# Tracking of Multiple Bouncing Balls Using Joint Probabilistic Data Association Particle Filters

Carl Daniel Ahlberg
ahlberg2@kth.se

January 24, 2021

**Abstract**

In this report the Joint Probabilistic Data association Algorithm is implemented in an attempt to track multiple bouncing balls. By using particle filters as the method of estimation the algorithm is capable of uniquely tracking multiple noisy targets, both represented as simulated balls and real tennis balls extracted from recordings.

## 1 Introduction

In the field of object tracking measurements are often noisy, filled with clutter or entirely non-existent. Hence, additional methods are required to determine the state of an object as the measurements cannot solely be relied on. Traditionally recursive state estimation, such as a particle filter (PF) [1], is used to obtain an estimation of an object state. This approach is sufficient when tracking a bouncing ball for instance. However, the problem becomes more complex when attempting to track multiple object simultaneously, such as several bouncing balls. Data association becomes a problem as each ball detection, or misdetection, has to be associated to the correct state estimation. Another problem regarding multi object tracking is the increased computational complexity stemming from the greater number of combinations between the measurements and the hypotheses. The Joint Probabilistic Data Association (JPDA) [2] and the Multiple Hypothesis Tracker (MHT) [3] algorithm are examples of algorithms capable of maintaining multiple hypothesis in environments with clutter and missing detections. JPDA is computationally more efficient compared to MHT, however it requires knowledge of the maximum number of targets, which MHT does not. This report tackles the problem of tracking a known number of bouncing balls, hence JPDA will be used. The original implementation of JPDA, using a Kalman filter for state estimation, has been modified to use a PF as an alternative estimation method in approaches such as [4] to track autonomous targets. This is referred to as Monte-Carlo JPDA (MC-JPDA) and this report will largely exploit this technique to estimate the position of multiple bouncing balls with noisy measurements without knowledge of the initial states.

## 2 Related work

The JPDA algorithm was originally proposed in [2] and builds on the Probabilistic Data Association (PDA) method where posterior association probabilities are calculated for all valid measurements of a target, which can be used to update the target state. A fundamental assumption of the PDA algorithm is the independence of a target to the remaining targets. JPDA instead calculates the posterior association probabilities jointly for the set of all targets and measurements. The benefit of a joint calculation is the ability for a single target to better account for false measurements, either clutter or measurements originating from other targets. The computational requirements of JPDA are high as the joint probability is computed for all targets and measurements. In the original implementation of JPDA, the authors proposed a *gating* procedure to reduce complexity by only considering measurements within a bound for each target. In [4] MC-JPDA was used to track 9 randomly wandering autonomous targets. An additional interaction graph between all targets was constructed, which in the tested simulated environment increased tracking efficiency. A unique PF was assigned to all targets, though this approach required a known initial position for each target.

Tracking of soccer players using MC-JPDA was implemented in [5]. 3 soccer players were tracked without any information regarding the initial state. A unique filter was assigned to each player by initializing a new PF only when the particles of the previous filter had converged below a certain covariance threshold. The tracking results from the MC-JPDA displayed the ability to cope with partial player occlusion and to recover after a temporary loss of measurements.

## 3 Data gathering

Data of multiple bouncing balls was acquired through both a simulated environment and recordings of real bouncing tennis balls.

### 3.1 Simulation

Ground truth data of the real bouncing balls was hard to extract from the recordings, therefore a simulated environment was instead used to measure the performance of the implemented MC-JPDA algorithm and to test specific scenarios which could not easily be replicated with real tennis balls. The state transition model of the simulated balls was represented in two ways: A bouncing and a non-bouncing state transition.

Parameters to simulate noisy measurements were included in the simulation in addition to randomizing the velocity after a bounce, thus emulating the behavior of a bouncing spinning ball in a simplistic way. The parameters of the simulation are:

| | |
|---|---|
| $\Sigma_{R_{sim}(2\times2)}$ | Measurement covariance |
| $\sigma^2_{\dot{x}}$ | Variance of $x$ velocity during bounce |
| $\sigma^2_{\dot{y}}$ | Variance of $y$ velocity during bounce |
| $P_{D_{sim}}$ | Probability of ball detection |
| $P_{FP_{sim}}$ | Probability of a false positive detection |

### 3.2 Real Data

Up to 4 bouncing tennis balls were recorded simultaneously at 40 fps. The balls were primarily extracted through a blob detection in OpenCV, which apart from the correct measurements produced a number of false positives.

## 4 Method

A PF was chosen as it is suggested in [1] that a PF can be preferable over Kalman filtering methods when dealing with data association problems. Additionally, the bounce of the balls in the used data can produce seemingly random change in velocities. This model uncertainty favors the use of a nonparametric filter such as the PF. Using Python as a programming language this project implements a similar JPDA algorithm to that proposed in [5], but a state transition model of a bouncing ball is used opposed to a random prediction of the next state. The state of a ball is described as

$$\boldsymbol{x}_t = \begin{pmatrix} x_t & y_t & \dot{x}_t & \dot{y}_t \end{pmatrix}, \tag{1}$$

where $x$ and $y$ is the position with the corresponding $\dot{x}$ and $\dot{y}$ velocities of a ball. The predicted state $\hat{\boldsymbol{x}}_t$ is calculated using the previous state $\boldsymbol{x}_{t-1}$ applied to the motion model with added process noise $P \sim \mathcal{N}(0, \Sigma_{P(4\times4)})$. Likewise, the measurements

$$\boldsymbol{z}_t = \begin{pmatrix} x_t & y_t \end{pmatrix} \tag{2}$$

contain noise described as $R \sim \mathcal{N}(0, \Sigma_{R(2\times2)})$. Given targets $\boldsymbol{X}_t = \{\boldsymbol{x}_t^1, ..., \boldsymbol{x}_t^K\}$ and observations $\boldsymbol{Z}_t = \{\boldsymbol{z}_t^1, ..., \boldsymbol{z}_t^M\}$ at time $t$, where $K$ is a known number of maximum targets, JPDA calculates the probability of a target $\boldsymbol{x}_t^k$ being associated to an observation $\boldsymbol{z}_t^j$. This can be denoted by the variable $\beta_{kj}$, which is thoroughly described in [4]. The likelihood of the observation $\boldsymbol{Z_t}$ given a target $\boldsymbol{x}_t^k$ can be calculated as

$$p(\boldsymbol{Z}_t|\boldsymbol{x}_t^k) = \sum_{j=0}^{M_t} \beta_{kj} p(\boldsymbol{z}_t^j|\boldsymbol{x}_t^k). \tag{3}$$

When $j = 0$, $z_t^0$ is modeled as clutter to account for situations when the target $\boldsymbol{x}_t^k$ is missing an observation. $p(\boldsymbol{Z}_t|\boldsymbol{x}_t^k)$ is calculated for each target and is incorporated in the weight assignment of the standard PF algorithm. As a part of the joint association probability calculation, two variables need to be approximated. These variables are the probability of a detection, $P_D$, and the probability of a false positive detection, $P_{FP}$.

To reduce the computational time of the implemented algorithm a gating procedure was added which removes measurements outside a specified radius for each target, thus reducing the number of calculated joint probabilities per target. This radius was set to a constant as it was sufficient for this project, although a more dynamic solution is to set this radius according to the covariance of the particles for a target.

The initial position of the balls are assumed to be unknown, which is made more difficult as several PF exists and each ball should map to a unique filter. To avoid two or more filters mapping to the same ball only one PF is initialized at the start of the data recording. When the particles of this filter has a covariance below a certain threshold a new filter is initialized. The procedure is identical for the rest of the filters until the number of filters have reached the known maximum number of targets.

# 5 Simulated results

A number of simulations were created, with varying number of balls, initial states and different degrees of process and measurement noise. A scenario with 3 balls will be used to evaluate the performance of the algorithm. The number of particles was set to 800 per filter and the simulation ran for 100 time steps. The following parameter values were used for the simulation and the MC-JPDA algorithm:

| | |
|---|---|
| $\Sigma_{R_{sim}}$ | diag(0.09, 0.09) |
| $\sigma^2_{\dot{x}}$ | 0.04 |
| $\sigma^2_{\dot{y}}$ | 0.04 |
| $P_{D_{sim}}$ | 0.95 |
| $P_{FP_{sim}}$ | 0.2 |
| $\Sigma_R$ | diag(0.09, 0.09) |
| $\Sigma_P$ | diag(0.04, 0.04, 0.04, 0.04) |
| $P_D$ | 0.9 |
| $P_{FP}$ | 0.1 |

Due to a slow convergence of the particles the value of $P_{FP}$ was set lower compared to $P_{FP_{sim}}$ and $P_D$ was higher than $P_{D_{sim}}$. The balls were simulated such that they would be close to each other with overlapping trajectories at certain time steps, as seen in Figure 1. This was tested to determine whether the algorithm could keep track of the correct target or if switching between different targets would occur. The mean absolute error (MAE) and the variance was calculated for all state variables and the results are portrayed in Table 1.

| | $x$ [m] | $y$ [m] | $\dot{x}$ [m/s] | $\dot{y}$ [m/s] |
|---|---|---|---|---|
| **Ball 1** | | | | |
| MAE | 0.338 | 0.365 | 0.415 | 1.362 |
| Variance | 0.782 | 0.077 | 0.339 | 4.477 |
| **Ball 2** | | | | |
| MAE | 0.472 | 0.425 | 0.664 | 1.908 |
| Variance | 1.948 | 0.745 | 0.431 | 5.590 |
| **Ball 3** | | | | |
| MAE | 0.191 | 0.673 | 1.354 | 3.270 |
| Variance | 0.014 | 1.189 | 0.329 | 8.357 |

Table 1: Results of the state estimation for 3 balls with the trajectory of the real balls shown in Figure 1.

A graph depicting the MAE for ball 2 is shown in Figure 2. What is notable in both Table 1 and Figure 2 is the high variance relative to the MAE, especially for $\dot{y}$. This is caused by two factors: the high error prior to the convergence of the filter to a target, and the error spikes which can be seen at time step 38, 62 and 95. These spikes are

caused by the bounce of the ball since a portion of all particles will have a negative $y$ velocity relative to the true velocity. Another noteworthy anomaly is the high $\dot{y}$ error between time step 38 and 62. This error coincides with a period where ball 1 and 2 are in very close proximity to each other, thus increasing the probability of an incorrect target localization. Though, this is not as heavily reflected in the other state variable errors. Overall, all filters managed to follow a unique target without any noticeable switching to incorrect targets. The MC-JPDA algorithm was additionally capable of handling the noisy measurements and false positives introduced in the simulation. However, the particles struggle to converge around the true state if too many false positives are present at a filter initialization.

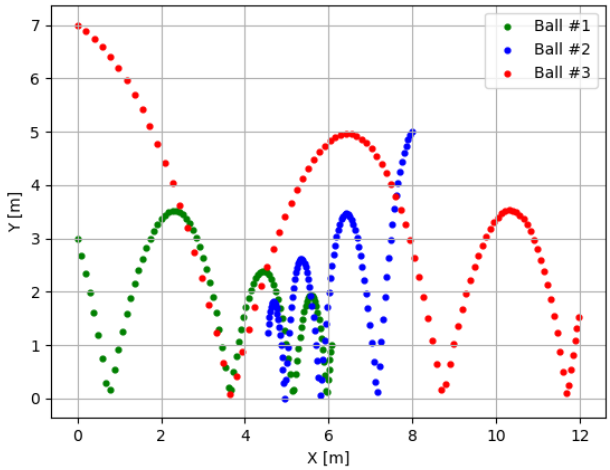



Figure 1: The trajectory of three bouncing balls, ball number 1 was the first to be localized, followed by number 2 and finally number 3.

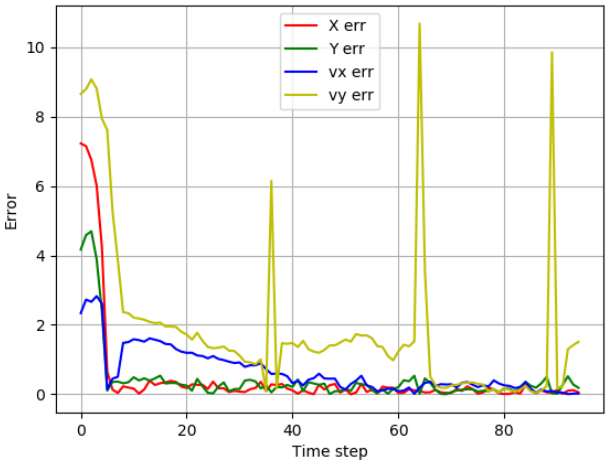



Figure 2: The error over time for ball 2 in Table 1. vx represents $\dot{x}$ and vy $\dot{y}$.

# 6 Real data results

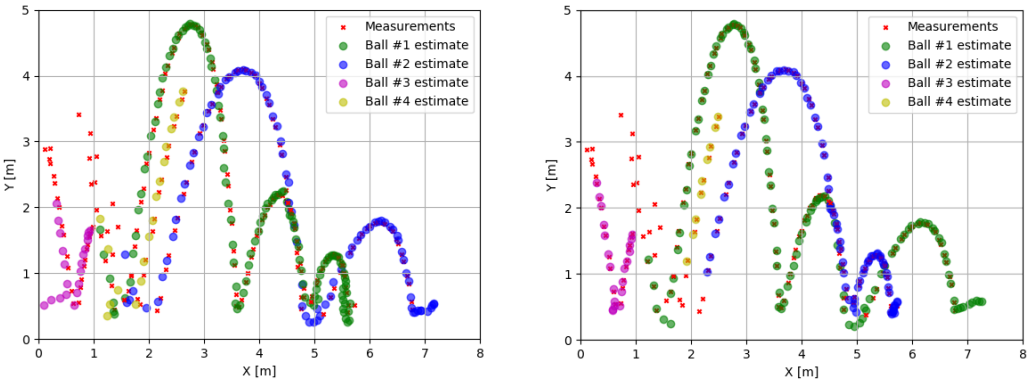
As the ground truth position of the balls were not extracted, the results from the real data is purely based on visual inspection and will focus on the ability of a filter to maintain a hypothesis for a unique ball.

Multiple recordings were tested and a partial trajectory of a recording with 4 balls is depicted in Figure 3. This recording contained 219 frames and the number of particles used per filter was 1000, the remaining MC-JPDA parameters were set as:

| | |
|---|---|
| $\Sigma_R$ | diag(0.04, 0.04) |
| $\Sigma_P$ | diag(0.09, 0.09, 0.09, 0.09) |
| $P_D$ | 0.92 |
| $P_{FP}$ | 0.01 |

What can be observed from Figure 3(a) is the ability of the algorithm to track a unique target, even when the trajectories of two balls overlap. However, when no measurements are available for a target the estimate quickly diverges from the true state due to a high $\Sigma_P$ value. Furthermore, the MC-JPDA algorithm used on the gathered recordings is highly sensitive to changes in parameter values. For instance, if the number of particles is reduced to 900 a switch of targets can occur during a trajectory overlap. This is visualized in Figure 3(b). The switch is likely caused by a high $\dot{y}$ error as the state values have not fully converged.

The algorithm had a mean processing time of 0.46 s per frame, though this processing time differs significantly depending on how many filters are initialized and how many observations are present in a frame. As an example, when 2 balls are tracked on the same recording the mean processing time decreased to 0.29 s per frame.




(a) Correct tracking

(b) Incorrect tracking

Figure 3: Trajectory of 4 bouncing tennis balls. Correct tracking seen in (a) using 1000 particles. A switch to the incorrect target at frame 82 between ball 1 and 2 displayed in (b) using 900 particles.

# 7 Conclusion

In this project the MC-JPDA algorithm was implemented to track bouncing balls in a simulation and in real recordings. The results gathered from the simulation were satisfactory as all balls are tracked correctly despite the presence of noise. The tracking of the real balls was successful, however if the error in state velocities is high a target switch can happen when trajectories intersect. Overall, the implemented algorithm can be used to track bouncing balls when measurements are frequent for each ball and the total number of balls are known. An improvement could be better track management, such as the removal of a filter if a target has disappeared for a number of frames. This leaves room for another filter to be initialized to track new targets.

Finally, the computational performance of the algorithm can be increased by dynamically changing the gate size of each PF and by transferring the implementation to a faster language, such a C++. These improvements might make the algorithm suitable for real-time multi ball tracking.

# References


[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probalistic robotics*. Emerald Group Publishing Limited, 2006.

[2] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983.

[3] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[4] Arsene Fansi Tchango, Vincent Thomas, Olivier Buffet, Alain Dutech, and Fabien Flacher. Tracking multiple interacting targets using a joint probabilistic data association filter. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

[5] M. Jaward, Lyudmila Mihaylova, N. Canagarajah, and D. Bull. Multiple object tracking using particle filters. volume 2006, page 8 pp., 01 2006.