correct assignments can be estimated in retrospect. Particle-set algorithms use multiple samples of the state to handle multiple hypotheses, and do not directly give a single estimated object position but require a batch-mode post-processing [1], which is obviously not suitable for online tracking.

Another disadvantage of traditional approaches is that motion model has to be expressed in a recursive way, that can be difficult or impossible. Usually, it is based on a local constancy assumption, like constant velocity and acceleration, and compensates the weakness of such models by adding white noise to the predicted positions. Nevertheless, that can be critical when occlusion or false negatives due to cluttered background happen in a large number of consecutive frames: The predicted position can be far from the actual position in case of agile motion for the target.

We propose a new formulation of probabilistic tracking, where the tracking is not performed frame by frame. Instead, a motion model is fitted to the detections in an interval of frames at the same time. This allow us to get a robust estimation of the motion. Furthermore the motion model can be arbitrarily complex, provided we can use enough frames forward and backwards at each time step. The usual motion models (with a recursive expression) can easily be expressed as required by our approach, so we can handle a larger class of motions than traditional recursive tracking.

Our new formulation computes the motion that maximizes a specified likelihood using a robust estimator to handle occlusion and false alarms. The mixture parameters involved in the likelihood expression are estimated iteratively and enforce the temporal coherence of the tracking. We will show how this can be performed online, with only a small delay between the acquisition of a frame and the corresponding output. This delay in our method can be compared to the delay involved in multiple hypothesis tracking to wait for hypothesis branches pruning [2, 3].

To summarize, our approach handles in the same formalism the following advantages:

- it easily deals with complex motion models ;
- it easily deals with abrupt motion changes ;
- it can easily handle a relatively large number of misdetections, even consecutive ones, and provides an accurate estimate of the target position even when a misdetection occurs ;
- the output of our algorithm is the actual target position, and not a density that requires a post-treatment.

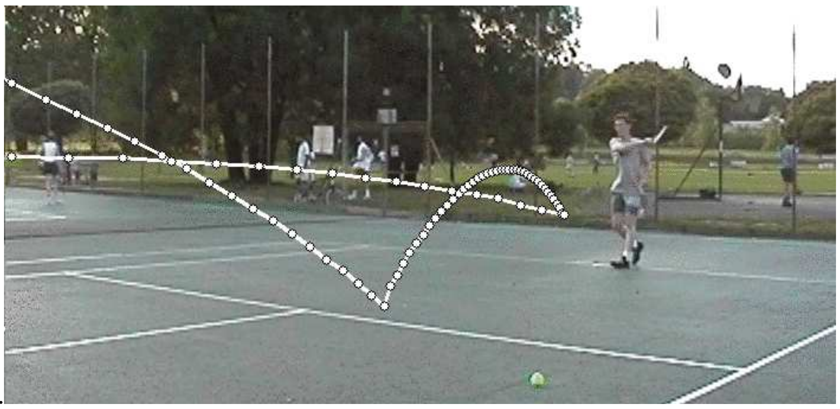
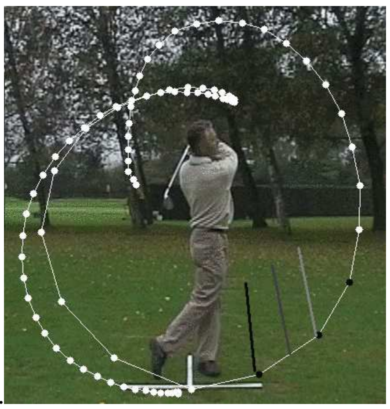
a.  b. 

Figure 1: Ball and golf club tracking with our method: (a) the ball tracking is performed without any manual initialization and is not perturbed by motion discontinuities which tend to break Kalman style approaches; (b) the golf club tracking is unperturbed by the mis-detections (corresponding to black disks) and the false-alarms, intentionally added to try to distract the tracker.

The paper is organized as follows. Section 2 introduces our approach. Section 3 demonstrates the applicability of our algorithm on two real world applications. Section 4 compares our approach with more traditional ones. Possible extensions of our method are discussed in the conclusion.

## 2. Robust Data Association

In this section we first state our problem and show how to robustly estimate the local motion. Then we discuss how the different steps of the algorithm can be performed online and we give a pseudo code description of this algorithm.

### 2.1. Problem Statement

Assuming that the time is discretized and frames are indexed by their acquisition time, let $\mathbf{z}_t = \{z_t^1 ... z_t^{n_t}\}$ be the set of hypotheses on the target generated by a detection algorithm for frame $t$. The $z_t^j$ are real vectors of dimension $N$. The true target location $y_t$ can be present in $\mathbf{z}_t$ or not, in case of failure of the detection algorithm.

We denote by $I_t$ the interval of frames acquired between time $t - n_B$ and $t + n_A$, and $\mathcal{Z}_t = \{\mathbf{z}_{t-n_B} \ldots \mathbf{z}_{t+n_A}\}$ the hypotheses for frames in $I_t$. We assume that the successive target positions $y_t$ satisfy a known motion model $\mathcal{M}$ , at least over the interval $I_t$. For each $t$, we want to determine which $z_t^j$ corresponds to the target $y_t$ or decide that a detection failure has occurred and give an estimate of $y_t$ in this case. We also want to do this by only considering the measures done over time to avoid any user interaction. In this paper we assume that only one target object satisfies $\mathcal{M}$.

### 2.2. Local Motion Definition

We want to estimate the target motion over $I_t$, i.e., the successive target positions $M_t = \{y_{t-n_B}, \ldots, y_{t+nA}\}$, or equivalently the motion model parameters. Since we consider only the frames in $I_t$, $M_t$ can be taken to be the motion with maximum posterior probability given the measures $\mathcal{Z}_t$, $\max_M p(M|\mathcal{Z}_t)$. According to Bayes' rule, we have:

$$M_t = \underset{M}{\operatorname{argmax}}\, p(M|\mathcal{Z}_t) = \underset{M}{\operatorname{argmax}} \left[ p(\mathcal{Z}_t|M) \frac{p(M)}{p(\mathcal{Z}_t)} \right]$$

We have some prior knowledge about the motion. Some motions, even if they satisfy the motion model, are physically impossible, e.g., because of unrealistic velocity or acceleration; in this case we set the prior $p(M) = 0$. For simplicity, all possible motions are given a uniform prior probability, but a better estimate could be used. $p(\mathcal{Z}_t)$ is constant, irrespective of $M$. So our new aim is to find

$$M_t = \underset{M}{\operatorname{argmax}}\, p(\mathcal{Z}_t|M)\delta(M)$$

where $\delta(M) = 1$ if the motion $M$ is possible and 0 otherwise.

### 2.3. Robust Estimation

A good way to estimate $M_t$ is to use a random sampling paradigm such as the robust estimators RANSAC [4] and MLESAC [5]. They are usually employed in computer vision for recovering epipolar geometry and estimating 3D motions from a set of matched points, but they are in fact general algorithms for robust estimation. Here, the random sampling approach is useful to handle mis-detections and false alarms.

The space of possible target motions is randomly sampled with $\mathsf{N}$ motion samples $M_\text{S}$ (with $\mathsf{N}$ being large enough, see section 2.7), computed from minimal sets of measures. Each set is made of $s$ randomly selected measures from different frames in $I_t$, which are utilised to predict a target trajectory $M_\text{S} = \{y_{t-n_B,\text{S}}, \ldots, y_{t+n_A,\text{S}}\}$ over

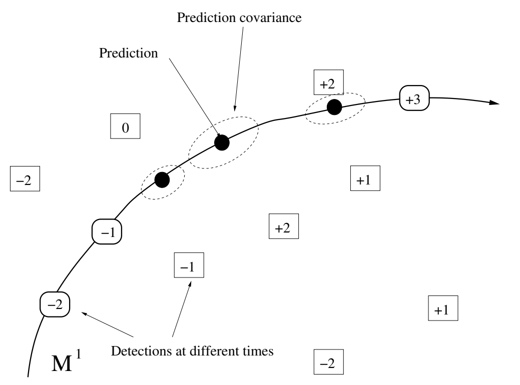


Figure 2: One motion used for the random motion space sampling, in the two-dimensional case. The motion has been computed from the hypotheses represented by rounded boxes.

$I_t$ as depicted by Figure 2. We keep the sample $\widetilde{M}$ that maximizes the "likelihood" over the motion space sampling:

$$\widetilde{M} = \underset{M_S}{\operatorname{argmax}}\, p(\mathcal{Z}_t|M_S)\delta(M_S). \qquad (1)$$

$\widetilde{M}$ is an initial estimate of $M_t$, computed from a minimal set of measures. It can be refined using more measures, as we will show section 2.6.

## 2.4. The Observation Model

As in classical tracking work, the measures are assumed to be independent, both mutually and with respect to the target motion, so that $p(\mathcal{Z}_t|M_S)$ in Equ. 1 can be written:

$$p(\mathcal{Z}_t|M_S) = \prod_{i=-n_B}^{+n_A} p(\mathbf{z}_{t+i}|y_{t+i,S}).$$

Note that $p(\mathbf{z}_{t+i}|y_{t+i,S})$ has the same form as the observation model in classical tracking methods, and we will refer here to common reasoning used in target tracking [6] to define its expression. In the following, the index $t+i$ is omitted for readability.

We make the following standard assumptions. The clutter is modeled with a Poisson process with spatial density $\lambda$ in the image. Any true target measurement is assumed to be unbiased and to have a normal distribution with covariance $\mathbf{\Lambda}_z$. We also consider $\mathbf{\Lambda}_y$, the covariance of $y_S$, since $y_S$ is computed from noisy measurements.

We compute the "innovation" $\nu^j$:

$$\nu^j = z^j - y_S$$

of all $n$ candidate measurements and their covariance

$$\mathbf{\Lambda}_\nu = \mathbf{\Lambda}_z + \mathbf{\Lambda}_y.$$

We finally get the observation density:

$$p(\mathbf{z}|y_S) = \gamma_0\lambda + \sum_{j=1}^{n} \gamma_j \frac{\exp\left(-\frac{1}{2}{\nu^j}^T \mathbf{\Lambda}_\nu^{-1} \nu^j\right)}{\sqrt{(2\pi)^N |\mathbf{\Lambda}_\nu|}},$$

where the $\gamma_j$ are mixture parameters (with $\sum_{j=0}^{n}\gamma_j = 1$).

## 2.5. Estimating The Mixture Parameters

The parameter $\gamma_{t,j}$ $(1 \leq j \leq n_t)$ can be interpreted as the probability that the target corresponds to measurement $z_t^j$ at time $t$, and $\gamma_{t,0}$ can be interpreted as the probability that a detection failure occurs at time $t$. A good initial guess for the $\gamma_{t,j}$ is: $\gamma_{t,0} = 1 - P_d$ if the target measurement is in $\mathbf{z}_t$ with probability $P_d$; $P_d$ reflects the performance of the target detector, and for $1 \leq j \leq n_t$, $\gamma_{t,j} = \frac{P_d}{n_t}$.

Next the $\gamma_{t,j}$ are re-estimated as follows. We introduce the indicator variables $\eta_t^j$, where $\eta_t^j = 1$ if the target corresponds to the $j$th measure at time $t$ ($\eta_t^0 = 1$ if a detection failure occurs), and $\eta_t^j = 0$ otherwise ($\eta_t^0 = 0$ if the target has been detected). We have:

$$\begin{aligned} p(\eta_t^j = 1|\gamma, \mathcal{Z}_t) &= \max_M p(\eta_t^j = 1, M|\gamma, \mathcal{Z}_t) \\ &= \max_M p(\eta_t^j = 1|M, \gamma, \mathcal{Z}_t) p(M|\gamma, \mathcal{Z}_t) \\ &\propto \max_M p(\eta_t^j = 1|M, \gamma, \mathcal{Z}_t) p(\mathcal{Z}_t|M, \gamma)\delta(M). \end{aligned}$$

The term $p(\mathcal{Z}_t|M,\gamma)\delta(M)$ is computed as before. $p(\eta_t^j = 1|M, \gamma, \mathcal{Z}_t)$ can be estimated as:

$$p(\eta_t^j = 1|M, \gamma, \mathcal{Z}_t) = \frac{p_j}{\sum_{k=0}^{n_t} p_k},$$

where $p_j$ is the likelihood of a measure given that is the target $(1 \leq j \leq n_t)$, and $p_0$ is the likelihood of a detection failure:

$$\begin{cases} p_0 = \gamma_0\lambda \\ p_j = \gamma_j \frac{\exp\left(-\frac{1}{2}{\nu^j}^T \mathbf{\Lambda}_\nu^{-1}\nu^j\right)}{\sqrt{(2\pi)^N|\mathbf{\Lambda}_z|}} \quad 1 \leq j \leq n_t. \end{cases}$$

The new estimates of the $\gamma_{t,j}$ are:

$$\gamma'_{t,j} = c.p(\eta_t^j = 1|\gamma, \mathcal{Z}_t),$$

in which $c$ ensures that $\sum_{j=0}^{n_t}\gamma'_{t,j} = 1$. This estimation should be iterated until convergence.

## 2.6. Estimating the Target Position

Once the final estimates $\gamma_j$ are computed, an initial estimate $M$ of the motion is computed using equation (1). This motion is computed from a minimal set of target positions and can be refined. Reference [5] minimises a robust cost function over all the input data. We prefer the method usually used after RANSAC: the motion is refined using the measures "close" to the predictions $y$ over the image interval $I_t$. A measure $z$ in a frame is considered to be close to the prediction $y$ in this frame if

$$\frac{\exp\left(-\frac{1}{2}(z-y)^T \mathbf{\Lambda}_\nu^{-1}(z-y)\right)}{\sqrt{(2\pi)^N |\mathbf{\Lambda}_\nu|}} < \chi^2(90\%, N).$$

If several measures are close to the prediction, only the closest one is kept. Finally, the refined motion provides a good prediction for the target position in frame $t$, and it is given as the output: if the target is not detected, it is a good estimation; otherwise this is usually a better estimation than the detection.

## 2.7. Number of Motion Samples

The number of motion samples $\mathsf{N}$ should be chosen sufficiently high to ensure (with a probability $p$) that at least one of the sets used for sampling contains only measures originating from the target. Literature on the RANSAC algorithm provides a formula to estimate $\mathsf{N}$ that we can easily adapt in our context: the probability that a selected measure in frame $t$ is originating from the target is $P_d/n_t$, and approximately equals $P_d/n$, with $n$ the average number of detections. Then at least $\mathsf{N}$ selections (i.e., $\mathsf{N}$ sets of $s$ measures) are required, where $(1-(P_d/n)^s)^{\mathsf{N}} = 1-p$, so $\mathsf{N}$ should be chosen as:

$$\mathsf{N} > \frac{\log(1-p)}{\log(1-(P_d/n)^s)}.$$

Numerical example: the values $p = 0.95, P_d = 0.9, n = 5, s = 3$ give $\mathsf{N} \geq 513$.

## 2.8. Algorithm

The recursive estimation of both the mixture parameters and the target position can be performed online, if we accept a small delay between the acquisition of a frame and the target position output. The number of re-estimations of the mixture parameters is fixed. For this description, we assume that only one re-estimation is done, but adding more iterations is straightforward. We also assume that $s = 3$ measures are required to estimate a motion.

Our algorithm can be summarized as follow. At time $t$:

- Detection is performed on frame $t$ ;
- Computation of the $\gamma'_j$ is performed on frame $t-n_A$ using the detections in frames $t-n_B-n_A$ to $t$ ;
- All the $\gamma'_j$ for frames between $t-n_B-2n_A$ and $t-n_A$ have been computed. Their values are employed to estimate the target position for frame $t-2n_A$.

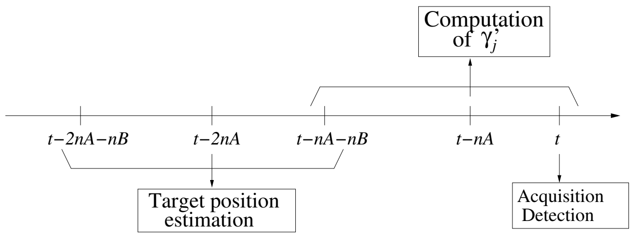


Figure 3: Actions performed at time $t$.

This is illustrated by Figure 3. A pseudo-code description is given below. For numerical stability, log likelihood motion samples is computed instead of likelihood, i.e., we compute:

$$\text{log-likelihood}(M) = \sum_{i=-n_B}^{+n_A} \log\left(p\left(\mathbf{z}_{t+i} | y_{t+i}, \mathrm{s}\right)\right).$$

```
Tracker()
  For each time $t$:
    Call detector on frame $t$ to generate $\mathbf{z}_t$
    ComputeThe$\gamma'$ForFrame($t - n_A$)
    ComputeTargetPositionForFrame($t - 2n_A$)
```

```
ComputeThe$\gamma'$ForFrame($t$)
  $\gamma'_{t,j} \leftarrow 0$   for $0 \leq j \leq n_t$
  Repeat N times:
    $M_\mathrm{S} \leftarrow$ GenerateOneMotionSampleForFrame($t$)
    If $M_\mathrm{S}$ is impossible (unrealistic velocity, etc...)
      continue
    $LLM_\mathrm{S} \leftarrow$ log-likelihood($M_\mathrm{S}$, $\gamma$)
    Compute the $p_j$, the measures likelihoods
    $sum \leftarrow \sum_{j=0}^{n_t} p_j$
    For $j = 0$ to $n_t$
      If $\gamma'_{t,j} < \frac{p_j}{sum}\exp(LLM_\mathrm{S})$
        $\gamma'_{t,j} \leftarrow \frac{p_j}{sum}\exp(LLM_\mathrm{S})$
  Normalise the $\gamma'_{t,j}$ such as $\sum \gamma'_{t,j} = 1$
```

```
ComputeTargetPositionForFrame($t$)
  $LLM_\mathrm{max} \leftarrow 0$
  Repeat N times:
    $M_\mathrm{S} \leftarrow$ GenerateOneMotionSampleForFrame($t$)
    If $M_\mathrm{S}$ is impossible (unrealistic velocity, etc...)
      continue
    $LLM_\mathrm{S} \leftarrow$ log-likelihood($M_\mathrm{S}$,$\gamma'$)
    If $LLM_\mathrm{S} > LLM_\mathrm{max}$
      $M \leftarrow M_\mathrm{S}$
      $LLM_\mathrm{max} \leftarrow LLM_\mathrm{S}$
  Refine $M$ (see section 2.6)
  Refined $M$ provides the target position in frame $t$
```

```
GenerateOneMotionSampleForFrame($t$)
    Select randomly 3 frames $(t_1, t_2, t_3)$ such as:
    $\begin{cases} t - n_B \leq t_k \leq t + n_A \quad k = 1,2,3; \\ t_1 \neq t_2, t_1 \neq t_3, t_2 \neq t_3. \end{cases}$
    Select randomly one measure $z_{t_k}$ for each $t_k$
    Compute $M$ from $(z_{t_1}, z_{t_2}, z_{t_3})$.
    Return $M$
```

## 3. Results

We show the applicability of our algorithm to two real world applications. The first one is tennis-ball tracking with a single camera, e.g., for broadcast enhancement purposes. The second one is golf-club tracking during a swing.

### 3.1. Tennis Ball Tracking

#### 3.1.1. Ball Detection

Ball detection consists of detecting moving objects, and keeping those that approximately match our expectation for color, size and aspect ratio for the ball. To this end we compute two masks. The first is derived by differencing current and previous frames and thresholding the result. The same operation is performed on current and next frames to obtain the second mask. A logical AND operation between these two masks gives the mask of the moving objects in the current frame. Some criteria on color and shape are applied to clusters in the resulting mask to keep only those that look like a tennis ball and to generate hypotheses on the ball position. Note that due to the relatively poor quality of the images the color test is not sufficient by itself to detect the typical tennis ball color.

#### 3.1.2. 2D-Motion Model of the Ball

We need to express the motion model and to compute the motion parameters from a set of measures. If the camera is in a "natural" position (image plane approximately perpendicular to the ground without tilt), the ball trajectory in the camera coordinate system can be expressed as:

$$\mathbf{M}(t) = \begin{bmatrix} X_0 & + & t\dot{X}_0 & & \\ Y_0 & + & t\dot{Y}_0 & - & t^2 g \\ Z_0 & + & t\dot{Z}_0 & & \end{bmatrix}$$

where $g$ is the acceleration due to gravity. The ball trajectory in the image is :

$$\begin{cases} u(t) = u_0 + \frac{\alpha_u (X_0 + t\dot{X}_0)}{Z_0 + t\dot{Z}_0} = u_0 + \frac{\alpha_u (\frac{X_0}{Z_0} + t\frac{\dot{X}_0}{Z_0})}{1 + t\frac{\dot{Z}_0}{Z_0}} \\ v(t) = v_0 + \frac{\alpha_v (Y_0 + t\dot{Y}_0 - t^2 g)}{Z_0 + t\dot{Z}_0} = v_0 + \frac{\alpha_v (\frac{Y_0}{Z_0} + t\frac{\dot{Y}_0}{Z_0} - t^2 \frac{g}{Z_0})}{1 + t\frac{\dot{Z}_0}{Z_0}} \end{cases}$$

where $u_0$, $v_0$, $\alpha_u$, $\alpha_v$ are the — unknown — camera internal parameters. In the following, we take $(u_0, v_0)$ to be at the image center. Given three ball 2D positions at different times $t_1$, $t_2$, $t_3$, the motion parameters can be computed by solving the linear system $\mathbf{AX} = \mathbf{B}$ with:

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 & 0 & 0 & 0 & -t_1 u(t_1) \\ 0 & 0 & 1 & t_1 & -t_1^2 & -t_1 v(t_1) \\ 1 & t_2 & 0 & 0 & 0 & -t_2 u(t_2) \\ 0 & 0 & 1 & t_2 & -t_2^2 & -t_2 v(t_2) \\ 1 & t_3 & 0 & 0 & 0 & -t_3 u(t_3) \\ 0 & 0 & 1 & t_3 & -t_3^2 & -t_3 v(t_3) \end{bmatrix}, \mathbf{X} = \begin{bmatrix} \alpha_u \frac{X_0}{Z_0} \\ \alpha_u \frac{\dot{X}_0}{Z_0} \\ \alpha_v \frac{Y_0}{Z_0} \\ \alpha_v \frac{\dot{Y}_0}{Z_0} \\ \alpha_v \frac{g}{Z_0} \\ \frac{\dot{Z}_0}{Z_0} \end{bmatrix}.$$

$$\mathbf{B} = [u(t_1) - u_0, v(t_1) - v_0, \cdots, v(t_3) - v_0]^T$$

The refined motion can be computed the same way, with more 2D positions as explained in section 2.6. Let $y = \Phi_t(\mathbf{B})$ be the ball position at time $t$ for this trajectory, the covariance matrix $\mathbf{\Lambda}_y$ of this position can be approximated as

$$\left[\frac{\partial \Phi_t}{\partial \mathbf{B}}\right] \begin{bmatrix} \sigma_u & & \\ & \ddots & \\ & & \sigma_v \end{bmatrix} \left[\frac{\partial \Phi_t}{\partial \mathbf{B}}\right]^T$$

where $\sigma_u$ and $\sigma_v$ are the standard deviation on measurements. The analytical expression for $\frac{\partial \Phi_t}{\partial \mathbf{B}}$ can then be easily derived using Maple, for example.

#### 3.1.3. Tracking Results

We demonstrate our ball tracker on a long video sequence composed of about 1000 deinterlaced frames. The tracking task is made difficult by the presence of different objects moving in the background. The average number of false detections is about 4 per frame. The ball sometimes goes near the camera, and the apparent displacement becomes large. On the other hand, the ball is sometimes far from the camera, and becomes too small to be detected. Since the ball goes in and outside the camera field, we use a simple test based on the likelihood on the refined motion: If it is lower than a threshold, we consider that the ball is not seen by the camera.

We use the values $n_B = n_A = 7$, and two re-estimations of the mixture parameters. As shown in Figure 5, this is usually enough for convergence. Figure 6 presents three motion samples used for a frame of the trajectory shown Figure 7.a, which have a likelihood respectively equal to 0.0077, 0.0036 and 0.00445. We verify on the whole sequence that the likelihoods of the samples that are close to the actual motion (like the first one) are effectively much higher than the likelihoods of the bad samples.

The tracking is performed well despite the appearance and disappearance of the ball and the motion discontinuities: The "future" frames allow to track the ball when it is appearing or just after it bounced. More traditionally, the ball can be tracked when disappearing or before it will

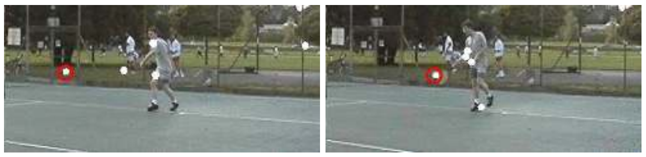


Figure 4: Two examples of ball detection results.

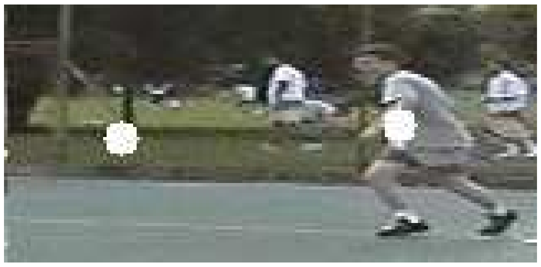


Figure 5: Estimation of the mixture parameters. The detected position at left corresponds to the ball and the right one is a false alarm. Their respective $\gamma_j$ parameter takes successively the values (0.45, 0.45), (0.72, 0.1), (0.73, 0.08).

bounce thankful to the considered "past" frames. Two recovered trajectories are shown in Figures 1.a and 7.

## 3.2. Golf Club Tracking

The second example is golf club tracking during a swing. The difficulties are first in the detection because the club is thin and reflective. Furthermore, during a swing, the club velocity is very high and its acceleration varies significantly during the transition between upswing and downswing.

### 3.2.1. Club and Club Motion Model

The golf literature provides a good swing model, called the double-pendulum model [7]. This model consists of two levers, hinged in the middle. The upper lever roughly corresponds to the golfer's shoulders and arms, while the lower lever corresponds to the club and the "hinge" between them corresponds to the wrists and hands. The hinge works only in a single plane in which the upper lever is swung about its fixed pivot at the top which is roughly located in the middle of the golfer's upper chest. The double-pendulum model is three-dimensional, and we consider here its orthographic projection in the image plane of a camera placed in front of the golfer. As shown in Figure 8, this projection can be parameterised by:
- the 2D point $C$, the fixed pivot;
- the 2D distance $R$ between $C$ and the second pivot;
- the 2D length $l$ of the club;
- the angle $\Psi$ between the upper lever and the $u$-axis;
- the angle $\varphi$ between the two levers.

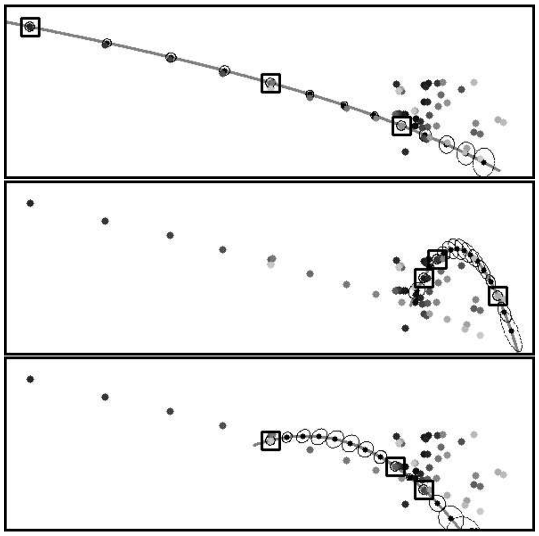


Figure 6: Three motion samples used for a frame of the trajectory shown in Figure 7, with likelihoods respectively equal to 0.0077, 0.0036 and 0.0045. The first sample, the one with the best likelihood, is close to the actual motion. The squares indicate the hypotheses employed to compute the sample, the black disks correspond to the predictions, and the ellipses represent their covariances. The gray disks correspond to the detections, their gray level indicating their detection time.

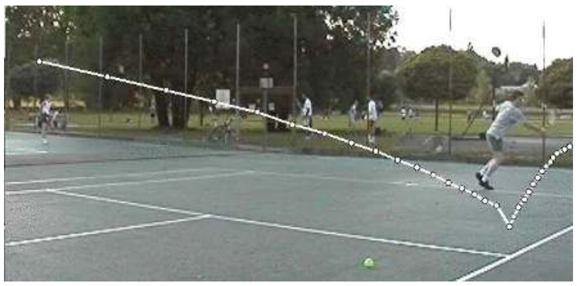


Figure 7: A recovered trajectory from a monocular sequence. Tracking is not lost after the ball has bounced.

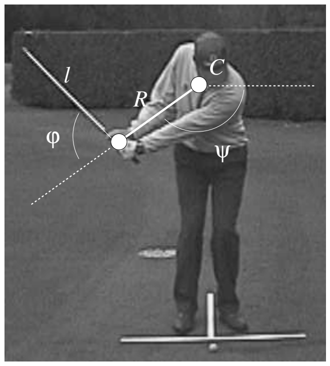


Figure 8: The double-pendulum model for a golf swing.

The point $C$ is assumed to be known, and the club measures are parametrised as $z = (R, l, \Psi, \varphi)$. We model the club dynamics as a second order process (including the 2D lengths $R$ and $l$: they are the projections of constant lengths in 3D and also varying). Given three club measures at different times $t_1$, $t_2$, $t_3$, the motion parameters can be computed by solving the linear systems $\mathbf{A}\mathbf{X}_p = \mathbf{B}_p$ with:

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix}$$

where $\mathbf{X}_p$ is a vector among

$$\left\{ \begin{bmatrix} R(0) \\ \dot{R}(0) \\ \ddot{R}(0) \end{bmatrix}, \begin{bmatrix} l(0) \\ \dot{l}(0) \\ \ddot{l}(0) \end{bmatrix}, \begin{bmatrix} \Psi(0) \\ \dot{\Psi}(0) \\ \ddot{\Psi}(0) \end{bmatrix}, \begin{bmatrix} \varphi(0) \\ \dot{\varphi}(0) \\ \ddot{\varphi}(0) \end{bmatrix} \right\}$$

and $\mathbf{B}_p$ the corresponding vector among

$$\left\{ \begin{bmatrix} R(t_1) \\ R(t_2) \\ R(t_3) \end{bmatrix}, \begin{bmatrix} l(t_1) \\ l(t_2) \\ l(t_3) \end{bmatrix}, \begin{bmatrix} \Psi(t_1) \\ \Psi(t_2) \\ \Psi(t_3) \end{bmatrix}, \begin{bmatrix} \varphi(t_1) \\ \varphi(t_2) \\ \varphi(t_3) \end{bmatrix} \right\}.$$

The covariance on the prediction $y(t)$ can be computed as in the previous example. As before, unrealistic predicted motions (based on the computed velocities and accelerations) are rejected.

### 3.2.2. Club Position Hypotheses Generation

The club detection is done as follows. First we detect the moving objects using the same technique as before. Segment detection is then performed on these objects. We consider only pairs of close parallel segments because they are a good cue for the club position in the image. Then the two segments of each pair are merged in one segment. Generally the resulting segments cover only a part of the shaft (even if they correspond to the club) because the segment detection does not work as well as we would like. The club extremities are then estimated by looking for color discontinuities along the segment. At this stage, we do not know which extremity corresponds to the club head, and each detection give us two hypotheses $(R_1, l, \Psi_1, \Phi_1)$ and $(R_2, l, \Psi_2, \Phi_2)$.

### 3.2.3. Golf Club Tracking Results

We have tested our algorithm on several sequences of various swings over a cluttered background. The average number of detections is about 5, and $P_d$ varies between 0.8 and 0.95. During a swing, the shaft acceleration is not constant and can take on a large range of values. We use the values $n_B = n_A = 5$, so the acceleration of the shaft is assumed to be approximately constant over an interval of 11 frames.

To demonstrate the robustness of our algorithm to successive mis-detections and false-alarms, we have manually replaced the correct detections by false-alarms when

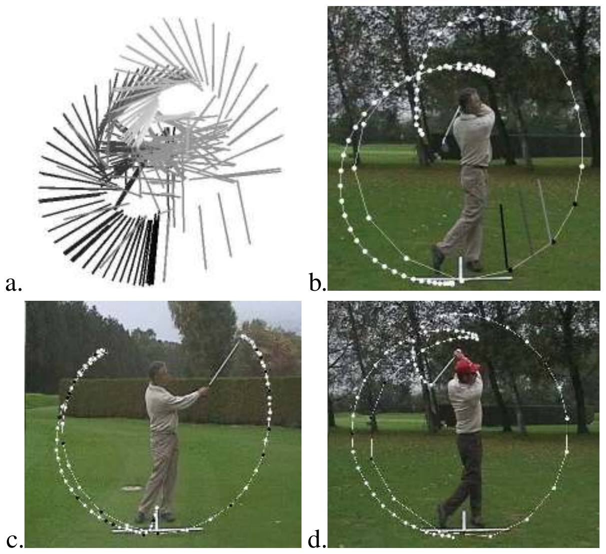


Figure 9: (a) detections over the first golf sequence and (b) the tracking result; (c) and (d) other results for different swings. The black disks corresponds to the club head when the club has not been detected.

the golf club velocity is particularly high for the sequence shown Figure 9.a and b. Figure 9.b presents the tracking result and the three added false alarms. In spite of this "trap" and the high velocity and high variations of the acceleration, the tracking performs remarkably well.

The Figure 9.b shows the robustness of the tracker to mis-detections and false-alarms (the club detections over time for this sequence are shown Figure 9.a). The movie attached to this paper shows the tracking and the detections frame by frame. Figure 9 also presents the recovered club head trajectory for different swings. Some correct detections have been randomly removed to test the robustness, but nevertheless the tracking is unperturbed. All the same parameters values have been used for these sequences.

## 4. Comparison with the Recursive Approach

In classical recursive approach, a state $\mathbf{x}_t$ represents the current estimate at time $t$ of the target parameters. In Kalman filter, this state is a single hypothesis of the target position and dynamics and an associated covariance matrix. In Multiple Hypothesis Tracking, it is represented by a mixture of Gaussians. In particle-set tracking [8], the particles both represent the multiple hypothesis and the state density.

In all cases, the tracking relies on the same principle: The states $\mathbf{x}_t$ embed information about target dynamics estimated from previous observations; the state in the coming frame is predicted using a recursive motion model and its density is estimated according to the observation in this frame. When the target is not detected or is occluded in successive frames, the prediction becomes less and less accurate, and the error estimate becomes grosser and grosser. In particle-based trackers, this also supposes a particularly large number of particles to represent the error. If then an abrupt change in the direction of motion then occurs, it would probably defeat the tracker.

On the contrary, our approach is robust enough to tolerate weak motion models, even under difficult conditions. For example, during the golf upswing and even more during the downswing, the constant acceleration motion model is a very poor approximation of reality. These motions feature large accelerations and an abrupt change in the direction of motion, that would probably defeat kalman-based approaches. Our approach also robust enough to handle the frequent false-negatives and false-positives that detecting a thin object such as a golf club entails.

## 5. Conclusion and Future Works

We have presented a new approach to tracking. It relies on fitting a local motion model to detections over time using a robust algorithm. We demonstrated this approach in an online tracking framework. Since our algorithm considers an interval of frames to estimate the target motion, it is more robust and accurate than classical recursive tracking algorithms. It can be employed for applications that allow a small delay between the acquisition of a frame and the tracker output for this frame.

Our first example consisted of the tracking of a tennis ball that goes into and outside the camera field of view and bounces when hitting the ground and the tennis racket. Our algorithm is not perturbed by these discontinuities and tracks the ball without any manual initialization. Our second example, the golf swing tracker, demonstrated that our method is also robust to successive mis-detections and false-alarms even when the target velocity is particularly large.

We can imagine a number of improvements to this work. In this paper we assume that the target is always visible. The fact that the target may not be visible should be integrated into the method. The tracker should also be able to take into account several targets. Another very promising extension is to consider multiple motion models: By recovering the actual motion model that fits the data, this method could be employed for motion recognition.

Since it relies on the data association principle, our algorithm considers only discrete targets. Nevertheless, it should be easily feasible to use view-based detection and consider more complex objects than in our examples, like human face or body, making our algorithm a reliable alternative to popular particle-based trackers.

## References

[1] M. Isard and A. Blake, "A smoothing fi lter for condensation," in *ECCV*, 1998, pp. 767–781.

[2] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, December 1979.

[3] I.J. Cox and S.L. Hingorani, "An effi cient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, February 1996.

[4] M.A Fischler and R.C Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[5] P.H.S. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[6] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, 1988.

[7] A. Cochran and J. Stobbs, *Search for the Perfect Swing*, Triumph books, 1996.

[8] M. Isard. and A. Blake, "CONDENSATION - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, August 1998.